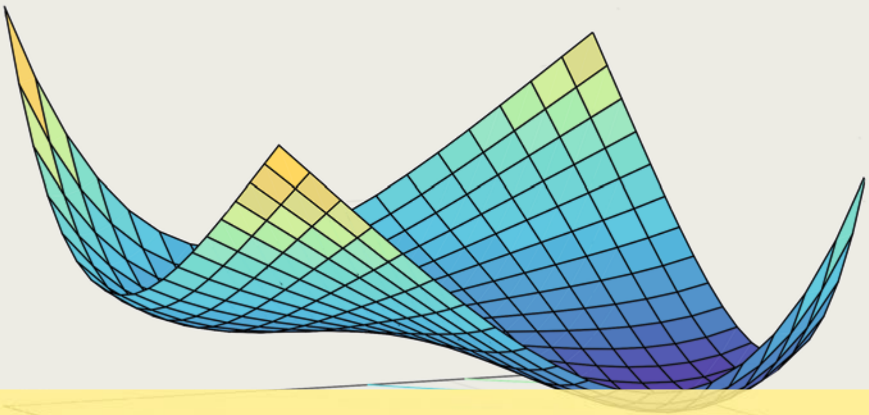


TEXTS IN COMPUTATIONAL SCIENCE AND ENGINEERING 13

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

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.

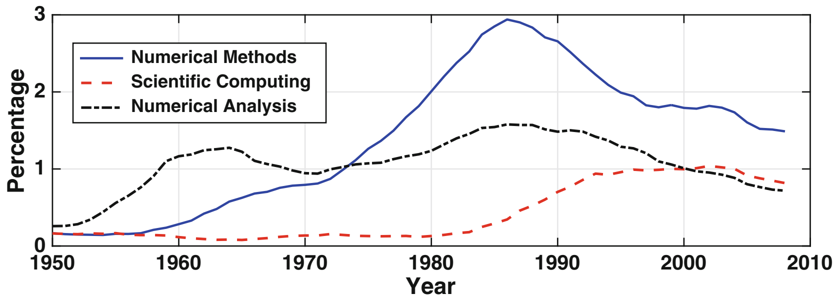


Figure 1 Historical record according to Google. The values are the number of instances that the expression appeared in a published book in the respective year, expressed as a percentage for that year, times 10^5 [Michel et al., 2011].

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.

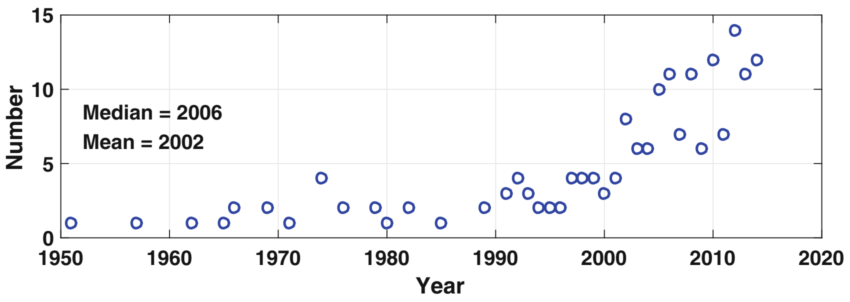


Figure 2 The number of references in this book, after 1950, as a function of the year they were published.

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

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

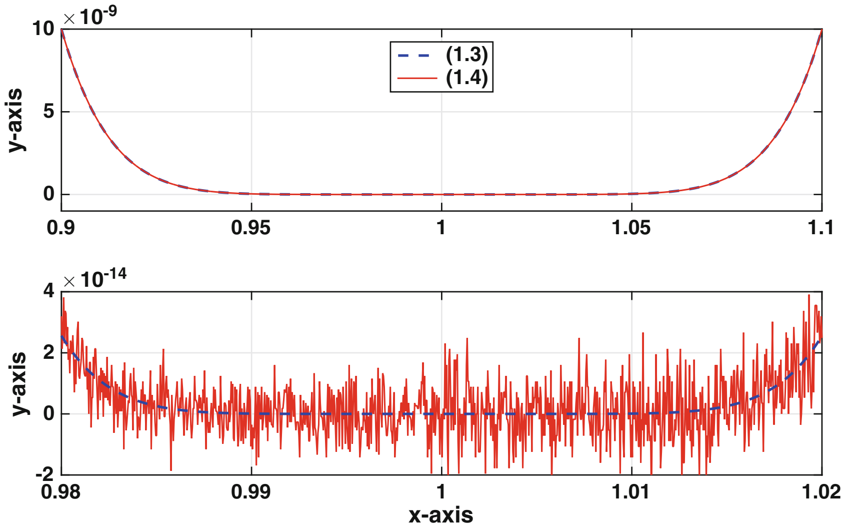


Figure 1.1 Plots of (1.4) and (1.3). Upper graph: the interval is $0.9 \leq x \leq 1.1$, and the two functions are so close that the curves are indistinguishable. Lower graph: the interval is $0.98 \leq x \leq 1.02$, and now they are not so close.

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

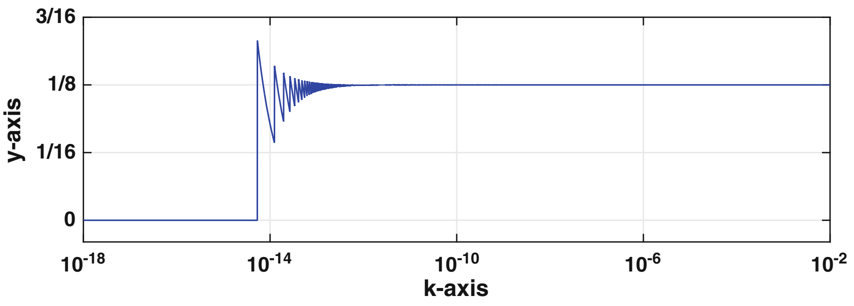


Figure 1.2 Plot of (1.5).

and then, after k drops a little below 10^{-14}, the computer decides that $y = 0$. It is also worth pointing out that ratios as in (1.5) arise in formulas for the numerical evaluation of derivatives, and this is the subject of Section 7.2. In particular, (1.5) comes from an approximation used to evaluate $f'(0)$, where $f(x) = \sqrt{16 + x}$. ■

Example 4

The final example concerns evaluating trigonometric functions, specifically, $\sin(x)$. It is well-known that if n is an integer, then $\sin(n\pi) = 0$. The integers of interest here are of the form $n = 2^k + 1$, where k is a positive integer. Using MATLAB one finds that if $k = 52$ then $\sin(n\pi) = -0.3021$, while if $k = 53$, then $\sin(n\pi) = -0.8926$. Obviously, neither value is close to zero. To investigate this further, the absolute value of $y = \sin((2^k + 1)\pi)$ is plotted as a function of k in Figure 1.3. Interestingly, the error grows almost monotonically, eventually getting close to one. It is also worth pointing out that this result is not limited to MATLAB and as an example, the same curve is obtained if the valves are computed with Python. ■

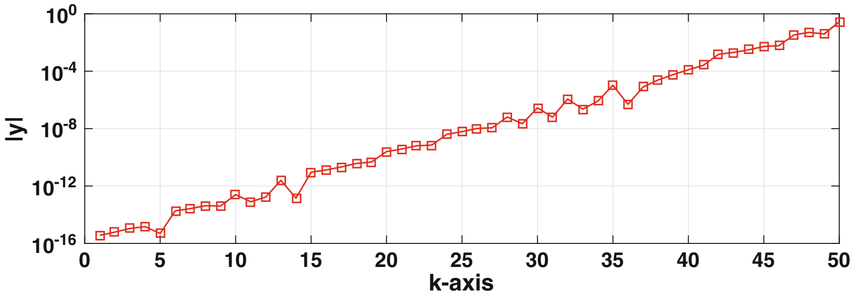


Figure 1.3 The absolute value of $y = \sin(n\pi)$, where $n = 2^k + 1$, as computed by MATLAB.

1.2 Floating-Point Number System

The problems illustrated in the above examples are minor compared to the difficulties that arose in the early days of computing. It was not unusual to get irreproducible results, in the sense that two different computers would calculate different answers to the same formula. To help eliminate this, a set of standards was established that computer manufactures were expected to comply with. The one of interest here concerns the floating-point system, and it is based on the IEEE-754 standard set in 1985. It consists of normal floats (described below), along with zero, ± *Inf*, and *NaN*.

1.2.1 Normal Floats

Normal (or normalized) floating-point numbers are real numbers that the computer has the exact value for. The form they are written in is determined by the binary nature of computer systems. Specifically, they have the form

$$x_f = (\pm)\, m \times 2^E, \tag{1.6}$$

where

$$m = 1 + \frac{b_1}{2} + \frac{b_2}{2^2} + \cdots + \frac{b_{N-1}}{2^{N-1}}\,. \tag{1.7}$$

In this representation m, E, and the b_i's have the following properties:

- m: This is the *mantissa*. The b_i's are either zero or one, and for this reason $1 \le m < 2$ (see Exercise 1.19).
- E: This is the *exponent* and it is an integer that satisfies $E_m \le E \le E_M$. For example, for double precision, $-1022 \le E \le 1023$. In general, according to the IEEE requirements, $E_m = -E_M + 1$ and $E_M = 2^{M-1} - 1$, where M is a positive integer.

As defined, a floating-point system requires specification of the two integers N and M, and from this the normal floats can be determined using (1.6) and (1.7). Some of the standard choices are listed in Table 1.2. The one of particular importance for scientific computing is double precision, for which $N = 53$ and $M = 11$.

Examples

1. $3 = 2 + 1 = \left(1 + \frac{1}{2}\right) \times 2$

In this case, $E = 1$, $b_1 = 1$, and the other b_i's are zero. This means that $x = 3$ is a floating-point number. ■

2. $-10 = -8 - 2 = -\left(1 + \frac{1}{2^2}\right) \times 2^3$

   In this case, $E = 3$, $b_2 = 1$, and the other b_i's are zero. This means that $x = -10$ is a floating-point number. ■

3. An irrational number is not a normal float. ■

It is worth looking at the values of m coming from (1.7). The smallest value occurs when all of the b_i's are zero, which gives $m = 1$. The next largest value occurs when $b_{N-1} = 1$ and all of the other b_i's are zero. In other words, the next largest is $m = 1 + \varepsilon$, where

$$\varepsilon = \frac{1}{2^{N-1}} . \tag{1.8}$$

The next largest value of m occurs when $b_{N-2} = 1$ and all of the other b_i's are zero, which gives us $m = 1 + 1/2^{N-2} = 1 + 2\varepsilon$. This pattern continues, and one ends up concluding that $m = 1, 1+\varepsilon, 1+2\varepsilon, 1+3\varepsilon, \cdots, 1+K\varepsilon$, where $K = 2^{N-1} - 1$. If you are curious how the value of K is determined, you should look at Exercise 1.19. Also, the number ε defined above plays a special role in the floating-point number system and it is called *machine epsilon*.

Examples

1. $x = 1$ is a floating-point number (take $m = 1$ and $E = 0$) and the floating-point number just to the right of $x = 1$ is $x_f = 1 + \varepsilon$, where ε is given in (1.8). ■

2. What is the floating-point number just to the left of $x = 1$?

   Answer: Between $x = 1/2$ and $x = 1$ the floats have the form $m \times 2^{-1}$. We need the largest value of m, which is $m = 1 + K\varepsilon$. Noticing that $K = \varepsilon^{-1} - 1$, then the float just to the left of $x = 1$ is

$$\begin{aligned} x_f &= (1 + (\varepsilon^{-1} - 1)\varepsilon) \times 2^{-1} \\ &= (2 - \varepsilon) \times 2^{-1} \\ &= 1 - \frac{1}{2}\varepsilon. \quad ■ \end{aligned}$$

Given any nonzero real number, the computer will attempt to approximate its value with the closest normal float. This requires a rule for rounding, which is explained in Section 1.2.3, and rules for what happens if the number is too big or very close to zero, which are explained in Section 1.2.4.

It is possible to have computer programs use multiple floating-point systems at the same time. For example, in C and FORTRAN you can declare variables to be either single or double precision (see Table 1.2). However, the default assumption in scientific computing is that double precision is used (this is what MATLAB uses).

1.2.2 Machine Epsilon

One floating-point number that plays a critical role in this textbook is known as *machine epsilon*. This is designated as ε, and it is given as

$$\varepsilon = \frac{1}{2^{N-1}}\,.$$

In the case of double precision, $\varepsilon \approx 2 \times 10^{-16}$. Because of its importance, most computer systems have a special variable set aside for ε. For example, in MATLAB machine epsilon is designated as `eps`.

Why is ε so important? The primary reason is that it is used to determine the relative accurately of a floating-point number, and this will be explained below. A related reason is that ε can be used to determine the spacing of the floating-point numbers. To explain, recall that the values for the mantissa are $m = 1, 1+\varepsilon, 1+2\varepsilon, 1+3\varepsilon, \cdots, 1+K\varepsilon$. As defined in (1.6), the floating-point numbers between $x = 1$ and $x = 2$ have the form $m \times 2^E$, where $E = 0$. This means they are a distance ε apart. Similarly, the floating-point numbers between $x = 2$ and $x = 2^2$ are a distance of $\varepsilon \times 2$ apart, and between $x = 2^2$ and $x = 2^3$ they are a distance of $\varepsilon \times 2^2$ apart. One consequence of this is that for large values of x the distance between the floats can be huge. For example, between $2^{100} \approx 10^{30}$ and $2^{101} \approx 2 \times 10^{30}$ they are a distance of $\varepsilon \times 2^{100} \approx 2.8 \times 10^{14}$ apart. The fact that they are so far apart can have dire consequences for some calculations, and a particular example will be considered in Section 1.2.6.

There are other normal floats that are occasionally useful enough that they should be mentioned.

- *largest positive*: This is $x_M = \left(1 - \frac{\varepsilon}{2}\right) \times 2^{E_M+1}$.
  In MATLAB this is denoted as `realmax`.
- *smallest positive*: This is $x_m = 2^{E_m}$.
  In MATLAB this is denoted as `realmin`.

Precision	**N**	**M**	E_m	E_M	**Smallest Positive** (x_m)	**Largest Positive** (x_M)	ε	**Decimal Digits**
Single	24	8	-126	127	1.2×10^{-38}	3.4×10^{38}	10^{-7}	7
Double	53	11	-1022	1023	2.2×10^{-308}	1.8×10^{308}	2×10^{-16}	16
Quadruple	113	15	-16382	16383	3.4×10^{-4932}	10^{4932}	10^{-34}	34

Table 1.2 Values for various floating-point systems specified by IEEE-754 and its extensions. The values for the smallest positive, largest positive, and machine epsilon are only given to one or two significant digits. Similarly, the number of decimal digits is approximate and is determined from the expression $N \log_{10} 2$. Also note that double precision is the default for scientific computing.

1.2.3 Rounding

Assuming x is a real number satisfying $x_m \le |x| \le x_M$ then in the computer this is rounded to a normal float x_f, and the relative error satisfies

$$\frac{|x - x_f|}{|x|} \le \frac{\varepsilon}{2}\,.$$

To do this it uses a "round to nearest" rule, which means x_f is the closest float to x. So, for example, in Figure 1.4 if $1 - \frac{\varepsilon}{4} < x < 1 + \frac{\varepsilon}{2}$ (Region II) then $x_f = 1$, while if $1 + \frac{\varepsilon}{2} < x < 1 + \frac{3\varepsilon}{2}$ (Region III) then $x_f = 1 + \varepsilon$. In the case of a tie, it uses a "round to even" rule, where it picks the nearest float with an even least significant digit.

1.2.4 Nonnormal Floats

To complete the floating-point system, a few additional terms are needed. The ones most relevant to our objective of numerical computing are described below.

Zero

It is not possible to represent zero using (1.6), and so it must be included as a special case. Correspondingly, there is an interval $-x_0 < x < x_0$ where any number in this interval is rounded to $x_f = 0$. The fact that a nonzero number is rounded to zero is the cause of many problems in numerical computing. For example, when this is done, an expression such as $1/x$ has no meaning. The exact value of x_0 is not of particular importance, although for MATLAB, $x_0 \approx 3 \times 10^{-324}$. The reason $x_0 < x_m$ is that there are additional floats between x_0 and x_m, what are called subnormal floats, that are to help reduce the divide by zero problem.

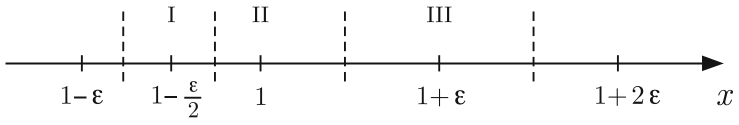


Figure 1.4 The two floating-point numbers just to the left and right of $x = 1$. The dashed lines are located half-way between the floats and any real number between them is rounded to the floating-point number in that subinterval.

Inf and NaN

Positive numbers larger than x_M are either rounded to x_M, if close enough, or assigned the value of `Inf`. The latter is a situation known as positive overflow. A similar situation occurs for very negative numbers, something called negative overflow and it produces the value of $-$`Inf`. For these situations when the calculated value is ill-defined, such as 0/0, the floating-point system assigns it a value of `NaN` (Not a Number). Needless-to-say, if you get a `NaN` then the calculation must be modified in some way.

Integers

Integers play an important role in programming, and examples include the counter used in a *for* or *do* loop, as well as the indices of a vector or matrix. Not all integers are part of the floating-point system and round-off in such cases is a problem. To avoid this, most computer systems have a way to treat integers as integers, where addition and subtraction are done exactly as long as the integers are not too big. For example, in C you can use the type declaration `int` to identify a variable as an integer while in FORTRAN it is understood that any variable beginning with the letters i, j, k, l, m, n is an integer. MATLAB does the typing automatically and will do integer arithmetic exactly whenever possible. It is able to do this as long as the values are less, in absolute value, than about 2^{53}.

There are aspects of the floating-point system that are not particularly important for developing the numerical algorithms considered in this text. For example, a number of textbooks describe the machine representation of a float, while others consider how the rules for arithmetic are affected using floats. As examples, if $x + y = z$ then you might wonder if it is true that $x_f + y_f = z_f$, or you might wonder if it is always true that $x_f + (y_f + z_f) = (x_f + y_f) + z_f$. For the record, the former is true while the latter is not. It is interesting to note that the non-associativity of floating-point addition has generated some difficulty in adapting algorithms to multicore processors. This is because the order of the numerical operations are affected by the way the problem is distributed between the cores, which means you can get different answers depending on how many cores you use. As you might expect, getting irreproducible results has generated considerable consternation [Shure, 2009]. To get some insight into how this problem is being solved, Demmel and Nguyen [2013] or Collange et al. [2015] should be consulted. For those interested in more detail related to floating-point arithmetic, they should consult ANSI/IEEE [1985], Goldberg [1991], Overton [2001], or Muller et al. [2010].

1.2.5 Flops

All numerical algorithms are judged by their accuracy and how long it takes to compute the answer. As one estimate of the time, an old favorite is to determine the flop count, where *flop* is an acronym for floating-point operation. To use this, it is necessary to have an appreciation of how long various operations take. In principle these are easy to determine. As an example, to determine the computing time for an addition one just writes a code where this is done N times, where N is a large integer, and then divides the total computing time by N. The outcomes of such tests are shown in Table 1.3, where the times are scaled by how long it takes to do an addition. Note that the actual times here are very short, with an addition taking approximately 6×10^{-10} sec. So, a program that involves 1.7 billion additions and multiplication should take less than a second. Because of this, even though $x = 3/2$ might take five times longer to compute than $x = 0.5 * 3$, it's really not necessary to worry about this (at least in the problems considered in this text).

A couple of comments need to be made about Table 1.3. First, using these numbers to accurately predict how long a calculation involving combinations of floats will take is difficult. Some systems have specialized instruction sets where certain operations are done in parallel. This includes simple combinations such as a multiply and addition, as well as dot products. A second comment is more of a question, and it relates to a problem in numerical computing. Namely, even though a computer provides values for functions like e^x and $\sin x$, just how accurate are these values? Most people who use computers pay little, if any, attention to this but, as will be explained next, this is something that is worth knowing about.

Operation	MATLAB Time	FORTRAN Time
Addition or Subtraction	1	1
Multiplication	1	1
Division	5	12
$\sqrt{x}$	24	18
$\sin x$	25	33
$\ln x$	50	18
e^x	19	19
x^n, for $n = 5, 10,$ or 100	134	15, 18, 28

Table 1.3 Approximate relative computing times for various floating-point operations in MATLAB (R2016a) and FORTRAN (gfortran v5.1.0). Note that they are normalized by the time it takes to do an addition.

1.2.6 *Functions*

Any computer system designed for scientific computing has routines to evaluate well-known or often used functions. This includes elementary functions like $\sqrt{x}$, transcendental functions like $\sin(x)$, e^x, and $\ln(x)$, and special functions like $\mathrm{erf}(x)$ and $J_\nu(x)$. To discuss how these fit into a floating-point system, these will be written in the generic form of $y = f(x)$. The ideal goal is that, letting y_f denote the computed value and assuming that $x_m \le |y| \le x_M$,

$$\frac{|y - y_f|}{|y|} \le \frac{\varepsilon}{2} .$$

Unfortunately, this does not apply to the current implementations of the floating-point system. It turns out that even for the elementary functions, guaranteeing correct rounding is difficult [Hanrot et al., 2007]. It is so difficult that it was intentionally left out of the IEEE-754 standard. The revised standard, IEEE-754 (2008) does consider this problem and makes recommendations for some of the elementary and transcendental functions. Note these are recommendations, or suggestions, and not requirements.

To illustrate how difficult it is to implement the IEEE-754 (2008) recommendations, suppose we want to evaluate $\sin(x)$ for larger values of x, say for $2^{53} < x < 2^{54}$. In this interval, using double precision, the distance between the floating-point numbers is $\varepsilon 2^{53} = 2$. This means that given x, the closest floating-point number the computer has for x is some number x_f in the interval $[x-1, x+1]$. It is very unlikely that the value of $\sin(x_f)$ is anywhere near the value of $\sin(x)$. This is the reason for the problem seen in Figure 1.3. To repeat the earlier example, using MATLAB one finds that $\sin((2^{52}+1)\pi) = -0.3021$, and $\sin((2^{53}+1)\pi) = -0.8926$. This type of error should be expected with any floating-point system (using double precision).

As illustrated in the above example, the low density of floating-point numbers for larger values of x makes it very difficult to accurately evaluate functions that oscillate over shorter distances. Fortunately, the situation for functions which are monotonic, such as $\exp(x)$ and $\ln(x)$, is much better. Those you might want to investigate some of the challenges related to accurate function evaluation should consult de Dinechin et al. [2004] or Muller [2005].

1.3 Arbitrary-Precision Arithmetic

Some applications, such as cryptography, require exact manipulation of extremely large integers. Because of their length, these integers are not representable using double, or even quadruple, precision. This has given rise to the idea of arbitrary-precision arithmetic, where the limitation is determined by the available memory for the computer. The price paid for this is that the computations are slower, with the computing time increasing fairly quickly as the size of the integers is increased.

As an example of the type of problem arbitrary-precision arithmetic is used for, there is the Great Internet Mersenne Prime Search (GIMPS). A Mersenne prime has the form $2^n - 1$, and considerable computing resources have been invested into finding them. The largest one currently known, which took 39 days to compute, has $n = 57{,}885{,}161$, which results in a prime number with 17,425,170 digits [GIMPS, 2015]. Just printing this number, with 3,100 digits per page, would take more than twelve times the pages in this text.

There are multiple computational challenges finding large prime numbers. One example is simply the difficulty of quickly multiplying large integers, and an illustration of how their binary representations can be used for this is touched on in Exercise 1.18. Those interested in the computational, and theoretical, underpinnings of computing primes should consult Crandall and Pomerance [2010].

1.4 Explaining, and Possibly Fixing, the Unexpected Results

The problem identified in Example 4, in Section 1.1, was discussed in Section 1.2.6. It is also analyzed in more depth in Exercise 1.17. What follows is a discussion related to the other examples that were presented in Section 1.1.

Example 1

The differences in the two sums are not unexpected when using double precision arithmetic. Also, the order of the error is consistent with the accuracy

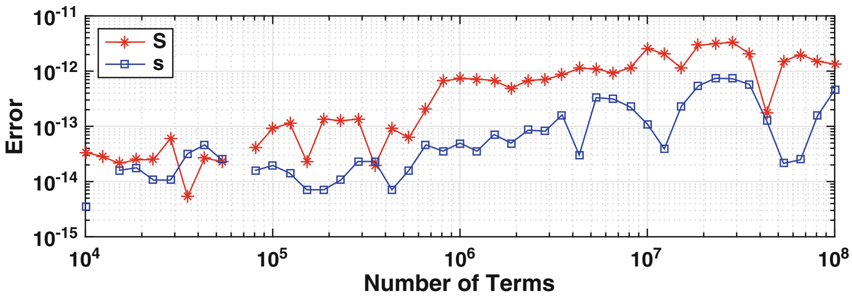


Figure 1.5 The error in computing the partial sum of the harmonic series using (1.1) and (1.2).

n	$E(n) - c(n)$	$E(n) - s(n)$
10^4	0	3.55×10^{-15}
10^5	0	1.95×10^{-14}
10^6	1.78×10^{-15}	4.97×10^{-14}
10^7	3.55×10^{-15}	1.10×10^{-13}
10^8	0	4.51×10^{-13}

Table 1.4 Comparison between compensated summation, as given in (1.9), and regular summation. Note $E(n)$ is the exact result, $c(n)$ is the value using compensated summation, and $s(n)$ is given in (1.2).

obtained for double precision. The question was asked about which sum might produce the more accurate result. One can argue that it is better to add from small to big. The reason being that if one starts with the larger terms, and the sum gets big enough, then the smaller terms are less able to have an affect on the answer. To check on this, it is necessary to know the exact value, or at least have an accurate approximate value. This can be found using something called the digamma function, from which one can show that for larger values of n,

$$\sum_{k=1}^{n} \frac{1}{k} = \ln(n+1) + \gamma - \frac{1}{2(n+1)} + O\left(\frac{1}{n^2}\right),$$

where $\gamma = 0.5772\cdots$ is Euler's constant. To investigate the accuracy of the two sums, the errors are shown in Figure 1.5. It is evident that for the most part, $s(n)$ serves as a more accurate approximation than $S(n)$. It is also seen that there is also a slow increase in the error for both, but this is not unusual when such a large number of floating-point calculations are involved (see Exercise 1.14).

Given the importance of summation in computing, it should not be surprising that numerous schemes have been devised to produce an accurate sum. A particularly interesting example is something called *compensated summation*. To explain how it works, consider the problem of calculating $\sum_{i=1}^{n} x_i$. The compensated summation procedure is as follows:

```
let:   $sum = 0$ and $err = 0$
loop:  for $i = 1, 2, 3, \cdots, n$
          $z = x_i + err$
          $q = sum$                                  (1.9)
          $sum = q + z$
          $err = z - (sum - q)$
       end
```

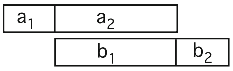
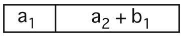
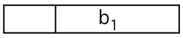
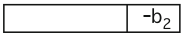
Operation	Floating-Point Result	Comments
a, b		mantissas for a and b are aligned for the addition
$s_f = (a+b)_f$		due to the fixed number of digits, b_2 is lost
$s_f - a$		a is removed from the sum
$(s_f - a) - b$		In removing b, the part that remains is $-b_2$

Table 1.5 Steps explaining how the error in floating-point addition is estimated for compensated summation. Adapted from Higham [1993].

The error in computing $s(n)$ when using this procedure, versus just adding the terms recursively, is given in Table 1.4. The improvement in the accuracy is dramatic. This happens because the method is based on an estimate of the error in a floating-point addition, and then compensates for this in the calculation. To explain, suppose we have two positive real numbers a and b, with $a > b$. The sequence of steps involved illustrating how the method works is given in Table 1.5. What it shows is that the part of b that is dropped in the addition can be approximated with $b - (s_f - a)$. In the loop in (1.9), the x_i's are added to produce the value of sum. In connection with Table 1.4, $a = sum$, $b = x_i$, and $b - (s_f - a) = err$. So err is the missing part of x_i, and it's added back in during the next iteration. This is the reason for setting $z = x_i + err$. There are variations on this procedure, and also limitations on its usefulness. Those interested in reading more about this should consult Demmel and Hida [2004] or Rump et al. [2008]. ■

Example 2

The first thing to notice is that the values of the function in the lower plot in Figure 1.2 are close to machine epsilon. The expanded version of the polynomial is required to take values $x = 1$ and combine them to produce a value close to zero. The errors seen here are consistent with arithmetic using double precision, and the fact that the values are sometimes negative also is not surprising.

It is natural to ask, given the expanded version of the polynomial (1.4), whether it is possible to find an algorithm for it that is not so sensitive to round-off error. There are procedures for the efficient evaluation of a polynomial, and two examples are *Horner's method* and *Estrin's method.* To explain how these work, Horner's method is based on the following observations:

$$\begin{aligned} a_2x^2 + a_1x + a_0 &= a_0 + (a_1 + a_2x)x, \\ a_3x^3 + a_2x^2 + a_1x + a_0 &= a_0 + (a_1 + (a_2 + a_3x)x)x, \\ a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0 &= a_0 + (a_1 + (a_2 + (a_3 + a_4x)x)x)x. \end{aligned}$$

Higher order polynomials can be factored in a similar manner, and the resulting algorithm for evaluating the nth degree polynomial $p(x) = a_0 + a_1x + \cdots + a_nx^n$ is

```
  let:  $p = a_n$
 loop:  for $i = 1, 2, 3, \cdots, n$
            $p = a_{n-i} + p * x$
        end
```

This procedure is said to be optimal because it uses the minimum number of flops to compute $p_n(x)$. In particular, it requires $2n$ adds and multiplies, while the direct method requires about $3n$.

Because of the reduced computing cost, Horner's method is often used in library programs for evaluating polynomials. For example, the library routines that are used by some computers to evaluate $\tan(x)$ and $\text{atan}(x)$ involve polynomials of degree 15 and 22 [Harrison et al., 1999], and having this done as quickly as possible is an important consideration. However, because additions and multiplications take only 6×10^{-10} sec, the speedup using Horner is not noticeable unless you are evaluating the polynomial at a huge number of points. The advantage using Horner is that it tends to be less sensitive to round-off error. To illustrate, using Horner to evaluate the polynomial in (1.4), the curve shown in Figure 1.6 is obtained. It clearly suffers the same oscillatory behavior the direct method has, which is shown in Figure 1.1. However, the amplitude of the oscillations is about half of what is obtained using the direct method. ■

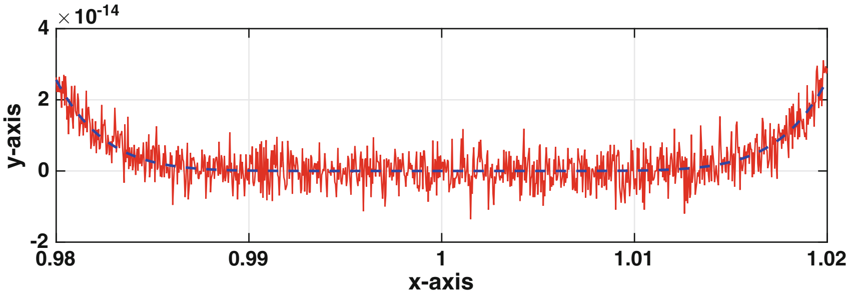


Figure 1.6 Plot of (1.4) when evaluated using Horner's method, solid (red) curve, and using (1.3), the dashed (blue) curve.

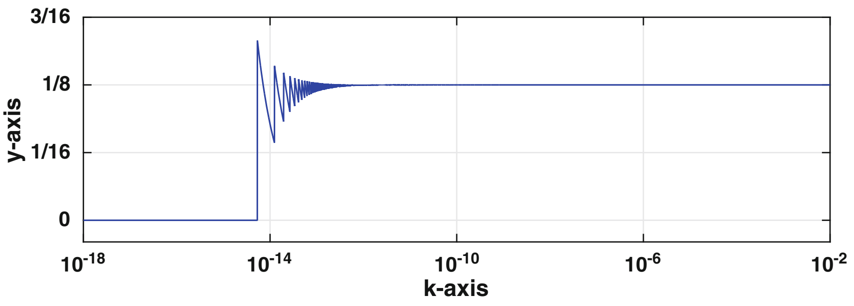


Figure 1.7 Plot of (1.10).

Example 3

The function considered was

$$y = \frac{\sqrt{16+k}-4}{k}, \tag{1.10}$$

and this is replotted in Figure 1.7. The reason for the problems seen in the graph is the value the computer assigns $\sqrt{16+k}$ for small values of k. To explain, assuming k is small then from Taylor's theorem

$$\sqrt{16+k} = 4 + \frac{1}{16}k + O(k^2). \tag{1.11}$$

In other words, the exact value of $\sqrt{16+k}$ is a little larger than 4. The floating-point numbers just to the right of $x = 4$ are shown in Figure 1.8. These are the values the computer has to pick from in this region. So, if the exact value of $\sqrt{16+k}$ falls in Region II, then the computer rounds the value to $4(1+\varepsilon)$. We will concentrate on Region I, which is the interval $4 < x < 4(1+\frac{1}{2}\varepsilon)$. In this case the computer will claim that $\sqrt{16+k} = 4$. It then takes this value, evaluates the numerator in (1.11) and concludes that $y = 0$. It is possible to estimate the value of k where the computer starts claiming that $y = 0$. This happens when $\sqrt{16+k} = 4(1+\frac{1}{2}\varepsilon)$. From this and (1.11), we have that $k \approx 32\varepsilon$. For double precision, $\varepsilon \approx 2.2 \times 10^{-16}$, and so the zero solution is produced for $k < 7 \times 10^{-15}$. Note that it is also

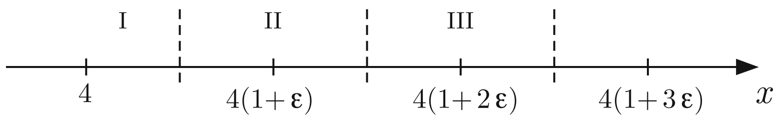


Figure 1.8 The floating-point numbers just to the right of $x = 4$. The dashed lines are located half-way between the floats and any real number between them is rounded to the floating-point number in that subinterval.

possible to explain the oscillations in the graph. The vertical drops in the curve in the vicinity of 10^{-14} come from the jumps in the computed value of $\sqrt{16+k}$. For example, the jump just to the right of 10^{-14} occurs because the value of $\sqrt{16+k}$ passes from Region III into Region II (see Figure 1.8). This means that the computer stops claiming that $\sqrt{16+k} = 4(1+2\varepsilon)$ and starts claiming that $\sqrt{16+k} = 4(1+\varepsilon)$. In Region II, the computer's evaluation of y yields

$$y = \frac{4\varepsilon}{k}\,.$$

As a function of k this produces a hyperbolic curve, and this can be seen in Figure 1.7 (it is the curve in the immediate vicinity of $k = 10^{-14}$).

As in the earlier examples, the question arises as to whether it is possible to evaluate this function and avoid the problems seen in Figure 1.7. It is, and one possibility is to transform the function by setting $z = \sqrt{16+k}$, so (1.10) becomes

$$\begin{aligned} y &= \frac{z-4}{z^2-16} \\ &= \frac{1}{z+4}\,. \end{aligned} \tag{1.12}$$

Using this expression, one obtains $y = 0.1250\cdots$ no matter how small one makes k. ■

1.5 Error and Accuracy

One of the most important words used in this text is *error* (and it is used a lot). There are different types of error that we will often make use of. For example, if x_c is a computed value, and x is the exact value, then

1. $|x - x_c|$ is the *error*,
2. $\dfrac{|x - x_c|}{|x|}$ is the *relative error* (assuming $x \neq 0$).

Because the relative error measures the difference relative to the size of x, it is a better measure of how many significant digits have been computed. To explain, if $|x-x_c|/|x| \approx 10^{-p}$, where p is a positive integer, then x_c should be correct to approximately p significant digits. The following examples illustrate the situation:

Examples

1. $x = 1{,}000{,}000$, $x_c = 1{,}000{,}001$
   In this case, $|x - x_c| = 1$ and $|x - x_c|/|x| = 10^{-6}$. Note that the relative error is reflective of the fact that the computed value agrees to 6 places with the exact value. ■

2. $x = 10^{-6}$, $x_c = 10^{-6} + 10^{-10}$
   In this case, $|x - x_c| = 10^{-10}$ and $|x - x_c|/|x| = 10^{-4}$. Similar to the last example, the relative error shows that the computed value agrees to 4 places with the exact value. ■

Note that the error, as defined above, has a significant flaw, which is that you need to know the exact solution to calculate it. For this reason, it will play an important role in the derivation of the numerical methods, and a less direct role in the implementation of the methods.

One of the problems of not knowing the error is that it can be difficult to know when to stop a computation. As an example, consider the problem of calculating the value of

$$s = 8 - \sum_{n=1}^{\infty} \frac{7}{8^n} \, .$$

Letting

$$s_k = 8 - \sum_{n=1}^{k} \frac{7}{8^n} \, , \tag{1.13}$$

one obtains the values shown in Table 1.6. A pattern is developing in the s_k's related to those digits that stop changing as more terms are added. For example, it appears that s_2 is correct to 2 digits, s_3 is correct to 3 digits, etc. It is possible to introduce a measure for the improvement in the value seen in this pattern by using the following:

1. $|s_k - s_{k-1}|$ is the *iterative error*,
2. $\dfrac{|s_k - s_{k-1}|}{|s_k|}$ is the *relative iterative error* (assuming $s_k \neq 0$).

The values for these quantities are given in Table 1.6. Similar to before, the relative iterative error is a more reflective measure for the number of correct digits in the computed answer.

The iterative error is easily computable, and used extensively in scientific computing. However, it too has a flaw, which is that just because s_k and s_{k-1} are close together, it does not necessarily follow that s_k is close to the exact value. There are various ways you can increase your confidence that s_k is close to the exact value, and an example would be to require that the computation continue until the condition that $|s_k - s_{k-1}| < tol$ is satisfied for three or

k	s_k	$\lvert s_k - s_{k-1}\rvert$	$\lvert s_k - s_{k-1}\rvert/\lvert s_k\rvert$
1	7.125000000000000		
2	7.015625000000000	1.1e−01	1.6e−02
3	7.001953125000000	1.4e−02	2.0e−03
4	7.000244140625000	1.7e−03	2.4e−04
5	7.000030517578125	2.1e−04	3.1e−05
6	7.000003814697266	2.7e−05	3.8e−06

Table 1.6 Values of s_k, which are given in (1.13), as they approach the exact value of $s = 7$. Also given are the iterative error and the relative iterative error.

four successive values of k. However, in the end, without some other piece of information, most numerical solutions have a certain level of uncertainty related to whether they have produced an accurate value for the solution. It is because of this that the theoretical underpinning of the method plays an important role in computing, because it can provide valuable insights into how the method should work. A consequence of this is that the theory can provide a tool for checking on whether the method has been implemented correctly.

1.5.1 Test Cases

The question that comes up with almost any computer code is, how do you know it is calculating the right answer? A good response to this is: well, we ran some tests and it worked just great. This requires the ability to find test cases you know the answer to, and which test the limits of your code. For some types of problems there are whole libraries of test problems, ones that are known to be rather difficult. For more run of the mill problems, the usual approach is to pick a solution and then find what problem it satisfies. It is that problem you then try your computer code on.

Examples

1. Matrix Equation

   Suppose you have written a code to solve matrix equations of the form $\mathbf{Ax} = \mathbf{b}$. In this case, to test if it works, you pick a matrix $\mathbf{A}$, and solution $\mathbf{x}$, calculate $\mathbf{b} = \mathbf{Ax}$, and then use your code to solve $\mathbf{Ax} = \mathbf{b}$ using this particular $\mathbf{A}$ and $\mathbf{b}$. It is then as easy matter to compare the exact solution with what the code computes. ■

2. Differential Equation

   Suppose you have written a code to solve $y'' + y' + y = f(t)$, where $y(0) = a$ and $y'(0) = b$. Instead of trying to find examples by solving the problem by hand, just pick a smooth function $y(t)$. With this choice, then $f(t) = y'' + y' + y$, $a = y(0)$ and $b = y'(0)$. Using this $f(t)$, a, and b in your code, you then can compare the computed values with the exact result. ■

Some care is needed when selecting test problems to make sure they are computable. As a case in point, for the matrix equation example above, the matrix should be well conditioned (this is explained in Chapter 3). In the case of nonlinear problems it is often the case that the solution is not unique, and it is possible to conclude that your numerical method has failed even though it has correctly computed a solution you were not aware of.

1.5.2 Over-Computing?

In using a numerical method, the question comes up as to how accurately to compute the answer. For example, numerical methods are used to solve problems in mechanics, and one often compares the computed values with data obtained experimentally. This begs the question, if the data are correct to only two or three digits, is it really necessary to obtain a numerical solution that is correct to 15 or 16 digits (the limit for double precision)? It is true that in many situations you do not need the accuracy provided using double precision, but this depends on the problem being solved. For example, in Chapter 3 it will be seen that when solving the matrix equation $\mathbf{Ax} = \mathbf{b}$ it is easily possible that 15 or 16 digits are needed just to guarantee that the computed solution is correct to one or two digits. This loss of accuracy is associated with what is called an ill-conditioned problem, which means that the problem tends to magnify small errors. On the other hand, some methods that will be considered actually try to take advantage of not over-computing the solution. A particular example is a search method used to find a minimum of a function, and this is explained in Section 8.7.2.

Exercises

1.1. Find nonzero numbers for x and y so MATLAB calculates x/y to be the stated result. Also, provide a short explanation why your example does this.

(a) Inf
(b) NaN
(c) 0
(d) 1 (with $x \neq y$)

1.2. Have MATLAB evaluate the following, and provide a plausible explanation for the answer.
(a) $10 * NaN$ and $0 * NaN$
(b) NaN/NaN
(c) Inf/Inf
(d) $Inf * 0$
(e) 0^{Inf} and $(Inf)^0$
(f) 1^{Inf}
(g) e^{-Inf} and e^{Inf}

1.3. Let x_f and y_f be adjacent floating-point numbers. You can assume they are positive and normal floats.
(a) What is the minimum possible distance between x_f and y_f?
(b) What is the maximum possible distance between x_f and y_f?
(c) How many double precision numbers lie between two consecutive single precision numbers? You can assume the single precision numbers are both positive.

1.4. In double precision, what is the distance from 32 to the next largest floating-point number?

1.5.
(a) Find the largest open interval about $x = 16$ so all real numbers from the interval are rounded to $x_f = 16$. That is, find the smallest value of L and largest value of R with $L < 16 < R$ so any number from the interval (L, R) is rounded to the floating-point number $x_f = 16$. Assume double precision is used.
(b) Redo part (a) for $x = 50$, that is, find the interval (L, R) that rounds to the floating-point number

$$x_f = 50 = \left(1 + \frac{1}{2} + \frac{1}{2^4}\right) \times 2^5.$$

1.6. Using compound interest, if an amount a is invested at an annual interest r and compounded n times per year then the amount A at the end of one year is

$$A = a\left(1 + \frac{r}{n}\right)^n.$$

It's not hard to show that the larger the value of n, the larger the amount at the end of the year. Assume that $a = 100$ and the interest rate is 1% so $r = 0.01$. Also assume there are 365 days in a year. Using MATLAB calculate A for the following cases:
(a) compounding every hour (so, $n = 365 * 24$),
(b) compounding every second,
(c) compounding every millisecond,

(d) compounding every nanosecond,
(e) compounding every picosecond.
(f) You should find that the values computed in (d) and (e) are incorrect. The question is why, that is, what causes the floating-point calculation to produce an incorrect value? Based on this, given a value of r (with $0 < r < 1$), at what value of n would you expect an incorrect result to be computed by MATLAB?

1.7. Consider the ratio

$$R = \frac{n(n-2)(n-4)\cdots 2}{(n-1)(n-3)(n-5)\cdots 1},$$

where n is even. It is known that if $n = 100$ then $R \approx 12.5645$ and if $n = 400$ then $R \approx 25.0820$.

(a) The commands below will, in theory, compute R. Try them and show that they work if $n = 100$ but not if $n = 400$ (for the latter, the first line must be changed). Explain why this happens.

```
$n = 100$
$T = 1;$
for $i = 2 : 2 : n$
    $T = T * i;$
end
$B = 1;$
for $i = 1 : 2 : n - 1$
    $B = B * i;$
end
$R = T/B$
```

(b) How can R be rewritten so MATLAB can be used to calculate R when $n = 400$? Prove it works by computing the result with MATLAB. Also, compute R for $n = 4{,}000{,}000$.

1.8. Compute the following. If you must modify the sum(s) in any way to obtain the answer, explain what you did and why.

(a) $\displaystyle\sum_{k=0}^{1000} \frac{e^k}{1+e^k}$

(b) $\displaystyle\sum_{k=0}^{1000} \frac{\cosh(k)}{1+\sinh(k)}$

(c) $\displaystyle\sum_{k=0}^{1000}\sqrt{3+e^k}-\sum_{n=0}^{1000}\sqrt{1+e^n}$

(d) $\displaystyle\frac{\sum_{k=0}^{1000}e^k}{\sum_{n=0}^{1000}ne^n}$

(e) $\displaystyle\sum_{k=1}^{1000}k\left[\sin\left(\pi(k^{10}+\frac{1}{k})\right)-\sin\left(\pi(k^{10}-\frac{1}{k})\right)\right]$

1.9. Homer Simpson, in the 1998 episode "The Wizard of Evergreen Terrace," claimed he had a counterexample to Fermat's Last Theorem, and it was that $3987^{12}+4365^{12}=4472^{12}$. This exercise considers whether it is possible to prove numerically that Homer is correct. Note that another (false) counterexample appeared in the 1995 episode "Treehouse of Horror VI."

(a) Calculate $3987^{12}+4365^{12}-4472^{12}$. If Homer is right, what should the answer be?

(b) Calculate $\left(3987^{12}+4365^{12}\right)^{1/12}-4472$. If Homer is right, what should the answer be?

(c) Calculate
$$\frac{3987^{12}+4365^{12}}{4472^{12}}.$$
If Homer is right, what should the answer be?

(d) Calculate $\left[\left(3987^{12}+4365^{12}\right)^{1/12}\right]^{12}-4472^{12}$. If Homer is right, what should the answer be?

(e) One argument that Homer could make is that (c) is the correct result and (a) and (b) can be ignored because if they are correct then you should not get a discrepancy between (a) and (d). Explain why MATLAB cannot be used to prove whether Homer is right or wrong.
Note: Homer's blackboard containing the stated formula, along with a few other gems, can be found in Singh [2013]. It also explains why Homer appears to have an interest in mathematics and physics.

1.10. The graph of the function
$$f(x)=\frac{\sqrt{1+x^2}-1}{x^2}$$
is shown in Figure 1.9 where the values of $f(x)$ were computed using MATLAB.

(a) Approximate $\sqrt{1+x^2}$ with a third degree Taylor polynomial expanded about $x=0$. Using this approximation, show that
$$\lim_{x\to 0}f(x)=\frac{1}{2}.$$

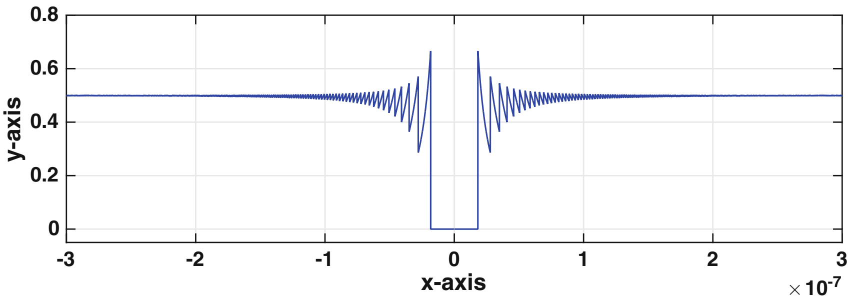


Figure 1.9 A plot of $f(x)$ for Exercise 1.10.

(b) What's causing the problem in MATLAB and why does the problem occur for the specific values of x shown. Also, why does MATLAB state that $f(x) = 0$ for small values of x?

1.11. The graph of the function

$$f(x) = \frac{e^x - 1}{x}$$

is shown in Figure 1.10.

(a) Approximate $e^x - 1$ with a third degree Taylor polynomial expanded about $x = 0$. Using this approximation, show that

$$\lim_{x \to 0} f(x) = 1 .$$

(b) From Figure 1.10 one would conclude that the limit in part (a) is zero. This is incorrect and what's causing the problem in MATLAB? Explain your reasoning and also state why the values of the function drop to zero near 10^{-16} and not, say, near 4×10^{-16}.

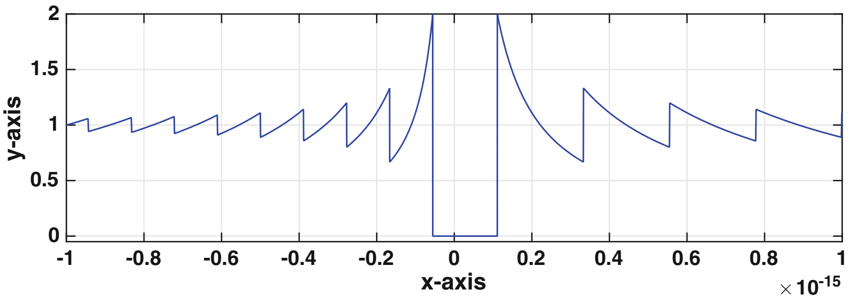


Figure 1.10 A plot of $f(x)$ for Exercise 1.11.

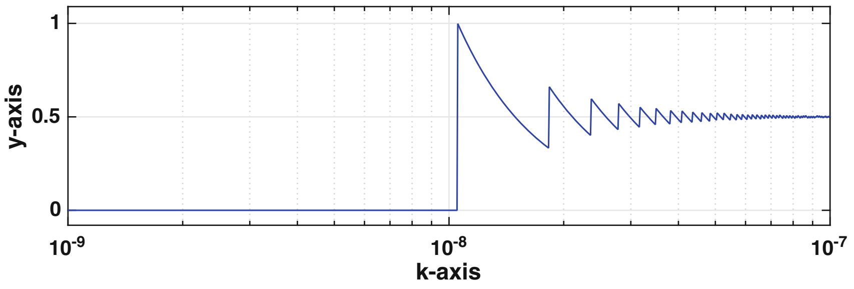


Figure 1.11 A plot of $f(x)$ for Exercise 1.12.

(c) The curve has jumps in the vicinity of 10^{-16} and 3×10^{-16}. Between these points what is the equation of the curve seen in Figure 1.10, and where do the jumps actually take place?

(d) Why are the jumps for $x < 0$ closer together that those for $x > 0$?

1.12. The graph of the function

$$f(x) = \frac{1 - \cos(x)}{x^2},$$

shown in Figure 1.11, was obtained using MATLAB. It is known that $\lim_{x \to 0} f(x) = 1/2$. Why does MATLAB claim the value of this limit is zero? Also, why does MATLAB start claiming that the value is zero for x at about 10^{-8}

1.13. Let

$$f(x) = \frac{\ln(1 - x)}{x}.$$

(a) Approximate $\ln(1 - x)$ with a third degree Taylor polynomial expanded about $x = 0$ Using this approximation, what value should you assign to $f(0)$?

(b) Using MATLAB, plot $f(x)$ for $-10^{-15} \le x \le 10^{-15}$. What value does MATLAB assign to $f(x)$ for x very near zero? The interval where the function is zero is not symmetric about $x = 0$. Why? Also, does this also explain why there are more oscillations on the right ($x > 0$) than on the left ($x < 0$)?

(c) Using MATLAB, plot $\ln(1-x)$ for $-5 \times 10^{-15} \le x \le 5 \times 10^{-15}$. The graph should resemble steps with the step containing $x = 0$ corresponding to the value of $\ln(1) = 0$. As x increases from $x = 0$, what determines the value of the first nonzero step? How do these steps explain the oscillations seen in the plot for part (b)?

1.14. This problem considers the following algorithm

```
$x = \sqrt{2}$
$s = 0$
for $i = 1, 2, 3, \cdots, N$
    $s = s + x$
end
$y = x - s/N$
```

It is assumed N is a prescribed (positive) integer.

(a) What is the exact value for y?

(b) Using MATLAB, if $N = 10^4$, one gets that $y \approx 1.4 \times 10^{-13}$, if $N = 10^8$, one gets that $y \approx 2.2 \times 10^{-9}$, and if $N = 10^{11}$, one gets that $y \approx 1.3 \times 10^{-6}$. Why is the error getting worse as N increases? Is there any correlation between the value of N and the value of y?

(c) Use compensated summation to compute this result and compare the values with those given in part (b).

1.15. The polynomial $p_n(x) = a_0 + a_1 x + \cdots + a_n x^n$ can be separated into the sum of two polynomials, one which contains even powers of x and the other involving odd powers. This problem explores the computational benefits of this. To make things simple, you can assume n is even, so $n = 2m$, where m is a positive integer.

(a) Setting $z = x^2$, find $f(z)$ and $g(z)$ so that $p_n(x) = f(z) + xg(z)$.

(b) What is the minimum flop count to compute the expression in part (a)? Also, explain why it is about half-way between the flop count for the direct method and the count using Horner's method.

(c) Evaluate (1.4) using the formula in part (a), and then plot the values for $0.98 \le x \le 1.02$ (use 1000 points in this interval). In comparison to the plot obtained using the direct method, does the reduced flop count reduce the error in the calculation?

1.16. This problem considers the consequences of rounding using double precision. Assume the "round to nearest" rule is used, and if there is a tie then the smaller value is picked (this rule for ties is used to make the problem easier).

(a) For what real numbers x will the computer claim the inequalities $1 < x < 2$ hold?

(b) For what real numbers x will the computer claim $x = 4$?

(c) Suppose it is stated that there is a floating-point number x_f that is the exact solution of $x^2 - 2 = 0$. Why is this not possible? Also, suppose $\bar{x}_f$ and $\bar{\bar{x}}_f$ are the floats to the left and right of $\sqrt{2}$, respectively. What does $\bar{\bar{x}}_f - \bar{x}_f$ equal?

1.17. This problem considers the error when evaluating $\sin x$, and the problem seen in Figure 1.3. It is assumed that x is a given real number that is not a floating-point number, and x_f is its floating-point approximation. Also, E is the integer so that $2^E < x < 2^{E+1}$ and $2^E \leq x_f \leq 2^{E+1}$.

(a) Use Taylor's theorem to show that $|\sin x - \sin x_f| \leq |x - x_f|$.

(b) The point x is between two floating-point numbers $\bar{x}_f$ and $\bar{\bar{x}}_f$, and either $x_f = \bar{x}_f$ or $x_f = \bar{\bar{x}}_f$. Explain why $|x - x_f| \leq |\bar{\bar{x}}_f - \bar{x}_f|/2$.

(c) Using parts (a) and (b) show that $|\sin x - \sin x_f| \leq \varepsilon 2^{E-1}$.

(d) Use the result in part (c) to show that if $|x| \leq L$ then

$$|\sin x - \sin x_f| \leq \frac{1}{4}\varepsilon L.$$

(e) When the computer evaluates $\sin x_f$ it produces a floating-point number s_f. Assuming that $|\sin x_f - s_f| \leq \varepsilon$, show that

$$|\sin x - s_f| \leq \frac{1}{4}\varepsilon(L+4).$$

(f) When using double precision, what interval $-L \leq x \leq L$ can you use and be able to guarantee that $|\sin x - s_f| \leq 10^{-8}$? How does this value of L compare with the corresponding result obtained from Figure 1.3?

1.18. This problem considers ways to compute x^n, where n is a positive integer. This problem arose from trying to explain MATLAB's rather large flop time in Table 1.3 for integer powers.

(a) Compare the total number of flops between computing $x^n = x*x*\cdots*x$, and computing

$$x^n = \begin{cases} y*y*\cdots*y & \text{if } n \text{ is even} \\ x*y*y*\cdots*y & \text{if } n \text{ is odd,} \end{cases}$$

where $y = x^2$. As examples of the last formula, $x^6 = y*y*y$, while $x^5 = x*y*y$.

(b) Suppose $n = 28$. Show that $28 = 2^4 + 2^3 + 2^2$, and

$$x^{28} = \left(\left(\left(x^2\right)^2\right)^2\right)^2 * \left(\left(x^2\right)^2\right)^2 * \left(x^2\right)^2.$$

What is the minimum number of flops required using this formula? Also explain why $2^4 + 2^3 + 2^2$ is the floating-point representation of 28. Note that this procedure is a version of the square-and-multiply algorithm.

(c) Suppose $n = 100$, so its floating-point representation is $(1 + \frac{1}{2} + \frac{1}{2^4}) \times 2^6$. Explain how to use the idea in part (b) to calculate x^{100}. How does the flop count compare with the two methods in part (a)?

(d) Another approach, assuming x is positive, is to write $x^n = e^{n \ln x}$. Based on the values in Table 1.3, what is the approximate flop time for this? How does it compare with the flop times found in parts (b) and (c)?

1.19. This problem considers how to determine the largest value of the mantissa.

(a) What values of the b_j's in (1.7) produce the largest value of m?

(b) Assuming $x \neq 1$, show that

$$\frac{1}{1-x} = 1 + x + x^2 + \cdots + x^n + \frac{x^{n+1}}{1-x}\,.$$

(c) Use the result from part (b) to show that the value of m from part (a) can be written as $m = 2 - \varepsilon$. From this show that the largest value of the mantissa is $m = 1 + K\varepsilon$, where $K = 2^{N-1} - 1$.

(d) Use part (c) to explain why the float just to the left of $x = 2$ is $2 - \varepsilon$. Also, explain why the float just to the right of $x = 2$ is $2(1 + \varepsilon)$.

Chapter 2
Solving A Nonlinear Equation

In this chapter one of the more common mathematical problems is studied, which is to find the solution, or solutions, of an equation of the form $f(x) = 0$. To illustrate the situation, we begin with a few examples.

2.1 Examples

The examples below are separated into physical and mathematical. The physical ones are typical in the sense that there are multiple parameters in the problem, and the variable to solve for is not x. It is also not clear if the parameters in the problem can mess things up, and either cause there to be no solution or possibly many solutions of the equation. The mathematical examples, on the other hand, are relatively simple, and the variable to solve for is always x. The objective in this case is to illustrate some of the mathematical complications that can arise when solving nonlinear equations.

2.1.1 Physical

1. A sphere falling through the air reaches a terminal velocity v, which is determined by a balance in the force of gravity and air resistance. From Newton's second law, it is possible to show that

$$v^2 = \frac{2mg}{\rho A c_D}\,, \tag{2.1}$$

where $A = \pi d^2/4$. In this expression, ρ is the density of air, m and d are the mass and diameter of the sphere, and g is the gravitational acceleration

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_2

constant. The term c_D is the drag coefficient, and it accounts for air resistance. It is known by anyone who sticks their arm out of a car window, the faster you go the greater the air resistance. In other words, c_D depends on v. For spheres, from experimental data the following formula has been proposed [White, 2005]

$$c_D = \frac{24}{Re} + \frac{6}{1+\sqrt{Re}} + \frac{2}{5}\,, \tag{2.2}$$

where $Re = \rho v d/\mu$ is known as the Reynolds number and μ is the (dynamic) viscosity of air. If this is substituted into (2.1), you end up with a rather complicated nonlinear equation to solve to determine the terminal velocity. One of the goals of this chapter is to derive methods to solve such equations. For those who might be interested, these methods are used to calculate the terminal velocity in Exercise 2.19. ■

2. The Michaelis-Menten model in biochemistry describes how an enzyme binds to a substrate and from this forms a new product molecule. The steps are illustrated in Figure 2.1. Using the law of mass action, and something called the quasi-steady-state approximation, one finds that the amount of the substrate S present at time t satisfies the differential equation

$$\frac{dS}{dt} = -\frac{v_m S}{K_M + S}\,, \tag{2.3}$$

where v_m and K_M are positive constants. Solving this, one obtains

$$K_M \ln(S/S_0) + S = S_0 - v_m t\,,$$

where S_0 is the amount at the beginning. So, determining how much of the substance S is present at any given value of t comes down to solving the above nonlinear equation for S. This is known as an implicit solution, and they are very common. Most of the more interesting problems that arise in science and engineering involve nonlinear differential equations. When it is possible to find solutions to such problems they are almost always in implicit form. How the methods developed in this chapter can be used to find S are explored in Exercise 2.17. ■

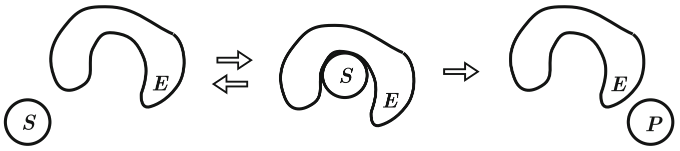


Figure 2.1 The steps in the Michaelis-Menten mechanism, where an enzyme, E, assists S in transforming into P [Holmes, 2009].

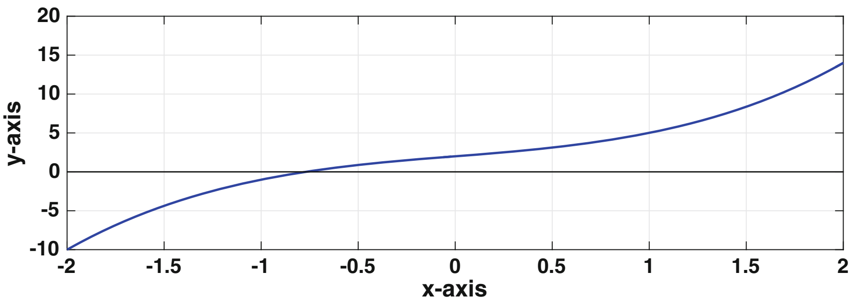


Figure 2.2 Plot of the function for Example 1.

2.1.2 Mathematical

1. Solve $x^3 + 2x + 2 = 0$.

   Setting $y = x^3 + 2x + 2$, then the plot of y is shown in Figure 2.2. It is seen that there is a solution of $y = 0$ between $x = -1$ and $x = -0.5$. ■

2. Solve $4x = 3\cos(2\pi x)$.

   It is not hard to sketch $y = 4x$ and $y = 3\cos(2\pi x)$, and this is done in Figure 2.3. The solutions of the equation correspond to where these curves intersect, and it shows that there are three solutions. ■

3. Solve $x = x^5 - x^4 + 1$.

   Setting $y = x^5 - x^4 + 1 - x$, then the plot of y is shown in Figure 2.4. What is distinctive about this example is that the solution at $x = 1$ is different than the others considered so far. In particular, the $x = 1$ solution is one-sided in the sense that the curve does not change sign as x passes though $x = 1$, which is what happens at the solution at $x = -1$. As will be seen, our numerical methods will assume the solution is not one-sided. ■

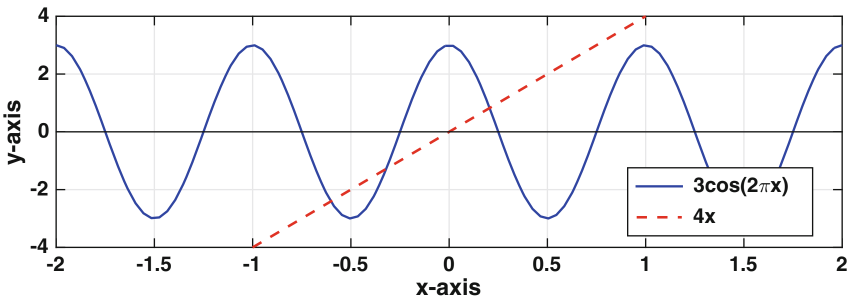


Figure 2.3 Plot of the two functions for Example 2.

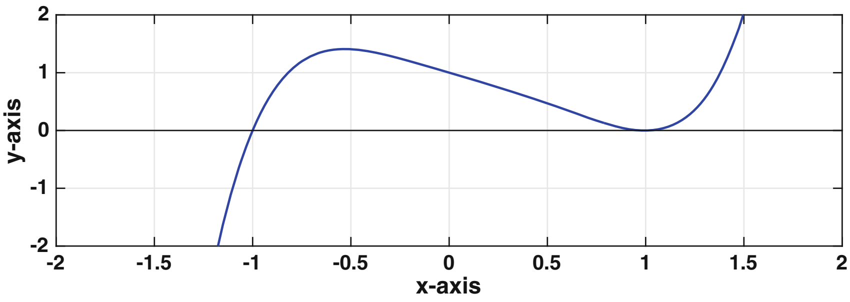


Figure 2.4 Plot of the function for Example 3.

4. Find where the function $F(x) = xe^{-x^2}$ attains its maximum value.

   This function is plotted in Figure 2.5(upper), and it's apparent the maximum is attained between $x = 0.5$ and $x = 1$. There are numerical methods for finding the location of the maximum (see Chapter 8), but the approach here is to simply find the points where $F'(x) = 0$. The graph of $f(x) = F'(x)$ is also shown in Figure 2.5. Not unexpectedly, the solutions of $f(x) = 0$ include the maximum location but they also include the location of the minimum. ■

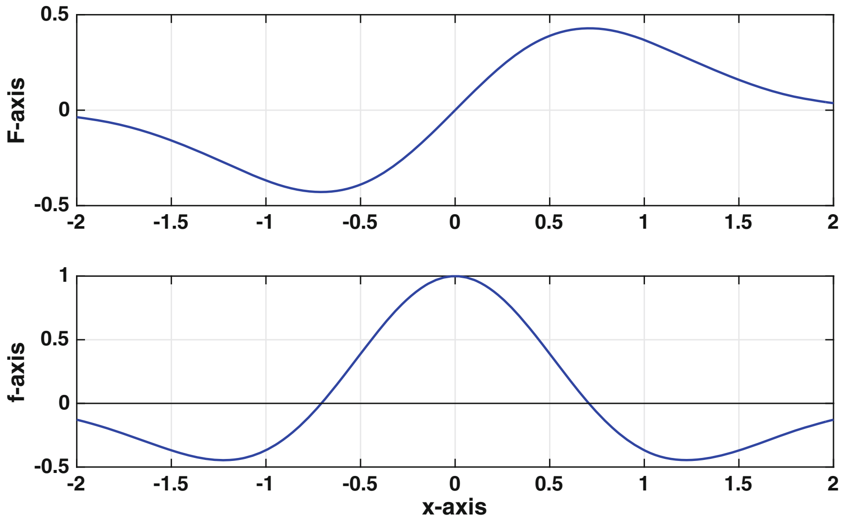


Figure 2.5 Plot of the functions for Example 4. Upper: plot of $F(x)$. Lower: plot of $f(x) = F'(x)$.

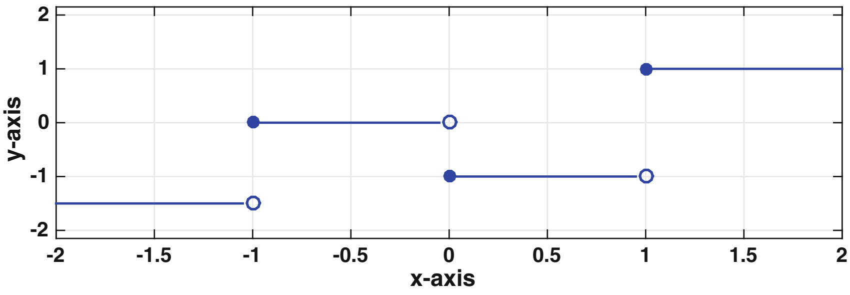


Figure 2.6 Plot of function for Example 4.

5. Solve $f(x) = 0$, where $f(x)$ is plotted in Figure 2.6.

   This function has a couple of complications we will avoid. First, every point in the interval $-1 \leq x < 0$ is a solution. In contrast, there are only a finite number of solutions in the other examples. Second, the function is not continuous. To explain why this is a problem, note that the function is negative at $x = 0.5$ and is positive at $x = 1.5$. If the function is continuous, then we would be guaranteed that there is at least one point in the interval $-0.5 < x < 1.5$ where $f(x) = 0$. For a non-continuous function, like the one in Figure 2.6, there is no such guarantee. ■

2.2 The Problem to Solve

In this chapter we will describe methods for finding a solution $\bar{x}$ of the equation $f(x) = 0$, where $f(x)$ is continuous. Two of the methods considered, Newton and secant, will include additional assumptions about $f(x)$.

2.3 Bisection Method

The easiest way to explain the steps in the bisection method is to consider an example, and so suppose we want to solve $x^3 + 2x + 2 = 0$. The function $f(x) = x^3 + 2x + 2$ is plotted in Figure 2.2. The bisection method is based on a simple observation, which is that if $f(a)$ and $f(b)$ have opposite sign (so one is positive and the other is negative), then there must be a solution of $f(x) = 0$ in the interval $a < x < b$. What the bisection method does is prescribe a systematic method for finding smaller and smaller trapping intervals.

Step 0: To get things started it is necessary to determine an interval that contains the solution. In looking at Figure 2.2, we could take $-2 < x < 0$, or $-1 < x < 0$, or $-2 < x < 2$. It doesn't make much difference which one is used, but it is essential that if the interval is $a_0 < x < b_0$ then $f(a_0)f(b_0) < 0$. From the Intermediate Value Theorem, this guarantees that there is at least one solution in the interval. This is illustrated in Figure 2.7 with $-2 < x < 0$

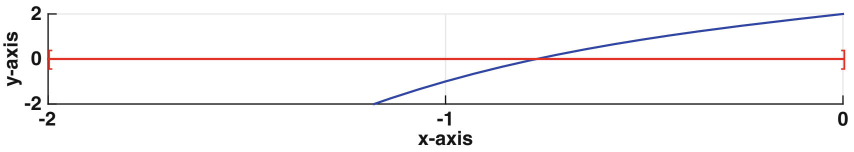


Figure 2.7 Graph of the function in Figure 2.2. Using bisection, the first interval $-2 < x < 0$ is cut in half and it is determined that the solution is in the right half $-1 < x < 0$.

Step 1: Cut the interval $a_0 < x < b_0$ in half and determine which half contains the solution. For example, if our initial interval is $-2 < x < 0$ then the midpoint is $x = -1$. Because $f(-1)f(0) < 0$ it must be that the solution is in the interval $-1 < x < 0$ (see Figure 2.8). So, this step has taken our previous interval (a_0, b_0) and produced a new interval (a_1, b_1) that is half the length and still contains the solution. To be specific, the length of this subinterval is $\ell_1 = (b_0 - a_0)/2$.

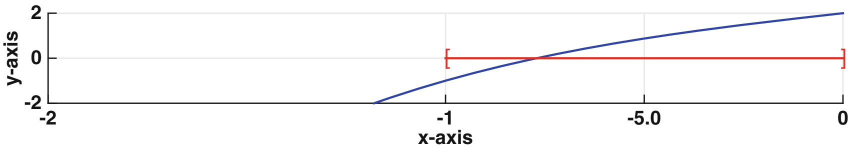


Figure 2.8 The interval $-1 < x < 0$ is cut in half and it is determined that the solution is in the left half $-1 < x < -1/2$.

Step 2: Cut the interval $a_1 < x < b_1$ in half and determine which half contains the solution. The midpoint of the interval is $c_1 = (a_1 + b_1)/2$, and the half containing the solution is the one in which $f(x)$ changes sign. In particular, if $f(a_1)f(c_1) < 0$ then the new interval is $a_2 < x < b_2$, where $a_2 = a_1$ and $b_2 = c_1$. Similarly, if $f(b_1)f(c_1) < 0$ then the new interval is $a_2 < x < b_2$, where $a_2 = c_1$ and $b_2 = b_1$. In either case, this step has taken the previous interval (a_1, b_1) and produced a new interval (a_2, b_2) that is half the length and still contains the solution. The length in this case is $\ell_2 = \ell_1/2 = (b_0 - a_0)/2^2$. For the example being considered, as illustrated in Figure 2.9, $c_1 = -1/2$, and this means that the new interval is $-1 < x < -1/2$.

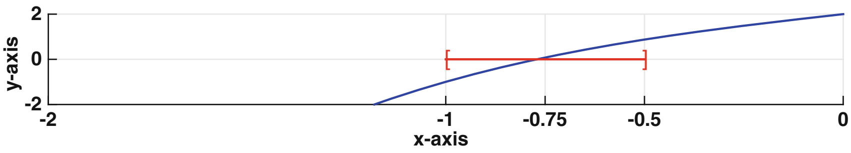


Figure 2.9 The interval $-1 < x < -0.5$ is cut in half and it is determined that the solution is in the left half $-1 < x < -0.75$.

The next steps continue in a similar manner. Note that in each step it is possible that the midpoint turns out to be the exact solution, in which case the calculation is stopped. To summarize the resulting procedure, assuming the interval (a_{i-1}, b_{i-1}) is known,

$$\begin{aligned}&\text{letting } c_{i-1} = (a_{i-1} + b_{i-1})/2\\ &\qquad \text{if } f(c_{i-1}) = 0, \text{ then stop}\\ &\qquad \text{else if } f(a_{i-1})f(c_{i-1}) < 0, \text{ then } a_i = a_{i-1},\ b_i = c_{i-1}\\ &\qquad \text{else } a_i = c_{i-1},\ b_i = b_{i-1}\end{aligned} \tag{2.4}$$

The length of the new subinterval is $\ell_i = b_i - a_i$, and given that this is half of the previous one, it follows that

$$\ell_i = \frac{1}{2^i}(b - a),$$

where (a, b) is the initial interval specified in Step 0.

Given that we are trying to find the solution of $f(x) = 0$, what point do we use from the subinterval (a_i, b_i) as the approximation for the solution $\bar{x}$? The solution could be anywhere in this interval, so the best choice for an approximation for $\bar{x}$ is the interval's midpoint $c_i = (a_i + b_i)/2$ (see Exercise 2.12). Making this choice, then the error in the approximation satisfies

$$|c_i - \bar{x}| \le \frac{1}{2}\ell_i. \tag{2.5}$$

Note that this is a worst-case result, and the actual error will be somewhat smaller than $\frac{1}{2}\ell_i$.

An algorithm for the bisection method is given in Table 2.1. It differs slightly from the procedure in (2.4) in that the endpoints are not indexed. Instead, the values of a and b are simply replaced with the newest values as the procedure proceeds.

Summarizing the above discussion, we have the following result:

Theorem 2.1. *If $f \in C[a,b]$, with $f(a)f(b) < 0$, then the midpoints c_0, c_1, c_2, $\cdots$ computed using the bisection method converge to a solution $\bar{x}$ of $f(x) = 0$, and the error satisfies*

$$|c_i - \bar{x}| \leq \frac{1}{2^{i+1}}(b-a). \tag{2.6}$$

This theorem is a rarity in scientific computing for two reasons. One, it states that the method always works as long as $f(a)f(b) < 0$. Second, it provides an explicit formula for the error. The latter is useful as it is possible to predict how many subintervals need to be computed even before the calculation is undertaken. Specifically, if one wants an error of no more than δ, then we need to take i large enough so that

$$\frac{1}{2^{i+1}}(b-a) \leq \delta.$$

Solving for i, the conclusion is that

$$i \geq -1 + \frac{\ln((b-a)/\delta)}{\ln 2}\,. \tag{2.7}$$

The price paid for the guarantee that the method always works is that it is slow compared to other methods we will consider. To explain, suppose we have computed c_i and the error in this approximation is 10^{-2}. If we would like to improve this and have an error of 10^{-3}, we would have to continue

```
  pick:   $a < b$ with $f(a)f(b) < 0$
          $tol > 0$
   let:   $err = (b-a)/2$
  loop:   while $err > tol$
             $c = (a+b)/2$
             if $f(c) = 0$, then stop
                else if $f(a)f(c) < 0$, then $b = c$
                else $a = c$
             end
             $err = (b-a)/2$
          end
answer:   $c = (a+b)/2$
```

Table 2.1 Algorithm for solving $f(x) = 0$ using the bisection method.

i	c_i	$\lvert c_i - \bar{x}\rvert$
0	−1	2.29e−01
1	−0.500000000000000	2.71e−01
2	−0.750000000000000	2.09e−02
3	−0.875000000000000	1.04e−01
4	−0.812500000000000	4.16e−02
⋮	⋮	⋮
20	−0.770916938781738	5.83e−08
21	−0.770917415618896	4.19e−07
⋮	⋮	⋮
49	−0.770916997059247	6.66e−16
50	−0.770916997059248	2.22e−16

Table 2.2 Solving $x^3 + 2x + 2 = 0$ using the bisection method given in (2.4). Note that 2.29e−01 $= 2.29 \times 10^{-1}$.

four more steps and compute c_{i+4}. The reason is that the error is reduced by a factor of 2 at each step, so we need to make four additional subdivision steps to reduce the error by at least a factor of 10. In contrast, for Newton's method, which is considered next, it is very possible that in just one step the error can go from 10^{-2} to 10^{-4}, and in the next step drop to 10^{-8}.

Example

If $f(x) = x^3 + 2x + 2$, and the initial interval is $(-2, 0)$, then the output using the bisection is shown in Table 2.2. What is also given is the error $e_i = |c_i - \bar{x}|$, where $\bar{x}$ is the exact solution. The latter is also plotted in Figure 2.10. It is no surprise that the method works, because this is guaranteed by Theorem 2.1. Also, to have an error of no more than 10^{-15}, then according to (2.7) we need to take $i \geq 50$, which is consistent with the results shown in Table 2.1. What might not be expected is the fact that the error does not necessarily improve with each step. However, this is easy to explain and it's due to our using the midpoint as the approximation. So, sometimes the exact solution is closer to the midpoint while other times it is farther away. However, overall the error follows an $\alpha/2^i$ decrease as the method proceeds, and to make this evident the curve $y = |c_1 - \bar{x}|/2^i$ is also plotted in Figure 2.10. ■

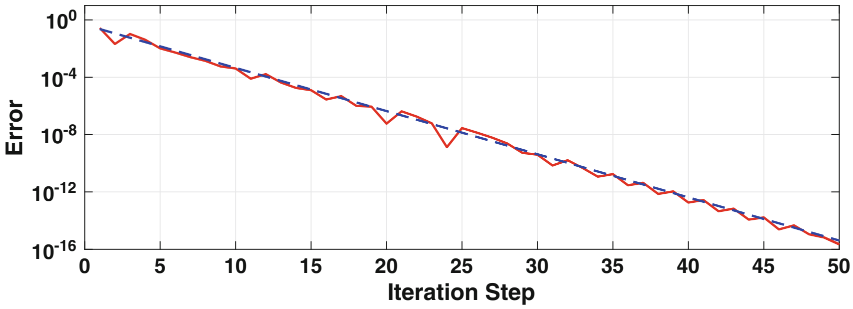


Figure 2.10 The solid (red) curve is the error $|c_i - \bar{x}|$, from Table 2.2, and the dashed (blue) curve is the function $\alpha/2^i$, where $\alpha = |c_1 - \bar{x}|$.

2.4 Newton's Method

We will now consider what is known as Newton's method, although you might also call it the tangent line method. It is easiest to introduce the ideas using an example, and we will again consider solving $x^3 + 2x + 2 = 0$. When we solved this with the bisection method the only information we used about the function is whether it was positive or negative. In Newton's method more information about the function is used.

The essential tool, as is often the case in numerical computing, is Taylor's theorem. What we are going to do is use Taylor's theorem to obtain a linear approximation of $f(x)$, for x near x_0. This is given in Appendix A, in equation (A.6), and from this we have that

$$f(x) \approx f(x_0) + f'(x_0)(x - x_0). \tag{2.8}$$

We are going to replace the equation $f(x) = 0$ with the equation $f(x_0) + f'(x_0)(x - x_0) = 0$. Solving this we get the solution

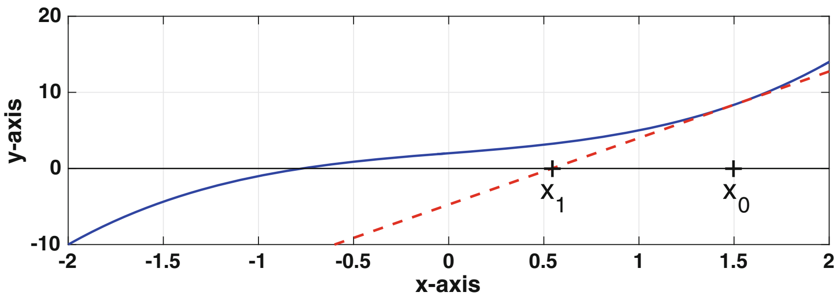


Figure 2.11 First step using Newton's method. The solid curve is $y = x^3 + 2x + 2$ and the dashed line is the line tangent to the curve at x_0.

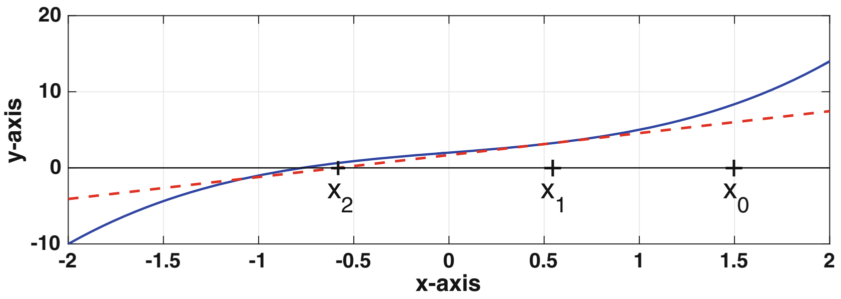


Figure 2.12 Second step using Newton's method. The solid curve is $y = x^3 + 2x + 2$ and the dashed line is the line tangent to the curve at x_1.

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \, . \tag{2.9}$$

A picture of this situation is shown in Figure 2.11. Since $f(x) = x^3 + 2x + 2$, then $f'(x) = 3x^2 + 2$. Also, recall from calculus that the equation for the line tangent to the curve at x_0 is $y = f(x_0) + f'(x_0)(x - x_0)$. This line is nothing more than the approximation we used in (2.8), and it is shown in Figure 2.11 in the case of when $x_0 = 1.5$. Where this line intersects the x-axis determines x_1, and from (2.9) we find that $x_1 = 0.5429\cdots$.

What we see in Figure 2.11 is that, starting with x_0, we have produced a point x_1 that is closer to the solution. We should be able to get even closer by doing this again, which means we approximate $f(x)$ for x near x_1 as $f(x) \approx f(x_1) + f'(x_1)(x - x_1)$. Using this approximation, and solving for x we get the solution

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} \, .$$

A picture of this situation is shown in Figure 2.12.

To summarize the resulting procedure, assuming the point x_i is known, the next point is calculated using the formula

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \, , \text{ for } i = 0, 1, 2, 3, \cdots \, . \tag{2.10}$$

This formula is *Newton's method*, and to use it to solve $f(x) = 0$ requires a start value x_0.

An algorithm for Newton's method is given in Table 2.3. It differs slightly from the procedure in (2.10) in that the values of x are not indexed. Instead, the value of x is overwritten as the procedure proceeds. Note that the stopping condition is based on the iteration error $|z| = |x_{i+1} - x_i|$. Assuming the solution is nonzero, one could instead use the relative iteration

```
pick:  $x$
       $tol > 0$
let:   $err = 3 * tol$
loop:  while $err > tol$
           $z = f(x)/f'(x)$
           $err = abs(z)$
           $x = x - z$
       end
```

Table 2.3 First version of algorithm for solving $f(x) = 0$ using Newton's method. The procedure stops when $|x_{i+1} - x_i| < tol$.

error $|(x_{i+1} - x_i)/x_{i+1}|$. This version of the algorithm does not account for the various ways Newton's method can fail, and how this can be done will be considered later (see Table 2.5).

Example

If $f(x) = x^3 + 2x + 2$, then (2.10) becomes

$$x_i = x_{i-1} - \frac{x_{i-1}^3 + 2x_{i-1} + 2}{3x_{i-1}^2 + 2}, \text{ for } i = 1, 2, 3, \cdots . \tag{2.11}$$

To determine the number and approximate locations of the solutions of $f(x) = 0$, we rewrite the equation as $x^3 = -2x - 2$. The left and right hand sides of this equation are plotted in Figure 2.13. It is apparent that there is only one solution, and it's in the interval $-1 < x < 0$. Consequently, a reasonable starting point is $x_0 = -1/2$. With this, and (2.11), it follows that

$$\begin{aligned} x_1 &= -\frac{1}{2} - \frac{7/8}{11/4} \\ &= -\frac{9}{11} \\ &\approx -0.81818. \end{aligned}$$

The remaining values are computed using MATLAB, and the results are given in Table 2.4. What is also given is the error $e_i = |x_i - \bar{x}|$, where $\bar{x}$ is the exact solution. The error is also plotted in Figure 2.14. It is clear that the method works, and it is notable how fast the error decreases when compared to the

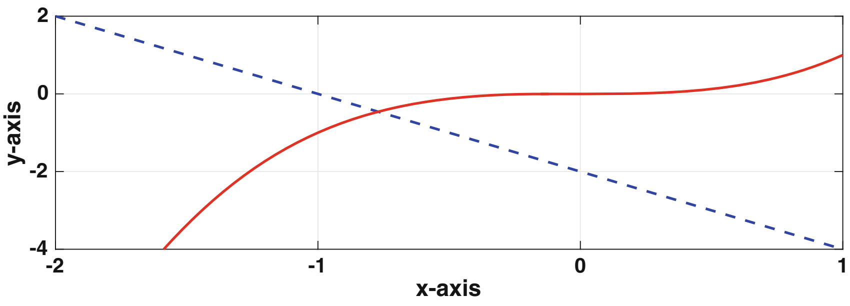


Figure 2.13 The functions $y = -2x - 2$, dashed (blue) line, and $y = x^3$, the solid (red) curve.

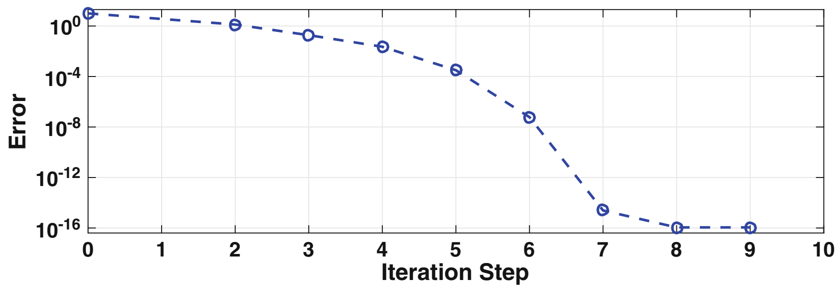


Figure 2.14 The error $|x_i - \bar{x}|$, from Table 2.4.

bisection method. The other observation to be made about the error is that once it gets down to about 10^{-16} it stops improving. This is as accurate as can be expected using double precision arithmetic. ■

i	x_i	$\|x_i - \bar{x}\|$	γ
0	−0.50000000000000	2.71e−01	
1	−0.81818181818182	4.73e−02	
2	−0.77225866916589	1.34e−03	2.1671
3	−0.77091809703576	1.10e−06	2.0745
4	−0.77091699705999	7.40e−13	2.0359
5	−0.77091699705925	1.11e−16	1.3152
6	−0.77091699705925	1.11e−16	1.0000

Table 2.4 Solving $x^3+2x+2 = 0$ using the Newton's method formula given in (2.11). Also given is the error $e_i = |x_i - \bar{x}|$, and the approximate order of convergence γ as determined from (2.15).

Example: Implicit Functions

The mathematical problem to be considered is to determine the value of a function $y(x)$ from an equation of the form $F(x, y) = 0$. Examples of this are

$$y^3 + x = \ln y,$$

and

$$y + x = e^{x-y}. \tag{2.12}$$

The Michaelis-Menten equation (2.3) is another example, where it's necessary to solve an equation of the form $F(t, S) = 0$ to find the solution S. It is relatively easy to use Newton's method to solve $F(x, y) = 0$ if it understood that x is fixed, or given, and we are solving the equation for y. The resulting iteration scheme is: after picking y_0, then

$$y_{j+1} = y_j - \frac{F(x, y_j)}{F_y(x, y_j)}, \tag{2.13}$$

where F_y is the partial derivative of F with respect to y. As an example, for (2.12) we get that

$$y_{j+1} = y_j - \frac{y_j + x - e^{x-y_j}}{1 + e^{x-y_j}}. \tag{2.14}$$

To use this, it is necessary to have a reasonable guess for the initial point y_0. To find such a value, the left and right hand sides of (2.12) are sketched in Figure 2.15 as a function of y (assuming that $x > 0$). Also shown are the values of the functions at $y = 0$. It is seen that the intersection point $\bar{y}$, which is the solution of the equation, is in the interval $0 < \bar{y} < A$. The value of A is determined by solving $e^{x-A} = x$, from which we get that $A = x - \ln(x)$. With this, $0 < \bar{y} < x - \ln(x)$. A reasonable starting point is the midpoint of this interval, which means $y_0 = \frac{1}{2}(x - \ln(x))$. For example, if $x = 3$, then $y_0 = (3 - \ln 3)/2 \approx 0.95$. Using (2.14), after 6 iteration steps, one finds that $y = 1.496\cdots$ with a relative iterative error on the order of machine ε.

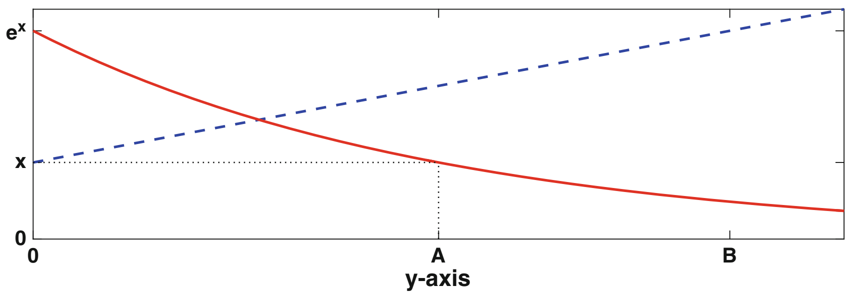


Figure 2.15 The functions $Y = y + x$, dashed (blue) line, and $Y = e^{x-y}$, the solid (red) curve, sketched as a function of y.

The method for finding A is useful enough that it is worth making an additional comment about it. Instead of using the solid (red) curve, we could have used the dashed (blue) curve. This produces a point B (see Figure 2.15), and it corresponds to when $B + x = e^x$. In other words, $B = e^x - x$. This was not mentioned earlier because the interval $0 < \bar{y} < A$ provides a better approximation than $0 < \bar{y} < B$ in the case of when $x = 3$. However, for values of x close to zero, B provides a better approximation. For example, when $x = 0$, $A = \infty$, and $B = 1$. The point being made here is that both A and B should be considered, and the smaller of the two used in determining y_0. ■

2.4.1 Order of Convergence

Inspecting the error in Table 2.4, it appears that except at the start and end, the error at step $i + 1$ is approximately the square of the error at step i. To investigate this, our observation implies that $e_{i+1} \approx Ce_i^\gamma$, where $\gamma \approx 2$. To see if this is true, we take the log of this expression to obtain $\ln e_{i+1} \approx \gamma \ln e_i + \ln C$. The closer the error gets to zero, the less important the term $\ln C$ contributes to this equation, and it can be dropped. In this case, we have that $\ln e_{i+1} \approx \gamma \ln e_i$. Solving for γ, we obtain

$$\gamma \approx \frac{\ln e_{i+1}}{\ln e_i} . \tag{2.15}$$

Based on what we see in Table 2.4, it is expected that $\gamma \approx 2$. To check on this, the computed value of $\ln e_{i+1}/\ln e_i$ is given in Table 2.4 and it does indeed appear that $\gamma \approx 2$, or at least it is approaching this value as x_i gets close to the exact solution.

The formula in (2.15) is based on a heuristic argument that came from observing what is computed using the method. Our observation, from Table 2.4, that the computed value for γ looks to be converging to 2 means that the method is second-order. However, it is impossible to prove this numerically because double precision arithmetic limits the resolution of the computation. The theoretical underpinning of this observation is developed in Section 2.4.3.

2.4.2 Failure

It is important to be aware that Newton's method might not work. For example, it is evident in (2.10) that if we ever get $f'(x_{i-1}) = 0$ then the method fails. There are other potential problems, and one is illustrated in the next example.

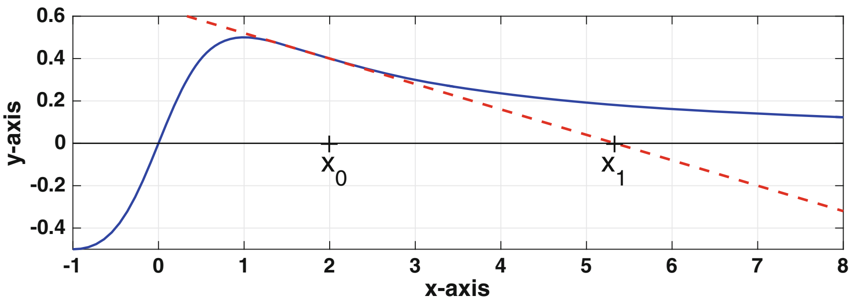


Figure 2.16 First step using Newton's method. The solid curve is $y = x/(1 + x^2)$, and the dashed line is the tangent line used by Newton's method when $x_0 = 2$.

Example

Suppose we use Newton's method to solve

$$\frac{x}{1+x^2} = 0.$$

The graph of the function is shown in Figure 2.16, as well as the tangent line when you take $x_0 = 2$. In this case the next point x_1 is father away from the solution. In fact, if you keep using Newton's method you would find that $x_i \to \infty$. Also note that Newton's method can be used to solve this equation, it is just that you need to pick x_0 near the solution. For example, if $x_0 = 1/2$ then the method will work just fine. ■

Other examples of how, or when, Newton's method fails are given in Exercises 2.24 and 2.25.

The algorithm for Newton's method should be revised so the calculation is stopped if the runaway situation shown in Figure 2.16 occurs. One way to do this is to put a bound on the value of x_i. For example, one picks a relatively large value M and if it ever happens that $|x_i| > M$ then it is assumed that runaway is occurring and the calculation is stopped. Another possibility is to simply limit the number of iteration steps to a number I. Note that when Newton's method does work it converges very quickly, so I does not need to be very large (e.g., $I = 20$). A revised algorithm for Newton's method incorporating these changes is given in Table 2.5.

2.4.3 Some Theory

To guarantee that Newton's method works you need to pick a starting point near the solution and you also need to require that $f'(\bar{x}) \neq 0$. It is possible to state this more formally, and this is done next. In doing this, recall that $\bar{x}$ is an exact solution of $f(x) = 0$. Also, it is always possible to get lucky and have $x_i = \bar{x}$, at which point the equation is solved and the iteration stopped. When this happens, we will say that x_0 possess the *finite termination property* (see Exercise 2.23).

Theorem 2.2. *Assume $f \in C^2(a,b)$, with $a < \bar{x} < b$ and $f'(x) \neq 0$ for $a < x < b$. In this case, for x_0 chosen close to $\bar{x}$, Newton's method, as given in (2.10), will converge to $\bar{x}$. Moreover, if $f''(\bar{x}) \neq 0$, and x_0 does not have the finite termination property, then*

$$|x_{i+1} - \bar{x}| = C_i |x_i - \bar{x}|^2, \tag{2.16}$$

where, as $i \to \infty$,

$$C_i \to \left| \frac{f''(\bar{x})}{2f'(\bar{x})} \right| . \tag{2.17}$$

Outline of Proof: The requirement that x_0 is close to $\bar{x}$, and Taylor's theorem, are the keys to proving this. Although the discussion to follow contains many of the steps of the proof, the objective is to explain how (2.17) is obtained.

```
pick:  $x$
       $tol > 0$
       $M > 0$
       $I > 0$
 let:  $err = 3 * tol$
       $i = 0$
loop:  while $err > tol$
          $z = f(x)/f'(x)$
          $err = abs(z)$
          $x = x - z$
          $i = i + 1$
          if $abs(x) > M$ or $I < i$, then stop
       end
```

Table 2.5 Revised algorithm for solving $f(x) = 0$ using Newton's method. The procedure stops when $|x_{i+1} - x_i| < tol$, or the method appears to fail.

Setting $e_i = x_i - \bar{x}$, then from (2.10) we have that $e_{i+1} = e_i - f(x_i)/f'(x_i)$. Since $x_i = \bar{x} + e_i$, then using Taylor's theorem (twice) we have that

$$f(x_i) = f(\bar{x} + e_i) = f(\bar{x}) + e_i f'(\bar{x}) + \frac{1}{2} e_i^2 f''(\bar{x}) + \cdots ,$$

and

$$f'(x_i) = f'(\bar{x} + e_i) = f'(\bar{x}) + e_i f''(\bar{x}) + \cdots .$$

In these expansions we are assuming that e_i is small, and this is based on the stated hypothesis that x_0 is close to $\bar{x}$. Since $f(\bar{x}) = 0$, and setting $z = f''(\bar{x})/f'(\bar{x})$, we have that

$$\begin{aligned} \frac{f(x_i)}{f'(x_i)} &= \frac{e_i f'(\bar{x}) + \frac{1}{2}e_i^2 f''(\bar{x}) + \cdots}{f'(\bar{x}) + e_i f''(\bar{x}) + \cdots} \\ &= e_i \frac{1 + \frac{1}{2}e_i z + \cdots}{1 + e_i z + \cdots} \end{aligned} \tag{2.18}$$

Note that, by assumption, $z \neq 0$ (what happens if it is zero is explained below). Recalling that if y is close to zero, then (see Section A.1)

$$\frac{1}{1+y} = 1 - y + y^2 - y^3 + \cdots .$$

This applies to the denominator in (2.18) with $y = e_i z + \cdots$, and so

$$\begin{aligned} \frac{f(x_i)}{f'(x_i)} &= e_i(1 + \frac{1}{2}e_i z + \cdots)(1 - e_i z + \cdots) \\ &= e_i - \frac{1}{2}e_i^2 z + \cdots . \end{aligned} \tag{2.19}$$

With this, $e_{i+1} = e_i - f(x_i)/f'(x_i)$ can be rewritten as $e_{i+1} = \frac{1}{2}ze_i^2 + \cdots$. This shows that as the method converges to the solution, the limiting expression is $e_{i+1} = \frac{1}{2}ze_i^2$, and this is where the conclusion in (2.17) comes from. □

Although it's certainly important to know that the method converges, the more useful piece of information in the above theorem is that, once the method starts to get close to the solution, then

$$|x_{i+1} - \bar{x}| \approx C|x_i - \bar{x}|^2, \tag{2.20}$$

where C is the positive constant given in (2.17). It's also possible to show that, in this case,

$$|x_{i+1} - x_i| \approx C|x_i - x_{i-1}|^2. \tag{2.21}$$

Knowing that the error, as given in (2.20), or the iterative error, as given in (2.21), should decrease quadratically is useful in checking that the algorithm is behaving as it should. This also means that our observation in Section 2.4.3 that $e_{i+1} \approx Ce_i^\gamma$ is correct. In particular, it means that γ, which is the *order of convergence* for Newton's method, is indeed 2.

To obtain (2.16) and (2.17) it was assumed $f''(\bar{x}) \neq 0$. In the case of when it is zero, Newton's method will converge faster than second-order. How this happens in explained in Exercise 2.27, but this is not typical, and for most problems the convergence is second-order.

One of the disappointing aspects of Theorem 2.2 is that it states that if you start close enough then the method works. There is little indication of what this means, and so you can end up simply guessing a starting value x_0, hoping it works. One way to avoid this uncertainty is to sketch the functions in the equation, and from this find to well-positioned starting points. This approach is used in many of the exercises at the end of the chapter.

Example

When using Newton's method it can be unpredictable what solution it will find. This is illustrated in Figure 2.17. In the lower graph the function is plotted, and it shows that there are four solutions of $f(x) = 0$. The upper graph gives the value of the root calculated using Newton's method as a function of the starting location. For example, taking a starting value of $x = 5$ the method converges to the root between 4 and 6. In fact, there are a wide range of starting values near $x = 5$ for which Newton's method produces the same result. The same is true for starting values near the other roots. Where the method appears to produce more unpredictable results is when the staring points are near the local max and min points of the function. As an example, for a starting points near $x = -1$ it is possible to have Newton's method converge to the root near $x = 2$ or the one between 4 and 6. ■

Example

Newton's method can be used to derive algorithms for calculating mathematical expressions using more elementary operations. As an example, Newton's method can be used to perform division using only subtraction and multiplication. To explain, given $\alpha > 0$, suppose we want to compute $1/\alpha$. In other words, we want to find the value of x which satisfies $x = 1/\alpha$ (without actually doing division). One might try rewriting the equation as $\alpha x - 1 = 0$, or as $\alpha^2 x^2 - 1 = 0$, but neither works. For example, using $f(x) = \alpha^2 x^2 - 1$, then (2.10) becomes

$$x_{i+1} = \frac{1}{2}x_i - \frac{1}{2\alpha^2 x_i},$$

and this requires a division. A choice that does work is to rewrite the equation as $\alpha = 1/x$, so $f(x) = \alpha - 1/x$. In this case (2.10) becomes

$$x_{i+1} = x_i(2 - \alpha x_i),$$

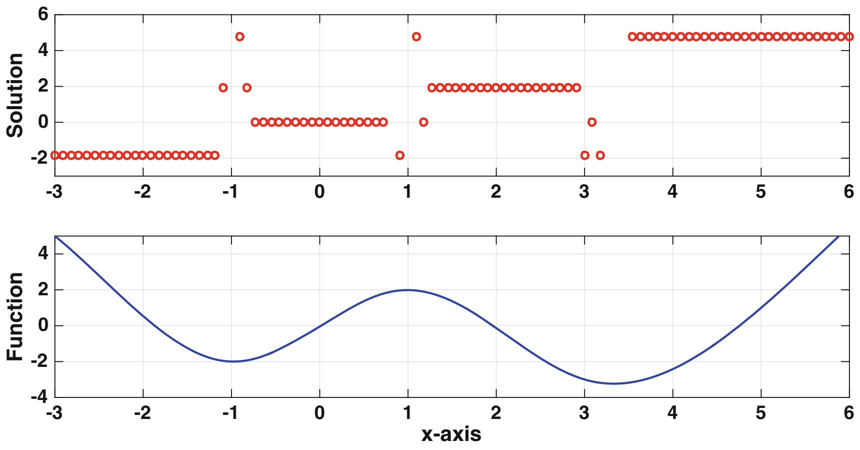


Figure 2.17 Lower: Plot of $f(x)$. Upper: Solution of $f(x) = 0$ obtained using Newton's method as a function of the starting value.

which consists of two multiplications and one subtraction (per iteration). This procedure has been used by more than one computer system to perform floating-point division. It is also possible to use Newton to compute $\sqrt{\alpha}$ using addition, subtraction, multiplication, and division (Exercise 2.6), and to do something similar for $\alpha^{1/3}$ (Exercise 2.7). ■

2.5 Secant Method

One of the complications with using Newton's method is that it requires computing the first derivative. For a function like $f(x) = x^3 + 2x + 2$ this is rather simple, but this is often not the case in applications. In such situations there is an alternative known as the secant method. This idea comes directly from calculus, where a secant line is used to introduce the idea of a tangent line to a curve. Namely, given x_0, one picks a nearby point x_1 and then draws the line passing through $(x_0, f(x_0))$ and $(x_1, f(x_1))$. This is illustrated in Figure 2.18. The formula for the secant line in this case is

$$y = f(x_0) + m_0(x - x_0), \tag{2.22}$$

where the slope is

$$m_0 = \frac{f(x_1) - f(x_0)}{x_1 - x_0}\,. \tag{2.23}$$

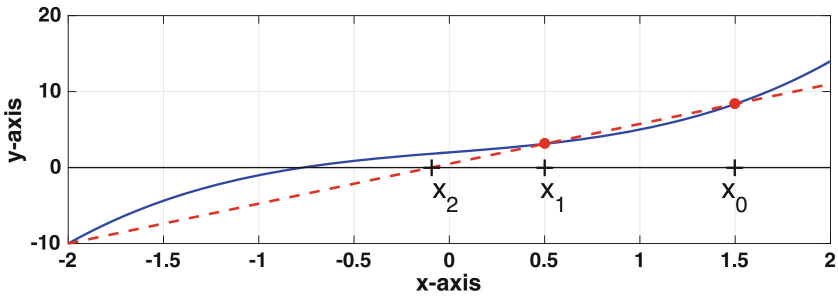


Figure 2.18 First step using the secant method. The solid curve is $y = x^3 + 2x + 2$ and the dashed line is the secant line using $x_0 = 1.5$ and $x_1 = 0.5$.

In comparing this to the tangent line, shown in Figure 2.11, it is evident that they are fairly close. Using a similar approach as in Newton's method, we now replace $f(x) = 0$ with $f(x_0) + m_0(x - x_0) = 0$, and solve to find

$$x_2 = x_0 - \frac{f(x_0)}{m_0},$$

or equivalently

$$x_2 = x_0 - \frac{f(x_0)(x_1 - x_0)}{f(x_1) - f(x_0)}.$$

It is now possible to construct a new secant line using x_1 and x_2, and then determine the next approximation x_3. It is found that

$$x_3 = x_1 - \frac{f(x_1)(x_2 - x_1)}{f(x_2) - f(x_1)}.$$

Generalizing this formula we have that

$$x_{i+1} = x_{i-1} - \frac{f(x_{i-1})(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})}.$$

It is not hard to show that this can be rewritten as

$$x_{i+1} = x_i - \frac{f(x_i)(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})}, \text{ for } i = 1, 2, \cdots. \tag{2.24}$$

This formula is known as the *secant method.* To use it, it is necessary to provide two starting values: x_0 and x_1.

An algorithm for the secant method is given in Table 2.6. It differs slightly from (2.24) in that the values of x are not indexed. Instead, as the procedure proceeds $X = x_{i-1}$ and $x = x_i$. Note that the stopping condition is based on the iteration error $|z| = |x_{i+1} - x_i|$. Assuming the solution is nonzero, one could instead use the relative iteration error $|(x_{i+1} - x_i)/x_{i+1}|$. Also, as with Newton's method, the constants M and I are used to stop the procedure if the method appears to be failing.

```
pick:  $x$ and $X$
       $tol > 0$
       $M > 0$
       $I > 0$
 let:  $err = 3 * tol$
       $i = 0$
loop:  while $err > tol$
           $z = f(x)(x - X)/(f(x) - f(X))$
           $err = abs(z)$
           $X = x$
           $x = x - z$
           $i = i + 1$
           if $abs(x) > M$ or $I < i$, then stop
       end
```

Table 2.6 Algorithm for solving $f(x) = 0$ using the secant method. The procedure stops when $|x_{i+1} - x_i| < tol$, or the method appears to fail.

Example

If $f(x) = x^3 + 2x + 2$, then (2.24) becomes

$$x_{i+1} = x_i - \frac{(x_i^3 + 2x_i + 2)(x_i - x_{i-1})}{(x_i^3 + 2x_i + 2) - (x_{i-1}^3 + 2x_{i-1} + 2)} \tag{2.25}$$

$$= x_i - \frac{x_i^3 + 2x_i + 2}{x_i^2 + x_i x_{i-1} + x_{i-1}^2 + 2}\,. \tag{2.26}$$

From Figure 2.13, there is only one solution and it is in the interval $-1 < x < 0$. So, it's reasonable to take $x_0 = -1/3$ and $x_1 = -2/3$. From (2.26),

$$\begin{aligned} x_2 &= -\frac{2}{3} - \frac{10/27}{25/9} \\ &= -\frac{4}{5} \\ &= -0.8. \end{aligned}$$

The remaining values are computed using MATLAB, and the results are given in Table 2.7. In addition, the values for error $e_i = |x_i - \bar{x}|$, where $\bar{x}$ is the exact solution, are given. The latter is also plotted in Figure 2.19. Not surprisingly, when comparing this with Table 2.2, the secant method is much

i	x_i	$\lvert x_i - \bar{x}\rvert$	γ
0	−0.33333333333333	4.38e−01	
1	−0.66666666666667	1.04e−01	
2	−0.80000000000000	2.91e−02	
3	−0.76904176904177	1.88e−03	1.7749
4	−0.77088382152809	3.32e−05	1.6426
5	−0.77091703510617	3.80e−08	1.6565
6	−0.77091699705848	7.71e−13	1.6325
7	−0.77091699705925	1.11e−16	1.3172
8	−0.77091699705925	1.11e−16	1.0000

Table 2.7 Solving $x^3 + 2x + 2 = 0$ using the secant method formula given in (2.25). Also given is the error $e_i = |x_i - \bar{x}|$, and the approximate order of convergence γ as determined from (2.15).

faster than bisection. In comparison to Newton's method, also shown in Figure 2.19, the secant method is slower but not significantly slower. To calculate the order of convergence we can use (2.15), and the values for this are given in Table 2.7. It appears that it is approaching a value of approximately 1.6, before it reaches the resolution of double precision. As will be stated below, the value determined from the theory is $\gamma = (1+\sqrt{5})/2 \approx 1.6180$. You might recognize that this is the golden ratio. How this could arise as the order of convergence can be explained by looking at the powers in the error values in Table 2.7. Specifically, the powers are increasing like a Fibonacci sequence, and the ratio of successive terms of a Fibonacci sequence approaches the golden ratio (this was proved by Kepler). This ratio is nothing more than the ratio given in (2.15) used to determine γ. The formal proof that it is the golden ratio is a bit more involved and it is outlined in Exercise 2.26. ■

2.5.1 Some Theory

Similar to what occurs with Newton's method, to guarantee that the secant method works you need to pick starting points near the solution. To state the result more formally, recall that $\bar{x}$ is an exact solution of $f(x) = 0$. Also, if it ever occurs that $x_i = \bar{x}$, then the equation is solved and the iteration stopped. When this happens, it is said that the starting points possess the finite termination property.

Theorem 2.3. *Assume $f \in C^2(a,b)$, with $a < \bar{x} < b$ and $f'(x) \neq 0$ for $a < x < b$. In this case, for x_0 and x_1 chosen close to $\bar{x}$, the secant method, as given in (2.24), will converge to $\bar{x}$. Moreover, if $f''(\bar{x}) \neq 0$, and x_0 and x_1 do not have the finite termination property, then*

$$|x_{i+1} - \bar{x}| = D_i |x_i - \bar{x}|^\gamma, \tag{2.27}$$

where $\gamma = (1+\sqrt{5})/2$, and as $i \to \infty$,

$$D_i \to \left| \frac{f''(\bar{x})}{2f'(\bar{x})} \right|^{\gamma-1}. \tag{2.28}$$

An outline of the proof, which explains where this particular value of γ comes from, is provided in Exercise 2.26. Because the rate of convergence is better than linear, which corresponds to $\gamma = 1$, but not as fast as quadratic, which corresponds to $\gamma = 2$, the secant method is said to be *superlinear*.

2.6 Other Ideas

The problem of solving $f(x) = 0$ is so old, and so simple to state, that many methods have been derived for solving it. For those curious about other possibilities, Traub [1982] discusses over 40 different methods. What is somewhat surprising is that research papers are still being published in this area, and those interested in this should consult Wilkins and Gu [2013] or Cordero et al. [2015]. One method of recent vintage that is particularly interesting uses what is called Chebyshev interpolation. The idea is to replace $f(x)$ with an interpolation polynomial (see Section 5.5.4), and then find its roots by solving an eigenvalue problem (see Chapter 4). In contrast to the methods considered in this chapter, the Chebyshev interpolation approach

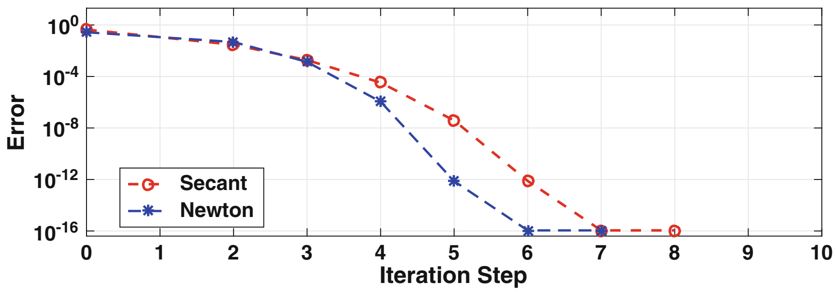


Figure 2.19 The error $|x_i - \bar{x}|$, from Table 2.7, for the secant method. For comparison, the curve for Newton's method is also given.

has the capability of simultaneously determining all of the roots in a given interval. More about this can be found in Boyd [2014].

Those who write general purpose codes tend to use a hybrid approach, which means they use a combination of methods. As an example, one could start with the bisection method to get close to the root, and then switch to the secant or Newton's method to speed things up. A well-known variation of this is the Van Wijngaarden–Dekker–Brent method, which is what MATLAB uses for the `fzero` command.

There is also the question of how do you solve problems with more than one equation. The easiest to extend is Newton's method, which uses the multi-dimensional version of Taylor's theorem, and this will be considered in Section 3.10. The bisection method is more difficult to extend because the idea of a trapping interval does not work in multi-dimensions. However, there are methods for multi-dimensional problems that have a lot in common with bisection, and one example is the Nelder-Mead algorithm described in Section 8.8. There is also a multi-dimensional analogue for the secant method and it is known as Broyden's method [Dennis and Schnabel, 1996].

2.6.1 Is Newton's Method Really Newton's Method?

Given that the quintessential numerical procedure is Newton's method, it is worth knowing if the name is appropriate. To set the stage, in the late 1600s one of the central research problems concerned computing the roots of polynomials. An example used by Newton is the equation $x^3 - 2x - 5 = 0$. Knowing the solution is near $x = 2$, the approach was to write $x = 2 + z$, and then consider $(2 + z)^3 - 2(2 + z) - 5 = 0$. Assuming z is small, so the z^3 and z^2 terms can be dropped, the conclusion is $z \approx 1/10$. How to improve this approximation was the problem considered at the time. Newton's idea, which was published in 1685, was to write $z = 0.1 + u$, substitute this into the equation for z, and then repeat the earlier argument to find u. Another proposal, made by Joseph Raphson in about 1690, was to write $x = 2.1 + u$ and then substitute this into the original equation for x [Raphson, 1690]. Mathematically, these methods are equivalent, and one gets the same value for u. However, because using the original equation is easier, it allowed Raphson to make a critical observation, which is that his method for solving $x^3 - 2x - 5 = 0$ can be written as

$$x = y - \frac{y^3 - 2y - 5}{3y^2 - 2},$$

where y is the previously calculated value for x. This is exactly what Newton's method, as given in (2.10), states should be done, where $x_0 = 2$. However, the calculus had not yet been developed, and as far as Raphson was concerned this was simply an algebraic approximation that worked. For this reason, he never extended the method to non-polynomial equations, even after the calculus was known [Kollerstrom, 1992; Ypma, 1995]. This has caused some historians to look elsewhere for the person responsible for (2.10). One candidate is Thomas Simpson, who published a book entitled "Essays on Several Curious and Useful Subjects in Speculative and Mix'd Mathematicks, Illustrated by a Variety of Examples" in which (2.10) does indeed appear. The case for Simpson is made very strongly by Kollerstrom [1992]. However, Simpson's book was published in 1740, which means it took over 40 years to make this connection. This is a long time, particularly for someone of Newton's capability. One argument made in support of Newton is that he used (2.10) to solve Kepler's equation $x - e\sin x = M$ before 1713 (when it appeared in the second edition of the *Principia*). Also, there are one or more letters, which were written in 1692, in which he discusses derivatives of equations [Wallis, 1699]. However, the Kepler solution can be obtained without using calculus [Ypma, 1995], and the letters are lost [Kollerstrom, 1992]. In the end, the real reason it is referred to as Newton's method is attributed to Fourier [1831], who simply said it was Newton's method (even if it might not be exclusively Newton's method).

Exercises

2.1. True or False: If Newton's method converges to a solution $\bar{x}$ for a particular choice of x_0, then it will converge to $\bar{x}$ for any starting point between $\bar{x}$ and x_0.

2.2. The problem involves using the bisection method on some of the examples at the beginning of the chapter.
(a) In Figure 2.3, what solution will the bisection method converge to if $a_0 = -1.5$ and $b_0 = 1$? What if $a_0 = -2$ and $b_0 = 1.5$?
(b) In Figure 2.4, what solution will the bisection method converge to if $a_0 = -2$ and $b_0 = 2$? What if $a_0 = -2$ and $b_0 = 4$? Also, explain how it is possible for the bisection method to find the solution $x = 1$.

2.3. The following questions concern the bisection method. You should assume that the method does not get lucky and ends up with $f(c_i) = 0$.
(a) How sensitive is the bisection method to the width of the initial interval? For example, in Table 2.2, if the initial interval is cut in half, what is the expected reduction in the number of iterations?
(b) Both $f_1(x) = x^3 - 2$ and $f_2(x) = e^x - 5\sin(x^3) - 3\cos(x)$ have one zero in the interval $0 < x < 2$. If the bisection takes 34 iterations to solve $f_1(x) = 0$, how many will it likely take to solve $f_2(x) = 0$?

A	$x^2 - 1 = e^x$ for $x \geq 0$
B	$\sin x = 1 - x^2$ for $x \geq 0$
C	$x^2 + 2x = 1/(1 + x^2)$ for $x \geq 0$
D	$e^{-x} = x^3$
E	$\ln x = 2 - x^2$
F	$\sin x = 2\sin(x + \pi/4)$ for $0 \leq x \leq \pi$

Table 2.8 Equations used in Exercise 2.4.

2.4. Each of the equations in Table 2.8 has one solution. Select an equation, and then do the following:
(a) Sketch the functions to determine the approximate location of the solution.
(b) For the bisection method, provide an initial interval that can be used to find the solution, and provide an explanation why it works. With this calculate, by hand, c_0 and c_1.
(c) What is Newton's iteration formula (2.10) for this equation? Also, provide a starting point x_0 for the solution, providing an explanation of why it is a good choice. With this calculate, by hand, x_1.
(d) What is the secant iteration formula (2.24) for this equation? Also, provide starting points x_0 and x_1 for the solution, providing an explanation of why they are a good choice. With this calculate, by hand, x_2.
(e) Compute the solution of the equation. Your answer should be correct to at least four significant digits. Make sure to state which numerical method was used, why you made this choice, and what error condition you used to stop the calculation.

2.5. This exercise concerns an implicitly defined function $y(x)$, defined through an equation of the form $F(x, y) = 0$. For the equations in Table 2.9, select one and then do the following:
(a) Sketch the functions as a function of y, and use this to show that for each x, there is one solution y.
(b) Use the sketch in part (a) to find a bounded interval for y that contains the solution (note the interval will possibly depend on x).
(c) What is Newton's iteration formula (2.10) for this equation? Also, provide a starting point y_0 for the solution, providing an explanation of why it is a good choice.
(d) What is the secant iteration formula (2.24) for this equation? Also, provide starting points y_0 and y_1 for the solution, providing an explanation of why they are a good choice.

A	$y^3 = x - y$ for $x > 0$
B	$1/(x^2 + y^2) = y$ for $x > 0$
C	$\ln(x + y) = -y^3$ for $x > 1$
D	$y^3 e^{2y} = x$ for $x > 0$
E	$x(1 - y^2) = \sqrt{y}$ for $x > 0$

Table 2.9 Equations used in Exercise 2.5.

(e) Compute $y(2)$. Your answer should be correct to at least four significant digits. Make sure to state which numerical method was used, why you made this choice, and what error condition you used to stop the calculation.
(f) Plot $y(x)$ for $2 \le x \le 10$. You should also provide an explanation of the algorithm you used to do this.

2.6. This problem examines how to use an iterative method to calculate the square root of a positive number α. In other words, the algorithm calculates $\sqrt{\alpha}$. The procedure can only contain the four elementary arithmetic operations (addition, subtraction, multiplication, and division). Also, you do not need to actually calculate anything, you just need to describe how to do this with the respective method.
(a) How can this be done using the bisection method?
(b) How can this be done using the Newton's method?

2.7. Show how to use Newton's method to evaluate $2^{1/3}$. Your procedure, or formula, should only contain additions, subtractions, multiplications, and/or divisions.

2.8. The values for the solution of $f(x) = 0$ in Table 2.10 were computed using MATLAB. What method was most likely used (bisection, Newton, secant)? Make sure to explain why it is the method you claim.

2.9. This exercise explores how to use Newton's method to evaluate an inverse function. To explain, given $y = g(x)$, then the inverse function satisfies $x = g^{-1}(y)$. The problem is, given y, what is the value of x?
(a) Assuming y is given, and setting $f(x) = y - g(x)$, show that Newton's method (2.10) gives

$$x_{i+1} = x_i + \frac{y - g(x_i)}{g'(x_i)}\ , \text{ for } i = 0, 1, 2, 3, \cdots$$

Iteration	Computed Solution
1	1.250000000000000
2	1.025000000000000
3	1.000304878048780
4	1.000000046461147
5	1.000000000000001

Table 2.10 Values for Exercise 2.8.

(b) Use the result from part (a) to evaluate e^2 and e^{-3} using the $\ln(x)$ function.
(c) Use the result from part (a) to evaluate arccos(1/2) and arccos(1/3).
(d) The error function is defined as

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-s^2} ds.$$

The inverse error function is denoted as $\operatorname{erf}^{-1}(x)$. Use the result from part (a) to evaluate $\operatorname{erf}^{-1}(1/2)$ and $\operatorname{erf}^{-1}(1/3)$. In doing this, you can use MATLAB's `erf` command to evaluate the error function.
(e) The complete elliptic integral of the first kind is defined as

$$\mathrm{K}(x) = \int_0^1 \frac{1}{\sqrt{(1-s^2)(1-xs^2)}} ds.$$

The inverse function is denoted as $\mathrm{K}^{-1}(x)$. Use the result from part (a) to evaluate $\mathrm{K}^{-1}(2)$ and $\mathrm{K}^{-1}(4)$. It is useful to know that

$$K'(x) = \frac{E(x) - (1-x)K(x)}{2x(1-x)},$$

where $E(x)$ is the complete elliptic integral of the second kind. In doing this, you can use MATLAB's `ellipke` command to evaluate K and E.

2.10. Four different methods were used to solve $f(x) = 0$, and the computed values for x_1, x_2, x_3, $\cdots$ are shown in Table 2.11.
(a) One of them is Newton's method. Which of the four is most likely Newton's method, and why?
(b) One of them is the bisection method. Which of the four is most likely the bisection method, and why?
(c) Suppose someone claimed they computed the solution using the secant method, and they obtained the results given for Method 3. Why would you tell them that they are most likely mistaken (i.e., they are wrong)?

i	Method 1	Method 2	Method 3	Method 4
1	1.10000000000000	1.02000000000000	1.05000000000000	1.03162277660168
2	1.01000000000000	1.00400000000000	1.02500000000000	1.00562341325190
3	1.00010000000000	1.00080000000000	1.01250000000000	1.00042169650343
4	1.00000001000000	1.00016000000000	1.00625000000000	1.00000865964323
5	1.00000000000000	1.00003200000000	1.00312500000000	1.00000002548297
6	1.00000000000000	1.00000640000000	1.00156250000000	1.00000000000407
7	1.00000000000000	1.00000128000000	1.00078125000000	1.00000000000000
8	1.00000000000000	1.00000025600000	1.00039062500000	1.00000000000000

Table 2.11 Values for Exercise 2.10.

2.11.(a) Suppose to solve $f(x) = 0$ one finds Newton's method takes 20 iterations and the secant method takes 30. When is it possible that the secant method takes less computing time? Make sure to explain your answer.

(b) Which of the curves in Figure 2.20 corresponds to Newton's method and which one to the bisection method? Make sure to justify your answers.

2.12. The exercise examines various choices that can be made for the approximate solution using the bisection method. Assume that the subinterval (a_i, b_i) has just been calculated, and the goal is to now determine what point $\bar{c}_i$ should be selected from this subinterval as the approximation for the solution $\bar{x}$.

(a) Whatever choice is made, the error is $\bar{c}_i - \bar{x}$. Sketch this as a function of $\bar{x}$, for $a_i \le \bar{x} \le b_i$. Explain why the minimum error occurs when $b_i - \bar{c}_i = \bar{c}_i - a_i$, and from this conclude that $\bar{c}_i = (b_i + a_i)/2$. In other words, one should select the midpoint of the subinterval.

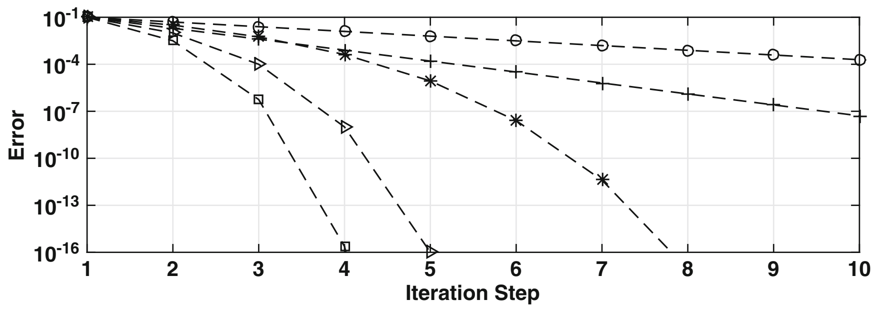


Figure 2.20 Graph for Exercise 2.11(b).

(b) Suppose one instead uses the relative error $(\bar{c}_i - \bar{x})/\bar{x}$. This requires a nonzero solution, so it is assumed here that $0 < a_i < b_i$. By sketching $(\bar{c}_i - \bar{x})/\bar{x}$, for $a_i \le \bar{x} \le b_i$, explain why the minimum error occurs when $\bar{c}_i = 2a_ib_i/(b_i + a_i)$.

(c) Does it make much difference which choice is made for $\bar{c}_i$? In answering this, assume that the stopping condition for the loop in Table 2.1 is the same irrespective of the choice for $\bar{c}_i$.

2.13. In this problem assume that $f(x)$ satisfies the conditions of Theorems 2.2 and 2.3.

(a) Suppose it is claimed that the values in Table 2.2 were produced using Newton's method. What argument can you make to refute this claim?

(b) Suppose it is claimed that the values in Table 2.7 were produced using the bisection method. What argument can you make to show this is unlikely?

(c) When using the secant method, does it make any difference which point is labeled x_0 and which is labeled x_1? In particular, what happens to x_2 and x_3 if you switch which point is labeled x_0 and x_1?

2.14. This problem concerns the configuration shown in Figure 2.21. There are four straight sides, of fixed length, that are free to rotate at the vertices. The bottom side, of length a, does not move. What is of interest is how the angle φ changes as θ changes. This is a situation that arises in kinematics, and it has been found that the two angles are related through the equation

$$A\cos\theta - B\cos\varphi + C = \cos(\theta - \varphi), \tag{2.29}$$

where $A = a/b$, $B = a/d$, and $C = (a^2 + b^2 - c^2 + d^2)/(2bd)$. In textbooks on the kinematics of machines this is known as the Freudenstein equation.

(a) If $\theta = 0$ then a triangle is produced. In this case, using the law of cosines, find φ and show that this is the same result obtained from (2.29).

In the rest of the problem let $a = 3/2$, $b = \sqrt{3}$, $c = 1$, and $d = 1/2$.

(b) Taking $\theta = 0$, plot the left and right hand sides of (2.29) for $0 \le \varphi \le 2\pi$ and show that there are two solutions. Explain geometrically why there are two, and identify which one corresponds to the configuration shown in Figure 2.21.

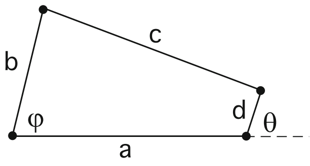


Figure 2.21 Figure for Exercise 2.14.

(c) Assuming Newton's method is used to find φ, what is (2.10) when applied to (2.29)? Use this to calculate φ for $\theta = \pi/6$ and $\theta = \pi/3$. Your values should be correct to six significant digits.

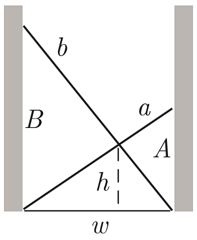


Figure 2.22 Figure for Exercise 2.15.

(d) As θ increases from $\theta = 0$ to $\theta = 2\pi$, the vertex connecting side b and c traces out a portion of a circle. Explain why, and find the maximum and minimum values of φ that determine this circular arc.

2.15. The crossing ladders problem is the following: Two ladders of length a and b, with $a \leq b$, are leaning across an alleyway between two buildings as shown in Figure 2.22. If they cross at a height h, what is the width w of the alleyway?

(a) Using similar triangles and the Pythagorean theorem show that $A^2+w^2 = a^2$, $B^2 + w^2 = b^2$, and
$$\frac{1}{h} = \frac{1}{A} + \frac{1}{B}.$$
In these formulas, A and B are the vertical heights of the two ladders. From this show that the problem reduces to solving the following equation for A:
$$h^2A^2 = (A - h)^2(b^2 - a^2 + A^2).$$
(b) Explain why $h \leq A \leq a$. Also, by sketching the functions in the equation from part (a), show that there are two positive solutions for A (assuming that $a < b$). Note that you might find it easier to first rewrite the equation before doing the sketch.
(c) Newton's method is going to be used to find A. What does (2.10) reduce to in this case? Based on part (b), what would be a good choice for the starting point in this case? Make sure to explain why.
(d) The exact solution is easy to determine in the case of when $a = b$. For this case, picking a value for b and h, use Newton's method to find A, and show that it gives the correct result.
(e) Taking $a = 20$, $b = 30$, and $h = 8$, use Newton's method to compute A and from this determine w. Your answers should be correct to at least eight significant digits. Also, state what you used for a starting value, and explain why you made this choice.

2.16. This problem considers finding α so the line $y = \alpha x$ is tangent to the curve $y = \cos(2\pi x)$. You can assume that $\alpha > 0$, and the tangency points occur for $x > 0$.

(a) By sketching $y = \alpha x$ and $y = \cos(2\pi x)$, explain why there are an infinite number of solutions for α. Also, use this sketch to determine the approximate locations where tangency occurs.
(b) Calculate the largest value of α. Make sure to explain what mathematical problem you solve to find α, as well as which numerical method you used, why you made this choice, and what error condition was used to stop the calculation.

2.17. From the Michaelis-Menten model for an enzyme-catalyzed reaction, the amount of the substrate S present at time t satisfies the differential equation

$$\frac{dS}{dt} = -\frac{v_m S}{K_M + S},$$

where v_m and K_M are positive constants. Solving this, one obtains

$$K_M \ln(S/S_0) + S = S_0 - v_m t, \tag{2.30}$$

where S_0 is the (nonzero) amount at the beginning. To find S, for any given value of t, it is necessary to solve this nonlinear equation, and how this might be done is considered in this exercise. How to find the numerical solution of this differential equation is considered in Exercise 7.20.
(a) As a function of S, sketch the two functions $K_M \ln(S/S_0)$ and $S_0 - v_m t - S$. Do this for $t = 0$ and for $t > 0$. Use this to explain why there is only one solution of (2.30).
(b) Use part (a) to explain why the solution satisfies $0 < S \leq S_0$.
(c) Suppose (2.30) is to be solved using Newton's method. What does (2.10) reduce to in this case? Based on parts (a) and (b), what would be a good choice for the starting point in this case? Make sure to explain why.
(d) Based on parts (a) and (b), what would be a good choice for a starting interval when using the bisection method to solve (2.30)? Make sure to explain why.
(e) Suppose (2.30) is to be solved using the secant method. What does (2.24) reduce to in this case? Based on parts (a) and (b), what would be a good choice for the two starting points in this case? Make sure to explain why.
(f) It is found that for sucrose, $v_m = 0.76$ mM/min and $K_M = 16.7$ mM [Johnson and Goody, 2011]. Also, assume that $S_0 = 100$ mM. Use one of the above methods from (c)–(e) to find the value of S at $t = 1$ min, at $t = 10$ min, and at $t = 100$ min. Your answers should be correct to at least four significant digits. Make sure to state which method was used, why you made this choice, and what error condition you used to stop the calculation.
(g) Using the parameter values given in part (f), and one of the above numerical methods from (c)–(e), plot S for $0 \leq t \leq 1000$.
(h) Explain how it is possible to produce the plot in part (g), from (2.30), without having to use a numerical solver to find S.

2.18. According to the Colebrook equation, the friction factor f for turbulent flow in a pipe is found by solving

$$\frac{1}{\sqrt{f}} = -2\log_{10}\left(\alpha + \frac{\beta}{\sqrt{f}}\right),$$

where α and β are constants that satisfy $0 < \alpha < 1$ and $0 < \beta < 1$. By setting $x = 1/\sqrt{f}$, this equation can be rewritten as

$$x = -2\log_{10}(\alpha + \beta x)\,, \tag{2.31}$$

where $0 < x < \infty$.

(a) Sketch the two functions in (2.31) for $0 < x < \infty$. Use this to explain why there is only one solution, and that it is in the interval $0 < x < 2\log_{10}(1/\alpha)$.
In the rest of the problem assume that $\alpha = 10^{-2}$ and $\beta = 10^{-4}$, which are typical values for these constants.
(b) What does (2.10) reduce to for (2.31)? Based on part (a), what would be a good choice for x_0? Make sure to explain why.
(c) Based on part (a), what would be a good choice for a_0 and b_0 when using the bisection method to solve (2.31)? Make sure to explain why.
(d) Based on part (a), what would be a good choice for x_0 and x_1 when using the secant method to solve (2.31)? Make sure to explain why.
(e) Use one of the methods from (b)–(d) to solve (2.31), and from this determine the value of f. Make sure to state which method was used, why you made this choice, and what error condition you used to stop the calculation.

2.19. The terminal velocity v of a sphere falling through the air satisfies

$$v^2 = \frac{2mg}{\rho A c_D}\,, \tag{2.32}$$

where $A = \pi d^2/4$. In this expression, ρ is the density of air, m and d are the mass and diameter of the sphere, and g is the gravitational acceleration constant. The term c_D is known as the drag coefficient, and from experimental data the following formula has been proposed [White, 2005]:

$$c_D = \frac{24}{Re} + \frac{6}{1+\sqrt{Re}} + \frac{2}{5}\,, \tag{2.33}$$

where $Re = \rho v d/\mu$ is known as the Reynolds number and μ is the (dynamic) viscosity of air. In this problem, (2.32) is rewritten in terms of Re, and it is then solved for Re. After this, the value of v is computed. Also, note that the velocity is positive.

(a) Show that it is possible to rewrite (2.32) as $(Re)^2 c_D = \alpha$, where α does not depend on v or Re. After substituting (2.33) into this equation, the

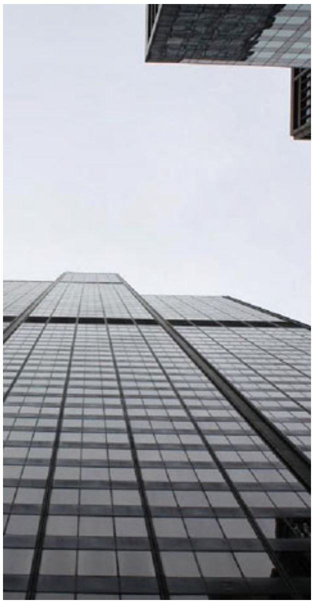


Figure 2.23 A baseball is going to get dropped. Whether you should try to catch it is considered in Exercise 2.19.

problem is rewritten so Re is the variable to solve for. Write down the resulting equation. Assuming the value of Re has been computed, explain how the velocity is calculated.

(b) Write the equation in part (a) as $c_D Re = \alpha/Re$, where c_D is given in (2.33). Sketch the left and right hand sides of this equation as a function of Re. From this show that there is one solution and it is in the interval $0 < Re < \alpha/24$.

(c) Assuming Newton's method is used to find Re, what is (2.10) when applied to the equation in part (a)? Based on your results from part (b), what would be a good choice for a starting point? Make sure to explain why.

(d) Write down the iteration formula if the secant method is used to find Re. Based on your results from part (b), what would be a good choice for the two starting points? Make sure to explain why.

(e) For air, $\mu = 1.8 \times 10^{-5}$, $\rho = 1.2$, $g = 9.8$, and for a baseball, $d = 0.075$ and $m = 0.14$ (using kg, m, s units). What is the terminal velocity of a baseball (assuming it is a perfect sphere)? Make sure to state which method was used, why you made this choice, and what error condition you used to stop the calculation. Also, how does this velocity compare to what is considered the velocity of a typical fastball in professional baseball?

(f) Is it possible to make a baseball out of something so its terminal velocity is approximately the speed of sound?

2.20. The ideal gas law states that $pv = nRT$, where p is the pressure, v is the volume, n is the amount of the substance (in moles), R is the gas constant, and T is the temperature (in Kelvin). An improved version of this is the van der Waals equation of state, and it is given as

$$\left(p + \frac{n^2 a}{v^2}\right)(v - nb) = nRT, \tag{2.34}$$

where a and b are positive constants. Also, p and v are positive.

(a) Explain why there is one solution for v. Note, one way to do this is to rewrite (2.34) as $p + n^2a/v^2 = nRT/(v - nb)$, and then sketch the left and right hand sides as functions of v.

(b) Using the sketch from part (a), explain why the solution satisfies $nb < v < nb + nRT/p$.

(c) Assuming Newton's method is used to find v, what is (2.10) when applied to (2.34)? Based on parts (a) and (b), what would be a good choice for a starting point? Make sure to explain why.

(d) Write down the iteration formula if the secant method is used to find v. Based on parts (a) and (b), what would be a good choice for the two starting points? Make sure to explain why.

(e) Assume that $p = 1$ atm, $n = 1$ mol, and recall that $R = 0.08205746$ L atm/(K mol). Also, for oxygen, which is the gas considered here, $a = 1.382$ and $b = 0.0319$. Note that the values for R, a and b are the exact values given in the *2012 CRC Handbook of Chemistry and Physics*. Using either (c) or (d), determine v at room temperature (you can assume this is 25° C). In your write-up, state why you picked the solver you used, and give your reason(s) for what value you selected for the error tolerance used to stop the solver. Also, explain why it isn't necessary to run the solver to the point that the error in the solution is on the order of machine ε.

(f) Using the values given in part (e), plot v as a function of T, for $0°C \leq T \leq 50°C$. In your write-up explain how you used your solver to do this.

2.21. It is not unusual to have to solve a problem involving a composite function, and such a situation is considered in this exercise. Suppose one wants to find the value(s) of x where $f = 0$. However, f is given in terms of a variable y, and there is a second equation that determines the value of y for any given value of x. In this exercise, let $f = y^3 + 3y + 1$, where

$$y + x = e^{-6y}.$$

So, given x, to evaluate f one first solves the above equation for y and then substitutes this into the formula for f. Although this might appear to an artificially complex question, as will be explained in Section 9.5, it is a fairly common question that arises in applications.

(a) Describe how to use the bisection method to find the value of x where $f = 0$. Make sure to explain how you find the initial interval.

(b) Writing f as $f(y(x))$, explain how to use Newton's method to find the value of x where $f = 0$.
(c) Using MATLAB, find the value of x where $f = 0$.

2.22. In the derivation of Newton's method, to determine the formula for x_{i+1}, the function $f(x)$ is approximated using a first-order Taylor approximation centered at x_i. This problem investigates what happens when you try to use a second-order Taylor approximation.
(a) Approximating $f(x)$ using a second-order Taylor approximation centered at x_i, what is the resulting formula for x_{i+1}? Note your formula will have a $\pm$ in it.
(b) In theory, a second-order Taylor approximation should be more accurate than a first-order Taylor approximation (at least when you are close to the solution). However, the formula in part (a) has several unpleasant complications that Newton's method doesn't have. Identify two of them.
(c) Given that x_{i+1} is close to x_i, what choice should be made for the $\pm$ in part (a)?
(d) One way to avoid the complications considered in part (b) is to note that the Taylor approximation used in part (a) contains a term of the form $(x_{i+1} - x_i)^2$. Explain why this can be approximated with $-(x_{i+1} - x_i)f(x_i)/f'(x_i)$. If this is done, what is the resulting formula for x_{i+1}? Note that the formula you are deriving is known as Halley's method, and it is an example of a third-order method.

2.23. This exercise considers Newton's method and the finite termination property. The equation to solve is $f(x) = 0$, where $f(x) = x(x^2 - 1)$.
(a) Sketch $f(x)$ and locate the three solutions.
(b) Suppose Newton's method is to be used to locate the solution $x = 1$. What does x_0 need to be so the problem is solved exactly in one step? Assume here that $x_0 \neq 1$.
(c) Using your sketch in part (a), and the result from part (b), explain where x_0 should be located (approximately) so the solution $x = 1$ is found in exactly two steps.

2.24. This problem considers solving $f(x) = 0$, where

$$f(x) = 3\cos(x) + \frac{3}{2} + \pi\sqrt{3}.$$

(a) Sketch $f(x)$ for $0 \leq x \leq 2\pi$. How many solutions of $f(x) = 0$ are there in this interval?
(b) Using Newton's method, take $x_0 = 2\pi/3$ and calculate x_1 (by hand). After this, calculate x_2 (by hand).
(c) Sketch the tangent lines for x_0 and x_1. Based on this, determine what the others x_i values will be.
(d) Explain why Newton's method will not converge for this problem.

2.25. This exercise considers solving $f(x) = 0$, where

$$f(x) = \frac{1 - e^{-10x}}{1 + e^{-10x}} .$$

This function is shown in Figure 2.24.
(a) Show that $f'(x) > 0$ for all x.
(b) Describe what happens if one uses Newton's method with $x_0 = 1$. Also, explain why essentially the same thing happens if you use $x_0 = -1$.
(c) Experiment with Newton's method, and find the largest (positive) value of x_0 that will result in Newton's method converging to the correct solution. You only need to find x_0 to two significant digits. Also, give the corresponding value of x_1. Note that keeping track of what happens to x_1 will be helpful for part (d).
(d) Explain why the value of x_0 you found in part (c) can be found by finding the positive solution of $2xf'(x) = f(x)$. What exactly is the relationship between x_0 and x_1 that gives rise to this equation?

2.26. This problem provides an outline of the proof of Theorem 2.3, using the ideas developed for the outline of the proof for Theorem 2.2.
(a) Setting $e_i = x_i - \bar{x}$, show that $e_{i+1} = e_i - f(x_i)(x_i - x_{i-1})/f'(x_i)$.
(b) Writing $x_i = \bar{x} + e_i$, show that

$$\frac{f(x_i)(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})} = \frac{e_i + \frac{1}{2}e_i^2 z + \cdots}{1 + \frac{1}{2}(e_i + e_{i-1})z + \cdots},$$

where $z = f''(\bar{x})/f'(\bar{x})$. You can assume that $z \neq 0$.
(c) Using the results from (a) and (b), show that $e_{i+1} = \frac{1}{2} z e_i e_{i-1} + \cdots$.
(d) To solve $|e_{i+1}| = \frac{1}{2}|z e_i e_{i-1}|$, let $E_i = \ln(|e_i|)$. Show that the equation becomes $E_{i+1} = Z + E_i + E_{i-1}$, where $Z = \ln(|z|/2)$. Also, show that $E_i = -Z + Ar_+^i + Br_-^i$, solves this equation, where $r_\pm$ are the two solutions of $r^2 - r - 1 = 0$, and A and B are arbitrary constants.

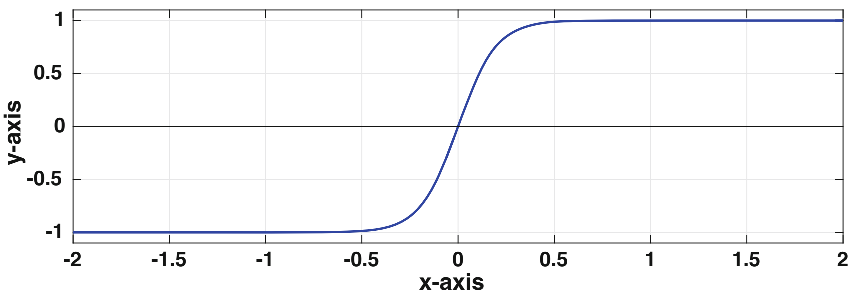


Figure 2.24 Plot of function for Exercise 2.25.

(e) Using the solution from part (d), explain why, as i increases, $E_i \approx (2/|z|)\exp(Ar_+^i)$.
(f) Find the value of γ so that $|e_{i+1}|/|e_i|^\gamma \to D$, and from this derive the result in (2.28).

2.27. This problem explores when Newton's method converges either faster or slower than stated in Theorem 2.2.
(a) Suppose $f''(\bar{x}) = 0$ but $f'''(\bar{x}) \neq 0$. Show that (2.19) is replaced with

$$\frac{f(x_i)}{f'(x_i)} = e_i - \frac{1}{3}e_i^3 w + \cdots ,$$

where $w = f'''(\bar{x})/f'(\bar{x})$. Explain why, in this case, Newton's method is third-order.
(b) Suppose the assumption that $f'(x) \neq 0$ for $a \leq x \leq b$ is replaced with $f'(x) \neq 0$ for $a \leq x \leq b$ except $f'(\bar{x}) = 0$. Show that (2.19) is replaced with

$$\frac{f(x_i)}{f'(x_i)} = \frac{1}{2}e_i + \cdots .$$

Explain why, in this case, Newton's method is first-order.

2.28. This problem explores how to use the floating-point representation to obtain a good starting point for Newton's method. To illustrate this idea, the equation to solve is $x^2 - a = 0$, where $a > 0$, which is used to evaluate $x = \sqrt{a}$.
(a) Show that the formula for Newton's method is

$$x_{i+1} = \frac{1}{2}\left(x_i + \frac{a}{x_i}\right).$$

(b) The floating-point approximation of a has the form $a_f = m \times 2^E$, where

$$m = 1 + \frac{b_1}{2} + \frac{b_2}{2^2} + \cdots + \frac{b_{N-1}}{2^{N-1}} .$$

Also, recall that for $0 \leq y < 1$,

$$\sqrt{1+y} = 1 + \frac{1}{2}y - \frac{1}{8}y^2 + \frac{1}{16}y^3 + \cdots .$$

Assuming E is even, and b_1 or b_2 are nonzero, use the above information to show that

$$\sqrt{a_f} \approx \left(1 + \frac{1}{4}b_1 + \frac{1}{8}b_2\right) \times 2^{E/2}.$$

In what follows it is assumed that this formula is used to determine x_0. Note that this expression only involves additions and multiplications (and not a square root).

(c) What does the approximation in part (b) reduce to for $a = 28 = (1 + 1/2 + 1/2^2) \times 2^4$? How close does x_0 come to $\sqrt{28}$?
(d) Modify the derivation in part (b) to find a starting value in the case of when $a = 50 = \big(1 + 1/2 + 1/2^4\big) \times 2^5$. Make sure to compare x_0 with the exact value. Also, you can assume the value of $\sqrt{2}$ is known.
(e) Write a MATLAB program that uses this idea to calculate $\sqrt{a}$, for any given $a > 1$. The stopping condition for Newton's method should be $|x_{i+1} - x_i|/|x_{i+1}| \leq 10^{-8}$. With this, compute $\sqrt{1.5}$, $\sqrt{33}$, $\sqrt{1001}$, and $\sqrt{0.1}$.

Chapter 3
Matrix Equations

This chapter concentrates on solving the matrix equation $\mathbf{Ax} = \mathbf{b}$, and the chapter to follow investigates various ways to compute the eigenvalues of a matrix $\mathbf{A}$. Together, they are central components of what is called numerical linear algebra. What will be evident from reading this material is the prominent role matrix factorizations play in the subject. To explain what this involves, given a matrix $\mathbf{A}$, one factors it as $\mathbf{A} = \mathbf{BC}$ or $\mathbf{A} = \mathbf{BCD}$. The factors, $\mathbf{B}$, $\mathbf{C}$, and $\mathbf{D}$, are matrices with nice, easy to compute with, properties. The time-consuming computational step is finding the factorization. There are many useful factorizations, and a listing of some considered in this text can be found in the index.

To help bring out the importance of matrix factorization, the first three sections of this chapter are written in reverse order, at least compared to most numerical textbooks. What is done here is to start with the final result, which is the description of the numerical method, then afterwards explain where or how one would come up with such an idea. There are several reasons for reversing the order, one being that the derivation of the method (Section 3.3) is useful mostly for showing where the idea comes from. Once you realize how the method works you also realize there is a more direct method to get the result (Section 3.2). Finally, there is the constructive component, where you actually solve problems with the method, and this is where we start.

3.1 An Example

Consider the following system of equations

$$\begin{aligned} 2x - 4y &= 2 \\ x + 7y &= 10. \end{aligned}$$

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_3

This can be written in matrix form as

$$\begin{pmatrix} 2 & -4 \\ 1 & 7 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 2 \\ 10 \end{pmatrix},$$

or equivalently as $\mathbf{Ax} = \mathbf{b}$, where

$$\mathbf{A} = \begin{pmatrix} 2 & -4 \\ 1 & 7 \end{pmatrix}, \qquad \mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix}, \quad \text{and} \quad \mathbf{b} = \begin{pmatrix} 2 \\ 10 \end{pmatrix}.$$

The goal of this chapter is to examine how matrix equations like the one above can be solved using a computer. The centerpiece of this is something called the *LU factorization method.* To show how this works, we will factor the matrix $\mathbf{A}$ so that

$$\mathbf{A} = \mathbf{LU}, \tag{3.1}$$

where $\mathbf{L}$ is a lower triangular matrix and $\mathbf{U}$ is an upper triangular matrix. It is not obvious how to do this, and it is also not clear why you would want to do this in the first place. In the next sections we will derive the above result and discuss some of its benefits and limitations. The purpose here is to provide an early example of how this result is used.

The most computationally expensive part of the method is to find the factorization. For this example we will simply state the result, which is that

$$\begin{pmatrix} 2 & -4 \\ 1 & 7 \end{pmatrix} = \begin{pmatrix} 2 & 0 \\ 1 & 3 \end{pmatrix}\begin{pmatrix} 1 & -2 \\ 0 & 3 \end{pmatrix}. \tag{3.2}$$

Consequently,

$$\mathbf{L} = \begin{pmatrix} 2 & 0 \\ 1 & 3 \end{pmatrix} \quad \text{and} \quad \mathbf{U} = \begin{pmatrix} 1 & -2 \\ 0 & 3 \end{pmatrix}.$$

Solving $\mathbf{Ax} = \mathbf{b}$ comes by noticing that the equation can now be written as

$$\mathbf{L}(\mathbf{Ux}) = \mathbf{b}.$$

Looking at this for a moment you will see that if we set $\mathbf{y} = \mathbf{Ux}$, then the equation can be broken down into two equations:

1. First: find the vector $\mathbf{y}$ that is the solution of $\mathbf{Ly} = \mathbf{b}$

2. Second: find the vector $\mathbf{x}$ that is the solution of $\mathbf{Ux} = \mathbf{y}$

The two equations listed above are much easier to solve than the original.

To demonstrate this procedure, consider our earlier example. The $\mathbf{Ly} = \mathbf{b}$ equation is

$$\begin{pmatrix} 2 & 0 \\ 1 & 3 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} 2 \\ 10 \end{pmatrix},$$

where $\mathbf{y} = (u, v)^T$. In component form,

$$\begin{aligned} 2u &= 2 \\ u + 3v &= 10. \end{aligned}$$

The lower triangular form of this matrix means we simply start at the top and solve for the various components of $\mathbf{y}$ (which is sometimes called forward substitution). In particular, $u = 1$ and $v = 3$. The $\mathbf{Ux} = \mathbf{y}$ equation is now

$$\begin{pmatrix} 1 & -2 \\ 0 & 3 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 1 \\ 3 \end{pmatrix}$$

or, in component form

$$\begin{aligned} x - 2y &= 1 \\ 3y &= 3 \end{aligned}$$

The upper triangular form of this matrix means we simply start at the bottom and solve for the various components of $\mathbf{x}$ (which is sometimes called back substitution), and one finds that $y = 1$ and $x = 3$.

It is hard to overstate the importance of the LU method in numerical computing. At the same time, for those who are seeing it for the first time, it looks strange. Also, those who take linear algebra are taught to solve matrix equations using Gaussian elimination, and the LU method looks to be something completely different. Actually, as we will see in Section 3.3, it comes directly from Gaussian elimination.

A comment is in order about the transpose notation used earlier. In stating that $\mathbf{y} = (u, v)^T$ it is meant that

$$\mathbf{y} = \begin{pmatrix} u \\ v \end{pmatrix}.$$

It is perhaps more consistent to write $\mathbf{y} = (u \;\; v)^T$, but the comma is used to help clarify the separation between the components of the vector.

3.2 Finding L and U

Once you realize that an LU factorization might be possible, finding it involves a fairly straightforward calculation. Basically, you simply assume the matrix can be factored and then attempt to find $\mathbf{L}$ and $\mathbf{U}$. To illustrate, using the earlier example we will assume that it is possible to write $\mathbf{A} = \mathbf{LU}$, which means we assume that

$$\begin{pmatrix} 2 & -4 \\ 1 & 7 \end{pmatrix} = \begin{pmatrix} \ell_{11} & 0 \\ \ell_{21} & \ell_{22} \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{pmatrix}.$$

Multiplying this out we have

$$\begin{pmatrix} 2 & -4 \\ 1 & 7 \end{pmatrix} = \begin{pmatrix} \ell_{11}u_{11} & \ell_{11}u_{12} \\ \ell_{21}u_{11} & \ell_{21}u_{12} + \ell_{22}u_{22} \end{pmatrix},$$

or in component form

$$\begin{aligned} \ell_{11}u_{11} &= 2 \qquad & \ell_{11}u_{12} &= -4 \\ \ell_{21}u_{11} &= 1 \qquad & \ell_{21}u_{12} + \ell_{22}u_{22} &= 7. \end{aligned}$$

The first thing to notice is that we have 4 equations and 6 unknowns, and so the LU factorization is not unique. We should pick two values and it does not make much difference what choice is made, other than being nonzero. Two possibilities, that are used frequently enough that they are given names, are the following:

- *Doolittle factorization*: choose $\ell_{11} = \ell_{22} = 1$
- *Crout factorization*: choose $u_{11} = u_{22} = 1$.

To reproduce the factorization in (3.2) we pick $\ell_{11} = 2$ and $\ell_{22} = 3$. From this one finds that $u_{11} = 1$, $u_{12} = -2$, $\ell_{21} = 1$, and $u_{22} = 3$.

A few questions arise from the above calculation. First, since the LU factorization is not unique you might wonder if the solution is unique. In particular, you might expect that if you use a different factorization that you will get a different solution. The answer is no, the non-uniqueness of the factorization does not mean the solution is non-unique. As a simple test, you might try a Doolittle or Crout factorization in the above example to demonstrate that you still get the same solution. A second question is, can you factor any square matrix, and the answer to this is given next.

3.2.1 What Matrices Have an LU Factorization?

Another question which arises is, is it possible for there not to be an LU factorization? To answer this we will try to factor a general 2×2 matrix using a Doolittle factorization. So, we consider the equation

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \ell_{21} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{pmatrix},$$

which in component form is

$$\begin{aligned} u_{11} &= a & u_{12} &= b \\ \ell_{21}u_{11} &= c & \ell_{21}u_{12} + u_{22} &= d. \end{aligned}$$

Assuming, for the moment, that $a \neq 0$, then $u_{11} = a$, $u_{12} = b$, $\ell_{21} = c/a$, and $u_{22} = d - bc/a$. If $a = 0$, then the above equations can still be solved if $c = 0$. In this case, any value for ℓ_{21} works. For example, taking $\ell_{21} = 0$, then $u_{12} = b$ and $u_{22} = d$.

The remaining case to consider is $a = 0$ and $c \neq 0$. To explain what to do, note that the associated matrix equation is

$$\begin{pmatrix} 0 & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}.$$

The component form of the above equation is

$$\begin{aligned} by &= b_1 \\ cx + dy &= b_2. \end{aligned}$$

This can be rewritten as

$$\begin{aligned} cx + dy &= b_2 \\ by &= b_1. \end{aligned}$$

Note that interchanging rows in this way is known as *pivoting*. The resulting matrix equation is

$$\begin{pmatrix} c & d \\ 0 & b \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} b_2 \\ b_1 \end{pmatrix}.$$

Because c is nonzero, using the result in the previous paragraph, we know that the above coefficient matrix has an LU factorization.

We have shown that any linear system involving two equations can be rearranged so the coefficient matrix has an LU factorization. This result is true for all square matrices and this is stated in the next result.

Theorem 3.1. *Every linear $n \times n$ system can be rearranged, using pivoting, so the coefficient matrix has an LU factorization.*

It is also worth knowing which matrix equations can be solved without requiring pivoting. One can prove that pivoting is not necessary if the matrix is strictly diagonally dominant, or if it is symmetric and positive definite. These properties are defined in Section 3.7, while the proof of this statement, as well as the proof of the above theorem, can be found in Bjöurck [2015]. Certain tri-diagonal matrices are also known to be solvable without pivoting, and this is explained in Section 3.8. These non-pivoting cases come with a caveat concerning the floating point approximation of the matrix, and this is explained in Section 3.11.1. Finally, if you pick a random square matrix, using MATLAB's `rand` command, it is almost impossible to produce a matrix that requires pivoting. The reason is that pivoting is only needed when the entries of the matrix satisfy specific equations. As an example, for the 2×2 matrix considered above, pivoting is only required when $a = 0$. The probability of producing exactly this value for a using the `rand` command is very small.

3.2.2 Factoring $n \times n$ Matrices

The method used to find an LU factorization for a general square matrix is the same as for the 2×2 case. Namely, you just assume it's possible and then calculate what's necessary. In the case of a Doolittle factorization, this means we assume that

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ldots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ \ell_{21} & 1 & \cdots & 0 \\ \vdots & \vdots & \ldots & \vdots \\ \ell_{n1} & \ell_{n2} & \cdots & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \ldots & \vdots \\ 0 & 0 & \cdots & u_{nn} \end{pmatrix}.$$

It is necessary to multiply the two matrices on the right and then equate the result with the matrix on the left. As an example, equating the first column on the left with the one on the right, we have that

$$\begin{pmatrix} a_{11} \\ a_{21} \\ \vdots \\ a_{n1} \end{pmatrix} = \begin{pmatrix} u_{11} \\ \ell_{21}u_{11} \\ \vdots \\ \ell_{n1}u_{11} \end{pmatrix}.$$

From this it follows that $u_{11} = a_{11}$, $\ell_{21} = a_{21}/a_{11}$, $\cdots$, $\ell_{n1} = a_{n1}/a_{11}$. This requires that $a_{11} \neq 0$. If it is zero, and at least one of the other a_{i1}'s is nonzero, then we would first need to use pivoting to get a nonzero entry in the $(1,1)$-position. There is also the possibility that all the entries in the first column are zero, a situation equivalent in the 2×2 case of when $a = c = 0$, and it can be handled in a similar manner. However, this situation will not arise if the matrix is invertible.

Once this step is complete, one then equates the second column, then the third, etc. The resulting algorithm for finding the nonzero entries in **L** and **U**, when no pivoting is needed, is given in Table 3.1.

3.2.3 Pivoting Strategies

In the above derivation, to deal with the case of when $a_{11} = 0$, one looks for a row with a nonzero entry in the first column and then interchanges that row with the first row. This operation is known as pivoting. Also, even though it was not demonstrated explicitly in the above derivation, each column has

```
for  $j = 1, 2, \cdots, n$
    $l(j, j) = 1$
    for  $i = 1, 2, \cdots, j$
        $u(i, j) = a(i, j)$
        for $k = 1, 2, \cdots, i - 1$
            $u(i, j) = u(i, j) - l(i, k)u(k, j)$
        end
    end
    for $i = j + 1, j + 2, \cdots, n$
        $l(i, j) = a(i, j)$
        for $k = 1, 2, \cdots, j - 1$
            $l(i, j) = l(i, j) - l(i, k)u(k, j)$
        end
        $l(i, j) = l(i, j)/u(j, j)$
    end
end
```

Table 3.1 Algorithm for finding a Doolittle factorization of **A**, assuming pivoting is not needed. It is understood that any loop with a larger starting value than ending value is skipped.

the potential for a zero divisor and so pivoting might be necessary multiple times in the calculation.

There are numerous variations on how to pivot, some worth considering and some which are a waste of time. As an example of the former, one could have a situation where $a_{11} \neq 0$ but a_{11} is so close to zero that dividing by it can cause numerical difficulties. In this case, even though a_{11} is nonzero, pivoting is necessary. Another variation concerns which row to pivot with. As described above, one looks for the first row with a nonzero entry and then performs the interchange. Instead, one looks at all of the rows in the first column and picks the one with the largest entry, in absolute value. This is known as partial pivoting. It is also possible to pivot using the columns of the matrix, although this requires reordering the entries in the solution vector. For those interested in exploring the various pivoting strategies that have been proposed, Golub and Van Loan [2013] should be consulted.

3.3 LU and Gaussian Elimination

One of the conventional methods for solving a linear system is Gaussian elimination. To do this one forms the augmented matrix and then row reduces to find the answer. The general form of this procedure, for $\mathbf{Ax} = \mathbf{b}$, is

$$[\,\mathbf{A} \mid \mathbf{b}\,] \rightarrow [\,\mathbf{U} \mid \mathbf{y}\,] \rightarrow [\,\mathbf{I} \mid \mathbf{z}\,],$$

where $\mathbf{U}$ is an upper triangular matrix and $\mathbf{I}$ is the identity matrix. From this, the solution is $\mathbf{x} = \mathbf{z}$. As an example, consider solving

$$\begin{pmatrix} 1 & 1 & 1 \\ 2 & 4 & 1 \\ 5 & -1 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 2 \end{pmatrix}. \tag{3.3}$$

The steps in the reduction are given in Table 3.2. The row operation used in each step is given in the second column. For example, $-2R_1+R_2 \rightarrow R_2$ means the first row is multiplied by -2, added to the second row, and the result put into the second row. The third column expresses the result in matrix form, using elementary matrices. For example,

$$\mathbf{E}_1 = \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

and it is easy to verify that

$$\mathbf{E}_1\mathbf{A} = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 2 & -1 \\ 5 & -1 & -1 \end{pmatrix}.$$

The other two elementary matrices are

$$\mathbf{E}_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -5 & 0 & 1 \end{pmatrix} \quad \text{and} \quad \mathbf{E}_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 3 & 1 \end{pmatrix}.$$

The conclusion we make from this calculation is that

$$\mathbf{E}_3\mathbf{E}_2\mathbf{E}_1\mathbf{A} = \mathbf{U},$$

Augmented Form	Row Operation	Matrix Form
$\left[\begin{array}{rrr\|r} 1 & 1 & 1 & 1 \\ 2 & 4 & 1 & 0 \\ 5 & -1 & -1 & 2 \end{array}\right]$		$\mathbf{Ax} = \mathbf{b}$
	$-2R_1 + R_2 \to R_2$	
$\to \left[\begin{array}{rrr\|r} 1 & 1 & 1 & 1 \\ 0 & 2 & -1 & -2 \\ 5 & -1 & -1 & 2 \end{array}\right]$		$\mathbf{E}_1\mathbf{Ax} = \mathbf{E}_1\mathbf{b}$
	$-5R_1 + R_3 \to R_3$	
$\to \left[\begin{array}{rrr\|r} 1 & 1 & 1 & 1 \\ 0 & 2 & -1 & -2 \\ 0 & -6 & -6 & -3 \end{array}\right]$		$\mathbf{E}_2\mathbf{E}_1\mathbf{Ax} = \mathbf{E}_2\mathbf{E}_1\mathbf{b}$
	$3R_2 + R_3 \to R_3$	
$\to \left[\begin{array}{rrr\|r} 1 & 1 & 1 & 1 \\ 0 & 2 & -1 & -2 \\ 0 & 0 & -9 & -9 \end{array}\right]$		$\mathbf{E}_3\mathbf{E}_2\mathbf{E}_1\mathbf{Ax} = \mathbf{E}_3\mathbf{E}_2\mathbf{E}_1\mathbf{b}$

Table 3.2 Summary of the steps when using an augmented matrix to solve (3.3).

where $\mathbf{U}$ is the upper triangular matrix given in the augmented matrix in the last step in Table 3.2. From this we conclude that

$$\mathbf{A} = \mathbf{E}_1^{-1}\mathbf{E}_2^{-1}\mathbf{E}_3^{-1}\mathbf{U},$$

in other words, $\mathbf{A} = \mathbf{LU}$, where

$$\mathbf{L} = \mathbf{E}_1^{-1}\mathbf{E}_2^{-1}\mathbf{E}_3^{-1}. \tag{3.4}$$

To complete the derivation we need some useful results from matrix algebra:

- If $\mathbf{E}$ is invertible and lower triangular, then $\mathbf{E}^{-1}$ is lower triangular.
- If $\mathbf{L}_1$ and $\mathbf{L}_2$ are lower triangular, then $\mathbf{L}_1\mathbf{L}_2$ is lower triangular.

Therefore, the matrix $\mathbf{L}$ in (3.4) is lower triangular.

The above discussion involved a particular 3×3 matrix but the conclusion is the same for the general $n \times n$ case. Namely, by keeping track of the steps involved with Gaussian elimination one effectively constructs an LU factorization of the original matrix, and uses this to solve the equation. This also provides the motivation for looking for the LU factorization in the first place. However, once you know that you can factor the matrix in this way, the augmented matrix approach is not needed to find the factorization.

3.4 LU Method: Summary

In what follows it is assumed that $\mathbf{A}$ is an $n \times n$ non-singular matrix, and $\mathbf{x}$ and $\mathbf{b}$ are n-vectors. Also, $\mathbf{U}$ is an upper triangular matrix and $\mathbf{L}$ is a lower triangular matrix (and both are $n \times n$).

To solve $\mathbf{Ax} = \mathbf{b}$ using the LU method, the following steps are taken:

1. Calculate factorization: $\mathbf{A} = \mathbf{LU}$
2. Solve for $\mathbf{y}$: $\mathbf{Ly} = \mathbf{b}$
3. Solve for $\mathbf{x}$: $\mathbf{Ux} = \mathbf{y}$

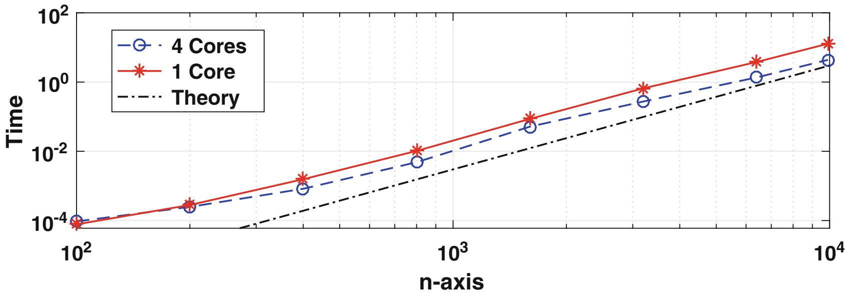


Figure 3.1 Computing time, in seconds, to find the LU factorization of an $n \times n$ matrix using MATLAB with a quad core, and with a single core, processor.

It is sometimes necessary to interchange rows to find the factorization (a process known as pivoting). Also, the factorization is not unique. This gives rise to certain choices for the diagonals in the factorization, and the two most commonly used are:

- *Doolittle factorization*: choose $\ell_{11} = \ell_{22} = \cdots = \ell_{nn} = 1$. In this case, $\mathbf{L}$ is said to be a *unit lower triangular matrix.*
- *Crout factorization*: choose $u_{11} = u_{22} = \cdots = u_{nn} = 1$. In this case, $\mathbf{U}$ is said to be a *unit upper triangular matrix.*

When solving an equation, only one of these is used when finding a factorization (i.e., you should not use both at the same time).

The number of flops (i.e., the additions, subtractions, multiplications, and divisions) needed to solve the problem is:

1. finding $\mathbf{L}$ and $\mathbf{U}$: $\frac{1}{6}n(n-1)(4n+1) = \frac{2}{3}n^3 - \frac{1}{2}n^2 + O(n)$
2. solving $\mathbf{Ly} = \mathbf{b}$ and $\mathbf{Ux} = \mathbf{y}$: $2n^2 + O(n)$

Therefore, for large matrices, a solution obtained using LU takes on the order of $\frac{2}{3}n^3$ flops. In comparison, solving the problem by first finding $\mathbf{A}^{-1}$ and then calculating $\mathbf{A}^{-1}\mathbf{b}$ takes on the order of $2n^3$ flops.

Note that the $\frac{2}{3}n^3$ flop count means that if the size of the matrix is doubled then the flops increase by a factor of about 8. To investigate this, the average computing times that MATLAB's LU command takes using a randomly generated $n \times n$ matrix are shown in Figure 3.1. The calculations were done first allowing MATLAB access to all four cores of the processor, and then again using only one core. The theory line in the plot is simply the curve $t = \frac{2}{3}n^3 t_0$, for a given value of t_0. What should happen, if the calculation follows the $\frac{2}{3}n^3$ rule, is that the curve for the computing time turns out to be parallel to the theory curve. This holds for the single core calculation, but not unexpectedly for the quad core curve. The computing times for the latter are, for the larger values of n, about a factor of 3 smaller than what is obtained for the single core calculation. The speedup for the multicore calculation is due to recent implementations of the LU factorization algorithm that take advantage of the hardware available, including memory hierarchy and multiple cores.

Note that for very large matrices, an important factor that can slow down the algorithm is the communication time. This is simply the time needed to move the data that is being used by the computer between memory locations, or between processors. With the current interest in solving physically realistic problems, and the very large matrices this can produce, research has been invested into how to redesign algorithms to reduce the communication time, perhaps at the expense of increasing the flops, to be able to reduce the overall computational time. Those interested in how this can affect the LU method should consult Ballard et al. [2011] or Yamazaki and Li [2012].

Example 1

Solve

$$\begin{aligned} x - y &= 2 \\ 3x + 2y &= 3 \end{aligned}$$

using an LU factorization. We will use a Doolittle factorization, and so we set

$$\begin{aligned} \begin{pmatrix} 1 & -1 \\ 3 & 2 \end{pmatrix} &= \begin{pmatrix} 1 & 0 \\ \ell_{21} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{pmatrix} \\ &= \begin{pmatrix} u_{11} & u_{12} \\ \ell_{21}u_{11} & \ell_{21}u_{12} + u_{22} \end{pmatrix}. \end{aligned}$$

From this it follows that $u_{11} = 1$, $u_{12} = -1$, $\ell_{21} = 3/u_{11} = 3$, and $u_{22} = 2 - \ell_{21}u_{12} = 5$. The next step is to solve $\mathbf{Ly} = \mathbf{b}$, which is

$$\begin{pmatrix} 1 & 0 \\ 3 & 1 \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} 2 \\ 3 \end{pmatrix}.$$

The solution is $u = 2$ and $v = -3$. Lastly, we solve $\mathbf{Ux} = \mathbf{y}$, which is

$$\begin{pmatrix} 1 & -1 \\ 0 & 5 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 2 \\ -3 \end{pmatrix}.$$

From this it follows that the solution of the problem is $y = -3/5$ and $x = 7/5$. ■

Example 2

Use the LU method to solve

$$\begin{pmatrix} 2 & 1 & 0 & 0 \\ 1 & 2 & 1 & 0 \\ 0 & 1 & 2 & 1 \\ 0 & 0 & 1 & 2 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 5 \end{pmatrix}.$$

The matrix is *tri-diagonal*, which means that only nonzero entries are on the diagonal and on the super- and sub-diagonals. It is a type of matrix that is very common in applications. Normally, finding the LU factorization of a 4×4 matrix by hand is a bit tedious, but for a tri-diagonal matrix it isn't so bad. This is because $\mathbf{L}$ and $\mathbf{U}$ are also tri-diagonal. In particular, using a Doolittle factorization, we assume that

$$\begin{aligned}\begin{pmatrix} 2 & 1 & 0 & 0 \\ 1 & 2 & 1 & 0 \\ 0 & 1 & 2 & 1 \\ 0 & 0 & 1 & 2 \end{pmatrix} &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ \ell_{21} & 1 & 0 & 0 \\ 0 & \ell_{32} & 1 & 0 \\ 0 & 0 & \ell_{43} & 1 \end{pmatrix}\begin{pmatrix} u_{11} & u_{12} & 0 & 0 \\ 0 & u_{22} & u_{23} & 0 \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{pmatrix} \\ &= \begin{pmatrix} u_{11} & u_{12} & 0 & 0 \\ \ell_{21}u_{11} & \ell_{21}u_{12} + u_{22} & u_{23} & 0 \\ 0 & \ell_{32}u_{22} & \ell_{32}u_{23} + u_{33} & u_{34} \\ 0 & 0 & \ell_{43}u_{33} & \ell_{43}u_{34} + u_{44} \end{pmatrix}.\end{aligned}$$

Starting with the first row, we conclude that $u_{11} = 2$ and $u_{12} = 1$. Dropping to the second row one finds that $\ell_{11} = 1/u_{11} = 1/2$. Finishing the second row and then continuing one finds that

$$\mathbf{U} = \begin{pmatrix} u_{11} & u_{12} & 0 & 0 \\ 0 & u_{22} & u_{23} & 0 \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{pmatrix} = \begin{pmatrix} 2 & 1 & 0 & 0 \\ 0 & \frac{3}{2} & 1 & 0 \\ 0 & 0 & \frac{4}{3} & 1 \\ 0 & 0 & 0 & \frac{5}{4} \end{pmatrix},$$

and

$$\mathbf{L} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ \ell_{21} & 1 & 0 & 0 \\ 0 & \ell_{32} & 1 & 0 \\ 0 & 0 & \ell_{43} & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ \frac{1}{2} & 1 & 0 & 0 \\ 0 & \frac{2}{3} & 1 & 0 \\ 0 & 0 & \frac{3}{4} & 1 \end{pmatrix}.$$

The next step is to solve $\mathbf{Ly} = \mathbf{b}$, which is

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ \frac{1}{2} & 1 & 0 & 0 \\ 0 & \frac{2}{3} & 1 & 0 \\ 0 & 0 & \frac{3}{4} & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 5 \end{pmatrix}.$$

One finds that $y_1 = y_2 = y_3 = 0$ and $y_4 = 5$. It remains to solve $\mathbf{Ux} = \mathbf{y}$, which is

$$\begin{pmatrix} 2 & 1 & 0 & 0 \\ 0 & \frac{3}{2} & 1 & 0 \\ 0 & 0 & \frac{4}{3} & 1 \\ 0 & 0 & 0 & \frac{5}{4} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 5 \end{pmatrix}.$$

The solution in this case is $x_4 = 4$, $x_3 = -3$, $x_2 = 2$, and $x_1 = -1$. ■

Example 3

We will now consider the $n \times n$ version of the above example, and the equation is

$$\begin{pmatrix} 2 & 1 & & & & \\ 1 & 2 & 1 & & \mathbf{0} & \\ & 1 & 2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & \mathbf{0} & & & & 1 \\ & & & & 1 & 2 \end{pmatrix} \mathbf{x} = \mathbf{b}, \tag{3.5}$$

where the vector $\mathbf{b}$ is selected so that $\mathbf{x} = (1, 1, \cdots, 1)^T$ is the exact solution. The matrix equation is going to be solved using the LU method, specifically a version that is specialized to tri-diagonal matrices as described in Section 3.8. The question considered here is how accurately we are able to compute the solution using double precision arithmetic. In other words, we are interested in how the exact solution $\mathbf{x}$ compares to the computed solution $\mathbf{x}_c$. To determine this we will compute the largest number, in absolute value, in the vector $\mathbf{x} - \mathbf{x}_c$. This number will be denoted as $||\mathbf{x} - \mathbf{x}_c||_\infty$. The results of the calculation are given in Table 3.3. For the smaller values of n the solution is as accurate as can be expected using double precision. For the larger values of n there is some loss of accuracy but the answer is still reasonably accurate. Note, for those who might be interested in just how long it takes a computer to solve a 160000×160000 matrix equation using the LU method, it is about 10^{-2} sec. However, the method takes maximum advantage of the tri-diagonal structure of this matrix. ■

n	$\|\|\mathbf{x}-\mathbf{x}_c\|\|_\infty$
4	6.66e−16
8	1.33e−15
12	1.78e−15
16	1.33e−15
160	6.26e−14
1600	1.35e−12
16000	4.97e−11
160000	3.01e−09

Table 3.3 Difference between the exact and computed solution in Example 3. Note that 6.66e−16 = 6.66×10^{-16}.

Example 4

The equation to be solved is

$$\begin{pmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & 1/2 & (1/2)^2 & \dots & (1/2)^{n-1} \\ 1 & 1/3 & (1/3)^2 & \dots & (1/3)^{n-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & 1/n & (1/n)^2 & \dots & (1/n)^{n-1} \end{pmatrix} \mathbf{x} = \mathbf{b}.$$

The vector $\mathbf{b}$ is selected so that $\mathbf{x} = (1, 1, \cdots, 1)^T$ is the exact solution. Note that the matrix in this example is known as the *Vandermonde matrix*, and we will see it again in Chapters 5 and 8. As in Example 3, to compare the exact solution $\mathbf{x}$ to the computed solution $\mathbf{x}_c$ we will compute the largest number, in absolute value, in the vector $\mathbf{x} - \mathbf{x}_c$. Also, as before, this number will be denoted as $||\mathbf{x} - \mathbf{x}_c||_\infty$. The results of the calculation are given in Table 3.4. The results are dramatically different than what were obtained in Example 3. Namely, although the accuracy is reasonable when $n = 4$, when $n = 16$ it is horrible, and it is not even defined when $n = 160$. ■

n	$\|\|\mathbf{x}-\mathbf{x}_c\|\|_\infty$
4	2.00e−15
8	4.23e−09
12	5.28e−03
16	25.6
160	NaN

Table 3.4 Difference between the exact and computed solution in Example 4.

What the last two examples show is that for some matrices the LU method works just great, but for others it is terrible. The reason is that solving matrix equations can be very sensitive to round-off error, whether one uses LU or some other method. What is required is to invest some effort in analyzing the error and how it depends on the properties of the matrix.

3.5 Vector and Matrix Norms

We need to be able to measure the size of vectors and matrices, and this will be done using the concept of a vector norm. Given $\mathbf{x} \in \mathbb{R}^n$, its norm is designated as $||\mathbf{x}||$. To qualify to be a norm, it must have the following three properties:

1. $||\alpha\mathbf{x}|| = |\alpha|\, ||\mathbf{x}||$
2. $||\mathbf{x}+\mathbf{y}|| \leq ||\mathbf{x}|| + ||\mathbf{y}||$
3. If $||\mathbf{x}|| = 0$, then $\mathbf{x} = \mathbf{0}$

It is required that these hold for all n-vectors $\mathbf{x}$ and $\mathbf{y}$, and numbers (scalars) α. It should be pointed out that the definition for a norm for a general vector space has additional requirements. For $\mathbf{x} \in \mathbb{R}^n$ these additional conditions are satisfied automatically, and this is considered in Exercise 3.13.

An example is the usual Euclidean definition of length, which is defined as

$$||\mathbf{x}||_2 \equiv \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}\,. \tag{3.6}$$

This is known as the *Euclidean norm* or the *2-norm*. Another norm we will often use is

$$||\mathbf{x}||_\infty \equiv \max\{\, |x_1|, |x_2|, \cdots, |x_n|\,\}\,, \tag{3.7}$$

which is known as the ∞-*norm*. A third norm that is used in scientific computing is

$$||\mathbf{x}||_1 \equiv |x_1| + |x_2| + \cdots + |x_n|\,, \tag{3.8}$$

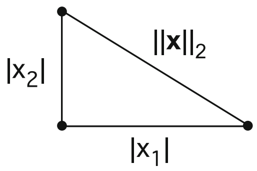


Figure 3.2 Components of the three vector norms in $\mathbb{R}^2$.

and this is known as the *1-norm*. Note that norms are dimensionally consistent with the vectors considered. For example, if $\mathbf{x}$ is a position vector, so the entries have the dimension of length, then the norm of $\mathbf{x}$ has the dimension of length.

Examples

1. If $\mathbf{x} = (2\,,\, -1)^T$, then $||\mathbf{x}||_2 = \sqrt{5}$, $||\mathbf{x}||_\infty = 2$, and $||\mathbf{x}||_1 = 3$. ■

2. If $\mathbf{x} \in \mathbb{R}^n$ with $\mathbf{x} = (1\,,\,1\,,\,1\,,\,\cdots\,,\,1)^T$, then $||\mathbf{x}||_2 = \sqrt{n}$, $||\mathbf{x}||_\infty = 1$, and $||\mathbf{x}||_1 = n$. ■

3. To show that $||\mathbf{x}||_\infty$ is a vector norm, note that

$$\begin{aligned} ||\alpha\mathbf{x}||_\infty &= \max\{\,|\alpha x_1|, |\alpha x_2|, \cdots, |\alpha x_n|\,\} \\ &= \max\{\,|\alpha|\,|x_1|, |\alpha|\,|x_2|, \cdots, |\alpha|\,|x_n|\,\} \\ &= |\alpha|\,\max\{\,|x_1|, |x_2|, \cdots, |x_n|\,\} = |\alpha|\,||\mathbf{x}||_\infty\,. \end{aligned}$$

Also, let j be the value where $||\mathbf{x}+\mathbf{y}||_\infty = |x_j + y_j|$. Because $|x_j + y_j| \le |x_j| + |y_j| \le ||\mathbf{x}||_\infty + ||\mathbf{y}||_\infty$, it follows that $||\mathbf{x}+\mathbf{y}||_\infty \le ||\mathbf{x}||_\infty + ||\mathbf{y}||_\infty$. Finally, it is clear that if $||\mathbf{x}||_\infty = 0$ then $\mathbf{x} = \mathbf{0}$. ■

If you are wondering how the norms compare, the answer can be inferred from the previous examples. Another way, for $\mathbf{x} \in \mathbb{R}^2$, can be obtained from Figure 3.2 using the usual relationship between the lengths of the sides of a right triangle. Namely, $\max\{\,|x_1|, |x_2|\,\} \le \sqrt{x_1^2 + x_2^2} \le |x_1| + |x_2|$. In other words,

$$||\mathbf{x}||_\infty \le ||\mathbf{x}||_2 \le ||\mathbf{x}||_1\,.$$

As demonstrated in the above example, for large vectors (big n), the three vector norms can produce significantly different values.

A natural question to ask is, why consider different vector norms, what's so bad for the old standard, which is the Euclidean norm given in (3.6)? In numerical computing, vector norms are often used to determine when to stop a calculation. To explain, many numerical methods produce a sequence of approximations $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$, $\cdots$ that converge to the desired solution. The usual way it is decided when to stop computing is through a requirement of the form $||\mathbf{x}_{m+1} - \mathbf{x}_m|| \leq tol$, where tol is a given error tolerance. One should pick a vector norm that best achieves this goal. For example, using the ∞-norm one is requiring that every element of $\mathbf{x}_{m+1} - \mathbf{x}_m$ is smaller than tol (in absolute value). This is a reasonable requirement, and does not take long to compute. The other norms are certainly used, and as an example the 1-norm is a natural way to measure error when approximating functions (see Exercise 6.25).

3.5.1 Matrix Norms

Each of the vector norms can be used to define a norm of a matrix. This is done by comparing the size of $\mathbf{Ax}$ to the size of $\mathbf{x}$. The definition is

$$||\mathbf{A}|| \equiv \max_{\mathbf{x}\neq\mathbf{0}} \frac{||\mathbf{Ax}||}{||\mathbf{x}||} . \tag{3.9}$$

Because matrices have the property that $\mathbf{A}(\alpha\mathbf{x}) = \alpha\mathbf{Ax}$, and norms have the property that $||\alpha\mathbf{x}|| = |\alpha| \cdot ||\mathbf{x}||$, this definition can be rewritten as

$$||\mathbf{A}|| = \max_{||\mathbf{x}||=1} ||\mathbf{Ax}|| . \tag{3.10}$$

Using the above formulas, the vector norm $||\mathbf{x}||_\infty$ gives rise to the matrix norm $||\mathbf{A}||_\infty$, and similarly for the other vector norms we have considered.

The definition in (3.9), or the version given in (3.10), is useful for the more theoretical aspects of the subject, but they are not particularly useful for calculating a matrix norm. Fortunately, it is possible to derive easier to use formulas for $||\mathbf{A}||_\infty$ and $||\mathbf{A}||_1$. In particular, one can show that the ∞-norm of a matrix reduces to

$$\begin{aligned} ||\mathbf{A}||_\infty &= \max \left\{ \sum_{j=1}^{n} |a_{1j}|, \ \sum_{j=1}^{n} |a_{2j}|, \cdots , \ \sum_{j=1}^{n} |a_{nj}| \right\} \\ &= \max_{1\leq i\leq n} \sum_{j=1}^{n} |a_{ij}| . \end{aligned} \tag{3.11}$$

In other words, the ∞-norm is determined by the largest row sum (of absolute values). In contrast, the 1-norm of a matrix is

$$\begin{aligned} ||\mathbf{A}||_1 &= \max\left\{\sum_{i=1}^{n}|a_{i1}|,\ \sum_{i=1}^{n}|a_{i2}|,\cdots,\ \sum_{i=1}^{n}|a_{in}|\right\} \\ &= \max_{1\le j\le n}\sum_{i=1}^{n}|a_{ij}|\,, \end{aligned} \tag{3.12}$$

which means that the 1-norm is determined by the largest column sum (of absolute values). One way to remember these two formulas is that ∞ is horizontal (rows), while 1 is vertical (columns). Also, unfortunately, there is no tidy little formula for the 2-norm of a matrix that is easy to calculate for large matrices. It is possible to connect the 2-norm with what are called singular values of the matrix, and this is considered in Section 4.5.3.

Example

If

$$\mathbf{A} = \begin{pmatrix} 1 & 2 & -3 \\ 4 & -5 & 6 \\ -7 & 8 & 9 \end{pmatrix},$$

then

$$\begin{aligned} ||\mathbf{A}||_\infty &= \max\{1+2+3\,,\,4+5+6\,,\,7+8+9\} \\ &= \max\{6,15,24\} = 24, \end{aligned}$$

and

$$\begin{aligned} ||\mathbf{A}||_1 &= \max\{1+4+7\,,\,2+5+8\,,\,3+6+9\} \\ &= \max\{12,15,18\} = 18. \quad \blacksquare \end{aligned}$$

As a final comment, matrix norms that are derived from a vector norm, which is the case in (3.9), are called *natural matrix norms.* There are matrix norms that are not derivable from a vector norm, what might be called unnatural norms, but they are not needed in what follows.

Basic Properties of a Matrix Norm

1. If $\mathbf{I}$ is the identity matrix, then $||\mathbf{I}|| = 1$.
2. $||\mathbf{Ax}|| \le ||\mathbf{A}|| \cdot ||\mathbf{x}||$
3. $||\mathbf{AB}|| \le ||\mathbf{A}|| \cdot ||\mathbf{B}||$

Proof: The first follows directly from (3.9). The second holds if $\mathbf{x} = \mathbf{0}$, and when $\mathbf{x} \neq \mathbf{0}$, the inequality follows from (3.9). As for the third, given that

$(\mathbf{AB})\mathbf{x} = \mathbf{A}(\mathbf{Bx})$, then using Property 2 (twice), $||(\mathbf{AB})\mathbf{x}|| \leq ||\mathbf{A}|| \cdot ||\mathbf{Bx}|| \leq ||\mathbf{A}|| \cdot ||\mathbf{B}|| \cdot ||\mathbf{x}||$. Assuming $\mathbf{x} \neq \mathbf{0}$, and rewriting the last inequality as,

$$\frac{||(\mathbf{AB})\mathbf{x}||}{||\mathbf{x}||} \leq ||\mathbf{A}|| \cdot ||\mathbf{B}|| \, ,$$

then the result follows from (3.9). □

3.6 Error and Residual

Letting $\mathbf{x}$ be the exact solution and $\mathbf{x}_c$ the computed solution:

- The *error vector* $\mathbf{e}$ is defined as

$$\mathbf{e} \equiv \mathbf{x} - \mathbf{x}_c$$

- The *residual vector* $\mathbf{r}$ is defined as

$$\mathbf{r} \equiv \mathbf{b} - \mathbf{A}\mathbf{x}_c$$

Although having zero error is desired, the reality is that when using floating point numbers the best we can expect are relative errors on the order of machine ε. Another complication is that for most problems we don't know $\mathbf{x}$ and are therefore not able to calculate $\mathbf{e}$. The residual, however, is something we can calculate. Everything that follows is based on the goal of using the residual to determine, or estimate, the error in the calculated solution. The first step is to realize that these two vectors are related through the formula

$$\mathbf{r} = \mathbf{A}\mathbf{e},$$

or equivalently

$$\mathbf{e} = \mathbf{A}^{-1}\mathbf{r}.$$

We would like to be able to state that if $\mathbf{r}$ is small then so is $\mathbf{e}$. However, as the above formula shows, it might happen that the multiplication by $\mathbf{A}^{-1}$ takes a small $\mathbf{r}$ and produces a large $\mathbf{e}$. How to relate these two vectors is given next.

Theorem 3.2. *Assuming* $\mathbf{A}$ *is non-singular and* $\mathbf{b}$ *is nonzero, then*

$$\frac{||\mathbf{e}||}{||\mathbf{x}||} \leq \kappa(\mathbf{A})\frac{||\mathbf{r}||}{||\mathbf{b}||} \, , \tag{3.13}$$

where $\kappa(\mathbf{A})$ *is the condition number of* $\mathbf{A}$ *and is defined as*

$$\kappa(\mathbf{A}) \equiv ||\mathbf{A}|| \cdot ||\mathbf{A}^{-1}|| \, .$$

Proof: First note that since $\mathbf{b}$ is nonzero, then $\mathbf{x}$ is nonzero. The conclusion of the theorem is a consequence of two inequalities. First, since $\mathbf{Ax} = \mathbf{b}$ then $||\mathbf{b}|| = ||\mathbf{Ax}|| \leq ||\mathbf{A}|| \cdot ||\mathbf{x}||$. From this we have the first inequality,

$$\frac{1}{||\mathbf{x}||} \leq \frac{||\mathbf{A}||}{||\mathbf{b}||} .$$

As for the second inequality, since $\mathbf{e} = \mathbf{A}^{-1}\mathbf{r}$, then $||\mathbf{e}|| \leq ||\mathbf{A}^{-1}|| \cdot ||\mathbf{r}||$. The theorem follows by simply combining the two inequalities. □

Note that

$$\frac{||\mathbf{e}||}{||\mathbf{x}||} \tag{3.14}$$

is the error relative to the value of the solution, while

$$\frac{||\mathbf{r}||}{||\mathbf{b}||}$$

is the residual relative to the right-hand side of the matrix equation. So, the above theorem is useful because it states that if the relative residual is small then the relative error is small. The requirement needed to make this conclusion is that the condition number is not very big. Matrices with large condition numbers are said to be *ill-conditioned.* Because condition numbers are one or greater (this is proved below), the requirement that $\mathbf{A}$ is ill-conditioned can be written as $1 \ll \kappa(\mathbf{A})$. Just how much bigger than one depends on the precision of the floating point system used and this will be explained in Section 3.6.3.

3.6.1 Significant Digits

It was explained in Section 1.5 how the relative error can be used to determine (approximately) the number of correct significant digits of a computed scalar quantity. For vectors, the relative error as given in (3.14) does not have such a straightforward connection with correct digits. To illustrate, suppose the exact value is $\mathbf{x} = (100, 1, 0)^T$ and the computed value is $\mathbf{x}_c = (100.1, 1.1, 0.1)^T$. Using the ∞-norm, the relative error is

$$\frac{||\mathbf{x} - \mathbf{x}_c||_\infty}{||\mathbf{x}||_\infty} = 10^{-3}.$$

While it is true that the first entry in $\mathbf{x}_c$ is correct to three digits, the second entry is correct to only one digit, and the concept of significant digits is not even applicable to the third entry. So, when dealing with a nonzero vector

$\mathbf{x}$, the connection of the relative error with the number of correct significant digits is guaranteed to only apply to the largest entry, in absolute value, of $\mathbf{x}$. This is also assuming that the ∞-norm is used.

3.6.2 The Condition Number

As illustrated in the above theorem, the condition number plays a central role in determining how accurately the matrix equation can be solved. We are using κ to designate this number but another common notation is cond($\mathbf{A}$). Also, some like to indicate which norm they are using and, as an example, will use $\kappa_\infty(\mathbf{A})$ or $\text{cond}_\infty(\mathbf{A})$ if they are using the ∞-norm. This will be done here as well. Specifically, if the formula applies for any norm, then there will be no subscript, while if the result depends on the norm used then a subscript will be employed.

Example 1

If

$$\mathbf{A} = \begin{pmatrix} a & b \\ c & d \end{pmatrix},$$

then

$$\mathbf{A}^{-1} = \frac{1}{ad-bc}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}. \tag{3.15}$$

This assumes, of course, that $ad - bc \neq 0$. With this

$$||\mathbf{A}||_\infty = \max\{\,|a|+|b|, |c|+|d|\,\},$$

and

$$||\mathbf{A}^{-1}||_\infty = \frac{1}{|ad-bc|}\max\{\,|d|+|b|, |c|+|a|\,\}.$$

So, for example, if

$$\mathbf{A} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix},$$

then $||\mathbf{A}||_\infty = 7$ and $||\mathbf{A}^{-1}||_\infty = 3$. Consequently, $\kappa_\infty(\mathbf{A}) = 21$. ■

Example 2

Suppose

$$\mathbf{A} = \begin{pmatrix} 1 & 2 \\ -1 & d \end{pmatrix}.$$

What is shown in Figure 3.3 is what the matrix $\mathbf{A}$ does to the circle $x^2+y^2=1$ for various choices for d. To explain, given a point $\mathbf{x}$ on the circle, then $\mathbf{Ax}$ is a point on the respective ellipse. The specific values are

a) $d=1$: In this case $\kappa_\infty(\mathbf{A})=3$
b) $d=5$: In this case $\kappa_\infty(\mathbf{A})=6$
c) $d=20$: In this case $\kappa_\infty(\mathbf{A})=21$

The important observation here is that for smaller values of the condition number, $\mathbf{A}$ does not distort the circle very much. However, for larger values the distortion becomes significant. As we will see below, it is easy to find matrices with very large condition numbers, such as 10^{10} and 10^{20}. The

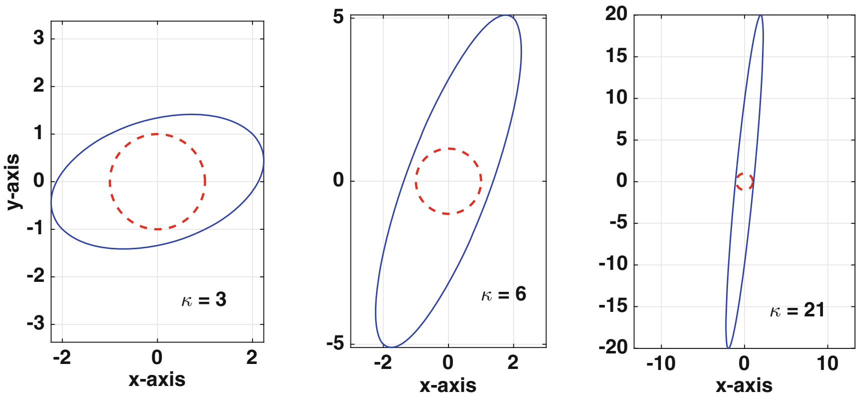


Figure 3.3 The dashed curve is the circle $x^2+y^2=1$, and the solid curve is what the matrix $\mathbf{A}$ from Example 2 transforms the circle into. The condition number used here is κ_∞.

resulting ellipse in such a situation is so distorted that any plot of the ellipse would look like a straight line. Why this is relevant to solving $\mathbf{Ax}=\mathbf{b}$ is discussed in Section 3.11.3. ■

Example 3

Suppose $\mathbf{A}$ is a diagonal matrix 3×3, which means it can be written as

$$\mathbf{A}=\begin{pmatrix} d_1 & 0 & 0 \\ 0 & d_2 & 0 \\ 0 & 0 & d_3 \end{pmatrix}.$$

In this case, $||\mathbf{A}||_\infty = \max\{\,|d_1|, |d_2|, |d_3|\,\}$. Also, assuming the diagonals are nonzero, then

$$\mathbf{A}^{-1} = \begin{pmatrix} 1/d_1 & 0 & 0 \\ 0 & 1/d_2 & 0 \\ 0 & 0 & 1/d_3 \end{pmatrix}.$$

From this it follows that

$$||\mathbf{A}^{-1}||_\infty = \max\{\,|1/d_1|, |1/d_2|, |1/d_3|\,\} = \frac{1}{\min\{\,|d_1|, |d_2|, |d_3|\,\}}.$$

Therefore,

$$\kappa_\infty(\mathbf{A}) = \frac{\max\{\,|d_1|, |d_2|, |d_3|\,\}}{\min\{\,|d_1|, |d_2|, |d_3|\,\}}.$$

This shows that for this matrix the condition number is not affected so much by how large or small the d_i's are but rather how different they are. For example, if $d_1 = d_2 = d_3 = 10^{-10}$ or if $d_1 = d_2 = d_3 = 10^{10}$, then $\kappa_\infty(\mathbf{A}) = 1$ and we have a well-conditioned matrix. However, if $d_1 = d_2 = 10^{-10}$ and $d_3 = 10^{10}$, then $\kappa_\infty(\mathbf{A}) = 10^{20}$ and we have an ill-conditioned matrix. ■

The condition number has several useful properties. Some of them are listed below, and they hold for any matrix norm.

Basic Properties of the Condition Number

Assuming that $\mathbf{A}$ and $\mathbf{P}$ are invertible, then the following hold.

1. $\kappa(\mathbf{I}) = 1$, where $\mathbf{I}$ is the identity matrix
2. $1 \le \kappa(\mathbf{A}) < \infty$
3. For any nonzero number α, $\kappa(\alpha\mathbf{A}) = \kappa(\mathbf{A})$
4. $\kappa(\mathbf{A}) = \kappa(\mathbf{A}^{-1})$
5. $\kappa(\mathbf{PA}) \le \kappa(\mathbf{P})\kappa(\mathbf{A})$

Proof: Note that Property 1 holds because $\mathbf{I}^{-1} = \mathbf{I}$ and $||\mathbf{I}|| = 1$. As for Property 2, since $\mathbf{AA}^{-1} = \mathbf{I}$, it follows that $||\mathbf{I}|| \le ||\mathbf{A}|| \cdot ||\mathbf{A}^{-1}||$. Property 3 holds because $||\alpha\mathbf{A}|| = |\alpha| \cdot ||\mathbf{A}||$ and $||(\alpha\mathbf{A})^{-1}|| = |\alpha|^{-1} \cdot ||\mathbf{A}^{-1}||$, and Property 4 is an immediate consequence of the definition of the condition number. Finally, for Property 5, given that $||\mathbf{PA}|| \cdot ||(\mathbf{PA})^{-1}|| = ||\mathbf{PA}|| \cdot ||\mathbf{A}^{-1}\mathbf{P}^{-1}||$, it follows that $||\mathbf{PA}|| \cdot ||(\mathbf{PA})^{-1}|| \le ||\mathbf{P}|| \cdot ||\mathbf{A}|| \cdot ||\mathbf{A}^{-1}|| \cdot ||\mathbf{P}^{-1}||$. □

3.6.3 A Heuristic

A rule of thumb has been developed that relates the condition number to the accuracy of the computed solution [Golub and Van Loan, 2013]. The heuristic is that

$$\frac{||\mathbf{x}-\mathbf{x}_c||}{||\mathbf{x}||} \approx \varepsilon\kappa(\mathbf{A}). \tag{3.16}$$

In other words, if $\varepsilon \approx 10^{-p}$ and $\kappa(\mathbf{A}) \approx 10^q$, then $\mathbf{x}_c$ is probably correct to no more than about $p-q$ digits. As an example, when using double precision, this difference is $16-q$. Note that when this difference is negative then the

n	$\lvert\lvert\mathbf{x}-\mathbf{x}_c\rvert\rvert_\infty$	$\kappa_\infty(\mathbf{A})$	$\varepsilon\kappa_\infty(\mathbf{A})$	$\lvert\lvert\mathbf{x}-\mathbf{x}_c\rvert\rvert_\infty$	$\kappa_\infty(\mathbf{A})$	$\varepsilon\kappa_\infty(\mathbf{A})$
4	6.66e−16	12	2.7e−15	2.00e−15	1.5e+03	3.4e−13
8	1.33e−15	40	8.9e−15	4.23e−09	4.5e+08	1.0e−07
12	1.78e−15	84	1.9e−14	5.28e−03	1.1e+15	2.3e−01
16	1.33e−15	144	3.2e−14	25.6	1.6e+18	3.5e+02
160	6.26e−14	1.3e+04	2.9e−12	*NaN*	*Inf*	*Inf*
1600	1.35e−12	1.3e+06	2.9e−10			
16000	4.97e−11					

Table 3.5 Error when computing solution in Example 3, on left, and in Example 4, on right, from Section 3.4. Because $||\mathbf{x}||_\infty = 1$, according to the heuristic, $\varepsilon\kappa_\infty(\mathbf{A})$ is an estimate of the error.

heuristic and probably the computed solution have no meaning. Also, as explained in Section 3.6.1, to make the connection with significant digits, it is appropriate to use the ∞-norm when using the heuristic.

The explanation of how (3.16) is arrived at involves a worst-case analysis applied to (3.13). Namely, even though you might solve the equation to the accuracy allowed using float-point arithmetic, so the relative error in the residual is on the order of machine ε, the relative error in the solution is as bad as permitted in (3.13).

Example

The table from Example 3 in Section 3.4, which was computed for a tri-diagonal matrix, is repeated in Table 3.5 (left side) but two columns are added, one giving the condition number and the other giving the value of $\varepsilon\kappa_\infty(\mathbf{A})$. Note the ∞-norm is used here, in which case $||\mathbf{x}||_\infty = 1$. Similarly, the table from Example 4, which was computed for the Vandermonde matrix, is repeated in Table 3.5 (right side). These results show that the heuristic is a bit pessimistic in the sense that the actual error is better than what is predicted by the heuristic. It is also clear that the Vandermonde matrix is ill-conditioned except for very small values of n. The tri-diagonal matrix in contrast is reasonably well-conditioned. ■

Note that the heuristic was not computed in the last row in Table 3.5. Although it is possible to let the computer run for possibly hours and eventually compute this number, this was not done to make a point. Computing the condition number for a large matrix is very time consuming. However, the information one can derive from knowing the condition number is important enough that considerable research has been invested into how to obtain an estimate of it relatively quickly, even for large matrices. An introduction to the various ways this can be done can be found in Higham [2002] and Golub and Van Loan [2013].

3.7 Positive Definite Matrices

One of the more common numerical problems that arises in continuum mechanics or electrodynamics involves solving equations with matrices such as

$$\begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{pmatrix} \quad \text{or} \quad \begin{pmatrix} 4 & -1 & 0 & -1 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & -1 & 4 & -1 & 0 & -1 \\ -1 & 0 & -1 & 4 & -1 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & -1 & 0 & -1 & 4 \end{pmatrix}.$$

These matrices have several properties that have a significant impact on the numerical methods that can be used. The two that are of interest here are that they are symmetric and positive definite. For those who are unfamiliar with the latter property, its definition is given next.

Definition 3.1. If $\mathbf{A}$ is an $n \times n$ symmetric matrix, then $\mathbf{A}$ is *positive definite* if either of the following holds:

1. $\mathbf{x}^T\mathbf{A}\mathbf{x} > 0, \ \forall\, \mathbf{x} \neq \mathbf{0}$, or
2. $\mathbf{A}$ has only positive eigenvalues.

To put this definition on solid ground, it is necessary to prove that any symmetric matrix that satisfies the first condition also satisfies the second condition (and vice versa). This is easy to do, and to illustrate, given an eigenvalue λ, and a corresponding eigenvector $\mathbf{x}$, then $\mathbf{x}$ is nonzero and $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$. With this, $\mathbf{x}^T\mathbf{A}\mathbf{x} = \lambda\mathbf{x} \cdot \mathbf{x}$. Since $\mathbf{x} \cdot \mathbf{x} > 0$, and if it is true that $\mathbf{x}^T\mathbf{A}\mathbf{x} > 0$, it then follows that $\lambda > 0$. The proof of the other direction requires a result from linear algebra about the eigenvalues for a symmetric matrix (see Theorem 4.1) and is left as an exercise.

The fact is that the above definition is not particularly useful even though the idea being defined is very important. It turns out that there are some easy to use tests for determining whether or not a matrix is positive definite. We begin with the negative results, namely ways to determine if a matrix does not have this property.

Theorem 3.3. *Assume* $\mathbf{A}$ *is a symmetric* $n \times n$ *matrix.*

1. $\mathbf{A}$ *is not positive definite if any diagonal entry is negative or zero.*
2. $\mathbf{A}$ *is not positive definite if the largest number in* $\mathbf{A}$*, in absolute value, is off the diagonal.*
3. $\mathbf{A}$ *is not positive definite if det*$(\mathbf{A}) \leq 0$.

Proof: In regard to the first statement, suppose $a_{ii} \leq 0$. Taking $\mathbf{x} = \mathbf{e}_i$, where $\mathbf{e}_i$ is the ith coordinate vector, then $\mathbf{x}^T\mathbf{A}\mathbf{x} = a_{ii}$. The latter number is not positive and so $\mathbf{A}$ is not positive definite. The second statement is proved in a similar manner, but using $\mathbf{x} = \mathbf{e}_i - \mathbf{e}_j$ and $\mathbf{x} = \mathbf{e}_i + \mathbf{e}_j$. The third statement follows from the result from linear algebra which states that if $\lambda_1, \lambda_2, \cdots, \lambda_n$ are the eigenvalues of $\mathbf{A}$ then $\det(\mathbf{A}) = \lambda_1\lambda_2\cdots\lambda_n$. Since the eigenvalues of a symmetric positive definite matrix are positive, then the product of the eigenvalues must be positive. $\square$

The first two conditions are easy to use even on very large matrices, while the usefulness of the third condition is limited to smaller or very simple matrices.

Examples

1. Because of the -4, the following matrix is not positive definite. One can also make this conclusion by showing that $\det(\mathbf{A}) < 0$.

$$\mathbf{A} = \begin{pmatrix} 4 & 1 \\ 1 & -4 \end{pmatrix}. \qquad \blacksquare$$

2. Because of the 4's off the diagonal, the following matrices are not positive definite because they violate the second condition.

$$\mathbf{A}_1 = \begin{pmatrix} 1 & 4 \\ 4 & 2 \end{pmatrix} \quad \text{and} \quad \mathbf{A}_2 = \begin{pmatrix} 1 & 4 \\ 4 & 4 \end{pmatrix}.$$

It should be pointed out that even though a four does appear on the diagonal in $\mathbf{A}_2$, the theorem states that it cannot appear anywhere else if the matrix is positive definite. $\blacksquare$

3. The theorem does not provide any insight about the following two symmetric matrices:

$$\mathbf{A}_3 = \begin{pmatrix} 4 & 3 & 3 \\ 3 & 1 & 3 \\ 3 & 3 & 1 \end{pmatrix} \quad \text{and} \quad \mathbf{A}_4 = \begin{pmatrix} 1 & -2 & -5 \\ -2 & 1 & 5 \\ -5 & 5 & 8 \end{pmatrix}.$$

   Both matrices only have positive numbers on their diagonals, and the largest number only appears on the diagonal. Also, $\det(\mathbf{A}_3) = 4$ and $\det(\mathbf{A}_4) = 26$. However, neither of them is positive definite. This is because the eigenvalues for $\mathbf{A}_3$ are -2, $4-3\sqrt{2}$, and $4+3\sqrt{2}$, while the eigenvalues for $\mathbf{A}_4$ are -1, -2, and 13. ■

There is a simple test to prove a matrix is positive definite and it concerns the size of the numbers on the diagonal compared to the other numbers in their respective rows. The property needed is defined next.

Definition 3.2. A matrix $\mathbf{A}$ is *strictly diagonal dominant* if, for every row,

$$|a_{ii}| > \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}|\,.$$

It is *diagonally dominant* if the above condition holds with $\geq$ instead of $>$.

Theorem 3.4. *A symmetric matrix* $\mathbf{A}$ *is positive definite if the diagonals are all positive and it is strictly diagonal dominant.*

Those interested in the proof of this theorem, or interested in other properties of positive definite matrices, should consult Süli and Mayers [2003].

Examples

1. Using the above theorem, the following matrices are easily shown to be positive definite.

$$\mathbf{A} = \begin{pmatrix} 4 & 1 \\ 1 & 4 \end{pmatrix} \qquad \mathbf{A} = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 3 & -1 \\ 0 & -1 & 2 \end{pmatrix}. \qquad ■$$

2. Although the matrix below is symmetric, because of the second row the matrix is not strictly diagonally dominant. In other words, the above theorem does not apply.

$$\mathbf{A} = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix}$$

It is possible to prove the matrix is positive definite by calculating the eigenvalues, and this is considered in Exercise 3.22. ■

As a final comment, a question that often comes up is where the idea of being positive definite comes from and why is it considered important. Many of the matrices that arise in applications come from approximations of differential equations. This includes Maxwell's equations in electrodynamics, or Navier-Stokes equations in fluid dynamics, or the heat equation in thermodynamics. The spatial terms in these equations often have a property known as ellipticity, and this helps guarantee that the solution is unique or the potential energy in the system behaves in a physically realistic manner. Saying something has ellipticity is a fancy way of saying it is positive definite. The matrices that come from these applications are simply inheriting the property from the original problem. What we are doing here is deriving numerical methods that take advantage of this property.

3.7.1 Cholesky Factorization

In the case of when the matrix is symmetric and positive definite, it is possible to find an LU factorization of the form

$$\mathbf{A} = \mathbf{U}^T\mathbf{U}, \tag{3.17}$$

where the diagonals of $\mathbf{U}$ are positive. This is known as the *Cholesky factorization*. Because this avoids having to calculate $\mathbf{L}$, the flop count is about half of the usual LU count. In other words, when using a Cholesky factorization the flop count is approximately $\frac{1}{3}n^3$. Also note that the procedure for solving the matrix equation is the same as before, it is just that now $\mathbf{L} = \mathbf{U}^T$.

In addition to a reduced flop count, a symmetric and positive definite matrix is always nonsingular. Moreover, the factorization can be carried out without having to use pivoting. These are some of the nice properties referred to earlier. What is not avoided, however, is the possibility that the matrix is ill-conditioned. The diagonal matrix used in Example 2 in Section 3.6.2 is positive definite as long as the diagonals are positive. As demonstrated in that example, the matrix can be either ill-conditioned or well-conditioned depending on the relative values of the diagonals.

Example

Use the Cholesky factorization to solve

$$\begin{pmatrix} 4 & 1 \\ 1 & 4 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -2 \\ 7 \end{pmatrix}.$$

It was earlier shown that the matrix is positive definite, and so the first step is to find the factorization. This is done by assuming that

$$\begin{aligned}\begin{pmatrix} 4 & 1 \\ 1 & 4 \end{pmatrix} &= \begin{pmatrix} u_{11} & 0 \\ u_{12} & u_{22} \end{pmatrix}\begin{pmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{pmatrix} \\ &= \begin{pmatrix} u_{11}^2 & u_{12}u_{11} \\ u_{12}u_{11} & u_{12}^2 + u_{22}^2 \end{pmatrix}.\end{aligned}$$

First, note that we need $u_{11}^2 = 4$, and so $u_{11} = 2$. The negative root is not considered because a Cholesky factorization requires the diagonals to be positive. With this one then finds that $u_{12} = \frac{1}{2}$, and $u_{22} = \frac{1}{2}\sqrt{15}$. The next step is to solve $\mathbf{U}^T\mathbf{y} = \mathbf{b}$, which is

$$\begin{pmatrix} 2 & 0 \\ \frac{1}{2} & \frac{1}{2}\sqrt{15} \end{pmatrix}\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} -2 \\ 7 \end{pmatrix}.$$

The solution is $u = -1$ and $v = \sqrt{15}$. Lastly, we solve $\mathbf{U}\mathbf{x} = \mathbf{y}$, which is

$$\begin{pmatrix} 2 & \frac{1}{2} \\ 0 & \frac{1}{2}\sqrt{15} \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} -1 \\ \sqrt{15} \end{pmatrix}.$$

From this it follows that the solution of the problem is $y = 2$ and $x = -1$. ■

It is interesting how Theorem 3.3 is used by MATLAB. When given the command **A**\b, MATLAB makes a series of tests to decide how to solve the problem. If it finds that the matrix is symmetric, contains only real numbers, and has positive diagonals, it will attempt a Cholesky factorization. It also knows that it is very possible that the matrix is not positive definite so there are built-in contingencies for what to do if the Cholesky factorization fails. What MATLAB is doing is not unreasonable because symmetric and positive definite matrices are so common in applications that having positive diagonals increases the likelihood that the matrix is positive definite to the point that the Cholesky factorization is worth trying.

3.8 Tri-Diagonal Matrices

Another type of matrix that often occurs in applications are those that are tri-diagonal. A standard LU factorization can be used on a tri-diagonal matrix and the procedure can be greatly simplified if one uses the fact that there are so many zeros in the matrix. To explain, a tri-diagonal matrix has the form:

$$\mathbf{A} = \begin{pmatrix} a_1 & c_1 & & & & \\ b_2 & a_2 & c_2 & & 0 & \\ & b_3 & a_3 & c_3 & & \\ & & \ddots & \ddots & \ddots & \\ & 0 & & & & c_{n-1} \\ & & & & b_n & a_n \end{pmatrix}. \tag{3.18}$$

The factorization of such a matrix involves tri-diagonal matrices. In particular, the factorization has the form

$$\begin{pmatrix} a_1 & c_1 & & & & \\ b_2 & a_2 & c_2 & & & \\ & b_3 & a_3 & c_3 & & \\ & & \ddots & \ddots & \ddots & \end{pmatrix} = \begin{pmatrix} \ell_{11} & & & & \\ \ell_{21} & \ell_{22} & & & \\ & \ell_{32} & \ell_{33} & & \\ & & \ddots & \ddots & \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & & & \\ & u_{22} & u_{23} & & \\ & & u_{33} & u_{34} & \\ & & & \ddots & \ddots \end{pmatrix}.$$

An example of this is given in Section 3.4. Keeping track of what is, or is not, zero, the entire LU method reduces to the algorithm given in Table 3.6. This is known as the *Thomas algorithm*. In comparison with a full LU, it

```
Set: $w = a_1,\ x_1 = \frac{z_1}{w}$
For $i = 2, 3, \ldots, n$
     $v_i = \frac{c_{i-1}}{w}$
     $w = a_i - b_i v_i$
     $x_i = \frac{z_i - b_i x_{i-1}}{w}$
End
For $j = n-1, n-2, \ldots, 1$
     $x_j = x_j - v_{j+1} x_{j+1}$
End
```

Table 3.6 Algorithm for solving $\mathbf{Ax} = \mathbf{z}$ when $\mathbf{A}$ is the tri-diagonal matrix given in (3.18).

requires only $8n-7$ flops! Moreover, it is only necessary to store the tri-diagonal portion of the matrix and so the entire method requires storage that amounts to approximately six n-vectors.

Note that tri-diagonal matrices can suffer the same problems more general matrices have. In particular, they can be singular and they can be ill-conditioned. This is evident in the algorithm given in Table 3.6 with the variable w. If w is zero, or nearly zero, then the algorithm will fail. In certain cases it is possible to determine very easily when $w \neq 0$, and this is given next.

Theorem 3.5. *The tri-diagonal matrix* $\mathbf{A}$ *in (3.18) is invertible, and the algorithm in Table 3.6 can be used to solve* $\mathbf{Ax} = \mathbf{z}$*, if either one of the following holds:*

1. $\mathbf{A}$ *is strictly diagonally dominant, or*
2. $\mathbf{A}$ *is diagonally dominant,* $c_i \neq 0 \ \forall i$*, and* $|b_n| < |a_n|$*.*

Outline of Proof: The only operation of concern in the algorithm is the division by w, and so the majority of the proof consists of showing this cannot be zero. For the second set of conditions, note that when $i = 2$, $|v_2| = |c_1/a_1| \leq 1$, where the inequality holds because the matrix is diagonally dominant. With this $|w| = |a_2 - b_2 v_2| \geq |a_2| - |b_2 v_2| \geq |a_2| - |b_2| > 0$, where the last inequality holds because the matrix is diagonally dominant and $c_2 \neq 0$. Continuing this argument, using induction, it is not hard to show that $|v_i| \leq 1$ and $|w| \geq |a_i| - |b_i| > 0$. As before, the last inequality holds, except for the last row of the matrix, because we are assuming $|a_i| \geq |b_i| + |c_i|$ and $c_i \neq 0$. The fact that it holds for $i = n$ is because we have explicitly assumed that $|b_n| < |a_n|$. Showing that w is nonzero when $\mathbf{A}$ is strictly diagonally dominant follows a similar induction proof. What remains is to prove that the vector computed by the algorithm is the solution of the equation, and this is left as an exercise. □

Examples

1. The matrix
$$\mathbf{A} = \begin{pmatrix} 2 & -1 & 0 \\ 1 & 2 & -1 \\ 0 & 1 & 2 \end{pmatrix}$$
is diagonally dominant, $c_1 = c_2 = -1$, and $|b_3| < |a_3|$. Therefore, according to the above theorem, it is invertible and the algorithm in Table 3.6 can be used with it. ■
2. Let
$$\mathbf{A}_1 = \begin{pmatrix} 2 & -1 & 0 \\ 1 & 1 & -1 \\ 0 & 1 & 2 \end{pmatrix}, \quad \mathbf{A}_2 = \begin{pmatrix} 2 & -1 & 0 \\ 1 & 2 & 0 \\ 0 & 1 & 2 \end{pmatrix}, \quad \mathbf{A}_3 = \begin{pmatrix} 2 & -1 & 0 \\ 1 & 1 & -1 \\ 0 & 2 & 2 \end{pmatrix}.$$

These do not satisfy the conditions because: $\mathbf{A}_1$ is not diagonally dominant (because of the second row), $\mathbf{A}_2$ violates the $c_i \neq 0 \; \forall i$ condition, and $\mathbf{A}_3$ violates $|b_3| < |a_3|$. ■

3.9 Sparse Matrices

Another type of matrix that often arises is one containing mostly zeros. These are said to be *sparse*. Although there is not a precise definition of what it means to be sparse, as a rule of thumb, a large $n \times n$ matrix is sparse if there are on the order of n nonzero entries. As an example, a large tri-diagonal matrix is sparse, and the reason is that there are no more than $3n$ nonzero entries in such a matrix. It is worth pointing out that matrices with few zero entries are said to be dense, which is another way of saying that the matrix is not sparse.

The question arises when solving a problem with a sparse matrix if it is possible to avoid having to store all those zero entries, and if it is possible to avoid calculations with them. There are such methods, but they usually require knowing something else about the matrix. For example, if the matrix is also symmetric and positive definite, then something called the sparse Cholesky factorization can be used. Another approach is to use a multifrontal method, which involves partitioning the matrix into smaller blocks [Liu, 1992]. It is also possible to use the conjugate gradient method, and this will be described in Section 8.6.

3.10 Nonlinear Systems

We now consider the problem of how to solve a nonlinear system of equations numerically. An example of this type of problem is to find the value(s) of x and y that satisfy

$$x^2 + 4y^2 = 1, \tag{3.19}$$

$$4x^2 + y^2 = 1. \tag{3.20}$$

Each of these equations defines a curve, and they are plotted in Figure 3.4. It is evident there are four solutions. We will use Newton's method to calculate these solutions, and it will make it easier if we first write the problem in the more general form of solving

$$f(x, y) = 0, \tag{3.21}$$

$$g(x, y) = 0. \tag{3.22}$$

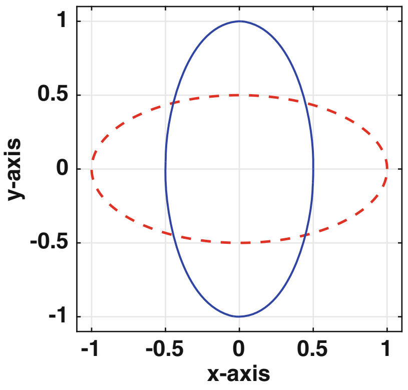


Figure 3.4 Curves coming from the equations in (3.19) and (3.20).

As usual with Newton, it's assumed that an initial guess (x_0, y_0) of the solution is provided. We then approximate the above functions for (x, y) near (x_0, y_0) using Taylor's theorem, which yields

$$\begin{aligned} f(x,y) &\approx f(x_0,y_0) + (x-x_0)f_x(x_0,y_0) + (y-y_0)f_y(x_0,y_0), \\ g(x,y) &\approx g(x_0,y_0) + (x-x_0)g_x(x_0,y_0) + (y-y_0)g_y(x_0,y_0). \end{aligned}$$

Note that both of these are the two variable versions of (2.8). As was done for the one variable case, we now replace (3.21) and (3.22) with

$$\begin{aligned} f(x_0,y_0) + (x-x_0)f_x(x_0,y_0) + (y-y_0)f_y(x_0,y_0) &= 0, \\ g(x_0,y_0) + (x-x_0)g_x(x_0,y_0) + (y-y_0)g_y(x_0,y_0) &= 0. \end{aligned}$$

Rearranging the terms in these equations, we obtain

$$\begin{aligned} xf_x(x_0,y_0) + yf_y(x_0,y_0) &= -f(x_0,y_0) + x_0f_x(x_0,y_0) + y_0f_y(x_0,y_0), \\ xg_x(x_0,y_0) + yg_y(x_0,y_0) &= -g(x_0,y_0) + x_0g_x(x_0,y_0) + y_0g_y(x_0,y_0). \end{aligned}$$

This can be written in matrix form as

$$\mathbf{J}_0\mathbf{x}_1 = -\mathbf{f}_0 + \mathbf{J}_0\mathbf{x}_0,$$

or equivalently as

$$\mathbf{x}_1 = \mathbf{x}_0 - \mathbf{J}_0^{-1}\mathbf{f}_0, \tag{3.23}$$

where

$$\mathbf{J}_0 = \begin{pmatrix} f_x(x_0,y_0) & f_y(x_0,y_0) \\ g_x(x_0,y_0) & g_y(x_0,y_0) \end{pmatrix}, \tag{3.24}$$

$$\mathbf{x}_0 = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}, \quad \mathbf{x}_1 = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix}, \quad \mathbf{f}_0 = \begin{pmatrix} f(x_0,y_0) \\ g(x_0,y_0) \end{pmatrix}.$$

The formula in (3.23) is the multi-variable version of (2.9). The matrix in (3.24) is the Jacobian for the problem, evaluated at $\mathbf{x}_0$.

Continuing this procedure, we obtain the general formula for Newton's method, which is that

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \mathbf{J}_i^{-1}\mathbf{f}_i \,, \text{ for } i = 0, 1, 2, 3, \cdots \tag{3.25}$$

where

$$\mathbf{J}_i = \begin{pmatrix} f_x(x_i, y_i) & f_y(x_i, y_i) \\ g_x(x_i, y_i) & g_y(x_i, y_i) \end{pmatrix}, \tag{3.26}$$

$$\mathbf{x}_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix}, \quad \mathbf{f}_i = \begin{pmatrix} f(x_i, y_i) \\ g(x_i, y_i) \end{pmatrix}. \tag{3.27}$$

The formula in (3.25) is the multi-variable version of (2.10). Also, the earlier requirement that $f'(x)$ is nonzero now becomes the condition that the Jacobian $\mathbf{J}$ is nonsingular.

For numerical reasons it is better to rewrite (3.25) to reduce the flop count. First note that it can be written as $\mathbf{J}_i\mathbf{x}_{i+1} = \mathbf{J}_i\mathbf{x}_i - \mathbf{f}_i$, and this can be written as $\mathbf{J}_i(\mathbf{x}_{i+1} - \mathbf{x}_i) = -\mathbf{f}_i$. Therefore, (3.25) can be broken into two steps, first one solves

$$\mathbf{J}_i\mathbf{z} = -\,\mathbf{f}_i \,, \tag{3.28}$$

and then

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{z} \,. \tag{3.29}$$

With this we have avoided having to determine $\mathbf{J}_i^{-1}$ and then calculating $\mathbf{J}_i^{-1}\mathbf{f}_i$. As a final comment, the above expressions were derived for the two-variable problem in (3.21) and (3.22). However, they apply to the more general problem of n nonlinear equations in n unknowns. In this case, $\mathbf{J}$ is the $n \times n$ Jacobian matrix, while $\mathbf{x}_i$ and $\mathbf{z}$ are n-vectors.

Example

For the equations in (3.19) and (3.20), $f(x, y) = x^2 + 4y^2 - 1$ and $g(x, y) = 4x^2 + y^2 - 1$. The Jacobian is therefore

$$\mathbf{J} = \begin{pmatrix} 2x & 8y \\ 8x & 2y \end{pmatrix}.$$

Setting $\mathbf{z} = (u, v)^T$, then (3.28) becomes

$$\begin{pmatrix} 2x_i & 8y_i \\ 8x_i & 2y_i \end{pmatrix}\begin{pmatrix} u \\ v \end{pmatrix} = -\begin{pmatrix} x_i^2 + 4y_i^2 - 1 \\ 4x_i^2 + y_i^2 - 1 \end{pmatrix}. \tag{3.30}$$

After solving this, using (3.29),

$$\begin{pmatrix} x_{i+1} \\ y_{i+1} \end{pmatrix} = \begin{pmatrix} x_i + u \\ y_i + v \end{pmatrix}. \tag{3.31}$$

i	x_i	y_i	Error
0	1	1	
1	0.6000000000	0.6000000000	4.00e−01
2	0.4666666667	0.4666666667	1.33e−01
3	0.4476190476	0.4476190476	1.90e−02
4	0.4472137791	0.4472137791	4.05e−04
5	0.4472135955	0.4472135955	1.84e−07
6	0.4472135955	0.4472135955	3.77e−14

Table 3.7 Solving (3.19) and (3.20) using the Newton's method formula given in (3.30) and (3.31). Also given is the iterative error $e_i = ||\mathbf{x}_i - \mathbf{x}_{i-1}||_\infty$.

The values obtained using this procedure are shown in Table 3.7. As expected when using Newton's method, the error shows that the method is second order. This is because once the method starts to get close to the solution, the error at step i is approximately the square of the error at step $i-1$. ■

Newton's method, as given in (3.28) and (3.29), is relatively easy to derive. We also saw that when it works, the error shows the second-order convergence that is the hallmark of the method. The fact is, however, that solving nonlinear equations with multiple variables using Newton's method, or any method for that matter, is challenging. One reason is that Newton's method requires a good guess for the solution, and these can be difficult to come by. It is natural to ask if there is a bisection type method that can be used to help locate good guesses. There is a method that has some similarity to bisection, but it requires the problem to be reformulated as a minimum problem. This is not hard to do, and as an example, one can rewrite (3.21) and (3.22) as follows: find the values of x and y that minimize

$$F(x, y) = f(x, y)^2 + g(x, y)^2.$$

The various numerical methods you can use to solve this are investigated in Chapter 8.

3.11 Some Additional Ideas

3.11.1 Yogi Berra and Perturbation Theory

Yogi Berra, an insightful baseball personality, once said "In theory there is no difference between theory and practice. In practice there is." This has particular applicability to computing. As an example, it was stated earlier that pivoting is not needed when finding an LU factorization of a symmetric positive definite matrix. Yet, it is possible to find a symmetric positive definite matrix that, when attempting to compute its factorization, the procedure fails (without pivoting). The reason is that it can happen that the floating point approximation of a matrix does not have the same properties as the original. As a case in point, it is possible to find a symmetric positive definite matrix whose floating point approximation, although symmetric, is not positive definite. Situations similar to this are considered in Exercise 3.15.

This helps explain the interest in what is called matrix perturbation theory. The idea here is that the original matrix $\mathbf{A}$ and its floating point approximation $\mathbf{A}_f$ are related through an equation of the form $\mathbf{A}_f = \mathbf{A} + \mathbf{P}$. The entries of $\mathbf{P}$ are proportional to machine ε, and are generally, although not always, much smaller than the entries in $\mathbf{A}$. In the vernacular of the subject, $\mathbf{P}$ is called a perturbation matrix. The question is, if $\mathbf{A}$ has a certain property, under what conditions, if any, will $\mathbf{A}_f$ have that same property? As an example, one can prove that if $\mathbf{A}$ is invertible, then $\mathbf{A}_f$ is invertible if the perturbation matrix is small enough that

$$\frac{||\mathbf{P}||}{||\mathbf{A}||} < \frac{1}{\kappa(\mathbf{A})} \, .$$

In other words, an invertible matrix will remain invertible if perturbed by a small enough amount that it satisfies the above inequality. The limitation of this statement is that, when computing, you do not know if the precision you are using is enough to guarantee that such an inequality is satisfied. Nevertheless, matrix perturbation theory plays a prominent role in the analysis of matrix algorithms, and more can be learned about this in Golub and Van Loan [2013] and Higham [2002].

3.11.2 Fixing an Ill-Conditioned Matrix

When stuck with having to solve a problem with an ill-conditioned matrix it is natural to try to modify the equation to improve the situation. For example, one might think that by multiplying the problem by a well-chosen constant α that the condition number can be lowered. However, as explained in Section 3.6.2, $\kappa(\alpha\mathbf{A}) = \kappa(\mathbf{A})$, so that idea will not work. Not giving up, the next attempt would be to multiply each equation making up the system by a different constant (e.g., the first equation by d_1, the second by d_2, etc). This can be expressed in matrix form by stating that the matrix equation is going to be multiplied by a diagonal matrix $\mathbf{D}$. In fact, if we are going to do this why not use the most general version and multiply by a matrix $\mathbf{P}$ that is not limited to being diagonal? Doing this, the problem becomes $\mathbf{Cx} = \mathbf{d}$, where $\mathbf{C} = \mathbf{PA}$ and $\mathbf{d} = \mathbf{Pb}$. The matrix $\mathbf{P}$ is called a *pre-conditioner*, and it is chosen so the new matrix $\mathbf{C}$ is not ill-conditioned. Note that it is possible to find a pre-conditioner so that $\kappa(\mathbf{C}) = 1$, which is the smallest value possible for the condition number. Namely, one can just take $\mathbf{P} = \mathbf{A}^{-1}$. Obviously, this is not a viable possibility because we do not know $\mathbf{A}^{-1}$. However, it does provide some idea of what one might look for, which is an easy to find approximation for $\mathbf{A}^{-1}$. Considerable research has been invested in how to do this and Benzi [2002] should be consulted to learn about this.

3.11.3 Insightful Observations About the Condition Number

As stated earlier, the condition number plays a central role in determining how accurately a matrix equation can be solved. The heuristic described in Section 3.6.3 helps to quantify its role in affecting the accuracy. However, there are other, more qualitative, ways to interpret its role. This means they are not particularly useful for evaluating it but they do provide insight into the impact of the condition number on the accuracy.

1. It is sometimes said that the condition number is a measure of the distortion associated with the matrix. This comment comes from the formula

$$\kappa(\mathbf{A}) = \frac{\max_{||\mathbf{x}||=1} ||\mathbf{Ax}||}{\min_{||\mathbf{x}||=1} ||\mathbf{Ax}||} .$$

   This expression was derived in Section 3.6.2 in the special case that the matrix is diagonal (also see Exercise 3.21). It shows that the larger the condition number the greater the distortion. To explain, the matrix norm is determined by what the matrix does to the vectors which satisfy $||\mathbf{x}|| = 1$. This is illustrated in Figure 3.3. If the transformed curve is close to a circle, then the max and min values of $||\mathbf{Ax}||$ are not too far apart and the condition number is not very big. However, if this transformed curve looks like an elongated ellipse, then the condition number gets a lot bigger. The greater the distortion, the larger κ becomes.

   A natural question to ask is, what does this distortion have to do with solving matrix equations accurately. To explain, consider the example in Section 3.6.2, where

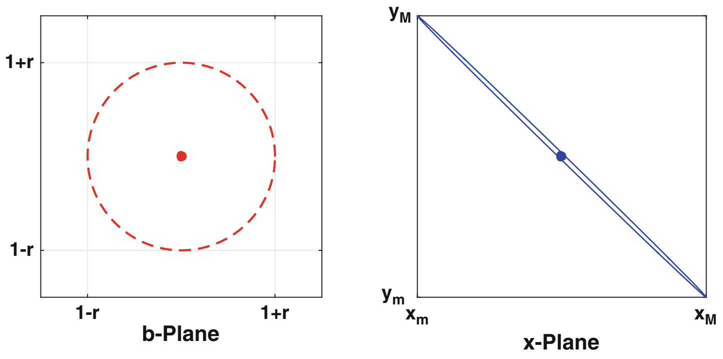


Figure 3.5 Left: The floating point approximation of **b** is located within a small circle centered at **b**. Right: The ellipse consists of those points that produce the circle on the left when evaluating **Ax**. The solid dot is the location of the exact solution.

$$\mathbf{A} = \begin{pmatrix} 1 & 2 \\ -1 & d \end{pmatrix}.$$

When solving $\mathbf{Ax} = \mathbf{b}$ numerically, **b** is replaced with its floating point approximation, which is located somewhere in a small circle centered at **b**. The radius of this circle is determined by machine epsilon, and this is illustrated in the left graph in Figure 3.5. In this example, $\mathbf{b} = (1, 1)^T$, $r = 10^{-6}$, and $d = -1.9$ (it is assumed for demonstration purposes that machine epsilon is on the order of 10^{-6}). The ellipse on the right is made up of those points that produce the circle on the left. In other words, if **x** is a point on the ellipse, then **Ax** is a point on the circle. This is significant because given the floating point vector $\mathbf{b}_f$, the solution of $\mathbf{Ax} = \mathbf{b}_f$ will be somewhere inside the ellipse. When the ellipse is very distorted, which happens when the condition number is large, and the computed solution $\mathbf{x}_f$ is towards one of the far ends of the ellipse, then $\mathbf{x}_f$ will be far away from the actual solution. For example, the target circle on the right in Figure 3.5 has $r = 10^{-6}$, but the major axis of the ellipse is approximately 7×10^{-5}. So, the accuracy in the solution does not match the accuracy in the floating point approximation for **b**. The situation gets worse as the condition number increases. To illustrate, if $d = -1.999999$ and r is the same as before, then the major axis of the ellipse is approximately 7, and $\kappa_\infty(\mathbf{A}) \approx 10^7$. In this case it could happen that the computed solution is not correct to any significant digit. Also, note that this conclusion is consistent with the heuristic described in Section 3.6.3.

2. Another often made comment is that the condition number is a measure of how close the matrix is to being singular (non-invertible). For example, MATLAB will issue the response "Warning: Matrix is close to singular or

badly scaled" in alarming orange text when given a matrix with a large condition number. The reason for this is the formula

$$\frac{1}{\kappa(\mathbf{A})} = \min\left\{\frac{||\mathbf{A}-\mathbf{B}||}{||\mathbf{A}||} : \mathbf{B} \text{ is singular}\right\}.$$

To decipher this, recall that the distance between two vectors can be written as $||\mathbf{x}-\mathbf{y}||$. In the same way the distance between two matrices can be written as $||\mathbf{A}-\mathbf{B}||$. Also recall how one calculates the distance between a point and an object such as a sphere. Namely, if $\mathbf{x}$ is the point, then the distance to the object is the smallest distance between $\mathbf{x}$ and the points making up the object. In other words, if S is the object, then the distance between $\mathbf{x}$ and S is $\min\{||\mathbf{x}-\mathbf{y}|| : \mathbf{y} \in S\}$. Based on this observation, the right-hand side of the above formula is the normalized distance between $\mathbf{A}$ and the set of singular matrices. What the formula shows is that the larger the condition number the closer the matrix is to the set of singular matrices.

3.11.4 Faster than LU?

As explained in Section 3.4, using LU takes approximately $\frac{2}{3}n^3$ flops. This is better than calculating $\mathbf{A}^{-1}\mathbf{b}$, which takes approximately $2n^3$ flops. The important observation for this discussion is that both methods are $O(n^3)$. This raises the question as to whether there are sub-cubic methods, in other words, can you find a method that requires $O(n^\omega)$ flops, where $\omega < 3$? Considerable research has been invested in this question, and the usual approach to answering it is to change the question. It can be proved that if you can multiply two $n \times n$ matrices using $O(n^\omega)$ flops then you can solve $\mathbf{Ax} = \mathbf{b}$ using $O(n^\omega)$ flops [Bunch and Hopcroft, 1974]. The usual method for multiplying two matrices requires n^3 multiplications and $n^3 - n^2$ additions, in other words, $O(n^3)$ flops. The cubic barrier for matrix multiplication was first broken by Strassen, who was able to produce an algorithm that uses $O(n^\omega)$ flops, where $\omega = \log_2 7 \approx 2.8$ [Strassen, 1969]. Others have worked on improving this, and the current best result is $\omega \approx 2.3727$ [Williams, 2012]. You might be wondering if anyone actually uses these methods to solve matrix equations. The answer is that these are mostly theoretical results, and the methods are rarely used in practice. The exception is Strassen's method, although it is not straightforward to implement [Bailey et al., 1991; Huss-Lederman et al., 1996]. Those who might be interested in this topic should consult Higham [2002].

3.11.5 Historical Comparisons

Although this discussion is going to be similar to when parents tell their children how hard it was "back in the day," it is worth knowing what early researchers in the area said about solving matrix problems. The particular individual is William Kahan, who was awarded the A.M. Turning Prize for his contributions in floating-point computations and his dedication to "making the world safe for numerical computations." Anyway, in Kahan [1966], when commenting about the difficulty of solving $\mathbf{Ax} = \mathbf{b}$ stated "On our computer (an IBM 7094-II at the University of Toronto) the solution of 100 linear equations in 100 unknowns can be calculated in about 7 seconds". This is where you are supposed to point out that we are now able to do this in less than 10^{-4} sec, which is no surprise. His next comment is more interesting, and he states that "This calculation costs about a dollar." Computers were treated like taxi-cabs, but instead of charging by the mile (or kilometer) they charged by the second. Fortunately, the introduction of UNIX ended this particular practice at most universities. He goes on to say, "to solve 10000 linear equations would take more than two months." Again, this is where it is necessary to comment that on current machines this takes about 1 sec. One might think the reason is the improvement in the speed of the processors, which is partly true. The more significant reason is memory. They were unable to store everything in active memory (RAM) and were forced into using what Kahan calls "bulk storage units, like magnetic tapes or disks." This eventually became known as the "swap to disk" problem, and as you would expect, the time required to transfer data back and forth to tape means you will be measuring computing time not with a stop-watch but with a calendar. Also, when a code can take months (or even days) a factor of two is actually significant. This forced them into using every possible trick available, like writing $0.5 * x$ instead of $x/2$.

The fact is that many of the concerns Kahan talks about are still with us, only the scale has changed. It is true that for most people, the capability of computers today is more than sufficient. However, for those working to solve the large multi-dimensional problems associated with physically realistic models that arise in many applications, the processing power is still not adequate. This is why they are in the process of building exascale computers, so they will be able to solve problems with trillions of unknowns. As often stated, these will be about 30 to 100 times faster than what we now have and will possibility be available in five years [Peckham, 2013]. To give Kahan the last word about these new computer systems, as he stated in 1966, "The time might come down to a day or so when machines 100 times faster than ours are produced, but such machines are just now being developed, and are most unlikely to be in widespread use within the next five years."

Exercises

3.1. The following are true or false. If true, provide an explanation why it is true, and if it is false provide an example demonstrating this along with an explanation why it shows the statement is incorrect.

(a) If $\mathbf{A}$ is strictly diagonally dominant, then $\mathbf{A}^T$ is strictly diagonally dominant.

(b) If $\mathbf{A}$ is strictly diagonally dominant, then $\alpha\mathbf{A}$, where α is a nonzero number, is strictly diagonally dominant.

(c) If $\mathbf{A}$ is symmetric, then $||\mathbf{A}||_\infty = ||\mathbf{A}||_1$.

(d) If a nonzero vector $\mathbf{x}$ can be found so $\mathbf{Ax} = \mathbf{0}$, where $\mathbf{A}$ is symmetric, then $\mathbf{A}$ is not positive definite.

(e) If $\mathbf{A}$ is positive definite, and symmetric, then $\mathbf{A}$ only has positive entries.

(f) If $\mathbf{A}$ is the 2×2 zero matrix and $\mathbf{A} = \mathbf{LU}$, then either $\mathbf{L}$ or $\mathbf{U}$ is the zero matrix.

(g) Because $||\mathbf{x}||_\infty \le ||\mathbf{x}||_1$ then it must be that $||\mathbf{A}||_\infty \le ||\mathbf{A}||_1$.

(h) Assuming $\mathbf{A}$ is 2×2, if $\mathbf{A}$ is symmetric and positive definite, then $\mathbf{A}^{-1}$ is symmetric and positive definite.

(i) A symmetric and positive definite matrix must be strictly diagonally dominant.

(j) An ill-conditioned matrix can be made well conditioned using pivoting (you can assume that the matrix is 2×2).

3.2. Using a Doolittle factorization, solve the following equations (by hand). Also, calculate $\kappa_\infty(\mathbf{A})$.

(a)

$$\begin{aligned} x + y &= 1 \\ x + 4y &= 7. \end{aligned}$$

(b)

$$\begin{aligned} x - 2y &= 0 \\ -x + 4y &= 1. \end{aligned}$$

(c)

$$\begin{aligned} -2x + y &= 3 \\ 4x - 6y &= -14. \end{aligned}$$

(d)

$$\begin{aligned} x + z &= 1 \\ x + y &= -1 \\ y + z &= 0. \end{aligned}$$

3.3. Find the Doolittle factorization of the following matrices:
(a)

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -2 & 2 \\ 0 & 2 & -1 \end{pmatrix}.$$

(b)

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \\ 1 & 2 & 3 \end{pmatrix}.$$

(c)

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}.$$

3.4. In this problem α is a small positive number. Sketch the two lines in the x, y-plane, and describe how they change, including the point of intersection, as α approaches zero. Also, calculate the condition number for the matrix, and describe how it changes as α approaches zero.
(a)

$$\begin{aligned} x - y &= -1 \\ -x + (1+\alpha)y &= 1. \end{aligned}$$

(b)

$$\begin{aligned} 2x + 4y &= 1 \\ (1-\alpha)x + 2y &= -1. \end{aligned}$$

3.5. Consider the matrix

$$\mathbf{A} = \begin{pmatrix} 1 & -1 & -1 \\ 1 & -4 & 0 \\ 0 & 1 & 0 \end{pmatrix}.$$

It is useful, but not essential, to know that this is a unimodular matrix.
(a) Find $\mathbf{A}^{-1}$.
(b) Find $\kappa_\infty(\mathbf{A})$.
(c) Find the Doolittle factorization of $\mathbf{A}$.

3.6. Consider the matrix

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}.$$

It is useful, but not essential, to know that this is a normal matrix.
(a) Find $\mathbf{A}^{-1}$.
(b) Find $\kappa_\infty(\mathbf{A})$.
(c) Find the Doolittle factorization of $\mathbf{A}$.

3.7. For each of the following matrices, explain why it is positive definite and then find the Cholesky factorization.

(a)
$$\mathbf{A} = \begin{pmatrix} 1 & 1 \\ 1 & 4 \end{pmatrix}.$$

(b)
$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 1 \\ 0 & 1 & 5 \end{pmatrix}.$$

(c)
$$\mathbf{A} = \begin{pmatrix} 4 & 6 & 0 \\ 6 & 25 & 0 \\ 0 & 0 & 16 \end{pmatrix}.$$

3.8. In this problem,
$$\mathbf{A} = \begin{pmatrix} 1 & -2 \\ -2 & 3 \end{pmatrix}.$$

(a) Find a vector $\mathbf{x}$ so that $||\mathbf{Ax}||_\infty = ||\mathbf{A}||_\infty$.
(b) Find a vector $\mathbf{x}$ so that $||\mathbf{Ax}||_1 = ||\mathbf{A}||_1$.
(c) One can show that $||\mathbf{A}||_2 = 9 + 4\sqrt{5}$. Find a vector $\mathbf{x}$ so that $||\mathbf{Ax}||_2 = ||\mathbf{A}||_2$.

3.9. In this problem assume that $\mathbf{A}$ is a 2×2 matrix
(a) Suppose the Doolittle and Crout factorizations produce the same result. This means, for example, that the lower triangular matrix found for each factorization is the same. What can you say about the entries in the matrix $\mathbf{A}$?
(b) Suppose the Doolittle, Crout and Cholesky factorizations produce the same result. What can you say about the entries in the matrix $\mathbf{A}$?

3.10. This problem considers the question, why not use UL instead of LU? You can assume that $\mathbf{A}$ is a 2×2 matrix.
(a) Find a Doolittle version of the factorization $\mathbf{A} = \mathbf{UL}$, where $\mathbf{L}$ is a unit lower triangular matrix and $\mathbf{U}$ is upper triangular. You can assume pivoting is not necessary.
(b) Describe the resulting algorithm for solving $\mathbf{Ax} = \mathbf{b}$.
(c) Is there any connection between the Doolittle factorization of $\mathbf{A}^T$ and the one you found in part (a)?

(d) Aside from an alphabetic advantage, is there any reason to prefer LU over UL?

3.11. A matrix often used to test the effectiveness of algorithms used to calculate eigenvalues is the Rosser matrix, which is given as

$$\mathbf{R} = \begin{pmatrix} 611 & 196 & -192 & 407 & -8 & -52 & -49 & 29 \\ 196 & 899 & 113 & -192 & -71 & -43 & -8 & -44 \\ -192 & 113 & 899 & 196 & 61 & 49 & 8 & 52 \\ 407 & -192 & 196 & 611 & 8 & 44 & 59 & -23 \\ -8 & -71 & 61 & 8 & 411 & -599 & 208 & 208 \\ -52 & -43 & 49 & 44 & -599 & 411 & 208 & 208 \\ -49 & -8 & 8 & 59 & 208 & 208 & 99 & -911 \\ 29 & -44 & 52 & -23 & 208 & 208 & -911 & 99 \end{pmatrix}.$$

Answer the following questions without using a computer.
(a) Is $\mathbf{R}$ symmetric?
(b) Is $\mathbf{R}$ positive definite? You must explain how you come to this conclusion.

3.12. This problem concerns the equation $\mathbf{Ax} = \mathbf{b}$, where

$$\mathbf{A} = \begin{pmatrix} -1 & 1 \\ 0 & \alpha \end{pmatrix}$$

and $\alpha > 0$.
(a) For what values of α is this matrix ill-conditioned? Make sure to identify what norm you are using.
(b) Suppose the residual $\mathbf{r}$ is small (but nonzero). For what values of α, if any, will the error $\mathbf{e} = \mathbf{x} - \mathbf{x}_c$ be large? Note $\mathbf{x}$ is the exact solution and $\mathbf{x}_c$ is the computed solution.
(c) Suppose the error $\mathbf{e}$ is small (but nonzero). For what values of α, if any, will the residual be large?

3.13. Use the three defining properties of a vector norm for the following.
(a) Show that if $\mathbf{x} = \mathbf{0}$, then $||\mathbf{x}|| = 0$.
(b) Show that for any $\mathbf{x}$, $||\mathbf{x}|| \geq 0$.
(c) Show that $||\mathbf{x}||_1$ is a vector norm.

3.14. Assuming pivoting is not necessary, then a symmetric matrix $\mathbf{A}$ can be factored as $\mathbf{A} = \mathbf{LDL}^T$, where $\mathbf{L}$ is a lower triangular matrix with ones on its diagonal and $\mathbf{D}$ is a diagonal matrix.
(a) Find the $\mathbf{LDL}^T$ factorization of the matrix

$$\mathbf{A} = \begin{pmatrix} -4 & 1 \\ 1 & 1 \end{pmatrix}.$$

(b) In Section 3.1, it was shown how an LU factorization results in solving two matrix equations (for $\mathbf{y}$ and $\mathbf{x}$). Explain how an LDL^T results in solving three matrix equations. Use this to solve

$$\begin{aligned} -4x + y &= 2 \\ x + y &= 1. \end{aligned}$$

(c) Explain why the flop count for solving $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is a symmetric $n \times n$ matrix, using an LDL^T factorization is approximately half of the flop count when using an LU factorization.

3.15. In this exercise, $\mathbf{A}$ is a 2×2 matrix, and $\mathbf{A}_f$ is its floating point approximation (using double precision).

(a) Give an example of an invertible matrix $\mathbf{A}$ where $\mathbf{A}_f$ is the zero matrix. Your example should also have $\kappa(\mathbf{A}) = 1$.

(b) Give an example of a symmetric and positive definite matrix $\mathbf{A}$ where $\mathbf{A}_f$ is symmetric but not positive definite.

(c) Give an example of a strictly diagonally dominant matrix $\mathbf{A}$ where $\mathbf{A}_f$ is not strictly diagonally dominant.

(d) Is it possible for $\mathbf{A}$ to be symmetric but $\mathbf{A}_f$ not symmetric?

3.16. This problem considers two ways to solve $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is an $n \times n$ magic matrix and the exact solution is $\mathbf{x} = (1, 1, \cdots, 1)^T$. The matrix $\mathbf{A}$ should be calculated in MATLAB using the `magic(n)` command, and calculate $\mathbf{b}$ using the formula $\mathbf{b} = \mathbf{Ax}$. In what follows $\mathbf{x}_M$ designates the solution computed using the MATLAB backslash operator, and $\mathbf{x}_I$ is the solution computed using the inverse formula $\mathbf{x}_I = \mathbf{A}^{-1}\mathbf{b}$. Use MATLAB to fill out Table 3.8 and then answer the following questions (note that $\mathbf{r} = \mathbf{b} - \mathbf{Ax}_M$). Also, the entries in the table only need to include two significant digits, and the norms refer to the infinity norm.

(a) Do you see any substantial differences between the two solution methods when they are compared using the relative error?

(b) Does a small residual indicate an accurate solution? Your answer should include a comment on the value of the condition number. What about any dependence on the size n of the matrix?

(c) How well does the last column predict the relative error?

3.17. This problem considers two ways to solve $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A} = 3\mathbf{P}$ and $\mathbf{P}$ is the $n \times n$ Pascal matrix. Also, the exact solution is $\mathbf{x} = (1, 1, \cdots, 1)^T$. The matrix $\mathbf{A}$ should be calculated in MATLAB using the `pascal(n)` command, and calculate $\mathbf{b}$ using the formula $\mathbf{b} = \mathbf{Ax}$. In what follows $\mathbf{x}_M$ designates the solution computed using the MATLAB backslash operator, and $\mathbf{x}_I$ is the solution computed using the inverse formula $\mathbf{x}_I = \mathbf{A}^{-1}\mathbf{b}$. Use MATLAB to fill out Table 3.9 and then answer the following questions (note that $\mathbf{r} = \mathbf{b} - \mathbf{Ax}_M$). Also, the entries in the table only need to include two significant digits, and the norms refer to the infinity norm.

n	$\dfrac{\lvert\lvert \mathbf{x}-\mathbf{x}_M \rvert\rvert}{\lvert\lvert \mathbf{x} \rvert\rvert}$	$\dfrac{\lvert\lvert \mathbf{x}-\mathbf{x}_I \rvert\rvert}{\lvert\lvert \mathbf{x} \rvert\rvert}$	$\lvert\lvert \mathbf{r} \rvert\rvert$	$\kappa(\mathbf{A})$	$\varepsilon\kappa(\mathbf{A})$
3					
6					
9					
12					

Table 3.8 Table for Exercise 3.16.

(a) Do you see any substantial differences between the two solution methods when they are compared using the relative error?
(b) Does a small residual indicate an accurate solution? Your answer should include a comment on the value of the condition number. What about any dependence on the size n of the matrix?
(c) How well does the last column predict the relative error?

3.18. This exercise considers the following three versions of the same problem:

$$\begin{array}{lll} ax+by=b_1 & cx+dy=b_2 & by+ax=b_1 \\ cx+dy=b_2\,, & ax+by=b_1\,, & dy+cx=b_2. \end{array}$$

Here a, b, c, d, as well as b_1 and b_2, are assumed known. Note that each version differs only in the order the equations are written down. A property of this problem is said to be fragile if it holds for one of the versions but not all of them.
(a) Is symmetry of the coefficient matrix a fragile property?
(b) Is uniqueness of the solution a fragile property?
(c) Is ill-conditionedness of the coefficient matrix a fragile property?
(d) Is invertibility of the coefficient matrix a fragile property?

3.19. This problem considers whether a Cholesky type factorization can be used on a matrix which is not positive definite. The assumption is that given an invertible symmetric matrix $\mathbf{A}$, then $\mathbf{A} = \mathbf{C}^T\mathbf{C}$ where $\mathbf{C}$ is upper triangular with possibly complex-valued entries. In this problem this will be referred to as a generalized Cholesky factorization. What will be shown is that a generalized Cholesky factorization is possible as long as the leading principal minors of $\mathbf{A}$ are nonzero. Note that a method that avoids the use of complex-valued factors is considered in Exercise 3.14.

n	$\frac{\lVert\mathbf{x}-\mathbf{x}_M\rVert}{\lVert\mathbf{x}\rVert}$	$\frac{\lVert\mathbf{x}-\mathbf{x}_I\rVert}{\lVert\mathbf{x}\rVert}$	$\lVert\mathbf{r}\rVert$	$\kappa(\mathbf{A})$	$\varepsilon\kappa(\mathbf{A})$
4					
8					
12					
16					

Table 3.9 Table for Exercise 3.17.

(a) The following matrix is invertible and symmetric but not positive definite. Find a matrix $\mathbf{C}$ satisfying the stated assumption.

$$\mathbf{A} = \begin{pmatrix} -4 & 1 \\ 1 & 1 \end{pmatrix}$$

(b) Using the factorization found in part (a), solve

$$\begin{aligned} -4x + y &= 2 \\ x + y &= 1. \end{aligned}$$

Do you obtain the same answer you would get if you did not use the factorization?

(c) What conditions must be imposed on the entries of the following symmetric matrix so it is invertible and has a generalized Cholesky factorization.

$$\mathbf{A} = \begin{pmatrix} a & b \\ b & c \end{pmatrix} .$$

(d) Answer the question posed in part (c) for a symmetric 3×3 matrix.

3.20. This exercise looks at some of the theorems about symmetric and positive definite matrices in the case of when

$$\mathbf{A} = \begin{pmatrix} a & b \\ b & c \end{pmatrix} .$$

As will be established in part (a), this matrix is positive definite if, and only if, $a > 0$ and $ac - b^2 > 0$. This result is then used to prove the various theorems we had about positive definite matrices.

(a) Show that the eigenvalues of the matrix are

$$\lambda_{\pm} = \frac{1}{2}\left[a + c \pm \sqrt{(a+c)^2 - 4(ac-b^2)}\right] .$$

Explain why the eigenvalues are positive if, and only if, $a + c > 0$ and $ac - b^2 > 0$. Also explain why these two conditions can be replaced with the requirements that $a > 0$ and $ac - b^2 > 0$.

(b) Use the result from part (a) to prove that $\mathbf{A}$ is not positive definite if any diagonal entry is negative or zero.

(c) Use the result from part (a) to prove that if $\mathbf{A}$ is positive definite, then the largest number in $\mathbf{A}$, in absolute value, can only appear on the diagonal.

(d) Use the result from part (a) to prove that if $\mathbf{A}$ is positive definite, then $\det(\mathbf{A}) > 0$.

(e) Use the result from part (a) to prove that $\mathbf{A}$ is positive definite if the diagonals are all positive and it is strictly diagonal dominant..

(f) What are the three equations for the u_{ij}'s that come from the Cholesky factorization? Explain why the assumption that $\mathbf{A}$ is positive definite is exactly what is needed to guarantee that you can find a real-valued solution to these equations.

3.21. This exercise shows that the formulas derived in Example 3 of Section 3.6.2 apply to any vector norm. Assume $\mathbf{D}$ is a diagonal $n \times n$ matrix, with diagonal entries d_1, d_2, $\cdots$, d_n.

(a) Show that $||\mathbf{A}||_\infty = \max\{\, |d_1|, |d_2|, \cdots, |d_n|\,\}$.

(b) Assuming the d_i's are not zero, show that

$$||\mathbf{A}^{-1}||_\infty = \frac{1}{\min\{\, |d_1|, |d_2|, \cdots, |d_n|\,\}}\,.$$

(c) Show that

$$\kappa_\infty(\mathbf{A}) = \frac{\max\{\, |d_i|\,\}}{\min\{\, |d_i|\,\}}\,.$$

3.22. This problem considers the following $n \times n$ tri-diagonal matrix:

$$\mathbf{A} = \begin{pmatrix} a & c & & & & \\ b & a & c & & \text{\Large 0} & \\ & b & a & c & & \\ & & \ddots & \ddots & \ddots & \\ & \text{\Large 0} & & b & a & c \\ & & & & b & a \end{pmatrix}.$$

It is possible to show that the eigenvalues of this matrix are

$$\lambda_i = a + 2\sqrt{bc}\cos\left(\frac{i\pi}{n+1}\right), \quad \text{for } \; i = 1, 2, \ldots, n.$$

Also, assume that $n \geq 3$.

(a) What is $||\mathbf{A}||_\infty$, and what is $||\mathbf{A}||_1$?

(b) In the case that $\mathbf{A}$ is symmetric (so, $c = b$), what inequality must be satisfied to guarantee that $\mathbf{A}$ is strictly diagonal dominant?

(c) Assuming the matrix is symmetric and a is positive with $2|b| \leq a$, explain why $\mathbf{A}$ is positive definite.

3.23. Consider the following nonlinear system of equations:

$$\begin{aligned} x^2 + y^2 &= 4, \\ y &= x^3. \end{aligned}$$

(a) Sketch the two curves and explain where (approximately) the solutions are located.
(b) What is $\mathbf{J}$ and $\mathbf{f}$, as given in (3.28), for this problem?
(c) What would be good starting values for the solutions you identified in part (a)? Make sure to provide an explanation for your choices.

3.24. Consider the following nonlinear system of equations:

$$\begin{aligned} 4x^2 + y^2 &= 16, \\ x^3 y &= 2. \end{aligned}$$

(a) Sketch the two curves and explain where (approximately) the solutions are located.
(b) What is $\mathbf{J}$ and $\mathbf{f}$, as given in (3.28), for this problem?
(c) What would be good starting values for the solutions you identified in part (a)? Make sure to provide an explanation for your choices.

3.25. This problem considers the situation of when the matrix is tri-diagonal, symmetric and positive definite. The equation to solve is $\mathbf{Ax} = \mathbf{z}$, where $\mathbf{A}$ is given in (3.18). Because the matrix is symmetric and positive definite, $a_i > 0$ and $b_i = c_{i-1}$.

(a) Show that the elements of the Cholesky factorization can be determined using an algorithm of the form

```
$d_1 = \sqrt{a_1}$
for $i = 2 : n$
    $v_{i-1} =$
    $d_i =$
end
```

where $d(i) = u(i,i)$ and $v(i) = u(i,i+1)$. Note that instead of working with the matrix $\mathbf{U}$, only the diagonal and upper diagonal entries are computed (since all the other entries are zero).

(b) Assuming the Cholesky factorization has been determined, show that the algorithm for solving the equation can be written as

```
$y_1 = z_1/d_1$
for $i = 2 : n$
    $y_i =$
end
$x_n = y_n/d_n$
for $i = n-1, n-2, \cdots, 1$
    $x_i =$
end
```

(c) Use your algorithm from parts (a) and (b) to solve the matrix equation in the case of when $a_i = 3$, $b_i = c_i = 1$, and n =100,000. Also, take $\mathbf{z} = \mathbf{Ax}$, where $\mathbf{x} = (1, 1, \cdots, 1)^T$. It is only necessary to report the values of $x(1)$ and $x(2)$ (to 16 digits). Also, report the computed value of $||\mathbf{r}||_\infty$ and $||\mathbf{e}||_\infty$.

(d) Using the same matrix as in part (c), use your algorithm to solve the matrix equation when $\mathbf{x} = (1, -1, 1, -1, \cdots, -1)^T$. It is only necessary to report the values of $x(1)$ and $x(2)$ (to 16 digits). Moreover, you must give a compelling explanation of why you believe you answer is correct (within the limits of double precision).

3.26. This problem considers solving a matrix equation using the Crout factorization.

(a) The algorithm for finding a Doolittle factorization of $\mathbf{A}$, assuming pivoting is not needed, is given in Table 3.1. Find a similar algorithm for the Crout factorization.

(b) Use your algorithm from part (a) to solve the matrix equation in the case of when $\mathbf{A}$ has diagonal entries $a_{ii} = 2$ and off-diagonal entries $a_{ij} = 1$. Also, $n = 1000$ and take $\mathbf{z} = \mathbf{Ax}$, where $\mathbf{x} = (1, 1, \cdots, 1)^T$. It is only necessary to report the values of $x(1)$ and $x(2)$ (to 16 digits). Also, report the computed value of $||\mathbf{r}||_\infty$ and $||\mathbf{e}||_\infty$.

(c) Using the same matrix as in part (b), use your algorithm to solve the matrix equation when $\mathbf{x} = (1, -1, 1, -1, \cdots, -1)^T$. It is only necessary to report the values of $x(1)$ and $x(2)$ (to 16 digits). Moreover, you must give a compelling explanation of why you believe you answer is correct (within the limits of double precision).

Chapter 4
Eigenvalue Problems

The problem considered in this chapter is: given an $n \times n$ matrix $\mathbf{A}$, find the number(s) λ and nonzero vectors $\mathbf{x}$ that satisfy

$$\mathbf{A}\mathbf{x} = \lambda \mathbf{x}. \tag{4.1}$$

This is an *eigenvalue problem*, where λ is an *eigenvalue* and $\mathbf{x}$ is an *eigenvector*. There are a couple of observations worth making about this problem. First, $\mathbf{x} = \mathbf{0}$ is always a solution of (4.1), and so what is of interest are the nonzero solutions. Second, if $\mathbf{x}$ is a solution, then $\alpha\mathbf{x}$, for any number α, is also a solution.

In linear algebra the procedure used to solve the eigenvalue problem consists of two steps:

1. Solve
$$\det(\mathbf{A} - \lambda \mathbf{I}) = 0, \tag{4.2}$$
where $\mathbf{I}$ is the identity matrix. This is known as the *characteristic equation*, and the left-hand side of this equation is an nth degree polynomial in λ.

2. For each eigenvalue λ, solve
$$(\mathbf{A} - \lambda \mathbf{I})\mathbf{x} = \mathbf{0}. \tag{4.3}$$

Note that Step 2 provides a way to determine an eigenvector once the eigenvalue is known. For some of the numerical methods used to solve (4.1), the eigenvectors are determined first. It is possible to determine the associated eigenvalue by multiplying both sides of (4.1) by an eigenvector $\mathbf{x}$. This yields the formula

$$\lambda = \frac{\mathbf{x} \cdot \mathbf{A}\mathbf{x}}{\mathbf{x} \cdot \mathbf{x}}\,, \tag{4.4}$$

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_4

which is known as *Rayleigh's quotient.* Also note that the "$\cdot$" appearing in this expression designates a dot product. As a reminder, if $\mathbf{x} = (x_1, x_2, \cdots, x_n)^T$ and $\mathbf{y} = (y_1, y_2, \cdots, y_n)^T$, then the dot product is defined as

$$\mathbf{x} \cdot \mathbf{y} \equiv x_1 y_1 + x_2 y_2 + \cdots + x_n y_n .$$

Examples

1. Consider the eigenvalue problem

$$\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \lambda \begin{pmatrix} x \\ y \end{pmatrix} . \tag{4.5}$$

The characteristic equation (4.2) is $\lambda^2 - 4\lambda + 3 = 0$, and so the eigenvalues are $\lambda_1 = 3$ and $\lambda_2 = 1$. For λ_1, (4.3) takes the form

$$\begin{pmatrix} -1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} .$$

From this it follows that the eigenvectors associated with this eigenvalue have the form $\mathbf{x} = \alpha \mathbf{x}_1$, where α is an arbitrary nonzero constant and

$$\mathbf{x}_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix} .$$

For λ_2, one finds that the eigenvectors have the form $\mathbf{x} = \beta \mathbf{x}_2$, where β is an arbitrary nonzero constant and

$$\mathbf{x}_2 = \begin{pmatrix} 1 \\ -1 \end{pmatrix} . \quad \blacksquare$$

2. Consider the eigenvalue problem

$$\begin{pmatrix} 3 & 0 \\ 0 & 3 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \lambda \begin{pmatrix} x \\ y \end{pmatrix} . \tag{4.6}$$

The characteristic equation is $(\lambda - 3)^2 = 0$, and so the only eigenvalue is $\lambda_1 = 3$. In this case, (4.3) takes the form

$$\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} .$$

All values of x and y satisfy this equation. It is possible to write this as $\mathbf{x} = \alpha\mathbf{x}_1 + \beta\mathbf{x}_2$, where α and β are arbitrary nonzero constants,

$$\mathbf{x}_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad \text{and} \quad \mathbf{x}_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \quad \blacksquare$$

3. Consider the eigenvalue problem

$$\begin{pmatrix} 1 & 2 \\ -\frac{1}{2} & 1 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \lambda \begin{pmatrix} x \\ y \end{pmatrix}.$$

The characteristic equation is $\lambda^2 - 2\lambda + 2 = 0$, and so the eigenvalues are $\lambda = 1 + i$ and $\lambda = 1 - i$. $\blacksquare$

4. The eigenvalue problem

$$\begin{pmatrix} 2 & 1 \\ 0 & 2 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \lambda \begin{pmatrix} x \\ y \end{pmatrix}$$

has only one eigenvalue $\lambda = 2$, and only one independent eigenvector. An $n \times n$ matrix that has fewer than n independent eigenvectors is said to be *defective*. So, the matrix of this example is defective, while the matrices for the three previous examples are not defective. $\blacksquare$

An important observation is that the first two matrices in the above examples are symmetric. They illustrate a result from linear algebra, which is stated next.

Theorem 4.1. *If* $\mathbf{A}$ *is a symmetric* $n \times n$ *matrix, then the following hold:*

1. *Its eigenvalues are real numbers.*
2. *If* $\mathbf{x}_i$ *and* $\mathbf{x}_j$ *are eigenvectors for different eigenvalues, then* $\mathbf{x}_i \cdot \mathbf{x}_j = 0$.
3. *It is possible to find a set of orthonormal basis vectors* $\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_n$, *where each* $\mathbf{u}_i$ *is an eigenvector for* $\mathbf{A}$.

In the last statement, for the vectors to be orthonormal it is required that $\mathbf{u}_i \cdot \mathbf{u}_j = 0$ if $i \neq j$ and $\mathbf{u}_i \cdot \mathbf{u}_i = 1$.

Example: Chain of Oscillators

Several applications involving eigenvalues and related ideas are considered in this chapter, including vibrating strings (Section 4.4.1), networks (Section 4.4.2), and image compression (Section 4.5.3). The particular example

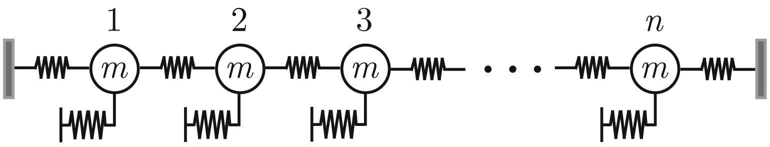


Figure 4.1 Chain of masses and springs.

described here involves a chain of oscillators as illustrated in Figure 4.1. What is shown in this figure are masses that are connected by springs, and each mass is also attached to its own spring (the other end of these springs is assumed to be held fixed). Letting $y_i(t)$ be the position of the ith mass, relative to its equilibrium location, then the resulting equation of motion is

$$my_i'' + ky_i = k_c(y_{i+1} - 2y_i + y_{i-1}), \text{ for } i = 1, 2, 3, \cdots, n \tag{4.7}$$

where $y_0 = y_{n+1} = 0$.

Although configuring masses and springs in this way looks like some complicated child's toy, this arises in numerous applications and has various names. A recent example is its use in the study of a chain of atoms that are subject to nearest neighbor interactions [Iooss and James, 2005]. However, the problem goes back centuries, to at least Bernoulli [1728], and has been used in a wide spectrum of applications, including the interactions of stars [Voglis, 2003] and the modeling of swarming motion [Erdmann et al., 2005]. It is also the basis of what is known as the Fermi-Pasta-Ulam (FPU) chain, which is used to study solitary waves [Ford, 1992; Berman and Izrailev, 2005].

The question considered here is, what are the time periodic solutions of this chain, what are called the normal modes of the system. This is answered by assuming $y_i(t) = x_i \exp(I\omega t)$, where $I = \sqrt{-1}$. Substituting this into (4.7), and writing the problem in matrix form, we obtain the eigenvalue equation $\mathbf{Ax} = \lambda\mathbf{x}$, where

$$\mathbf{A} = \begin{pmatrix} a & -1 & & & & \\ -1 & a & -1 & & 0 & \\ & -1 & a & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & 0 & & & & -1 \\ & & & & -1 & a \end{pmatrix}, \tag{4.8}$$

$a = 2 + k/k_c$ and $\lambda = m\omega^2/k_c$. Using the formula from Exercise 3.22, the eigenvalues of this matrix are

$$\lambda_i = a + 2\cos\left(\frac{i\pi}{n+1}\right), \quad \text{for } \; i = 1, 2, \ldots, n. \tag{4.9}$$

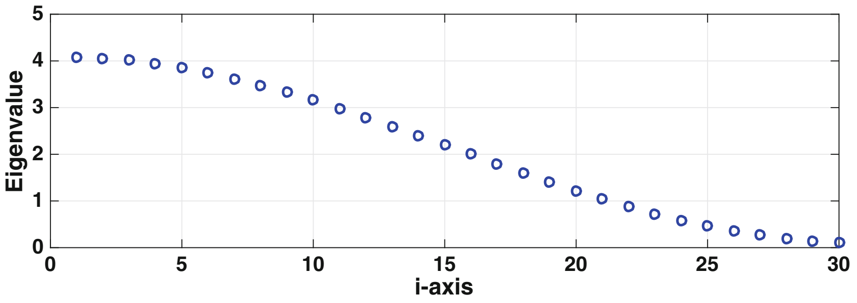


Figure 4.2 Eigenvalues of an oscillator chain in the case of when $n = 30$.

The values obtained from this formula in the case of when $k = 1$, $k_c = 1/10$, and $n = 30$ are shown in Figure 4.2. In the applications mentioned earlier, n is usually quite large, and as an example $n = 3000$ in James et al. [2013]. The eigenvalues in this case follow the curve seen in Figure 4.2, but they are much closer together. As will be explained later, this has a detrimental effect on how fast the eigenvalue solvers considered in this chapter are able to compute the eigenvalues. ■

We are now going to begin considering how to compute the solution of an eigenvalue problem. To be honest, depending on how many eigenvalues are going to be computed, this is not an easy task. The characteristic equation, as given in (4.2), involves a determinant, and these require on the order of $O(n^3)$ flops to compute. This is significant because the roots of (4.2) can be very sensitive to round-off error, and an illustration of this is given in Figure 1.1. What this means is that unless n is very small, we will not use (4.2) to determine the eigenvalues.

All of the numerical methods to be considered require, to prove they work, that the matrix is not defective. As you might recall, this means that the $n \times n$ matrix has n linearly independent eigenvectors. For some of the methods it is also necessary to include the additional requirement that the eigenvectors are orthogonal. According to Theorem 4.1, symmetric matrices satisfy this requirement. Symmetric matrices also have the benefit of being easy to determine, and they are also very common in applications. In comparison, defective matrices are rare.

4.1 Power Method

This method will provide a procedure for calculating what is known as the *dominant eigenvalue*. It comes from a simple observation. If you multiply $\mathbf{Ax} = \lambda\mathbf{x}$ by $\mathbf{A}$, you get $\mathbf{A}^2\mathbf{x} = \lambda\mathbf{Ax} = \lambda^2\mathbf{x}$. Multiplying by $\mathbf{A}$ again gives

$\mathbf{A}^3\mathbf{x} = \lambda^3\mathbf{x}$, and in general $\mathbf{A}^k\mathbf{x} = \lambda^k\mathbf{x}$. The observation is that as k increases the right-hand side will be dominated by the largest eigenvalue, or more precisely the one that is largest in absolute value. To illustrate how to take advantage of this we consider an example.

For the eigenvalue problem in (4.2), the matrix is

$$\mathbf{A} = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}. \tag{4.10}$$

To use the power method to find the largest eigenvalue $\lambda_1 = 3$, it's necessary to guess a nonzero starting vector $\mathbf{y}_0$. We found earlier that two linearly independent eigenvectors for this matrix are

$$\mathbf{x}_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad \text{and} \quad \mathbf{x}_2 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}.$$

These can be used as a basis, which means that for any choice of $\mathbf{y}_0$, it is possible to find α_1 and α_2 so that

$$\mathbf{y}_0 = \alpha_1\mathbf{x}_1 + \alpha_2\mathbf{x}_2. \tag{4.11}$$

With this,

$$\begin{aligned} \mathbf{y}_1 &= \mathbf{A}\mathbf{y}_0 \\ &= \alpha_1\mathbf{A}\mathbf{x}_1 + \alpha_2\mathbf{A}\mathbf{x}_2 \\ &= \alpha_1\lambda_1\mathbf{x}_1 + \alpha_2\lambda_2\mathbf{x}_2, \end{aligned}$$

and $\mathbf{y}_2 = \mathbf{A}\mathbf{y}_1 = \alpha_1\lambda_1^2\mathbf{x}_1 + \alpha_2\lambda_2^2\mathbf{x}_2$. At the kth step,

$$\begin{aligned} \mathbf{y}_k &= \mathbf{A}\mathbf{y}_{k-1} \\ &= \alpha_1\lambda_1^k\mathbf{x}_1 + \alpha_2\lambda_2^k\mathbf{x}_2 \\ &= \lambda_1^k\left(\alpha_1\mathbf{x}_1 + \alpha_2\omega^k\mathbf{x}_2\right), \end{aligned} \tag{4.12}$$

where $\omega = \lambda_2/\lambda_1 = 1/3$. After calculating $\mathbf{y}_k$, we can use the Rayleigh quotient in (4.4) to obtain an approximation v_k of the corresponding eigenvalue. The result is

$$\begin{aligned} v_k &= \frac{\mathbf{y}_k \cdot \mathbf{A}\mathbf{y}_k}{\mathbf{y}_k \cdot \mathbf{y}_k} \\ &= \frac{(\alpha_1\mathbf{x}_1 + \alpha_2\omega^k\mathbf{x}_2)\cdot(\alpha_1\lambda_1\mathbf{x}_1 + \alpha_2\omega^k\lambda_2\mathbf{x}_2)}{(\alpha_1\mathbf{x}_1 + \alpha_2\omega^k\mathbf{x}_2)\cdot(\alpha_1\mathbf{x}_1 + \alpha_2\omega^k\mathbf{x}_2)} \\ &= \lambda_1\frac{1 + \alpha^2\omega^{2k+1}}{1 + \alpha^2\omega^{2k}}, \end{aligned} \tag{4.13}$$

```
Pick:  random $\mathbf{y}$
       $tol > 0$
Let:   $\mathbf{z} = \mathbf{A}\mathbf{y}$
       $v_0 = (\mathbf{y} \cdot \mathbf{z})/\mathbf{y} \cdot \mathbf{y}$
Loop   For $k = 1, 2, 3, \cdots$
         $\mathbf{y} = \mathbf{z}/||\mathbf{z}||_2$
         $\mathbf{z} = \mathbf{A}\mathbf{y}$
         $v_k = \mathbf{y} \cdot \mathbf{z}$
         If $|v_k - v_{k-1}|/|v_k| < tol$ then stop
       End
```

Table 4.1 Power method for calculating the dominant eigenvalue of **A**. Note that v_k is the computed value for the eigenvalue.

where $\alpha = \alpha_2/\alpha_1$. Because $\omega < 1$, then $\omega^{2k} \to 0$ as $k \to \infty$. Therefore, from (4.13) we conclude that $v_k \to \lambda_1$ as $k \to \infty$.

The formula in (4.13) has some useful information related to how fast the method converges. To explain, we have that

$$v_k - \lambda_1 = c\lambda_1 \frac{\omega^{2k}}{1 + \alpha^2\omega^{2k}} .$$

where $c = \alpha^2(\omega - 1)$. This means that as k increases,

$$\frac{v_k - \lambda_1}{\lambda_1} \approx c\,\omega^{2k}. \tag{4.14}$$

Since this also means that $v_{k-1} - \lambda_1 \approx c\lambda_1\omega^{2(k-1)}$, it then follows that

$$v_k - \lambda_1 \approx \omega^2(v_{k-1} - \lambda_1). \tag{4.15}$$

This is a particularly useful result because it states that once v_k starts to get close to λ_1, the error $|v_k - \lambda_1|$ is approximately a factor of $|\lambda_2/\lambda_1|^2$ smaller than the previous error $|v_{k-1} - \lambda_1|$. Moreover, using the same approximation used to derive (4.15), it is possible to show that

$$v_k - v_{k-1} \approx \omega^2(v_{k-1} - v_{k-2}). \tag{4.16}$$

So, the iterative error $|v_k - v_{k-1}|$ is approximately a factor of $|\lambda_2/\lambda_1|^2$ smaller than the previous error $|v_{k-1} - v_{k-2}|$. In comparison, the convergence of the eigenvector is slower. In (4.12), the reduction of the $\mathbf{x}_2$ contribution decreases by a factor of $|\lambda_2/\lambda_1|$ with each iteration step.

Although the method can be used to find λ_1, there is a potential numerical problem with the formula for $\mathbf{y}_k$ in (4.12) because the term λ_1^k becomes very large as k increases. This can be avoided by scaling the vectors. Specifically, at the kth step, one first computes $\mathbf{z}_k = \mathbf{A}\mathbf{y}_{k-1}$, and then lets $\mathbf{y}_k = \mathbf{z}_k/||\mathbf{z}_k||_2$, where $||\mathbf{z}_k||_2 = \sqrt{\mathbf{z}_k \cdot \mathbf{z}_k}$. What this does is effectively removes the λ_1^k coefficient in (4.12). The resulting algorithm is given in Table 4.1. Note that the values of $\mathbf{y}$ and $\mathbf{z}$ are not indexed, but are overwritten as the procedure proceeds.

Example

As determined earlier, the eigenvalues of

$$\mathbf{A} = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$$

are $\lambda_1 = 3$ and $\lambda_2 = 1$. Applying the power method to $\mathbf{A}$, as given in Table 4.1, let $\mathbf{y}_0 = (-3, 2)^T$. In this case,

$$\mathbf{z}_0 = \mathbf{A}\mathbf{y}_0 = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}\begin{pmatrix} -3 \\ 2 \end{pmatrix} = \begin{pmatrix} -4 \\ 1 \end{pmatrix}$$

and

$$v_0 = \frac{\mathbf{y}_0 \cdot \mathbf{z}_0}{\mathbf{y}_0 \cdot \mathbf{y}_0} = \frac{14}{13} \approx 1.0769.$$

To improve on this, we calculate the following

$$\begin{aligned} \mathbf{y}_1 &= \frac{\mathbf{z}_0}{\sqrt{\mathbf{z}_0 \cdot \mathbf{z}_0}} = \frac{1}{\sqrt{17}}\begin{pmatrix} -4 \\ 1 \end{pmatrix}, \\ \mathbf{z}_1 &= \mathbf{A}\mathbf{y}_1 = \frac{1}{\sqrt{17}}\begin{pmatrix} -7 \\ -2 \end{pmatrix}, \\ v_1 &= \mathbf{y}_1 \cdot \mathbf{z}_1 = \frac{26}{17} \approx 1.5294. \end{aligned}$$

The next steps are calculated using MATLAB, with the results given in Table 4.2. It is evident that v_k is approaching the dominant eigenvalue $\lambda_1 = 3$. According to (4.15), as the iteration proceeds, the error should decrease by a factor $|\lambda_2/\lambda_1|^2 = 1/9 \approx 0.1111$. The same is true, according to (4.16), for the iterative error. To verify this, the ratios for these two error measures are also given in Table 4.2. Finally, the method also produces an eigenvector for λ_1, and from the MATLAB calculation it is found that $\mathbf{y}_7 = (-0.7087, -0.7055)^T$. The exact result is a multiple of $\mathbf{x}_1$, but the entries in $\mathbf{y}_7$ only agree to two digits. The error for the eigenvalue approximation v_7, in contrast, is about 3×10^{-6}. The reason for the poorer approximation

k	v_k	$\left\lvert\dfrac{v_k-\lambda_1}{v_{k-1}-\lambda_1}\right\rvert$	$\left\lvert\dfrac{v_k-v_{k-1}}{v_{k-1}-v_{k-2}}\right\rvert$
0	1.076923076923		
1	1.529411764706	7.65e−01	
2	2.528301886792	3.21e−01	2.21e+00
3	2.933687002653	1.41e−01	4.06e−01
4	2.992408138476	1.14e−01	1.45e−01
5	2.999153603954	1.11e−01	1.15e−01
6	2.999905920605	1.11e−01	1.12e−01
7	2.999989546297	1.11e−01	1.11e−01

Table 4.2 v_k is the value for the dominant eigenvalue computed using the power method applied to (4.10). The exact value is $\lambda_1 = 3$. Also given are the error ratios given (4.15) and (4.16), both of which approach $|\lambda_2/\lambda_1|^2 = 1/9 \approx 0.1111$.

for the eigenvector can be found in (4.12). When going from k to $k+1$, the contribution of $\mathbf{x}_2$ is removed by another factor of ω. The eigenvalue improvement, according to (4.14), is by a factor of ω^2. ■

As shown in (4.14), the speed at which the method converges depends on the ratio λ_2/λ_1. To investigate this observation, suppose we want the relative error in the eigenvalue to satisfy $|v_k - \lambda_1|/|\lambda_1| \le 10^{-4}$. From (4.14) this will happen if, approximately,

$$|c\,\omega^{2k}| \le 10^{-4},$$

where $\omega = \lambda_2/\lambda_1 < 1$. First, note that $c = (\omega - 1)(\alpha_2/\alpha_1)^2$, where α_1 and α_2 are the coefficients in (4.11). This shows that if you unfortunately pick a starting vector $\mathbf{y}_0$ with a small contribution from $\mathbf{x}_1$, so α_1 is small, then the power method will take a large number of iterations to converge. The exact number depends on the ratio λ_2/λ_1. To illustrate the sensitivity of the number on this ratio, suppose $c \approx 1$, so from the above inequality we get that

$$k \ge -\frac{2}{\log \omega}\,.$$

Since $\omega = 1/3$, this means $k > 4$. This is a small number of iteration steps, but this happens because the two eigenvalues are relatively far apart. As an example of when this is not the case, if $\lambda_1 = 1000$ and $\lambda_2 = 1001$, then $\omega = 1000/1001$ and we need $k > 4607$. This observation that the speed at which the method converges slows down if the eigenvalues are relatively close together will apply to all of the methods we will consider.

To conclude the introduction of the power method, it should be pointed out that there is a complication concerning what eigenvector the method produces. To explain, suppose $\lambda_1 = -3$ and $\lambda_2 = 1$. We still obtain (4.12), but because of the minus sign, the coefficient of $\mathbf{y}_k$ switches sign as k changes. This also happens for the normalized version of the method given in Table 4.1. In this case, the sequences $\mathbf{y}_0$, $\mathbf{y}_2$, $\mathbf{y}_4$, $\cdots$ and $\mathbf{y}_1$, $\mathbf{y}_3$, $\mathbf{y}_5$, $\cdots$ are both converging to an eigenvector for λ_1, but the vectors they converge to differ by a minus sign. This annoying $\pm$ problem only arises if the dominant eigenvalue is negative.

4.1.1 General Formulation

The derivation of the power method was illustrated using a specific 2×2 matrix, and to use it on general $n \times n$ matrices it is necessary to know what conditions are necessary so it will work. First, in (4.13) and (4.15) we used the inequality $|\lambda_2/\lambda_1| < 1$ to be able to conclude that $\omega^k \to 0$ as $k \to \infty$. For a more general matrix, suppose λ_1, λ_2, $\cdots$, λ_n are the eigenvalues of $\mathbf{A}$, and they are labeled so that $|\lambda_1| \geq |\lambda_2| \geq \cdots \geq |\lambda_n|$. This gives us the following definition.

Definition 4.1. λ_1 is the *dominant eigenvalue* of $\mathbf{A}$ if $|\lambda_1| > |\lambda_2|$.

The strict inequality in this definition is what enables us to guarantee that $|\lambda_i/\lambda_1|^k \to 0$ as $k \to \infty$ for $i = 2, 3, \cdots, n$. Also note that because absolute values are used here, the dominant eigenvalue is not necessarily the largest eigenvalue.

The second thing we needed was that the eigenvectors of $\mathbf{A}$ can be used as a basis. In the $n \times n$ case this means that there are n linearly independent eigenvectors. According to Theorem 4.1, this is guaranteed if $\mathbf{A}$ is symmetric.

The third, and final, requirement is that $\alpha_1 \neq 0$ in (4.12). In other words, the initial guess $\mathbf{y}_0$ must include a part of the eigenvector for the dominant eigenvalue.

To summarize the above discussion, we have the following theorem for the power method given in Table 4.1.

Theorem 4.2. *Assume that $\mathbf{A}$ is a nonzero symmetric $n \times n$ matrix, with a dominant eigenvalue λ_1. If the initial guess for $\mathbf{y}$ includes some part of an eigenvector for λ_1, then the v_k's converge to λ_1. Moreover, if the initial guess contains some portion of an eigenvector for λ_2, then the error decreases as follows:*

$$|v_k - \lambda_1| = \omega_k^2 |v_{k-1} - \lambda_1|, \tag{4.17}$$

and

$$|v_k - v_{k-1}| = \overline{\omega}_k^2 |v_{k-1} - v_{k-2}|, \tag{4.18}$$

where as $k \to \infty$, both ω_k and $\overline{\omega}_k$ approach

$$\left| \frac{\lambda_2}{\lambda_1} \right| . \tag{4.19}$$

The error in the associated eigenvector decreases as $O(\omega_k)$.

By using a random number (vector) generator to produce the initial guess for $\mathbf{y}$, as prescribed in Table 4.1, it is almost certain that it contains some part of an eigenvector for λ_1, as well as some portion of an eigenvector for λ_2. To explain, if $\mathbf{x}_i = (a_1, a_2, \cdots, a_n)^T$ is an eigenvector, and $\mathbf{y}_0 = (r_1, r_2, \cdots, r_n)^T$, then $\mathbf{y}_0$ contains no contribution from $\mathbf{x}_i$ when $\mathbf{y}_0 \cdot \mathbf{x}_i = 0$. This means that the randomly chosen numbers r_1, r_2, $\cdots$, r_n must be such that

$$r_1 a_1 + r_2 a_2 + \cdots + r_n a_n = 0.$$

The probability of picking n random numbers that sum to zero in this way is extremely low. However, in the very unlikely case of when $\mathbf{y}_0$ contains no contribution from an eigenvector for λ_2, but does contain a component from an eigenvector for λ_3, then the limiting value in (4.19) is replaced with $|\lambda_3/\lambda_1|$. Similar modifications are necessary for the other possible situations. In what follows, when discussing the convergence of the power method, it is assumed that (4.19) holds.

As stated earlier, (4.17) is an important result because it states that once v_k starts to get close to λ_1, the error $|v_k - \lambda_1|$ is approximately a factor of $|\lambda_2/\lambda_1|^2$ smaller than the previous error $|v_{k-1} - \lambda_1|$. The iterative error also decreases in this manner, and this is because of (4.18). A consequence of this is that the power method can converge very quickly if $|\lambda_2| \ll |\lambda_1|$, but it can also be very slow if $|\lambda_2|$ is not much different than $|\lambda_1|$.

The above theorem holds in the more general case of when $\mathbf{A}$ is an $n \times n$ matrix, with n linearly independent eigenvectors. The requirement is that the eigenvalue λ_1 with the largest magnitude is real-valued and $-\lambda_1$ is not also an eigenvalue. However, without the assumption of symmetry, the error in (4.17) and (4.18) is guaranteed to decrease as $O(\omega_k)$ rather than the stated $O(\omega_k^2)$. It is also possible to use shifting (which is explained in the next section) so the power method will work on any symmetric matrix (see Exercise 4.8).

Example

Suppose $\mathbf{A}$ is the $n \times n$ matrix

k	v_k	$\left\lvert\dfrac{v_k-\lambda_1}{v_{k-1}-\lambda_1}\right\rvert$	$\left\lvert\dfrac{v_k-v_{k-1}}{v_{k-1}-v_{k-2}}\right\rvert$
0	3.000300728189		
1	3.000633149675	1.00e+00	
2	3.001168862204	9.99e−01	1.61e+00
3	3.002095543148	9.99e−01	1.73e+00
4	3.003727977465	9.98e−01	1.76e+00
⋮	⋮	⋮	⋮
22	3.991584072355	5.66e−01	5.73e−01
23	3.995248545996	5.65e−01	5.68e−01
⋮	⋮	⋮	⋮
34	3.999991483933	5.63e−01	5.63e−01
35	3.999995209694	5.63e−01	5.63e−01

Table 4.3 Dominant eigenvalue $\lambda_1 = 4$ of (4.20) as computed using the power method. Also shown are the error ratios obtained from (4.17) and (4.18), both of which approach $|\lambda_2/\lambda_1|^2 \approx 0.5625$.

$$\mathbf{A} = \begin{pmatrix} a & & & & 1 \\ & a & & & \\ & & \ddots & & \\ & & & a & \\ 1 & & & & a \end{pmatrix}. \tag{4.20}$$

Specifically, if a_{ij} denotes the entries in $\mathbf{A}$, then $a_{ij} = 0$ except that $a_{ii} = a$ and $a_{n1} = a_{1n} = 1$. The eigenvalues of $\mathbf{A}$ are a, $a+1$, and $a-1$ (see Exercise 4.12). Taking $a = 3$, $n = 200$, and a random starting vector $\mathbf{y}_0$, the power method produces the results given in Table 4.3. For this matrix, the dominant eigenvalue is $\lambda_1 = 4$. The convergence is not as fast as in Table 4.2, and the reason is that $\lambda_2 = 3$, and $\lambda_2/\lambda_1 = 3/4$. According to (4.19), once the method starts to get close to the exact solution, the error should be reduced by a factor of $(3/4)^2 \approx 0.56$ at each step. This reduction holds for the error ratio coming from (4.17), given in the third column, and for the iterative error ratio coming from (4.18), given in the fourth column. ■

There are numerous variations of the power method, and one of the more interesting involves using probabilistic measures of the error, and this is discussed more in Kuczyński and Woźniakowski [1992].

4.2 Extensions of the Power Method

It is relatively easy to modify the power method to find some of the other eigenvalues and eigenvectors. This requires knowing the right formulas from linear algebra, and two we will now need are the following.

Theorem 4.3. *Suppose the eigenvalues of an $n \times n$ matrix $\mathbf{A}$ are λ_1, λ_2, $\cdots$, λ_n.*

1. *Setting $\mathbf{B} = \mathbf{A} - \omega\mathbf{I}$, where ω is a constant, then the eigenvalues of $\mathbf{B}$ are $\lambda_1 - \omega$, $\lambda_2 - \omega$, $\cdots$, $\lambda_n - \omega$. Also, if $\mathbf{x}_i$ is an eigenvector for $\mathbf{A}$ that corresponds to λ_i, then $\mathbf{x}_i$ is an eigenvector for $\mathbf{B}$ that corresponds to $\lambda_i - \omega$.*
2. *If $\mathbf{A}$ is invertible, then the eigenvalues of $\mathbf{A}^{-1}$ are $1/\lambda_1$, $1/\lambda_2$, $\cdots$, $1/\lambda_n$. Also, if $\mathbf{x}_i$ is an eigenvector for $\mathbf{A}$ that corresponds to λ_i, then $\mathbf{x}_i$ is an eigenvector for $\mathbf{A}^{-1}$ that corresponds to $1/\lambda_i$.*

4.2.1 Inverse Power Method

To find the smallest eigenvalue of $\mathbf{A}$, in absolute value, one can use the power method with $\mathbf{A}^{-1}$. The explanation why involves Statement 2 in Theorem 4.3. Suppose that the eigenvalues of $\mathbf{A}$ are λ_1, λ_2, $\cdots$, λ_{n-1}, λ_n and they satisfy

$$|\lambda_1| \ge |\lambda_2| \ge \cdots \ge |\lambda_{n-1}| > |\lambda_n| > 0.$$

With this, the eigenvalues of $\mathbf{A}^{-1}$ are λ_1^{-1}, λ_2^{-1}, $\cdots$, λ_{n-1}^{-1}, λ_n^{-1}, and they satisfy

$$\left|\frac{1}{\lambda_n}\right| > \left|\frac{1}{\lambda_{n-1}}\right| \ge \cdots \ge \left|\frac{1}{\lambda_1}\right| .$$

Consequently, $|1/\lambda_n|$ is the dominant eigenvalue of $\mathbf{A}^{-1}$ and it is determined by the eigenvalue of $\mathbf{A}$ that is closest to zero.

To use the power method with $\mathbf{A}^{-1}$, the two lines in Table 4.1 that state $\mathbf{z} = \mathbf{A}\mathbf{y}$ take the form $\mathbf{z} = \mathbf{A}^{-1}\mathbf{y}$. It is possible to avoid having to calculate the inverse matrix by realizing that one can find $\mathbf{z}$ by solving $\mathbf{A}\mathbf{z} = \mathbf{y}$. In this way, the LU factorization method can be used, and $\mathbf{A}$ only needs to be factored once. Also, in terms of the speed of convergence, according to (4.17)–(4.19), the error for a symmetric matrix is reduced each iteration step by a factor of about ω^2, where $\omega = \lambda_n/\lambda_{n-1}$.

```
Set:   $\mathbf{B} = \mathbf{A} - \mu\mathbf{I}$
Pick:  random $\mathbf{y}$
       $tol > 0$
Find:  $\mathbf{B} = \mathbf{LU}$
       $\mathbf{Bz} = \mathbf{y}$
Let:   $v_0 = (\mathbf{y}\cdot\mathbf{z})/\mathbf{y}\cdot\mathbf{y}$
Loop   For $k = 1, 2, 3, \cdots$
          $\mathbf{y} = \mathbf{z}/||\mathbf{z}||_2$
          $\mathbf{Bz} = \mathbf{y}$
          $v_k = \mathbf{y}\cdot\mathbf{z}$
          If $|v_k - v_{k-1}|/|v_k| < tol$ then stop
       End
```

Table 4.4 Inverse iteration for calculating the eigenvalue of $\mathbf{A}$ closest to μ. At completion the eigenvalue is $\lambda = \mu + 1/v_k$. Note that $\mathbf{Bz} = \mathbf{y}$ means that the equation is solved, using the known LU factorization, for $\mathbf{z}$.

4.2.2 Inverse Iteration

The method that is going to be described next is based on the following question: if we have an approximation for an eigenvalue, can we use this to speed up the convergence of the method used to compute the exact value? To answer this, it is assumed that the eigenvalues of $\mathbf{A}$ are $\lambda_1, \lambda_2, \cdots, \lambda_n$. Also, suppose we know that eigenvalue λ_i is approximately μ. More specifically, it is assumed that μ is closest to λ_i, and so $|\lambda_i - \mu| < |\lambda_j - \mu|$, for $j \neq i$. This means, according to Statement 1 in Theorem 4.3, that the eigenvalue of $\mathbf{B} = \mathbf{A} - \mu\mathbf{I}$ that is closest to zero is $\lambda_i - \mu$. Adapting the inverse power method described above to $\mathbf{B}$ we get the algorithm given in Table 4.4. The v_k's in this case are converging to $1/(\lambda_i - \mu)$, so at completion $\lambda_i = \mu + 1/v_k$. As for the speed of convergence, suppose that the second closest eigenvalue to μ is λ_j. According to (4.17)–(4.19), the error for a symmetric matrix is reduced each iteration step by a factor of about ω^2, where $\omega = (\lambda_i - \mu)/(\lambda_j - \mu)$. Consequently, if the approximation is good enough that $|\lambda_i - \mu| \ll |\lambda_j - \mu|$, then the method will be speeded up considerably.

One of the reasons for the popularity of inverse iteration is because there are efficient methods for finding eigenvalues that are not so efficient for finding eigenvectors (one of these will be considered in Section 4.3.3). With good approximations for the eigenvalues, inverse iteration provides a very good method for finding the eigenvectors.

Example

The eigenvalues of

$$\mathbf{A} = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} \tag{4.21}$$

are $\lambda_1 = 3$ and $\lambda_2 = 1$. Inverse iteration will be used to compute $\lambda_2 = 1$ by starting with the approximation of $\mu = 1/2$. The shifted matrix is then

$$\mathbf{B} = \mathbf{A} - \frac{1}{2}\mathbf{I} = \begin{pmatrix} 3/2 & 1 \\ 1 & 3/2 \end{pmatrix}.$$

Taking $\mathbf{y}_0 = (1,0)^T$, and solving $\mathbf{B}\mathbf{z}_0 = \mathbf{y}_0$, one finds that

$$\mathbf{z}_0 = \frac{2}{5}\begin{pmatrix} 3 \\ -2 \end{pmatrix}.$$

With this $v_0 = (\mathbf{y}_0 \cdot \mathbf{z}_0)/(\mathbf{y}_0 \cdot \mathbf{y}_0) = 6/5$, and this leads to the approximation

$$\lambda_1 \approx \mu + 1/v_0 = \frac{4}{3} \approx 1.3333.$$

To improve on this, we calculate

$$\mathbf{y}_1 = \frac{\mathbf{z}_1}{||\mathbf{z}_1||_2} = \frac{1}{\sqrt{13}}\begin{pmatrix} 3 \\ -2 \end{pmatrix}$$

k	λ_2	Error
0	1.333333333333	3.33e−01
1	1.015873015873	1.59e−02
2	1.000639795266	6.40e−04
3	1.000025599672	2.56e−05
4	1.000001023999	1.02e−06
5	1.000000040960	4.10e−08
6	1.000000001638	1.64e−09
7	1.000000000066	6.55e−11
8	1.000000000003	2.62e−12

Table 4.5 Result of using inverse iteration to calculate the eigenvalue $\lambda_2 = 1$ of (4.21), along with the corresponding error.

and, from solving $\mathbf{B}\mathbf{z}_1 = \mathbf{y}_1$,

$$\mathbf{z}_1 = \frac{2\sqrt{13}}{65}\begin{pmatrix} 13 \\ -12 \end{pmatrix}.$$

With this $v_1 = (\mathbf{y}_1{\cdot}\mathbf{z}_1)/(\mathbf{y}_1{\cdot}\mathbf{y}_1) = 126/65$, and this leads to the approximation

$$\lambda_1 \approx \mu + 1/v_1 = 64/63 \approx 1.01587.$$

The remaining approximations are calculated using MATLAB, with the results given in Table 4.5. ■

Example

The eigenvalues of the $n \times n$ matrix given in (4.20) are a, $a-1$, and $a+1$. Taking $a = 3$, and $n = 200$, the resulting values computed using the inverse iteration method are given in Table 4.6. Two shifts were tried, one was $\mu = 4.8$ and the second was $\mu = 4.4$. Comparing these to the values computed using the power method, which are given in Table 4.3, it is evident that shifting does indeed reduce the number of iteration steps. It is also evident that the better the shift the faster the method works. ■

k	λ_1 with $\mu = 4.8$	Error	λ_1 with $\mu = 4.4$	Error
0	3.023084747050		3.016681187905	
1	3.106937685868	8.93e−01	3.176228115668	8.24e−01
2	3.377432051969	6.23e−01	3.724004514014	2.76e−01
3	3.754250014560	2.46e−01	3.969823111025	3.02e−02
4	3.939532257759	6.05e−02	3.997466432162	2.53e−03
5	3.987446631422	1.26e−02	3.999792697636	2.07e−04
6	3.997495089407	2.50e−03	3.999983074187	1.69e−05
7	3.999504206314	4.96e−04	3.999998618281	1.38e−06
8	3.999902026471	9.80e−05	3.999999887207	1.13e−07
9	3.999980645683	1.94e−05	3.999999990792	9.21e−09
10	3.999996176866	3.82e−06	3.999999999248	7.52e−10

Table 4.6 Dominant eigenvalue $\lambda_1 = 4$ of (4.20) as computed using the inverse iteration method given in Table 4.4, using two different shifts.

One of the interesting aspects of inverse iteration is that you could reasonably expect that it will fail if you have a very good estimate of the eigenvalue. The reason is that the better the guess, the more ill-conditioned the matrix $\mathbf{B} = \mathbf{A} - \mu\mathbf{I}$ becomes. In fact, at one time it was actually advised not to use an accurate shift to avoid producing an ill-conditioned matrix. However, good shifts do not cause the method to fail, and an explanation can be found in Peters and Wilkinson [1979].

4.2.3 Rayleigh Quotient Iteration

A second reason why inverse iteration is important is because of a method derived from it. This modification involves improving the shift as the iteration proceeds. To explain, let $\mathbf{y}_0$ be a starting vector. Using the Rayleigh's quotient (4.4), an approximate eigenvalue is

$$\mu_0 = \frac{\mathbf{y}_0 \cdot \mathbf{A}\mathbf{y}_0}{\mathbf{y}_0 \cdot \mathbf{y}_0}\,.$$

This provides the first shift, and we take $\mathbf{B}_0 = \mathbf{A} - \mu_0\mathbf{I}$, and from this one finds $\mathbf{z}_0$ by solving $\mathbf{B}_0\mathbf{z}_0 = \mathbf{y}_0$. As usual, this is normalized to produce $\mathbf{y}_1 = \mathbf{z}_0/||\mathbf{z}_0||_2$. This enables us to find a somewhat better approximation for the eigenvalue, which is

$$\mu_1 = \frac{\mathbf{y}_1 \cdot \mathbf{A}\mathbf{y}_1}{\mathbf{y}_1 \cdot \mathbf{y}_1}\,.$$

This brings us to the better shifted matrix $\mathbf{B}_1 = \mathbf{A} - \mu_1\mathbf{I}$. This process of using inverse iteration, but improving the approximation for the shift, is known as Rayleigh quotient iteration, and it is summarized in Table 4.7.

The usual criticism of the Rayleigh quotient iteration is that a new LU factorization is needed at each iteration step to find $\mathbf{z}$. In contrast, the shift in the inverse iteration method does not change, so only one LU is required. Although this is potentially a drawback, as will be demonstrated in the example below, Rayleigh converges so quickly that this is often not a particular issue.

Example

The eigenvalues of the $n \times n$ matrix given in (4.20) are a, $a-1$, and $a+1$. Taking $a = 3$, and $n = 200$, the resulting values computed using the Rayleigh quotient iteration are shown in Table 4.8. The contrast between this and what is obtained using the power method on the same matrix, which is given in Table 4.3, is striking. It is also much faster than the inverse iteration results given in Table 4.6. It is possible to prove that the order of convergence, which

```
Pick:  random $\mathbf{y}$ with $||\mathbf{y}||_2 = 1$
       $tol > 0$
Let:   $v_0 = \mathbf{y} \cdot \mathbf{A}\mathbf{y}$
Loop   For $k = 1, 2, 3, \cdots$
           $\mathbf{B} = \mathbf{A} - v_{k-1}\mathbf{I}$
           $\mathbf{B}\mathbf{z} = \mathbf{y}$
           $\mathbf{y} = \mathbf{z}/||\mathbf{z}||_2$
           $v_k = \mathbf{y} \cdot \mathbf{z}$
           If $|v_k - v_{k-1}|/|v_k| < tol$ then stop
       End
```

Table 4.7 Rayleigh quotient iteration for calculating an eigenvalue of $\mathbf{A}$. Note that $\mathbf{Bz} = \mathbf{y}$ means that the equation is solved for $\mathbf{z}$.

is explained in Section 2.4.1, is $\gamma = 3$. This means that if the error at step k is 10^{-m}, then at the next step you should expect the error will be about 10^{-3m}. The computed values for γ, using (2.15), are given in Table 4.8. The difficulty with this is that the convergence is so fast that you quickly reach the resolution possible with double precision. The consequence is that you get the predicted value at $k = 2$, but after that improvement in the answer is not possible and the computed values for γ are approximately equal to one. ■

As seen in the above example, the Rayleigh quotient iteration can be third order. The proof of this requires the assumption that the matrix is symmetric or normal [Parlett, 1998]. There has been an effort to extend the method to

k	v_k	$\|v_k - \lambda\|$	γ
0	0.050740429652		
1	3.000515277178	5.15e−04	−7.42e−01
2	3.000000000139	1.39e−10	3.00e+00
3	3.000000000000	2.66e−15	1.48e+00
4	3.000000000000	1.78e−15	1.01e+00

Table 4.8 Eigenvalue computed using Rayleigh quotient iteration for matrix (4.20), with $a = 3$. Also shown are the error and the computed value for the order of convergence.

nonnormal matrices [Parlett, 1974], but it is known that the method can fail if the matrix is not symmetric [Batterson and Smillie, 1989].

Also, nothing has been said about which eigenvalue the Rayleigh quotient iteration converges to. In the above example, it converged to the middle eigenvalue, versus the dominant one or the one closest to zero. Presumably, the one that appears depends on the eigenvalue representation in the starting vector, and which eigenvalue the approximate shifts are closest to as the iteration proceeds. Those interested in pursuing this question should consult Beattie and Fox [1989] and Pantazis and Szyld [1995].

4.3 Calculating Multiple Eigenvalues

Having derived methods for finding particular eigenvalues, we now consider how to calculate several of them at the same time. In this discussion it is assumed $\mathbf{A}$ is a symmetric $n \times n$ matrix, so Theorem 4.1 applies. It is also assumed that if λ is a nonzero eigenvalue, then $-\lambda$ is not an eigenvalue.

Suppose the power method has been used to calculate the dominant eigenvalue λ_1, and an associated eigenvector $\mathbf{x}_1$. The question arises if it is possible to calculate λ_2 in a similar way. It is, and to explain how, consider the example used to introduce the power method in Section 4.1. We begin, as before, and guess a nonzero starting vector $\mathbf{y}_0$. Also, as in (4.11), it is possible to write $\mathbf{y}_0 = \alpha_1\mathbf{x}_1 + \alpha_2\mathbf{x}_2$, where, because the eigenvectors are orthogonal,

$$\alpha_1 = \frac{\mathbf{x}_1 \cdot \mathbf{y}_0}{\mathbf{x}_1 \cdot \mathbf{x}_1} \quad \text{and} \quad \alpha_2 = \frac{\mathbf{x}_2 \cdot \mathbf{y}_0}{\mathbf{x}_2 \cdot \mathbf{x}_2} .$$

If we use the power method without modification, then the $\mathbf{x}_1$ part of $\mathbf{y}_0$ will grow and eventually lead us to the dominant eigenvalue. The modification we will make to prevent this is to remove this term from $\mathbf{y}_0$ by taking $\mathbf{w}_0 = \mathbf{y}_0 - \alpha_1\mathbf{x}_1$. The next step is as before,

$$\begin{aligned} \mathbf{y}_1 &= \mathbf{A}\mathbf{w}_0 \\ &= \alpha_2\lambda_2\mathbf{x}_2 . \end{aligned}$$

By subtracting out the $\mathbf{x}_1$ components of the vectors we will produce a power method that will converge to λ_2. More precisely, it will converge to λ_2 for this example. The qualification is needed because for some matrices it produces a different result. To illustrate, for the eigenvalue equation in (4.6) there is one eigenvalue and two linearly independent eigenvectors. The first application of the power method will produce the eigenvalue $\lambda = 3$ and an associated eigenvector. The second application, using the above modification, will produce the same eigenvalue but it will now produce a second eigenvector that is perpendicular to the first.

In principle, the method used to find λ_2 can be generalized to compute all of the eigenvalues of a symmetric $n \times n$ matrix, one at a time. There are problems with doing this because round-off error for the first eigenvalues can significantly affect the accuracy when calculating the last few eigenvalues. How to fix this problem is considered next.

4.3.1 Orthogonal Iteration

It is possible to modify the power method and calculate all, or just a few, of the eigenvalues simultaneously. To use our earlier example, which is given in (4.5), to find λ_1 and λ_2, we need start-off guesses $\mathbf{y}_0$ (for λ_1) and $\mathbf{z}_0$ (for λ_2). Individually, the power method would then calculate $\mathbf{y}_1 = \mathbf{A}\mathbf{y}_0$ and $\mathbf{z}_1 = \mathbf{A}\mathbf{z}_0$. These can be calculated together by writing

$$\mathbf{B}_1 = \mathbf{A}\mathbf{B}_0, \tag{4.22}$$

where $\mathbf{B}_1 = (\mathbf{y}_1 \;\; \mathbf{z}_1)$ is the matrix with column vectors $\mathbf{y}_1$ and $\mathbf{z}_1$, and $\mathbf{B}_0 = (\mathbf{y}_0 \;\; \mathbf{z}_0)$ with column vectors $\mathbf{y}_0$ and $\mathbf{z}_0$. We also know that we need to subtract out the $\mathbf{x}_1$ contribution to $\mathbf{z}_0$. Unlike before, we do not know $\mathbf{x}_1$ but we do know that as the method proceeds the first column of $\mathbf{B}_k = (\mathbf{y}_k \;\; \mathbf{z}_k)$ will converge to $\mathbf{x}_1$ (or a multiple of it). We would also like the second column to converge to a multiple of $\mathbf{x}_2$. To help make this happen, we will force the columns of $\mathbf{B}_k$ to have the same properties as the eigenvectors, which is that they are orthonormal. So, given $\mathbf{y}_0$ and $\mathbf{z}_0$, we calculate orthogonal vectors $\mathbf{e}_1$ and $\mathbf{e}_2$ as follows:

$$\begin{aligned} \mathbf{e}_1 &= \mathbf{y}_0, \\ \mathbf{e}_2 &= \mathbf{z}_0 - \frac{\mathbf{z}_0 \cdot \mathbf{e}_1}{\mathbf{e}_1 \cdot \mathbf{e}_1}\, \mathbf{e}_1. \end{aligned} \tag{4.23}$$

In the power method we normalized the iteration vectors, and we will do the same here. In other words, we do the following:

$$\begin{aligned} \mathbf{q}_1 &= \mathbf{e}_1/||\mathbf{e}_1||, \\ \mathbf{q}_2 &= \mathbf{e}_2/||\mathbf{e}_2||. \end{aligned}$$

The procedure used to turn $\mathbf{y}_0$ and $\mathbf{z}_0$ into $\mathbf{q}_0$ and $\mathbf{q}_1$ is known as the *Gram-Schmidt* method for orthonormalizing a set of vectors (this is described in more detail later). Having done this, then instead of (4.22), we have that

$$\mathbf{B}_1 = \mathbf{A}\mathbf{Q}_0, \tag{4.24}$$

where $\mathbf{Q}_0 = (\mathbf{q}_0 \;\; \mathbf{q}_1)$ and $\mathbf{B}_1 = (\mathbf{y}_1 \;\; \mathbf{z}_1)$. Now Gram-Schmidt must be applied to $\mathbf{y}_1$ and $\mathbf{z}_1$ to find $\mathbf{Q}_1$. Once this is done, then $\mathbf{B}_2 = \mathbf{A}\mathbf{Q}_1$. The other steps

```
Pick:  random $\mathbf{B}_0$
       $\mathbf{Q}_0 = GS(\mathbf{B}_0)$
Loop   For $k = 1, 2, 3, \cdots$
         $\mathbf{B}_k = \mathbf{A}\mathbf{Q}_{k-1}$
         $\mathbf{Q}_k = GS(\mathbf{B}_k)$
       End
```

Table 4.9 Orthogonal iteration for finding the eigenvalues of a symmetric matrix $\mathbf{A}$. Note $\mathbf{Q}_k = GS(\mathbf{B}_k)$ means that Gram-Schmidt is applied to the columns of $\mathbf{B}_k$ to produce $\mathbf{Q}_k$. Also, $\mathbf{B}_0$ is an $n \times m$ matrix, where $1 \leq m \leq n$.

in the iteration are computed in a similar manner, and the procedure we have derived is known as *orthogonal iteration.*

The algorithm for orthogonal iteration, for an $n \times n$ symmetric matrix $\mathbf{A}$, is given in Table 4.9. To guarantee that the method works, it is required that if λ_i is a nonzero eigenvalue then $-\lambda_i$ is not an eigenvalue, and the columns of $\mathbf{B}_0$ must be independent. As the method converges, the columns of $\mathbf{Q}_k$ serve as approximations for the eigenvectors for $\mathbf{A}$, and the corresponding eigenvalues can be determined using Rayleigh's quotient (4.4). Note that given the ith and $i+1$st column vectors of $\mathbf{Q}_k$, the associated eigenvalues satisfy $|\lambda_{i+1}| \leq |\lambda_i|$. Also, this procedure has the same annoying $\pm$ eigenvector property that the power method has. Namely, for any negative eigenvalue, the associated column in $\mathbf{Q}_k$ contains a $(-1)^k$, and so it alternates between positive and negative.

It's important to point out that it's not required that $\mathbf{B}_0$ have n column vectors. If $\mathbf{B}_0$ has m column vectors, with $1 \leq m \leq n$, then $\mathbf{Q}_k$ has m column vectors. Therefore, orthogonal iteration provides a way to compute some, or all, of the eigenvalues of a symmetric matrix.

Example

To use orthogonal iteration to calculate the eigenvalues of

$$\mathbf{A} = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$$

take as a starting matrix

$$\mathbf{B}_0 = \begin{pmatrix} -1 & 2 \\ 3 & -1 \end{pmatrix}.$$

k	λ_1	λ_2
0	1.40000000	2.60000000
1	2.38461538	1.61538462
2	2.90588235	1.09411765
3	2.98908595	1.01091405
4	2.99878142	1.00121858
5	2.99986453	1.00013547
6	2.99998495	1.00001505
7	2.99999833	1.00000167
8	2.99999981	1.00000019

Table 4.10 Calculation of the eigenvalues $\lambda_1 = 3$ and $\lambda_2 = 1$ of (4.5) using the orthogonal iteration given in Table 4.9.

To apply Gram-Schmidt to the columns of this matrix, we use (4.23) and set

$$\mathbf{e}_1 = \begin{pmatrix} -1 \\ 3 \end{pmatrix} \quad \text{and} \quad \mathbf{e}_2 = \begin{pmatrix} 2 \\ -1 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} -1 \\ 3 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 3 \\ 1 \end{pmatrix} .$$

Normalizing these vectors we obtain the orthogonal matrix $\mathbf{Q}_0 = (\mathbf{q}_1 \;\; \mathbf{q}_2)$, where

$$\mathbf{q}_1 = \frac{1}{\sqrt{10}}\begin{pmatrix} -1 \\ 3 \end{pmatrix} \quad \text{and} \quad \mathbf{q}_2 = \frac{1}{\sqrt{10}}\begin{pmatrix} 3 \\ 1 \end{pmatrix} .$$

The corresponding approximations for the eigenvalues are

$$\lambda_1 \approx \mathbf{q}_1 \cdot \mathbf{A}\mathbf{q}_1 = \frac{7}{5} = 1.4$$

and

$$\lambda_2 \approx \mathbf{q}_2 \cdot \mathbf{A}\mathbf{q}_2 = \frac{13}{5} = 2.6$$

The next step is to compute $\mathbf{B}_1 = \mathbf{A}\mathbf{Q}_0$, and then repeat what was done above. These steps are calculated using MATLAB, with the results given in Table 4.10. As expected, once the answer gets close to the solution, the error for λ_1 drops by a factor of about $|\lambda_2/\lambda_1|^2 = 1/9$ with each iteration step. ■

Example

The eigenvalues of the $n \times n$ matrix in (4.20) are a, $a+1$ and $a-1$ (for $n \geq 3$). Moreover, there are $n-2$ linearly independent eigenvectors for a. Therefore,

k	λ_1	λ_2	λ_2	λ_3
1	1.45774533	1.00017912	1.00039921	4.93e−32
2	1.77151254	1.00012718	1.00022273	2.47e−32
3	1.93106506	1.00004630	1.00007248	1.23e−32
4	1.98182668	1.00001287	1.00001947	2.94e−33
5	1.99539389	1.00000331	1.00000496	1.14e−34
6	1.99884448	1.00000083	1.00000124	1.23e−32
7	1.99971087	1.00000021	1.00000031	1.23e−32

Table 4.11 Calculation of the eigenvalues of (4.20), when $a = 1$ and $n = 10$ using orthogonal iteration. Note the exact values are $\lambda_1 = 2$, $\lambda_2 = 1$, and $\lambda_3 = 0$.

taking $a = 1$, then the first column of $\mathbf{Q}_k$ converges to an eigenvector for $\lambda_1 = 2$, the next $n - 2$ columns converge to eigenvectors for $\lambda_2 = 1$, and the last column converges to an eigenvector for $\lambda_3 = 0$. The resulting values for the eigenvalues computed using the Rayleigh quotient, in the case of when $n = 10$, are given in Table 4.11. The value for λ_2 in the third column is computed using the second column of $\mathbf{Q}_k$ and the λ_2 in column four is computed using the seventh column of $\mathbf{Q}_k$. Similar values for λ_2 are obtained for the other six columns. ■

The most challenging step using orthogonal iteration is calculating the $\mathbf{Q}_k$'s. As introduced, these can be determined using the Gram-Schmidt process and this is discussed in more depth below. There is another method for finding these matrices, using what is known as a QR factorization. This is explained in the section following this one.

4.3.1.1 Regular and Modified Gram-Schmidt

To use orthogonal iteration, as given in Table 4.9, it is necessary to use the Gram-Schmidt process. To write down this procedure, assume $\mathbf{B}$ has m linearly independent column vectors $\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_m$, where $1 \leq m \leq n$. The first step is to construct m orthogonal vectors from the $\mathbf{c}_i$'s, and this is done as follows:

$$\begin{aligned}
\mathbf{e}_1 &= \mathbf{c}_1 \\
\mathbf{e}_2 &= \mathbf{c}_2 - \frac{\mathbf{c}_2 \cdot \mathbf{e}_1}{\mathbf{e}_1 \cdot \mathbf{e}_1}\mathbf{e}_1 \\
\mathbf{e}_3 &= \mathbf{c}_3 - \frac{\mathbf{c}_3 \cdot \mathbf{e}_1}{\mathbf{e}_1 \cdot \mathbf{e}_1}\mathbf{e}_1 - \frac{\mathbf{c}_3 \cdot \mathbf{e}_2}{\mathbf{e}_2 \cdot \mathbf{e}_2}\mathbf{e}_2 \\
&\vdots \\
\mathbf{e}_m &= \mathbf{c}_m - \sum_{i=1}^{m-1} \frac{\mathbf{c}_m \cdot \mathbf{e}_i}{\mathbf{e}_i \cdot \mathbf{e}_i}\mathbf{e}_i \,.
\end{aligned} \tag{4.25}$$

The vectors are now normalized by setting $\mathbf{q}_i = \mathbf{e}_i/||\mathbf{e}_i||$, and from this we have that $GP(\mathbf{B}) = \mathbf{Q}$, where $\mathbf{Q} = (\mathbf{q}_1 \;\; \mathbf{q}_2 \;\; \cdots \;\; \mathbf{q}_m)$.

The formulas in (4.25) are referred to as regular Gram-Schmidt, and they are what is usually given in a linear algebra course. In terms of computing, however, the procedure can be sensitive to round-off error for larger values of n. To explain, due to round-off, $\mathbf{e}_2$ will not be exactly perpendicular to $\mathbf{e}_1$. Since $\mathbf{e}_2$ appears in all of the subsequent formulas, this error affects the orthogonality for the remaining $\mathbf{e}_j$'s. The same can be said for the consequences of round-off error made with every $\mathbf{e}_j$.

An alternative is to use what is known as *modified Gram-Schmidt*, which reduces, but does not eliminate, the sensitivity. To explain how this is done, (4.25) calculates $\mathbf{e}_j$ using the following steps (for $j > 1$):

```
Set $\mathbf{e}_j = \mathbf{c}_j$
For $k = 1, 2, \cdots, j-1$
    $r = \mathbf{c}_j \cdot \mathbf{e}_k / \mathbf{e}_k \cdot \mathbf{e}_k$
    $\mathbf{e}_j = \mathbf{e}_j - r\mathbf{e}_k$
End
```

For the modified Gram-Schmidt method, one instead does the following:

```
Set $\mathbf{e}_j = \mathbf{c}_j$
For $k = 1, 2, \cdots, j-1$
    $r = \mathbf{e}_j \cdot \mathbf{e}_k / \mathbf{e}_k \cdot \mathbf{e}_k$
    $\mathbf{e}_j = \mathbf{e}_j - r\mathbf{e}_k$
End
```

Note that when using exact arithmetic, these two procedures produce the same result. However, in the regular version, orthogonality is obtained by removing the parts of $\mathbf{c}_j$ coming from the earlier computed $\mathbf{e}_k$'s. In the modified version, orthogonality is obtained by removing $\mathbf{e}_k$ from the currently computed value of $\mathbf{e}_j$.

n	GS Error	MGS Error	$\kappa(\mathbf{B})$
Random			
2	1.67e−16	1.67e−16	3.25e+00
4	1.72e−15	1.22e−15	1.86e+01
8	6.64e−15	9.44e−16	5.33e+01
32	9.21e−13	2.30e−14	4.14e+03
128	1.60e−11	1.90e−13	1.40e+04
512	8.26e−11	1.43e−12	1.07e+05
1024	3.85e−10	1.47e−12	1.02e+05
Vander			
2	4.44e−16	4.44e−16	8.00e+00
4	2.90e−13	2.05e−14	1.55e+03
8	8.36e−02	2.90e−09	4.52e+08
12	1.00e+00	1.31e−02	1.06e+15
16	1.00e+00	6.19e−01	2.55e+18

Table 4.12 Values of the dot product error for the Gram-Schmidt (GS) and modified Gram-Schmidt (MGS) on two different matrices.

Example

It is of interest to see just what sort of improvement is obtained when using modified Gram-Schmidt, and a comparison is given in Table 4.12. To explain how the error is computed, given an $n \times n$ matrix $\mathbf{B}$, Gram-Schmidt is applied to its columns to compute $\mathbf{Q}$. After this, the dot products $\mathbf{q}_j \cdot \mathbf{q}_k$, where $\mathbf{q}_j$ and $\mathbf{q}_k$ are the columns of $\mathbf{Q}$ with $j \neq k$, are computed. If exact arithmetic is done, then these should all be zero. What is reported in Table 4.12 is the largest value of the dot products in absolute value. The same thing was done using the modified Gram-Schmidt procedure. Also, two matrices were tried, one was obtained using random numbers while the second is the Vandermonde matrix. The conclusion drawn from this is that the two methods produce similar results for the matrix that is well conditioned. On the badly conditioned matrix, both fail but the modified Gram-Schmidt procedure does not fail as quickly as the condition number increases. It is possible to find better modifications of Gram-Schmidt, and those interested in this should consult Giraud et al. [2005]. ■

4.3.2 QR Factorization

It is worth reconsidering the relationship between $\mathbf{B}$ and $\mathbf{Q}$ in orthogonal iteration (see Table 4.9). It's assumed, to get things started, that $\mathbf{B}$ is an $n \times n$ matrix with independent column vectors. Although it was not stated earlier, it is possible to write

$$\mathbf{B} = \mathbf{QR}, \tag{4.26}$$

where $\mathbf{R}$ is an upper triangular matrix. As an example, suppose

$$\mathbf{B} = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}.$$

Applying Gram-Schmidt to the columns of this matrix, one finds that

$$\mathbf{Q} = \frac{1}{\sqrt{5}} \begin{pmatrix} 1 & 2 \\ 2 & -1 \end{pmatrix}.$$

To find $\mathbf{R}$, we solve (4.26) to obtain

$$\begin{aligned} \mathbf{R} &= \mathbf{Q}^{-1}\mathbf{B} \\ &= \frac{1}{\sqrt{5}} \begin{pmatrix} 5 & 4 \\ 0 & 3 \end{pmatrix}. \end{aligned} \tag{4.27}$$

The verification that $\mathbf{R}$ is, in general, upper triangular is easy to show using the triangular form of the equations involved with Gram-Schmidt, as given in (4.25).

Nothing was said about $\mathbf{R}$ earlier because it was not needed in the derivation of the orthogonal iteration method. However, the formula in (4.26) is significant, and it is an example of what is called a *QR factorization*. It is reminiscent of the LU factorization used to solve a matrix equation. However, the matrix $\mathbf{Q}$ is not necessarily lower triangular, and instead it has the property of being an orthogonal matrix. To make this clear, we have the following definition.

Definition 4.2. An *orthogonal matrix* is a square matrix whose columns are orthonormal vectors.

Some of the more important properties of an orthogonal matrix are listed in the next theorem.

Theorem 4.4. *Suppose* $\mathbf{Q}$ *is an* $n \times n$ *matrix.*

1. $\mathbf{Q}$ *is an orthogonal matrix if and only if* $\mathbf{Q}^{-1} = \mathbf{Q}^T$.
2. $\mathbf{Q}$ *is an orthogonal matrix if and only if its rows are orthonormal vectors.*
3. *If* $\mathbf{Q}$ *is an orthogonal matrix, then* $|\det(\mathbf{Q})| = 1$.

The above theorem is useful when computing a QR factorization. To explain, once $\mathbf{Q}$ is determined, then $\mathbf{R}$ is computed using the formula

$$\mathbf{R} = \mathbf{Q}^T\mathbf{B}.$$

This follows, because if $\mathbf{B} = \mathbf{QR}$, then $\mathbf{R} = \mathbf{Q}^{-1}\mathbf{B} = \mathbf{Q}^T\mathbf{B}$. It is not difficult to show that $\mathbf{R}$ is an upper triangular matrix.

For us, an important question is, given $\mathbf{B}$, how do you compute $\mathbf{Q}$? Assuming the matrix has independent columns, the straightforward approach is to apply Gram-Schmidt to the columns as described in the previous section. There are other possibilities, and the more prominent is a method using what are called Householder transformations, and another method using Givens rotations [Golub and Van Loan, 2013; Higham, 2002]. Both of these methods have the advantage that they work even if the matrix does not have n independent column vectors. However, finding a computationally efficient method requires some unique challenges and this is discussed at the end of the next section.

It is possible to find those who recommend using the QR factorization to solve matrix equations, as compared to the LU method considered in Chapter 3 [Trefethen and Bau, 1997]. One of the reasons used to argue against doing this is that QR requires approximately twice the flops LU takes. To investigate this, the computing times for these factorizations are given in Table 4.13 using MATLAB's commands for these factorizations (using version R2016a). Included in this comparison is another factorization known as the SVD, which is explained in Section 4.5. The fact that QR takes longer than the expected factor of two probably has to do with the need to manipulate large matrices using QR, and this adds to the computational overhead needed to carry out the method.

n	LU (sec)	QR	SVD
200	0.0003	2.6	23.4
400	0.0009	2.9	27.9
600	0.0025	2.9	24.1
800	0.0050	2.8	23.5
1000	0.0094	2.8	23.1
2000	0.1067	1.8	24.1
4000	0.4107	3.3	34.7

Table 4.13 Computing time for an LU, QR, and SVD factorization of a random $n \times n$ matrix using MATLAB. The times for QR and SVD are in terms of multiples of the time the LU takes for that value of n.

As a final comment, the QR factorization is not limited to square matrices. It is straightforward to generalize the factorization to any matrix which has more rows than columns. This makes it useful for the orthogonal iteration method described earlier. It is also useful for certain least squares problems, and this is discussed in more detail in Section 8.3.2.2. Those interested in a more expanded discussion of the QR factorization should consult Bjöurck [2004] or Higham [2002].

4.3.3 The QR Method

In the case of when it is necessary to compute all of the eigenvalues of a symmetric matrix, the following procedure can be used:

```
Set $\mathbf{C}_0 = \mathbf{A}$
For $k = 0, 1, 2, \cdots$
    $\mathbf{Q}_k\mathbf{R}_k = \mathbf{C}_k$          % find $\mathbf{Q}_k$ and $\mathbf{R}_k$
    $\mathbf{C}_{k+1} = \mathbf{R}_k\mathbf{Q}_k$      % calculate $\mathbf{C}_{k+1}$
End
```

This is a strange looking result in that it states after factoring $\mathbf{C}_k$, you then multiply the factors in reverse order to calculate $\mathbf{C}_{k+1}$. Although it is not obvious, it is possible to prove that if the method converges, then $\mathbf{C}_k$ approaches a diagonal matrix, and the diagonals are the eigenvalues of $\mathbf{A}$. The eigenvalues will appear on the diagonal according to how many independent eigenvectors they have (i.e., their geometric multiplicity). To guarantee that the QR method converges, it is required that for any nonzero eigenvalue λ, $-\lambda$ is not also an eigenvalue. This procedure is called the *QR method.* A proof of convergence, including the extension of the method to nonsymmetric matrices, can be found in Watkins [2008] and Golub and Van Loan [2013]. However, some insight into how the method works is given in Exercise 4.17.

Example

Consider the eigenvalue problem in (4.5), where

$$\mathbf{A} = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}.$$

Applying Gram-Schmidt to the columns of $\mathbf{C}_0 = \mathbf{A}$, as given in (4.25),

$$\mathbf{e}_1 = \begin{pmatrix} 2 \\ 1 \end{pmatrix} \quad \text{and} \quad \mathbf{e}_2 = \begin{pmatrix} 1 \\ 2 \end{pmatrix} - \frac{4}{5}\begin{pmatrix} 2 \\ 1 \end{pmatrix} = \frac{3}{5}\begin{pmatrix} -1 \\ 2 \end{pmatrix}.$$

Normalizing these vectors we obtain

$$\mathbf{Q}_0 = \frac{1}{\sqrt{5}}\begin{pmatrix} 2 & -1 \\ 1 & 2 \end{pmatrix}.$$

From this we have that

$$\mathbf{R}_0 = \mathbf{Q}_0^T\mathbf{C}_0 = \frac{1}{\sqrt{5}}\begin{pmatrix} 5 & 4 \\ 0 & 3 \end{pmatrix},$$

and

$$\mathbf{C}_1 = \mathbf{R}_0\mathbf{Q}_0 = \frac{1}{5}\begin{pmatrix} 14 & 3 \\ 3 & 6 \end{pmatrix}.$$

The remaining $\mathbf{C}_k$ matrices are calculated using MATLAB, with the result that

$$\mathbf{C}_2 = \begin{pmatrix} 2.9756 & 0.21951 \\ 0.21951 & 1.0244 \end{pmatrix} \qquad \mathbf{C}_3 = \begin{pmatrix} 2.9973 & 7\text{e}{-02} \\ 7\text{e}{-02} & 1.0027 \end{pmatrix}$$

$$\mathbf{C}_4 = \begin{pmatrix} 2.9997 & 2\text{e}{-02} \\ 2\text{e}{-02} & 1.0003 \end{pmatrix} \qquad \mathbf{C}_5 = \begin{pmatrix} 3.0000 & 8\text{e}{-03} \\ 8\text{e}{-03} & 1.0000 \end{pmatrix}$$

$$\mathbf{C}_6 = \begin{pmatrix} 3.0000 & 3\text{e}{-03} \\ 3\text{e}{-03} & 1.0000 \end{pmatrix} \qquad \mathbf{C}_7 = \begin{pmatrix} 3.0000 & 9\text{e}{-04} \\ 9\text{e}{-04} & 1.0000 \end{pmatrix}$$

$$\mathbf{C}_8 = \begin{pmatrix} 3.0000 & 3\text{e}{-04} \\ 3\text{e}{-04} & 1.0000 \end{pmatrix} \qquad \mathbf{C}_9 = \begin{pmatrix} 3.0000 & 1\text{e}{-04} \\ 1\text{e}{-04} & 1.0000 \end{pmatrix}$$

It is seen that the off-diagonal entries are converging to zero, while the diagonal entries are approaching the eigenvalues $\lambda_1 = 3$ and $\lambda_2 = 1$. A more expansive list of the computed values for the two eigenvalues is given in Table 4.14. As with orthogonal iteration, once the answer gets close to the solution, the error in $\lambda_1 = 3$ drops by a factor of about $|\lambda_2/\lambda_1|^2 = 1/9$ with each iteration step. ■

It is worth comparing the QR and orthogonal iteration methods. Both require finding $\mathbf{Q}$, and the QR method also requires finding $\mathbf{R}$. As is evident in comparing Tables 4.11 and 4.10, they have the same rate of converge. There are some significant differences. First, for the QR method the $\mathbf{C}_k$'s converge to a matrix containing the eigenvalues, while the $\mathbf{B}_k$'s for orthogonal iteration contain the eigenvectors. For QR, once the eigenvalues are computed then the associated eigenvectors can be computed easily using inverse iteration (Section 4.2.2). For orthogonal iteration, once the eigenvectors are known, Rayleigh's quotient (4.4) can be used to compute the eigenvalues.

Other differences include the observation that orthogonal iteration can be used to compute a few or all of the eigenvalues, while the QR method computes all of them. Also, the QR method is required to start with $\mathbf{C}_0 = \mathbf{A}$, while orthogonal iteration can be used with an initial matrix with m columns, where $1 \le m \le n$.

One property that the QR method and orthogonal method have in common is that they are computationally intensive. In both algorithms, to calculate all of the eigenvalues of the matrix, each step requires $O(n^3)$ flops, and the improvement in the eigenvalue approximations at each step depends on ratios of the form $|\lambda_i/\lambda_j|^2$, where $|\lambda_i| < |\lambda_j|$. Consequently, if two of the eigenvalues are close together, it can take a large number of iterations to compute them accurately. This has given rise to the development of more efficient implementations for the QR method than the simple factor and re-multiply version given earlier. The one most often used in practice uses what is called the implicitly shifted QR method, which is also known as a bulge-chasing method. A nice explanation of this can be found in Watkins [2008]. Another method that is used for larger symmetric matrices, and potentially faster than QR, involves a divide and conquer procedure. Implementing this idea efficiently is challenging and more can be learned about this in Demmel [1997] and Nakatsukasa and Higham [2013].

As a final comment, the QR method has been stated to be one of the "10 algorithms with the greatest influence on the development and practice of science and engineering in the 20th century" [Dongarra and Sullivan, 2000]. What is interesting is that John Francis, who was responsible for deriving and then naming it the QR method, was completely unaware, and amazed, of how significant his work had become. He missed this because shortly after

k	λ_1	λ_2
1	2.8	1.2
2	2.97560975609756	1.02439024390244
3	2.9972602739726	1.0027397260274
4	2.99969521487351	1.00030478512649
5	2.99996613039797	1.00003386960203
6	2.99999623665423	1.00000376334577
7	2.99999958184977	1.00000041815023
8	2.99999995353885	1.00000004646114
9	2.99999999483765	1.00000000516235

Table 4.14 Calculation of the eigenvalues of (4.5) using the QR method. The exact values are $\lambda_1 = 3$ and $\lambda_2 = 1$.

deriving the method he went to work designing and building industrial computer systems. As he recalled, “there had been no reaction, none whatsoever” when his papers appeared [Golub and Uhlig, 2009].

4.3.4 Are the Computed Values Correct?

For many real word applications it is often not known whether the matrix has the required properties for the eigenvalue solver to work, and in such cases one usually simply tries it and sees what happens. What is considered here are simple tests that can be used to check on the correctness, or accuracy, of the computed values.

The principal test will involve the trace of the matrix, and what this is defined next.

Definition 4.3. If $\mathbf{A}$ is an $n \times n$ matrix with entries a_{ij}, then the trace $\text{tr}(\mathbf{A})$ is defined as

$$\text{tr}(\mathbf{A}) = \sum_{i=1}^{n} a_{ii}.$$

In other words, the trace is the sum of the diagonal entries of the matrix.

The second piece of information that is needed comes from a theorem in linear algebra, which is stated next.

Theorem 4.5. *Let $\mathbf{A}$ be an $n \times n$ matrix, with eigenvalues λ_1, λ_2, $\cdots$, λ_n (listed according to their algebraic multiplicities), then*

$$tr(\mathbf{A}) = \sum_{i=1}^{n} \lambda_i,$$

$$tr(\mathbf{A}^k) = \sum_{i=1}^{n} \lambda_i^k,$$

where k is a positive integer.

To demonstrate how the formulas in this theorem are used, we consider a few examples.

Examples

1. For the matrix

$$\mathbf{A} = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix},$$

it was determined earlier that the eigenvalues are $\lambda_1 = 3$ and $\lambda_2 = 1$. Also, $\text{tr}(\mathbf{A}) = 2 + 2 = 4$, and this does indeed equal $\lambda_1 + \lambda_2$. ■

2. The matrix
$$\mathbf{A} = \begin{pmatrix} 3 & 0 \\ 0 & 3 \end{pmatrix}$$
has the one eigenvalue $\lambda_1 = 3$, but it has two linearly independent eigenvectors. Because of this, it has an algebraic multiplicity of two, and so the sum to be considered is $\lambda_1 + \lambda_1 = 6$, It is easy to check that this equals $\text{tr}(\mathbf{A})$. ■

3. The (non-symmetric) matrix
$$\mathbf{A} = \begin{pmatrix} 1 & 2 \\ -\frac{1}{2} & 1 \end{pmatrix}$$
has eigenvalues $\lambda_1 = 1 + i$ and $\lambda_2 = 1 - i$. It is easy to verify that $\text{tr}(\mathbf{A}) = \lambda_1 + \lambda_2$. ■

The above theorem is useful when computing all of the eigenvalues of a matrix, which includes using the QR method and a full matrix with orthogonal iteration. Once these methods are finished, then the values can be checked by comparing their sum with the value for the trace of the matrix (which is very easy to compute). In Section 4.4.2, this test will be used to help determine which of two answers is correct.

There are numerous other ways to check on the accuracy of the computed eigenvalues. One possibility is to use what are called Gershgorin circles [Süli and Mayers, 2003; Varga, 2004]. These provide bounds for the eigenvalues, although they are of limited use for determining the accuracy of a numerical computation. Another possibility is to use shifting. This is very effective at eliminating the possibility that the eigenvalues appear as ± pairs, and this will be demonstrated in Section 4.4.2.

4.4 Applications

Two examples are presented below that illustrate how eigenvalue problems arise in applications. One is from mechanics and the second is from network theory.

4.4.1 Natural Frequencies

Mechanical systems like a mass and spring, and an elastic string, are capable of free vibrations. The frequencies of these vibrations are called natural frequencies, and they play an important role in determining the motion of the system. Finding these frequencies reduces to solving an eigenvalue problem, and this will be illustrated for an elastic string. An explanation of how the elastic string problem can be reduced to a matrix eigenvalue problem is given in the next paragraph. This can be skipped, if desired, and instead you can jump to the subsequent paragraph, where the resulting matrix problem is given.

It is assumed that the string is held at its ends, which are located at $x = 0$ and $x = \ell$. If $u(x,t)$ denotes the vertical displacement of the string, then the equation of motion is

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2} ,$$

where c is a positive constant. The natural frequencies ω are found by assuming that $u(x,t) = v(x)\exp(I\omega t)$, where $I = \sqrt{-1}$. Substituting this into the above partial differential equation, one obtains the ordinary differential equation

$$c^2 \frac{d^2 v}{dx^2} = -\omega^2 v. \tag{4.28}$$

Because the ends are fixed, the solution is required to satisfy $v(0) = v(\ell) = 0$. As with all eigenvalue problems, $v \equiv 0$ is a solution. Consequently, the question is, what values of ω will result in a nonzero solution? These can be computed by replacing the second derivative with the numerical approximation derived in Section 7.2.3. In doing this, the spatial interval $0 \le x \le \ell$ is subdivided into $n+1$ subintervals, as illustrated in Figure 4.3. The function $v(x)$ is then replaced with the vector consisting of its values at the grid points x_1, x_2, $\cdots$, x_n. This yields $\mathbf{v} = (v_1, v_2, \cdots, v_n)^T$, where v_i is the numerical approximation of $v(x_i)$.

Using the approximations discussed in the previous paragraph, finding the natural frequencies of an elastic string reduces to solving the eigenvalue problem

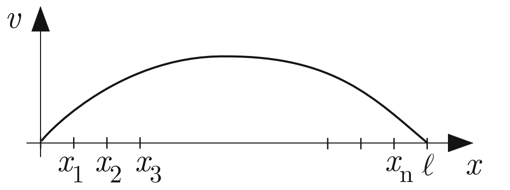


Figure 4.3 Deflection $v(x)$ of string, and the spatial points at which it is computed.

$$\mathbf{A}\mathbf{v} = \lambda\mathbf{v}, \tag{4.29}$$

where

$$\mathbf{A} = \begin{pmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & 0 & \\ & -1 & 2 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & 0 & & & & -1 \\ & & & & -1 & 2 \end{pmatrix}. \tag{4.30}$$

Note that $\mathbf{A}$ is an $n \times n$ symmetric tri-diagonal matrix. Once the eigenvalues λ are computed, then the corresponding natural frequencies of the string are given as $\omega = (n+1)c\sqrt{\lambda}/\ell$.

In most applications, one is often only interested in the lower natural frequencies. Based on this, we want to find the first m eigenvalues of $\mathbf{A}$ that are closest to zero. One way to do this is to use orthogonal iteration with $\mathbf{A}^{-1}$. The algorithm, which comes from using $\mathbf{A}^{-1}$ in the procedure given in Table 4.9, is given in Table 4.15. Since we are interested in finding the first m eigenvalues, we will take $\mathbf{B}$ to be an $n \times m$ matrix. The results of the computation are given in Table 4.16. The stopping condition used in the calculation was $|v_k - v_{k-1}|/|v_k| < 10^{-6}$, which resulted in the method taking 18 iteration steps. Also, the exact frequencies for the string are $\omega = i\pi c$, where $i = 1, 2, 3, \cdots$. Since $\omega = (n+1)c\sqrt{\lambda}$, then the exact values can be written as

$$\bar{\lambda}_i = \left(\frac{i\pi}{n+1}\right)^2, \text{ for } i = 1, 2, 3, \cdots . \tag{4.31}$$

The relative difference between these values and the computed values is given in Table 4.16.

```
Pick:  random $\mathbf{B}$
       $\mathbf{Q} = GS(\mathbf{B})$
       $\mathbf{A} = \mathbf{LU}$
Loop   For $k = 1, 2, 3, \cdots$
          $\mathbf{AB} = \mathbf{Q}$
          $\mathbf{Q} = GS(\mathbf{B})$
       End
```

Table 4.15 Inverse orthogonal iteration used to calculate the m eigenvalues of $\mathbf{A}$ closest to zero. Note $\mathbf{Q} = GS(\mathbf{B})$ means that Gram-Schmidt is applied to the columns of $\mathbf{B}$ to find $\mathbf{Q}$, and $\mathbf{AB} = \mathbf{Q}$ means that the equation is solved for $\mathbf{B}$ using the LU factorization. Also, $\mathbf{B}$ is an $n \times m$ matrix.

i	λ_i	$\lvert\lambda_i - \bar{\lambda}_i\rvert/\lvert\bar{\lambda}_i\rvert$
1	1.566558555547e−04	1.31e−05
2	6.265988811618e−04	5.22e−05
3	1.409755457697e−03	1.17e−04
4	2.506004005922e−03	2.08e−04
5	3.915169058548e−03	3.26e−04

Table 4.16 Calculation of the five smallest eigenvalues of (4.30) when $n = 250$ using inverse orthogonal iteration. Also given is the relative error $|\lambda_i - \bar{\lambda}_i|/|\bar{\lambda}_i|$.

There are two somewhat subtle questions that should be considered before leaving this example. The first is, what needs to be done to obtain more accurate values for the natural frequencies of the string? The stopping condition using orthogonal iteration was $|v_k - v_{k-1}|/|v_k| < 10^{-6}$, yet the relative error for the natural frequencies, as given in Table 4.16, is only about 10^{-4}. To answer this, the eigenvalue equation in (4.29) is an approximation of the problem in (4.28). If we want better approximations to the natural frequencies, then we need to make (4.29) a better approximation of (4.28), and this is accomplished by making n bigger. As an example, to have the relative error of all five eigenvalues no more than 10^{-5}, one can take, approximately, $n = 1500$.

The second question comes from the observation that the matrix in (4.30) looks similar to the one for the oscillator chain in (4.8). This is a concern because the eigenvalues for (4.8) get very close together as n increases, which causes our eigenvalue solvers to converge very slowly. This does not seem to happen for the string problem, and the question is, why not? The answer comes from the formula for the eigenvalues for both matrices, which is given in (4.9). The speed of convergence of our eigenvalue solvers depends on the ratios

$$\frac{\lambda_{i-1}}{\lambda_i} = \frac{a + 2\cos((i-1)\theta)}{a + 2\cos(i\theta)}\,, \text{ for } i = 2, 3, 4, \cdots, n, \tag{4.32}$$

where $\theta = \pi/(n+1)$. It is also worth noting that the higher frequencies correspond to small values of i, and the lower frequencies come from the larger values of i. The values of (4.32) are given in Figure 4.4 for the oscillator chain, where $a = 2.1$, and for the string, where $a = 2$. What is seen is that the ratios for the oscillator chain are close to one, particularly for the larger value of n. For the string, the ratios are close to one for the higher frequencies, but not for the lowest frequencies. Because we limited the calculation to the first five frequencies for the string, our inverse orthogonal iteration procedure had no trouble converging relatively quickly.

4.4.2 Graphs and Networks

A network, or graph, consists of nodes and lines connecting them. Two examples are shown in Figure 4.5. The lines in this case indicate the connections between the respective nodes. For example, the nodes could identify computers and the lines identifying how they are connected. Other possibilities are that the nodes are cities and the lines are airline routes (see Exercise 4.25), or that the nodes are atoms and the lines indicate their bonds (see Exercise 4.26).

Eigenvalues provide useful information about the properties of the network, and this is done using what is called an *adjacency matrix*. This matrix for the network on the left in Figure 4.5 is

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}. \tag{4.33}$$

To explain how this is determined, first note that if there are n nodes, then the adjacency matrix is $n \times n$. The entries of $\mathbf{A}$ are zero except when node i is connected to node j, in which case $a_{ij} = a_{ji} = 1$. For example, for the

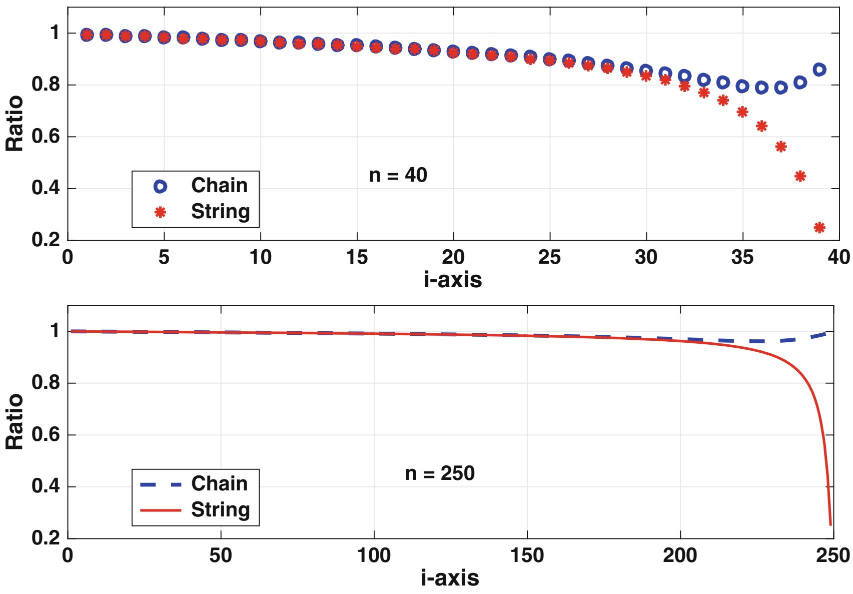


Figure 4.4 Eigenvalue ratios (4.32) for the oscillator chain and for the string, when $n = 40$ and $n = 250$.

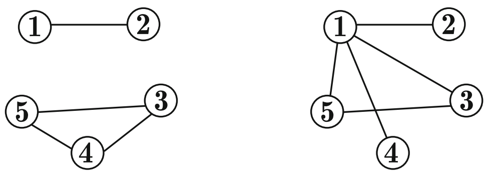


Figure 4.5 Examples of two networks with five nodes.

graph on the left in Figure 4.5, the third and fifth nodes are connected, so $a_{35} = a_{53} = 1$. Also, since the first and fifth nodes are not connected, then $a_{15} = a_{51} = 0$.

We are considering what is known as an undirected graph, or network, which means that if $a_{ij} = 1$ then $a_{ji} = 1$. In other words, the adjacency matrix is symmetric. The eigenvalues of the adjacency matrix play an important role in determining the properties of a network. As an example, suppose the eigenvalues for the adjacency matrix are $\lambda_1, \lambda_2, \cdots, \lambda_n$. If P_m is the number of paths through the graph that take m steps and end up back at the node they started at, it is possible to show that

$$P_m = \sum_{i=1}^{n} \lambda_i^m. \tag{4.34}$$

As an example, for the network on the left in Figure 4.5, the paths consisting of two steps are $1 \to 2 \to 1$, $2 \to 1 \to 2$, $5 \to 4 \to 5$, etc. Using the terminology of graph theory, these are called closed paths of length two. There are a total of eight closed paths of length two, so $P_2 = \lambda_1^2 + \cdots + \lambda_5^2 = 8$.

In our example, we will assume there are n nodes, with node 1 connected to node 2. For the others, node i is connected to node $i + 1$, except that node n is connected to node 3. Such a network for $n = 5$ is shown on the left in Figure 4.5, and the resulting adjacency matrix is given in (4.33). To test our eigenvalue solvers we will take $n = 10$. The eigenvalues computed using orthogonal iteration, as outlined in Table 4.9, are

$$\mathbf{v} = \begin{pmatrix} -1.8262 \\ -0.54037 \\ -0.24545 \\ -0.16485 \\ 9.2445\text{e}{-33} \\ 3.6978\text{e}{-32} \\ 0.18969 \\ 0.22062 \\ 0.54037 \\ 1.8262 \end{pmatrix}. \tag{4.35}$$

Although all seems fine, there is a potentially serious problem with this result. The first and last two eigenvalues appear to be negatives of each other, and this is the one situation that orthogonal iteration likely fails. It is possible to check on whether the computation is correct by using shifting. To check, orthogonal iteration is going to be applied to $\mathbf{B} = \mathbf{A} - 3\mathbf{I}$. If the result in (4.35) is correct, then the eigenvalues of $\mathbf{B}$ should not have $\pm$ pairs. Doing this, and then shifting back, one gets the eigenvalues

$$\mathbf{v} = \begin{pmatrix} -2 \\ -1.4142 \\ -1.4142 \\ -1 \\ 0 \\ 0 \\ 1 \\ 1.4142 \\ 1.4142 \\ 2 \end{pmatrix}. \tag{4.36}$$

This shows that applying orthogonal iteration directly to $\mathbf{A}$ is even worse than we originally thought because the above solution shows that every nonzero eigenvalue appears as a $\pm$ pair.

The above situation is interesting because two different answers are computed and it is necessary to determine which, if any, is correct. The question is, how do you know which one to pick. This is where the theory plays an important role. From Section 4.3.4, we know that $\mathrm{tr}\mathbf{A} = \lambda_1 + \lambda_2 + \cdots + \lambda_{10}$. We also know, from (4.34), that for an adjacency matrix, $P_m = \sum \lambda_i^m$. Although we might know the exact value of the P_m's, we do know that they are positive integers. Using the values computed using orthogonal iteration, from (4.35),

$$P_2 = 12.317, \qquad P_3 = 0.13494,$$

while, from (4.36),

$$P_2 = 18, \qquad P_3 = 50.$$

According to this, (4.36) is the correct answer.

If theoretical checks are not possible, then one could run some numerical tests. For example, use orthogonal iteration with different shifts to see if one continues to obtain (4.36).

4.5 Singular Value Decomposition

One of the more significant results in linear algebra is the spectral decomposition theorem. For those who might not remember this, it is given next.

Theorem 4.6. *If* $\mathbf{A}$ *is a symmetric matrix, then it is possible to factor* $\mathbf{A}$ *as*

$$\mathbf{A} = \mathbf{QDQ}^T, \tag{4.37}$$

where $\mathbf{D}$ *is a diagonal matrix and* $\mathbf{Q}$ *is an orthogonal matrix.*

The definition and basic properties of an orthogonal matrix are given in Section 4.3.2.

To explain where the matrices in this theorem come from, let $\mathbf{u}_1$, $\mathbf{u}_2$, $\cdots$, $\mathbf{u}_n$ be orthonormal eigenvectors for $\mathbf{A}$, with corresponding eigenvalues λ_1, λ_2, $\cdots$, λ_n (see Theorem 4.1). In this case, $\mathbf{D}$ is the diagonal matrix

$$\mathbf{D} = \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{pmatrix},$$

and the ith column of $\mathbf{Q}$ is $\mathbf{u}_i$. Diagrammatically, the factorization in (4.37) can be written as

$$\mathbf{A} = \begin{pmatrix} \uparrow & \uparrow & & \uparrow \\ \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_n \\ \downarrow & \downarrow & & \downarrow \end{pmatrix} \begin{pmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{pmatrix} \begin{pmatrix} \leftarrow \mathbf{u}_1 \rightarrow \\ \leftarrow \mathbf{u}_2 \rightarrow \\ \vdots \\ \leftarrow \mathbf{u}_n \rightarrow \end{pmatrix}.$$

Example

For the matrix $\mathbf{A}$ in (4.5) we found that $\lambda_1 = 3$, with $\mathbf{u}_1 = (1,1)^T/\sqrt{2}$, and $\lambda_2 = 1$, with $\mathbf{u}_2 = (1,-1)^T/\sqrt{2}$. Accordingly, the factorization in the above theorem is

$$\begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix} = \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{pmatrix} \begin{pmatrix} 3 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{pmatrix}. \quad \blacksquare$$

Given that it is possible to carry out such a factorization, the next question is, why do this? One answer is that this shows that it is possible to change the basis so the matrix is diagonal, and diagonal matrices are very easy to work with. For example, it is much easier to solve $\mathbf{Dx} = \mathbf{b}$ than it is to solve $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is a full $n \times n$ matrix. The limitation is that the above theorem requires the matrix to be symmetric. What we are going to consider is how it might be possible to derive a similar result for non-symmetric matrices, and even for matrices that are not $n \times n$. It needs to be pointed out that the resulting factorization will have some distinct differences from (4.37), even in the case of when the matrix is symmetric. What these are will be explained once the factorization has been derived.

4.5.1 Derivation of the Singular Value Decomposition

The way we will approach finding a factorization is the same method we used to find the LU factorization, namely we will simply assume it's possible and see if we can make it work. The assumption here is that $\mathbf{A}$ is $m \times n$, with $n \leq m$. The hypothesis is that there is a factorization of the form

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T, \tag{4.38}$$

where $\mathbf{U}$ is an $m \times m$ orthogonal matrix, $\mathbf{V}$ is an $n \times n$ orthogonal matrix, and $\mathbf{\Sigma}$ is an $m \times n$ matrix that has the form

$$\mathbf{\Sigma} = \begin{pmatrix} \mathbf{S} \\ \mathbf{O} \end{pmatrix}, \tag{4.39}$$

where $\mathbf{S}$ is a diagonal $n \times n$ matrix of the form

$$\mathbf{S} = \begin{pmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n \end{pmatrix}, \tag{4.40}$$

and $\mathbf{O}$ is an $(m-n) \times n$ matrix containing only zeros. The key step for finding these matrices is to consider $\mathbf{A}^T\mathbf{A}$ and $\mathbf{A}\mathbf{A}^T$. It should be pointed out that it is possible to prove that a factorization is possible in a short paragraph [Golub and Van Loan, 2013]. The approach used here is more constructive, albeit longer, with the objective of being able to show how the three matrices in the factorization can be determined.

It is not hard to show that $\mathbf{A}^T\mathbf{A}$ is an $n \times n$ symmetric matrix. So, from (4.37) we have that

$$\mathbf{A}^T\mathbf{A} = \mathbf{Q}_L\mathbf{D}_L\mathbf{Q}_L^T, \tag{4.41}$$

where $\mathbf{D}_L$ is a diagonal matrix formed from the eigenvalues for $\mathbf{A}^T\mathbf{A}$, and $\mathbf{Q}_L$ is an orthogonal matrix (and both are $n \times n$). On the other hand, if the factorization is (4.38) is possible, then

$$\begin{aligned} \mathbf{A}^T\mathbf{A} &= (\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T)^T\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \\ &= (\mathbf{V}\mathbf{\Sigma}^T\mathbf{U}^T)\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \\ &= \mathbf{V}\mathbf{D}_V\mathbf{V}^T, \end{aligned} \tag{4.42}$$

where $\mathbf{D}_V = \mathbf{\Sigma}^T\mathbf{\Sigma}$ is an $n \times n$ diagonal matrix given as

$$\mathbf{D}_V = \begin{pmatrix} \sigma_1^2 & & & \\ & \sigma_2^2 & & \\ & & \ddots & \\ & & & \sigma_n^2 \end{pmatrix}. \tag{4.43}$$

In comparing (4.41) and (4.42) we come to the conclusion that for the factorization in (4.38) we can take $\mathbf{V} = \mathbf{Q}_L$. It also means that $\mathbf{D}_V = \mathbf{D}_L$, and so the diagonals in $\mathbf{D}_V$ are the eigenvalues for $\mathbf{A}^T\mathbf{A}$. More specifically, in (4.40), $\sigma_i = \sqrt{\lambda_i}$, where λ_i is an eigenvalue for $\mathbf{A}^T\mathbf{A}$.

In a similar vein, given that $\mathbf{A}\mathbf{A}^T$ is an $m \times m$ symmetric matrix, then

$$\mathbf{A}\mathbf{A}^T = \mathbf{Q}_R\mathbf{D}_R\mathbf{Q}_R^T, \tag{4.44}$$

where $\mathbf{D}_R$ is a diagonal matrix formed from the eigenvalues for $\mathbf{A}\mathbf{A}^T$, and $\mathbf{Q}_R$ is an orthogonal matrix (and both are $m \times m$). According to (4.38), $\mathbf{A}\mathbf{A}^T = \mathbf{U}\mathbf{D}_U\mathbf{U}^T$, and for this to agree with (4.44) we take $\mathbf{U} = \mathbf{Q}_R$, and $\mathbf{D}_U = \mathbf{\Sigma}\mathbf{\Sigma}^T$ is an $m \times m$ diagonal matrix of the form

$$\mathbf{D}_U = \begin{pmatrix} \sigma_1^2 & & & & & \\ & \ddots & & & & \\ & & \sigma_n^2 & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{pmatrix}. \tag{4.45}$$

Moreover, this matrix must equal $\mathbf{D}_R$, which means the diagonals in $\mathbf{D}_U$ are the eigenvalues for $\mathbf{A}\mathbf{A}^T$.

We now know how to determine the orthogonal matrices $\mathbf{U}$ and $\mathbf{V}$ in (4.38). However, the σ_i's in (4.40) are connected to both the eigenvalues of $\mathbf{A}^T\mathbf{A}$, as expressed in (4.43), and to the eigenvalues of $\mathbf{A}\mathbf{A}^T$, as expressed in (4.45). To ensure that these conditions do not contradict each other we need the following information.

Lemma 1 *Assuming $\mathbf{A}$ is an $m \times n$ matrix, with $n \leq m$, then:*

1. *$\mathbf{A}^T\mathbf{A}$ and $\mathbf{A}\mathbf{A}^T$ are symmetric with non-negative eigenvalues.*
2. *If λ is an eigenvalue of $\mathbf{A}\mathbf{A}^T$, with eigenvector $\mathbf{u}$, then either λ is an eigenvalue for $\mathbf{A}^T\mathbf{A}$ with eigenvector $\mathbf{A}^T\mathbf{u}$, or else $\lambda = 0$ and $\mathbf{A}^T\mathbf{u} = \mathbf{0}$.*

The proofs of these statements are not particularly difficult and are left as an exercise. From the first statement, it follows that $\sigma_i \geq 0$ in (4.43) and (4.45). From the second statement, $\mathbf{A}\mathbf{A}^T$ and $\mathbf{A}^T\mathbf{A}$ share nonzero eigenvalues, and the additional eigenvalues of $\mathbf{A}\mathbf{A}^T$ are just zero, hence the extra zeros on the diagonal in (4.45).

The final detail concerns getting the sign correct. Everything we have ascertained about the factorization comes from considering $\mathbf{A}\mathbf{A}^T$ and $\mathbf{A}^T\mathbf{A}$. These products are the same for $\mathbf{A}$ and $-\mathbf{A}$, and there is nothing in the derivation to account for the difference in sign. To do this, suppose $\mathbf{v}_j$ is the eigenvector that is going to be used in the jth column in $\mathbf{V}$, and it has corresponding eigenvalue λ_j. Also note that, being an eigenvector, we could just as well pick $-\mathbf{v}_j$. Letting $\mathbf{u}_j$ be the eigenvector to be used in the jth column of $\mathbf{U}$, then from (4.38) we have

$$\mathbf{A}\mathbf{v}_j = \sigma_j \mathbf{u}_j, \quad \text{for } j = 1, 2, \cdots, n.$$

By assumption, the singular values are non-negative, so $\sigma_j = \sqrt{\lambda_j}$. This means we pick the signs for the eigenvectors for $\mathbf{V}$ and $\mathbf{U}$ to be consistent with the above equation.

4.5.2 Summary of the Singular Value Decomposition

Assuming $\mathbf{A}$ is a nonzero $m \times n$ matrix, with $n \leq m$, then the singular value decomposition (SVD) of $\mathbf{A}$ has the form

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T, \tag{4.46}$$

where the matrices appearing here are

$\mathbf{U}$: This is an $m \times m$ orthogonal matrix of the form

$$\mathbf{U} = \begin{pmatrix} \uparrow & \uparrow & & \uparrow \\ \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_m \\ \downarrow & \downarrow & & \downarrow \end{pmatrix},$$

where the column vectors $\mathbf{u}_i$ are orthonormal eigenvectors for $\mathbf{A}\mathbf{A}^T$.

$\mathbf{V}$: This is an $n \times n$ orthogonal matrix of the form

$$\mathbf{V} = \begin{pmatrix} \uparrow & \uparrow & & \uparrow \\ \mathbf{v}_1 & \mathbf{v}_2 & \cdots & \mathbf{v}_n \\ \downarrow & \downarrow & & \downarrow \end{pmatrix},$$

where the column vectors $\mathbf{v}_i$ are orthonormal eigenvectors for $\mathbf{A}^T\mathbf{A}$.

$\mathbf{\Sigma}$: This is an $m \times n$ matrix of the form

$$\mathbf{\Sigma} = \begin{pmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n \\ 0 & & \cdots & 0 \\ \vdots & & & \vdots \\ 0 & & \cdots & 0 \end{pmatrix} .$$

This can be written in more compact form as

$$\mathbf{\Sigma} = \begin{pmatrix} \mathbf{S} \\ \mathbf{O} \end{pmatrix} , \tag{4.47}$$

where $\mathbf{S}$ is a diagonal $n \times n$ matrix of the form

$$\mathbf{S} = \begin{pmatrix} \sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_n \end{pmatrix} , \tag{4.48}$$

and $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$. The σ_i's are called the *singular values* for $\mathbf{A}$, and they are given as $\sigma_i = \sqrt{\lambda_i}$, where λ_i is an eigenvalue for $\mathbf{A}^T\mathbf{A}$. Also, the columns of $\mathbf{U}$ and $\mathbf{V}$ are consistent with the ordering of the σ_i's. In particular, $\mathbf{A}\mathbf{v}_j = \sigma_j \mathbf{u}_j$ for $j = 1, 2, \cdots, n$. Finally, $\mathbf{O}$ in (4.47) is an $(m-n) \times n$ matrix containing only zeros.

Although the factorization resembles the one in (4.37), and it certainly uses that result in its derivation, there are differences. The most obvious one is that the factorization in (4.46) works on matrices that are not square. However, consider the case of when $\mathbf{A}$ is symmetric (and therefore square). The diagonals of $\mathbf{D}$ are the eigenvalues of $\mathbf{A}$, and they can be positive, negative, or zero. The singular values listed in $\mathbf{S}$, on the other hand, are non-negative. One might guess that the singular values in this case are just the absolute values of the nonzero eigenvalues of $\mathbf{A}$, and this is correct. However, this conclusion is limited to symmetric matrices, and as will be shown in an example below, it does not need to hold for square but non-symmetric matrices.

Examples

1. Suppose

$$\mathbf{A} = \begin{pmatrix} 2 & -1 \\ 1 & 1 \\ -1 & 2 \end{pmatrix} .$$

In this case

$$\mathbf{A}^T\mathbf{A} = \begin{pmatrix} 6 & -3 \\ -3 & 6 \end{pmatrix} .$$

The eigenvalues of this matrix are $\lambda_1 = 9$ and $\lambda_2 = 3$, with corresponding orthonormal eigenvectors $\mathbf{v}_1 = (-\sqrt{2}/2, \sqrt{2}/2)^T$ and $\mathbf{v}_2 = (\sqrt{2}/2, \sqrt{2}/2)^T$. Consequently,

$$\mathbf{V} = \begin{pmatrix} -\frac{1}{2}\sqrt{2} & \frac{1}{2}\sqrt{2} \\ \frac{1}{2}\sqrt{2} & \frac{1}{2}\sqrt{2} \end{pmatrix} .$$

Similarly, the matrix

$$\mathbf{A}\mathbf{A}^T = \begin{pmatrix} 5 & 1 & -4 \\ 1 & 2 & 1 \\ -4 & 1 & 5 \end{pmatrix}$$

has eigenvalues $\lambda_1 = 9$, $\lambda_2 = 3$ and $\lambda_3 = 0$. The corresponding orthonormal eigenvectors are $\mathbf{u}_1 = (-\sqrt{2}/2, 0, \sqrt{2}/2)^T$, $\mathbf{u}_2 = (\sqrt{6}/6, \sqrt{6}/3, \sqrt{6}/6)^T$, and $\mathbf{u}_3 = (\sqrt{3}/3, -\sqrt{3}/3, \sqrt{3}/3)^T$. Consequently,

$$\mathbf{U} = \begin{pmatrix} -\frac{1}{2}\sqrt{2} & \frac{1}{6}\sqrt{6} & \frac{1}{3}\sqrt{3} \\ 0 & \frac{1}{3}\sqrt{6} & -\frac{1}{3}\sqrt{3} \\ \frac{1}{2}\sqrt{2} & \frac{1}{6}\sqrt{6} & \frac{1}{3}\sqrt{3} \end{pmatrix} .$$

The resulting SVD is

$$\mathbf{A} = \mathbf{U}\begin{pmatrix} 3 & 0 \\ 0 & \sqrt{3} \\ 0 & 0 \end{pmatrix}\mathbf{V}^T .$$

The singular values for $\mathbf{A}$ in this case are $\sigma_1 = 3$ and $\sigma_2 = \sqrt{3}$. ■

2. In the case of when

$$\mathbf{A} = \begin{pmatrix} 2 & -1 \\ 1 & -2 \end{pmatrix} ,$$

one finds that

$$\mathbf{A} = \mathbf{U}\begin{pmatrix} 3 & 0 \\ 0 & 1 \end{pmatrix}\mathbf{V}^T ,$$

where

$$\mathbf{U} = \begin{pmatrix} \frac{1}{2}\sqrt{2} & \frac{1}{2}\sqrt{2} \\ \frac{1}{2}\sqrt{2} & -\frac{1}{2}\sqrt{2} \end{pmatrix} \quad \text{and} \quad \mathbf{V} = \begin{pmatrix} \frac{1}{2}\sqrt{2} & \frac{1}{2}\sqrt{2} \\ -\frac{1}{2}\sqrt{2} & \frac{1}{2}\sqrt{2} \end{pmatrix} .$$

The singular values for $\mathbf{A}$ in this case are $\sigma_1 = 3$ and $\sigma_2 = 1$. In comparison, the eigenvalues of $\mathbf{A}$ are $\lambda_1 = \sqrt{3}$ and $\lambda_2 = -\sqrt{3}$. Note that the singular values of $\mathbf{A}$ are not the absolute values of its eigenvalues. ■

There are some useful facts about the SVD that should be stated explicitly, so they can be referred to later. The first concerns the SVD for a symmetric matrix, which was discussed earlier.

Theorem 4.7. *If $\mathbf{A}$ is symmetric, then its singular values are the absolute value of its eigenvalues. For its SVD, if $\mathbf{u}_i$ is the ith column of $\mathbf{U}$, then the ith column of $\mathbf{V}$ is $\pm\mathbf{u}_i$, where the $-$ is used if the corresponding eigenvalue is negative, otherwise the $+$ is used.*

This result is a direct consequence of the fact that for a symmetric matrix, $\mathbf{A}\mathbf{A}^T = \mathbf{A}^T\mathbf{A} = \mathbf{A}^2$.

The second useful fact involves the rank of a matrix. As you might recall from linear algebra, the number of independent rows in a matrix is equal to the number of independent columns, and this number is defined as the rank r of the matrix.

Theorem 4.8. *The matrix $\mathbf{A}$ has rank r if, and only if, $\mathbf{A}$ has exactly r nonzero singular values.*

This follows from a result in linear algebra which states the rank of a diagonal matrix equals the number of nonzero diagonals, from which it can be shown that the rank of $\mathbf{A}$ equals the rank of $\mathbf{\Sigma}$.

4.5.2.1 Computing a SVD

It is more than apparent how much work is needed to find the SVD for a matrix. Using a direct approach, it is necessary to calculate $\mathbf{A}\mathbf{A}^T$ and $\mathbf{A}^T\mathbf{A}$, then find the eigenvalues and eigenvectors for each of these matrices. All of these steps are doable using one or more of the methods described earlier in this chapter. However, more efficient procedures have been developed, and the standard approach is to use what is known as the Golub-Reinsch algorithm, which involves transforming the problem to make it more amenable to a computational solution [Cline and Dhillon, 2007]. The number of flops for a square matrix is, according to Golub and Van Loan [2013], approximately $21n^3$. Said another way, the flops for the SVD is about a factor of 31 more than for the LU. In looking at the times in Table 4.13, it appears the algorithm used by MATLAB is a bit better than this, at least for the smaller matrices. Even though it does take significantly longer than the LU calculation, the actual computing time is not intolerable. For example, an SVD factorization for a 4000×4000 matrix takes less than 30 seconds. It is also apparent that the time increases quickly as the size of the matrix increases, and this has generated interest in finding efficient algorithms for computing an approximate SVD for very large matrices. More information about this can be found in Halko et al. [2011].

4.5.3 Application: Image Compression

One application for the SVD is image compression. To explain, the grayscale image shown in Figure 4.6 consists of a 1528×1225 array of integers, each with a value from 0 to 255. This number represents the intensity of gray for that pixel, with 0 corresponding to black and 255 corresponding to white. The SVD is going to be applied to this array, but before doing this note that the SVD factorization in (4.46) can be multiplied out and written as

$$\mathbf{A} = \sum_{i=1}^{r} \sigma_i \mathbf{W}_i, \tag{4.49}$$

where

$$\mathbf{W}_i = \begin{pmatrix} \uparrow & \uparrow & & \uparrow \\ v_{i1}\mathbf{u}_i & v_{i2}\mathbf{u}_i & \cdots & v_{in}\mathbf{u}_i \\ \downarrow & \downarrow & & \downarrow \end{pmatrix}.$$

The sum in (4.49) includes only the nonzero singular values, and it is assumed that σ_r is the smallest nonzero singular value for $\mathbf{A}$. Also, $\mathbf{W}_i$ is an $m \times n$ matrix with its columns determined using the ith column of $\mathbf{U}$ and the ith column of $\mathbf{V}$. This is often written as an outer product, which in this case takes the form $\mathbf{W}_i = \mathbf{u}_i \mathbf{v}_i^T$. What is important in (4.49) is what is not there. Namely, even though there are m column vectors in $\mathbf{U}$, only the first r of them make a nonzero contribution to $\mathbf{A}$.

The idea underlying image compression using the SVD is to consider if it is necessary to include all r terms in (4.49). In particular, is it possible to use an approximation of the form

$$\mathbf{A}_k = \sum_{i=1}^{k} \sigma_i \mathbf{W}_i, \tag{4.50}$$

where $k < r$? To answer this, note that the entries in the $\mathbf{W}_i$ matrices satisfy $-1 \leq w \leq 1$. Consequently, the contributions of the terms in the above sum are mostly determined by the size of the respective singular value σ_i. The singular values for this image are plotted in Figure 4.7. Certainly given how large the first few singular values are, it would be expected that they need

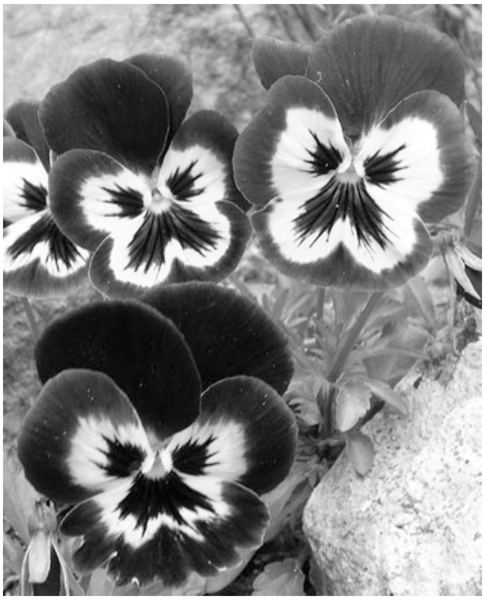


Figure 4.6 Grayscale image of pansies, before being compressed using the SVD. They don't look to be too excited about being compressed.

to be included. However, the singular values decrease, and presumably their respective contributions to the image also decrease. The question is, just how many do we need to get an acceptable reproduction of the original image. To investigate this, the resulting images when using $\mathbf{A}_{10}$, $\mathbf{A}_{25}$, $\mathbf{A}_{50}$, and $\mathbf{A}_{100}$ are shown in Figure 4.8. Although the first two (on the top row) are not great, the one on the bottom left, which corresponds to $k = 50$, is not bad. The one on lower right, which is the $k = 100$ case, is even better. These observations are consistent with the singular values plotted in Figure 4.7. Namely, there is a dramatic change in the singular values as k increases, up to about $k = 40$,

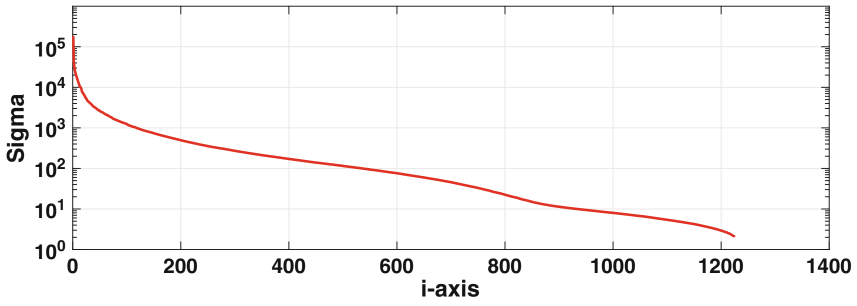


Figure 4.7 Singular values σ_i for the image in Figure 4.6.

after which their values more slowly decrease. What we are seeing here is that the improvements in the image follow a similar pattern, and this is why there are more noticeable differences between $k = 25$ and $k = 50$ than there are between $k = 50$ and $k = 100$.

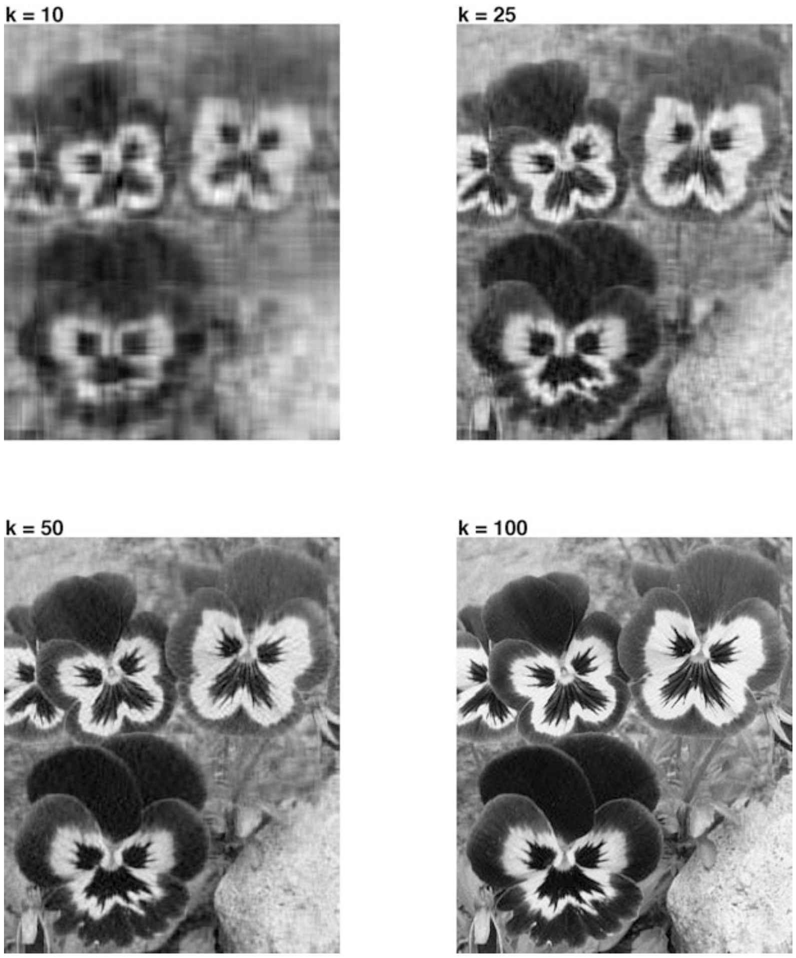


Figure 4.8 Resulting image when using (4.50) to approximate the original image: for $k = 10$, $k = 25$, $k = 50$, and $k = 100$.

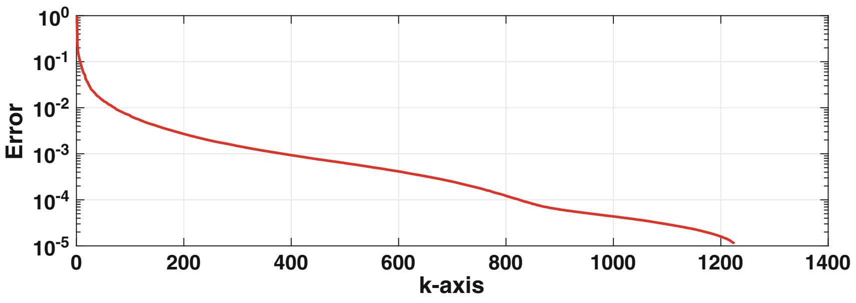


Figure 4.9 Relative error (4.51) when using $\mathbf{A}_k$, given in (4.50), to approximate $\mathbf{A}$ for the pansy picture.

4.5.3.1 Eckart-Young Theorem and Error

It is possible to provide a mathematical explanation for the improvement in the images in Figure 4.7 as k increases. This requires the next result, which is known as the Eckart-Young theorem.

Theorem 4.9. *Assuming* $\mathbf{A}$ *is an* $m \times n$ *matrix, with* $\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$*, then*

$$||\mathbf{A}||_2 = \sigma_1,$$

and

$$||\mathbf{A} - \mathbf{A}_k||_2 = \sigma_{k+1},$$

where $\mathbf{A}_k$ *is given in (4.50), and* $1 \le k < r$.

The proof of this can be found in Golub and Van Loan [2013]. Also, when matrix norms were introduced in Section 3.5.1 we only considered square matrices. The definition, given in (3.9) or (3.9), applies without change to $m \times n$ matrices, with the understanding that $\mathbf{x}$ is an n-vector.

The usefulness of the above theorem for us is that it shows that the relative error in using $\mathbf{A}_k$ to approximate a nonzero matrix $\mathbf{A}$ is

$$\begin{aligned} E(k) &\equiv \frac{||\mathbf{A} - \mathbf{A}_k||_2}{||\mathbf{A}||_2} \\ &= \frac{\sigma_{k+1}}{\sigma_1} \, . \end{aligned} \tag{4.51}$$

The resulting graph for E for the pansy picture is shown in Figure 4.9. This verifies our earlier conclusion, which is that the approximation improves quickly as k increases up to about 50, after which the improvement slows. For example, using the first 50 terms, the error is approximately 1.4×10^{-2}, and adding an additional 50 terms drops the error to just 7×10^{-3}.

What benefits, in terms of storage, are achieved by using this type of compression? Well, the full image requires storing 1,871,800 integers, while the ith term in (4.49) requires storage of an m-vector and an n-vector, which in this case means 2,753 floating point numbers. So, for $\mathbf{A}_{100}$, the $\mathbf{W}_i$ matrices require storage of 275,300 floating point numbers. This is a factor of about 0.15 smaller than the original. This is certainly an improvement, but note that using the SVD in this way produces a "lossy" compression because information is lost in the approximation. To its credit, however, the SVD method has an adjustable parameter, k, that allows for various resolutions of the image.

Exercises

4.1. The symmetric matrix

$$\mathbf{A} = \begin{pmatrix} 2 & 2 \\ 2 & -1 \end{pmatrix}$$

has eigenvectors $\mathbf{x}_1 = (2, 1)^T$ and $\mathbf{x}_2 = (1, -2)^T$.
(a) Is this matrix positive definite?
(b) What are the corresponding eigenvalues?
(c) Assuming the starting vector $\mathbf{y}_0 = (1, 1)^T$, what eigenvalue will the power method converge to and what will be the resulting eigenvector?
(d) Assuming the calculation in part (c) is finished, explain how to use shifting to compute the other eigenvalue.

4.2. The symmetric matrix

$$\mathbf{A} = \begin{pmatrix} -7 & -6 \\ -6 & 2 \end{pmatrix}$$

has eigenvectors $\mathbf{x}_1 = (1, -2)^T$ and $\mathbf{x}_2 = (-2, 1)^T$.
(a) Is this matrix positive definite?
(b) What are the corresponding eigenvalues?
(c) Assuming the starting vector $\mathbf{y}_0 = (1, -1)^T$, what eigenvalue will the power method converge to?
(d) Assuming the calculation in part (c) is finished, explain how to use shifting to compute the other eigenvalue.

4.3. The symmetric matrix

$$\mathbf{A} = \begin{pmatrix} 11 & 7 & -4 \\ 7 & 11 & 4 \\ -4 & 4 & -10 \end{pmatrix}.$$

has eigenvectors $\mathbf{x}_1 = (1, 1, 0)^T$, $\mathbf{x}_2 = (1, 0, 2)^T$, and $\mathbf{x}_3 = (0, -2, 1)^T$.

(a) Is this matrix positive definite?
(b) What are the corresponding eigenvalues?
(c) Assuming the starting vector $\mathbf{y}_0 = (1,1,1)^T$, what eigenvalue will the power method converge to and what will be the resulting eigenvector?

4.4. The symmetric matrix

$$\mathbf{A} = \begin{pmatrix} 47 & 32 & 8 \\ 32 & -1 & -16 \\ 8 & -16 & 59 \end{pmatrix} .$$

has eigenvectors $\mathbf{x}_1 = (2,1,0)^T$, $\mathbf{x}_2 = (1,-2,10)^T$, and $\mathbf{x}_3 = (2,-4,-1)^T$.
(a) Is this matrix positive definite?
(b) What are the corresponding eigenvalues?
(c) Assuming the starting vector $\mathbf{y}_0 = (1,1,1)^T$, what eigenvalue will the power method converge to and what will be the resulting eigenvector?

4.5. Suppose $\mathbf{A}$ is a symmetric 5×5 matrix with eigenvalues -2, -1, 0, 1, 2.

(a) Explain why the power method will likely fail with this matrix.
(b) Explain how shifting can be used so the power method can be used to calculate the ± 2 eigenvalues of $\mathbf{A}$.
(c) Explain why the power method applied to $\mathbf{B} = \mathbf{A} + 10^4\mathbf{I}$ can be used to find the largest eigenvalue of $\mathbf{A}$, but it is a poor choice to, say, using $\mathbf{B} = \mathbf{A} + 10\mathbf{I}$.

4.6. Consider the symmetric matrix

$$\mathbf{A} = \begin{pmatrix} 2 & 2 \\ 2 & -1 \end{pmatrix} .$$

(a) Assuming $\mathbf{B}_0 = \mathbf{A}$, using orthogonal iteration, what are the $k = 0$ approximations for the eigenvalues?
(b) Continuing part (a), find $\mathbf{B}_1$ when using orthogonal iteration and the resulting approximations for the eigenvalues.
(c) Assuming $\mathbf{C}_0 = \mathbf{A}$, find $\mathbf{C}_1$ when using the QR method.
(d) When using the QR method, what matrix does the $\mathbf{C}_k$ matrices converge to?

4.7. Consider the symmetric matrix

$$\mathbf{A} = \begin{pmatrix} -7 & -6 \\ -6 & 2 \end{pmatrix} .$$

(a) Assuming $\mathbf{B}_0 = \mathbf{A}$, using orthogonal iteration, what are the $k = 0$ approximations for the eigenvalues?

(b) Continuing part (a), find $\mathbf{B}_1$ when using orthogonal iteration and the resulting approximations for the eigenvalues.
(c) Assuming $\mathbf{C}_0 = \mathbf{A}$, find $\mathbf{C}_1$ when using the QR method.
(d) When using the QR method, what matrix does the $\mathbf{C}_k$ matrices converge to?

4.8. Consider the symmetric matrix

$$\mathbf{A} = \begin{pmatrix} -2 & 0 \\ 0 & 2 \end{pmatrix}.$$

(a) What are the eigenvalues for this matrix? Explain why this matrix does not have a dominant eigenvalue.
(b) Suppose the power method is applied to $\mathbf{A}$. What does (4.13) reduce to? Explain why the method does not converge.
(c) Let $\mathbf{B} = \mathbf{A} - \mu\mathbf{I}$. Explain why the power method, when applied to $\mathbf{B}$, will converge for any nonzero value for μ. Also, explain how to pick values for μ to compute the two eigenvalues for $\mathbf{A}$ using the power method for each.

4.9. This exercise explores how to use the results from the power method to estimate $|\lambda_2|$.
(a) Explain how to use the computed values for v_k and the iteration error in Table 4.2 to estimate $|\lambda_2|$.
(b) The power method was used on a positive definite 3×3 matrix and the results are given in Table 4.17. What is, approximately, the second largest eigenvalue?
(c) Explain how the result from part (b), and the inverse power method, can be used to compute λ_2.

k	v_k	$\lvert v_k - v_{k-1}\rvert$
1	1.050490429082	
2	2.923452914041	1.87e+00
3	3.351559013702	4.28e−01
4	3.396446378825	4.49e−02
5	3.408195210234	1.17e−02
6	3.412146406763	3.95e−03
7	3.413503657073	1.36e−03
8	3.413969884337	4.66e−04
9	3.414129935923	1.60e−04
10	3.414184865200	5.49e−05

Table 4.17 Data for Exercise 4.9.

4.10. Suppose $\mathbf{A}_1$ and $\mathbf{A}_2$ are symmetric and positive definite 100×100 matrices with the following eigenvalues:

$\mathbf{A}_1$: $\lambda_1 = 100$, $\lambda_2 = 99$, $\lambda_3 = 98$, $\cdots$, $\lambda_{100} = 1$
$\mathbf{A}_2$: $\lambda_1 = 100$, $\lambda_2 = 10$, $\lambda_3 = 1$, $\cdots$, $\lambda_{100} = 10^{-97}$

(a) Will the power method most likely converge faster for $\mathbf{A}_1$ or $\mathbf{A}_2$?
(b) Assume that $\mathbf{A}$ is a symmetric positive definite $n \times n$ matrix with eigenvalues $\lambda_1 > \lambda_2 > \cdots > \lambda_n > 0$. If λ_1 is known, explain why, in theory, λ_n can be computed by applying the power method to $\mathbf{B} = \mathbf{A} - \lambda_1 \mathbf{I}$.
(c) To compute λ_{100} for $\mathbf{A}_1$, one can use the method from part (b) or the inverse power method. Which will most likely converge faster?
(d) Explain why the method in part (b) will likely fail when applied to $\mathbf{A}_2$.
(e) Show that, in theory, the method in part (b) converges faster than the inverse power method if $\lambda_1 < \lambda_{n-1} + \lambda_n$.

4.11. For the symmetric matrices in (4.5), (4.6), and (4.20), the dominant eigenvalue turns out to equal $||\mathbf{A}||_\infty$. Is this always true for a symmetric matrix?

4.12. By writing out the equation $\mathbf{Ax} = \lambda \mathbf{x}$ in component form, show that the eigenvalues of (4.20) are a, $a+1$, and $a-1$. What are the corresponding eigenvectors?

4.13. Suppose $\mathbf{A}$ is a symmetric 3×3 matrix with eigenvalues λ_1, λ_2, and λ_3, with $|\lambda_3| < |\lambda_2| < |\lambda_1|$. Suppose the starting vector for the power method contains a portion of an eigenvector for λ_1. Write out what happens to (4.12) and (4.13) in this 3×3 case. Also, what happens to the error (4.15) and the iterative error (4.16) formulas?

4.14. Suppose the eigenvalues of a symmetric matrix $\mathbf{A}$ satisfy $\lambda_1 > \lambda_2 \geq \cdots \geq \lambda_{n-1} > \lambda_n > 0$. To calculate λ_1, the power method is going to be applied to the shifted matrix $\mathbf{B} = \mathbf{A} - \omega \mathbf{I}$.
(a) What condition(s) must be imposed on ω so the method will converge to λ_1.
(b) Explain why the choice $\omega = (\lambda_2 + \lambda_n)/2$ will result in the fastest convergence of the power method.

4.15. This problem concerns what is known as the Rosser matrix, which is given as

$$\mathbf{R} = \begin{pmatrix} 611 & 196 & -192 & 407 & -8 & -52 & -49 & 29 \\ 196 & 899 & 113 & -192 & -71 & -43 & -8 & -44 \\ -192 & 113 & 899 & 196 & 61 & 49 & 8 & 52 \\ 407 & -192 & 196 & 611 & 8 & 44 & 59 & -23 \\ -8 & -71 & 61 & 8 & 411 & -599 & 208 & 208 \\ -52 & -43 & 49 & 44 & -599 & 411 & 208 & 208 \\ -49 & -8 & 8 & 59 & 208 & 208 & 99 & -911 \\ 29 & -44 & 52 & -23 & 208 & 208 & -911 & 99 \end{pmatrix}.$$

The eigenvalues of $\mathbf{R}$ are: 0, 1000, 1020, $\pm 10\sqrt{10405}$, $510 \pm 100\sqrt{26}$ [Rosser et al., 1951]. Note that the eigenvalue $\lambda = 1000$ has two linearly independent eigenvectors. Also, $\mathbf{R}$ is symmetric but it's not positive-definite.

(a) Show that four of the eigenvalues, in magnitude, are very close (although unequal).
(b) Explain why the power method will likely fail on this matrix.
(c) One can try shifting, and let $\mathbf{B} = \mathbf{A} - \omega\mathbf{I}$. If $\omega = -1$, what eigenvalue of $\mathbf{B}$ will the power method converge to, and what is the associated eigenvalue of $\mathbf{A}$?
(d) Using the shift from part (c), according to Theorem 4.2, by what factor is the error reduced by at each iteration step? How many iteration steps are needed to reduce the error by a factor of 10?
(e) The power method is to be applied to the shifted matrix $\mathbf{B} = \mathbf{A} - \omega\mathbf{I}$. What conditions need to be imposed on ω so it converges to $10\sqrt{10405} - \omega$?

4.16. In this problem the QR method for computing eigenvalues is considered. Select one of the following matrices and then answer the questions that follow.

$$\mathbf{A}_1 = \begin{pmatrix} -1 & 1 \\ 1 & 2 \end{pmatrix} \qquad \mathbf{A}_2 = \begin{pmatrix} 2 & 2 \\ 2 & 2 \end{pmatrix} \qquad \mathbf{A}_3 = \begin{pmatrix} 2 & -1 \\ -1 & 1 \end{pmatrix}$$

(a) Find $\mathbf{C}_1$.
(b) Find $\mathbf{C}_2$.
(c) What matrix does the $\mathbf{C}_k$'s converge to?

4.17. This exercise explores the connections between the QR method and the matrices in the factorization $\mathbf{A} = \mathbf{QDQ}^T$ (see Theorem 4.6). It's assumed that $\mathbf{A}$ is symmetric.

(a) In the QR method, show that $\mathbf{C}_1 = \mathbf{Q}_1^T\mathbf{A}\mathbf{Q}_1$, $\mathbf{C}_2 = \mathbf{Q}_2^T\mathbf{Q}_1^T\mathbf{A}\mathbf{Q}_1\mathbf{Q}_2$, and in general, $\mathbf{C}_k = \mathbf{P}_k^T\mathbf{A}\mathbf{P}_k$, where $\mathbf{P}_k = \mathbf{Q}_1\mathbf{Q}_2 \cdots \mathbf{Q}_k$. Note that $\mathbf{P}_k$ is an orthogonal matrix (you do not need to show this).
(b) Assuming that $\mathbf{C}_k$ converges to a diagonal matrix, explain why the QR method is a procedure for computing the matrix $\mathbf{D}$ in Theorem 4.6. Also explain how two lines of code can be added to the QR algorithm given in Section 4.3.3 so that, when finished, you have a matrix containing the eigenvectors.

4.18. Consider the matrix

$$\mathbf{A} = \begin{pmatrix} 1 & 2 \\ 1 & 2 \end{pmatrix}.$$

(a) Find a SVD for $\mathbf{A}$.
(b) Find the expansion for $\mathbf{A}$ given in (4.49).
(c) Find a SVD for $\mathbf{A}^T$.

(d) Find a SVD for $2\mathbf{A}$.
(e) Find a SVD for $-\mathbf{A}$.
(f) Find $||\mathbf{A}||_2$.

4.19. Consider the symmetric matrix

$$\mathbf{A} = \begin{pmatrix} 4 & 4 \\ 4 & -2 \end{pmatrix} .$$

(a) Find a factorization as given in (4.38).
(b) Find a SVD for $\mathbf{A}$.
(c) Find the expansion for $\mathbf{A}$ given in (4.49).
(d) Find $||\mathbf{A}||_2$.

4.20. Pick one of the following matrices, and then find the requested quantities.

$$i)\ \ \mathbf{A} = \begin{pmatrix} 0 & 1 \\ 0 & 2 \\ 0 & 0 \end{pmatrix} \qquad ii)\ \ \mathbf{A} = \begin{pmatrix} 1 & 2 \\ 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad iii)\ \ \mathbf{A} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$

(a) Find a SVD for $\mathbf{A}$.
(b) Find the expansion for $\mathbf{A}$ given in (4.49).
(c) Find a SVD for $3\mathbf{A}$.
(d) Find a SVD for $-\mathbf{A}$.
(e) Find $||\mathbf{A}||_2$.

4.21. The pixel values for different images are given in Table 4.18 (they each contain 4 pixels). Select one of them and then answer the questions that follow.
(a) What is the SVD of the corresponding matrix?
(b) Find $\mathbf{W}_1$ and $\mathbf{W}_2$, as defined in (4.49).
(c) Suppose the approximation $\mathbf{A} \approx \mathbf{A}_1$ is used, where $\mathbf{A}_1 = \sigma_1 \mathbf{W}_1$. Find $\mathbf{A}_1$, and determine the relative error is using this matrix to approximate $\mathbf{A}$.

4.22. This problem considers the following $n \times n$ symmetric tri-diagonal matrix:

A)

2	1
0	1

B)

0	1
1	2

C)

1	0
1	2

Table 4.18 Data for Exercise 4.21.

$$\mathbf{A} = \begin{pmatrix} 0 & c_1 & & & & \\ c_1 & 0 & c_2 & & & \\ & c_2 & 0 & c_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & c_{n-2} & 0 & c_{n-1} \\ & & & & c_{n-1} & 0 \end{pmatrix},$$

where

$$c_i = \frac{i}{\sqrt{(2i-1)(2i+1)}}\,.$$

The eigenvalues, and eigenvectors, of this matrix play a role in Gaussian quadrature (see Section 6.4.4).

(a) Taking $n = 2$, use orthogonal iteration to calculate the eigenvalues of $\mathbf{A}$. The relative iteration error for the nonzero eigenvalues should be less than 10^{-12}. Also, calculate the eigenvalues by hand to make sure your computed answers are correct. Comment on any modifications of the orthogonal iteration procedure you make to compute the correct result.

(b) Let $\mathbf{q}_1 = (q_{11}, q_{12})^T$ and $\mathbf{q}_2 = (q_{21}, q_{22})^T$ be the orthonormal eigenvectors calculated in part (a). As explained in Section 6.4.2, when $n = 2$ one should have $2q_{11}^2 = 1$ and $2q_{21}^2 = 1$. How close does your result come to satisfying these conditions?

(c) Taking $n = 10$, use orthogonal iteration to calculate the eigenvalues of $\mathbf{A}$. The relative iteration error for the nonzero eigenvalues should be less than 10^{-12}. Explain why you are confident that the values are correct.

(d) Assume that the orthonormal eigenvectors are $\mathbf{q}_1, \mathbf{q}_2, \cdots, \mathbf{q}_n$, where $\mathbf{q}_j$ is the eigenvector for eigenvalue λ_j. If $\mathbf{q}_j = (q_{j1}, q_{j2}, \cdots, q_{jn})^T$, calculate the quantity $\overline{w}_j = 2q_{j1}^2$ for each eigenvalue. Although this looks a little odd, the $\overline{w}_j$'s are used to determine the weights in the Gaussian quadrature formula.

4.23. The equation for a string on an elastic foundation is

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2} - ku\,,$$

where c and k are positive constants. The problem for finding the natural frequencies of the string can be reduced to solving $\mathbf{Ax} = \lambda \mathbf{x}$, where

$$\mathbf{A} = \begin{pmatrix} a & -1 & & & & \\ -1 & a & -1 & & 0 & \\ & -1 & a & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & 0 & & & & -1 \\ & & & & -1 & a \end{pmatrix}.$$

Note that $\mathbf{A}$ is an $n \times n$ symmetric tri-diagonal matrix, and

$$a = 2 + \frac{k}{c^2(n+1)^2} .$$

Also, by solving the string equation one finds that the exact values for the λ's are

$$\bar{\lambda}_i = \frac{k + c^2(i\pi)^2}{c^2(n+1)^2} , \text{ for } i = 1, 2, 3, \cdots .$$

(a) Explain why **A** is positive definite.
(b) Use inverse orthogonal iteration to compute the five smallest eigenvalues of **A**. To do this take $n = 250$ and $c = k = 1$. Also, compute the relative error as in Table 4.16.
(c) The exact eigenvalues of **A** are $\lambda_i = a + 2\cos(i\theta)$, for $i = 1, 2, \ldots, n$, where $\theta = \pi/(n+1)$. Taking $n = 250$ and $c = k = 1$, as in Figure 4.4, plot the ratios λ_{i-1}/λ_i and use this to explain why inverse orthogonal iteration should converge relatively quickly when computing the smaller eigenvalues.

4.24. This exercise involves finding the natural frequencies for the coupled oscillators in Figure 4.10. The equation of motion in this case is $\mathbf{y}'' + \mathbf{K}\mathbf{y} = \mathbf{0}$, where $\mathbf{y} = (y_1(t), y_2(t), y_3(t))^T$ and

$$\mathbf{K} = \begin{pmatrix} 1 + k_{12} + k_{13} & -k_{12} & -k_{13} \\ -k_{12} & 1 + k_{12} + k_{23} & -k_{23} \\ -k_{13} & -k_{23} & 1 + k_{13} + k_{23} \end{pmatrix} .$$

In the above matrix, k_{ij} is the spring constant for the spring connecting the ith and jth oscillator, and it is positive (these are the three smaller springs shown in Figure 4.10). Assuming $\mathbf{y} = \mathbf{x}e^{I\omega t}$, where $I = \sqrt{-1}$, then the problem reduces to solving $\mathbf{K}\mathbf{x} = \lambda\mathbf{x}$, where $\lambda = \omega^2$.
(a) Show that **K** is positive definite.
(b) Using orthogonal iteration, compute the eigenvalues of **K** in the case of when $k_{ij} = 1/10$. Also, state what stopping condition you used, and how many iteration steps were required.

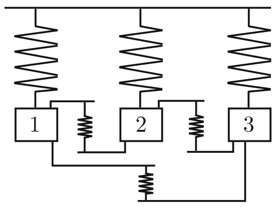


Figure 4.10 Three oscillators that are coupled by springs, as an example of the problem considered in Exercise 4.24.

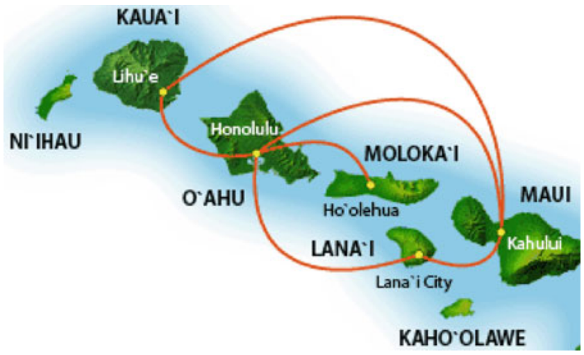


Figure 4.11 Six air routes, and five cities, used to construct the adjacency matrix in Exercise 4.25.

(c) What is the eigenvalue for $\mathbf{K}$ when the oscillators are uncoupled? Note that this means that the k_{ij}'s are zero.

(d) Using orthogonal iteration, compute the eigenvalues of $\mathbf{K}$ in the case of when $k_{ij} = 1/1000$. You should use the same stopping condition as in part (b). How many iteration steps were required, and if significantly different than the number for part (b), explain why.

4.25. The routes for a small airline are shown in Figure 4.11. In terms of a network, the five cities are the vertices or nodes, and the six air routes are the connections, in a similar manner to those shown in Figure 4.11.

(a) Number the cities from 1 to 5, and from this write down the corresponding adjacency matrix.

(b) Compute the eigenvalues of the matrix you found in part (a). Also, it can be proved that if you sum up the eigenvalues of an adjacency matrix, you get zero (see Section 4.3.4). Do your values satisfy this condition? If not, provide a reason why.

(c) Number the cities in a different order than you used in part (a) and compute the eigenvalues of the resulting adjacency matrix. How do these differ from what you computed in part (b)?

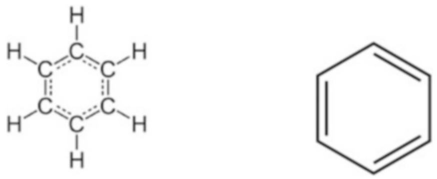


Figure 4.12 Diagrammatic representations of benzene. The representation on the right is the one used to construct the adjacency matrix in Exercise 4.26.

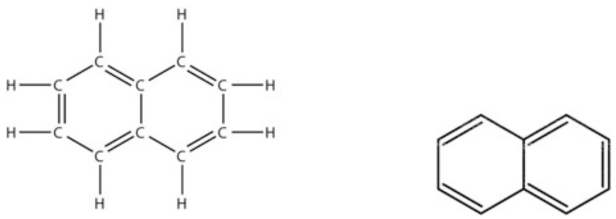


Figure 4.13 Diagrammatic representations of naphthalene. The representation on the right is the one used to construct the adjacency matrix in Exercise 4.26.

4.26. Adjacency matrices arise in quantum chemistry in the form of what are called Hückel Hamiltonian matrices. To illustrate, the atomic configuration for benzene is shown in Figure 4.12. The atoms are located at the vertices, and they are connected by nearest neighbor bonds (which form the edges of the polygon). The Hückel Hamiltonian matrix for benzene is nothing more than the adjacency matrix for this graph. In this context, the eigenvalues are associated with the energy states of the system, and the eigenvectors provide information about the corresponding orbital structure for that particular energy.

(a) Using the representation on the right in Figure 4.12, number the atoms (vertices) from 1 to 6 and write down the corresponding adjacency matrix for benzene. The double lines in this graph should be treated the same as the single line connections.

(b) Compute the eigenvalues of the matrix you found in part (a). Also, it can be proved that if you sum up the eigenvalues of an adjacency matrix, you get zero (see Section 4.3.4). Do your values satisfy this condition? If not, provide a reason why.

(c) The atomic configuration for naphthalene is shown in Figure 4.13. Number the atoms (vertices) from 1 to 10 and write down the corresponding adjacency matrix. From this compute the eigenvalues of the matrix. Also, it can be proved that if you sum up the eigenvalues of an adjacency matrix, you get zero. Do your values satisfy this condition? If not, provide a reason why.

4.27. This problem considers finding the eigenvalues of an $n \times n$ symmetric matrix $\mathbf{A}$ whose entries are given as

$$a_{ij} = \alpha e^{-\pi\alpha^2(i-j)^2},$$

where $\alpha = 20/(n-1)$. In this problem, take $n = 100$.

(a) By computing the needed quantities in Theorem 3.4, show the matrix is positive definite. Also, explain why the matrix is symmetric.

(b) Use the power method to calculate the dominant eigenvalue to four significant digits. Make sure to state your stopping condition, and how many iterations it took to find the eigenvalue.

(c) Explain how to use the iterative error in part (b) to estimate the second largest eigenvalue λ_2. Using this estimate, and shifting, calculate λ_2 to four significant digits. Make sure to state your stopping condition, and how many iterations it took to find the eigenvalue.
(d) Compute the smallest eigenvalue of $\mathbf{A}$ to four significant digits. Make sure to state the method you used, the stopping condition, and how many iterations it took to find the eigenvalue.

4.28. This problem considers finding the eigenvalues of an $n \times n$ symmetric matrix $\mathbf{A}$ whose entries are given as $a_{ij} = |i - j|$.
(a) Write out the matrix in the case of when $n = 5$, and explain why the matrix is not positive definite.
(b) Taking $n = 1000$, use the power method, and shifting if necessary, to calculate the largest and the smallest eigenvalue of $\mathbf{A}$. You should calculate each eigenvalue to four significant digits. Also, explain why you know you have calculated the correct eigenvalues.
(c) Taking $n = 1000$, calculate the eigenvalue of $\mathbf{A}$ that is closest to zero. You should calculate the eigenvalue to four significant digits. Also, explain why you know you have calculated the correct eigenvalue.

4.29. An often used age-structured model of a population divides the number of females into age groups $g_1, g_2, \cdots, g_n$. Here g_1 is the number in the youngest group, g_2 is the number in the second youngest group, etc. It is assumed that after a given time interval, $s_i g_i$ of those in g_i survive and move to age group g_{i+1}. Also, over the same time interval, the females in g_i produce $b_i g_i$ female babies. Setting $\mathbf{g} = (g_1, g_2, \cdots, g_n)^T$, with $\mathbf{g}_k$ being the value at time step t_k and $\mathbf{g}_{k+1}$ being the value at time step t_{k+1}, then $\mathbf{g}_{k+1} = \mathbf{A}\mathbf{g}_k$, where

$$\mathbf{A} = \begin{pmatrix} b_1 & b_2 & b_3 & & \cdots & b_n \\ s_1 & 0 & 0 & & \cdots & 0 \\ 0 & s_2 & 0 & & \cdots & 0 \\ \vdots & & \ddots & & & \vdots \\ 0 & \cdots & & s_{n-2} & 0 & 0 \\ 0 & \cdots & & 0 & s_{n-1} & 0 \end{pmatrix}.$$

This is known as a Leslie matrix. It is assumed that the s_i's are positive, b_1 and b_n are non-negative, and the other b_i's are positive. In this case, the dominant eigenvalue λ_1 of $\mathbf{A}$ is positive. If $\lambda_1 > 1$, then the population increases, and it decreases if $\lambda_1 < 1$. Also, note that this matrix is not symmetric, but it has n linearly independent eigenvectors.
(a) Deer can survive up to 20 years in the wild. Taking $n = 20$, and assuming their survivability deceases with age, let $s_i = \exp(-i/100)$. Also, assuming 3/4 of the females in each age group have one offspring each year, with equal probability of being male or female, then $b_i = (3/4)(1/2) = 3/8$. The exception is the youngest group, and for this assume that $b_1 = 0$. Does the population increase or decrease?

(b) If $\lambda_1 = 1$, then the population approaches a constant value as time increases. For the deer in part (a), assuming $b_1 = 0$ and $b_2 = b_3 = \cdots = b_n = b$, what does b have to be so this happens? The value you find should be correct to six significant digits.

4.30. An important factorization in mechanics is the polar decomposition. For an $n \times n$ matrix $\mathbf{A}$ with a positive determinant, the factorization is $\mathbf{A} = \mathbf{QP}$, where $\mathbf{Q}$ is an orthogonal matrix and $\mathbf{P}$ is a symmetric positive definite matrix.

(a) Assuming the SVD of $\mathbf{A}$ has been computed, explain how this can be used to compute $\mathbf{Q}$ and $\mathbf{P}$. Make sure to explain why the formulas you derive for $\mathbf{Q}$ and $\mathbf{P}$ guarantee that they have their required properties.

(b) According to Theorem 4.4, $\det(\mathbf{Q}) = \pm 1$. An orthogonal matrix with $\det(\mathbf{Q}) = -1$ corresponds to a reflection, and these are considered to be unphysical. For this reason, in mechanics one is interested in having $\det(\mathbf{Q}) = 1$, which corresponds to what is known as a proper orthogonal matrix and physically they correspond to rotations. Explain how to modify, if necessary, your algorithm or formulas in part (a) so that $\mathbf{Q}$ is a proper orthogonal matrix.

4.31. This problem considers the following $n \times n$ tri-diagonal matrix:

$$\mathbf{A} = \begin{pmatrix} a & c & & & & \\ b & a & c & & 0 & \\ & b & a & c & & \\ & & \ddots & \ddots & \ddots & \\ & 0 & & b & a & c \\ & & & & b & a \end{pmatrix}.$$

It is assumed that $bc > 0$ and $n \geq 3$.

(a) Assuming $c = b$, the eigenvalues of $\mathbf{A}$ are $\lambda_i = a + 2|b|\cos(i\theta)$, for $i = 1, 2, \ldots, n$, where $\theta = \pi/(n+1)$. Show that

$$\mathbf{x}_i = (\sin(\phi), \sin(2\phi), \sin(3\phi), \cdots, \sin(n\phi))^T$$

is an eigenvector for λ_i, where $\phi = i\theta$.

(b) The matrix in part (a) is symmetric. Explain why it is positive definite if $a \geq 2|b|$.

(c) For the eigenvectors in part (a), show that $\mathbf{x}_i \cdot \mathbf{x}_j = 0$ if $i \neq j$, and $\mathbf{x}_i \cdot \mathbf{x}_i = (n+1)/2$.

(d) The eigenvalues of $\mathbf{A}$ are $\lambda_i = a + 2\sqrt{bc}\cos(i\theta)$, for $i = 1, 2, \ldots, n$, where $\theta = \pi/(n+1)$. Show that

$$\mathbf{x}_i = (\kappa \sin(\phi), \kappa^2 \sin(2\phi), \kappa^3 \sin(3\phi), \cdots, \kappa^n \sin(n\phi))^T$$

is an eigenvector for λ_i, where $\phi = i\theta$ and $\kappa = \sqrt{b/c}$.

Chapter 5
Interpolation

5.1 Information from Data

The topic of this chapter is interpolation, which relates to passing a curve through a set of data points. To put this in context, extracting information from data is one of the central objectives in science and engineering and exactly what or how this is done depends on the particular setting. Two examples are shown in Figure 5.1. Figure 5.1(L) contains data obtained from measurements of high redshift type supernovae. As is often the case with computerized testing systems, there are many data points and there is some scatter in the values obtained. Because of this one would not be interested in finding a function that passes through all of these points, but rather a function that behaves in a qualitatively similar manner as the data. In this case one uses a fitting method, like least squares, to make the connection more quantitative. Exactly how this might be done will be considered in Chapter 8.

In comparison, the data in Figure 5.1(R) have a well-defined shape and for this reason are more amenable to interpolation. This is also true for the data shown in Figure 5.2. The hand data is typical of what arises in CAD applications, while the data on the right relates to a more mathematical application. To explain, the data points are obtained by evaluating the function

$$f(x) = \frac{1}{3} + \sum_{n=1}^{\infty} \frac{4(-1)^n}{n^{4/3}\pi^2} \cos(n\pi x) \tag{5.1}$$

at 10 points from the interval $-1 \leq x \leq 1$. What is seen is that the above relatively complicated function does not have a correspondingly complicated graph. This raises the question if we might be able to replace the function with a much simpler expression that would serve as a respectable approximation of the original.

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_5

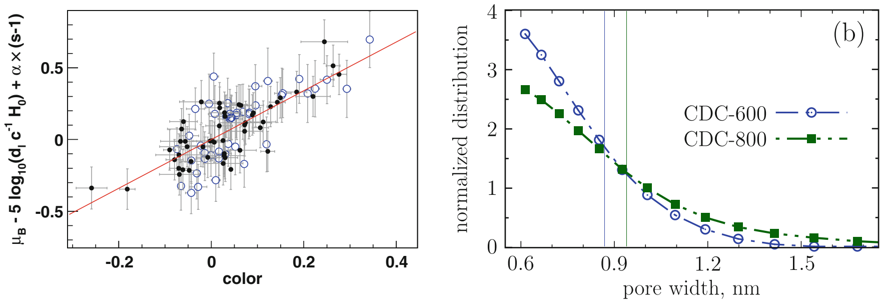


Figure 5.1 Left: data related to a supernovae redshift [Astier et al., 2006]. Right: data for nanopores in a supercapacitor [Kondrat et al., 2012].

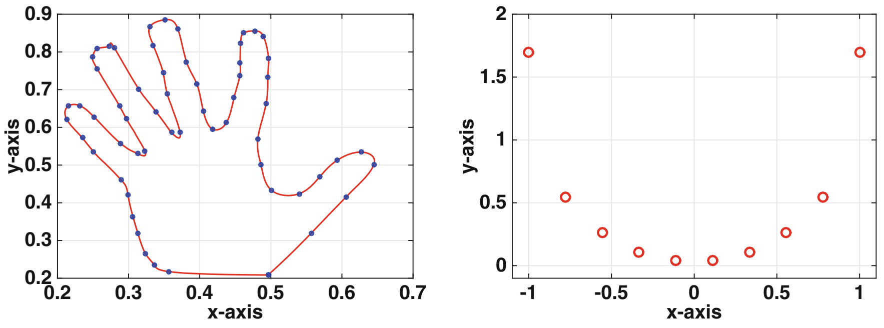


Figure 5.2 Left: geometric representation of a hand using interpolation [Burkardt, 2015]. Right: values of (5.1) at selected points in interval $-1 \leq x \leq 1$.

One of the better ways to test how well an interpolation method works is to try it out on different data sets. In what follows we will use the sets shown in Figure 5.3. Each consists of 12 equally spaced points over the interval $-1 \leq x \leq 1$. The top two were generated using functions; the one in the upper left comes from the 5th order polynomial

$$y(x) = (x + 0.9)(x + 0.1)^2(x - 0.2)(x - 0.8),$$

while the one on the upper right consists of points that lie on the circle $x^2 + y^2 = 1$. The lower two are used to mimic or resemble a periodic function and one with jumps. It is recommend that you spend a moment or two and sketch in what you think would be an acceptable interpolation function for each data set. This will help later when we see what the standard interpolation methods produce.

It is of interest to know that many of the interpolation methods derived in this chapter are summarized in Appendix C, Table C.1.

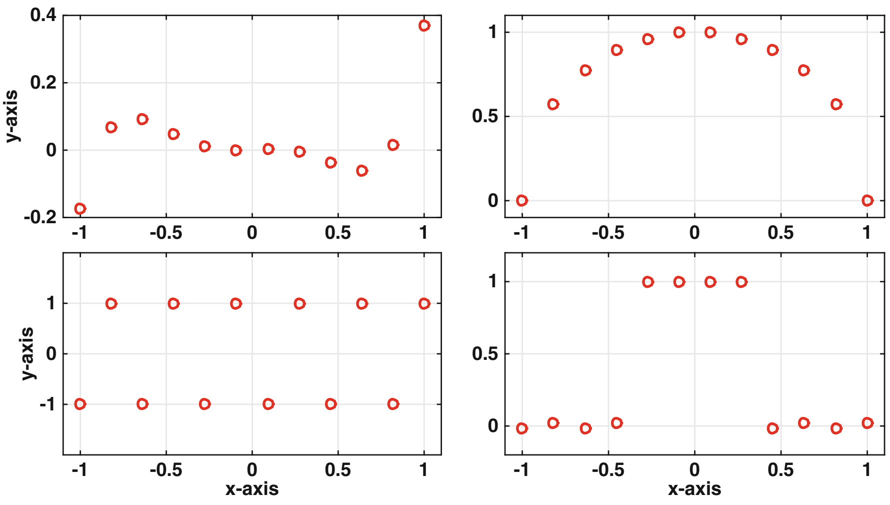


Figure 5.3 Data sets that are used to test the various interpolation methods.

5.2 Global Polynomial Interpolation

If one wants to find a function to connect two data points, the easiest choice is to use a straight line, in other words a linear function. Similarly, given three data points one would likely use a quadratic. To generalize this idea, suppose there are $n+1$ points and they are $(x_1, y_1), (x_2, y_2), \cdots, (x_{n+1}, y_{n+1})$, where $x_1 < x_2 < \cdots < x_{n+1}$. We are going to find a single nth degree polynomial $p_n(x)$ that passes through each and every point. This is not particularly difficult and there are several ways to find the interpolation polynomial. However, as is often the case in numerical computing, some methods are much more sensitive to round-off error than other methods.

5.2.1 Direct Approach

Taking the direct approach, the simplest choice is to take

$$p_n(x) = a_0 + a_1 x + \cdots + a_n x^n. \tag{5.2}$$

The interpolation requirement is that $p_n(x_1) = y_1$, $p_n(x_2) = y_2$, $\cdots$, $p_n(x_{n+1}) = y_{n+1}$. Using the above polynomial this produces the equations

$$\begin{aligned} a_0 + a_1 x_1 + \cdots + a_n x_1^n &= y_1 \\ a_0 + a_1 x_2 + \cdots + a_n x_2^n &= y_2 \\ &\vdots \\ a_0 + a_1 x_{n+1} + \cdots + a_n x_{n+1}^n &= y_{n+1} \end{aligned}$$

This can be rewritten in matrix form as $\mathbf{Va} = \mathbf{y}$, where $\mathbf{a} = (a_0, a_1, \cdots, a_n)^T$, $\mathbf{y} = (y_1, y_2, \cdots, y_{n+1})^T$, and

$$\mathbf{V} = \begin{pmatrix} 1 & x_1 & x_1^2 & \dots & x_1^n \\ 1 & x_2 & x_2^2 & \dots & x_2^n \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_{n+1} & x_{n+1}^2 & \dots & x_{n+1}^n \end{pmatrix}. \tag{5.3}$$

This is a *Vandermonde matrix*. Given that it has a name you should not be surprised that it plays an important role in interpolation, but as will be seen shortly not all of its contributions are good.

Example

Each of the test data sets in Figure 5.3 contains 12 points. Fitting $p_{11}(x)$ to each set produces the curves shown in Figure 5.4. The top two look to be reasonable fits to the data while the bottom two are not. The over- and under-shoots seen in the bottom two curves often appear with higher degree polynomials and one of the drawbacks of using a global polynomial with larger data sets. ■

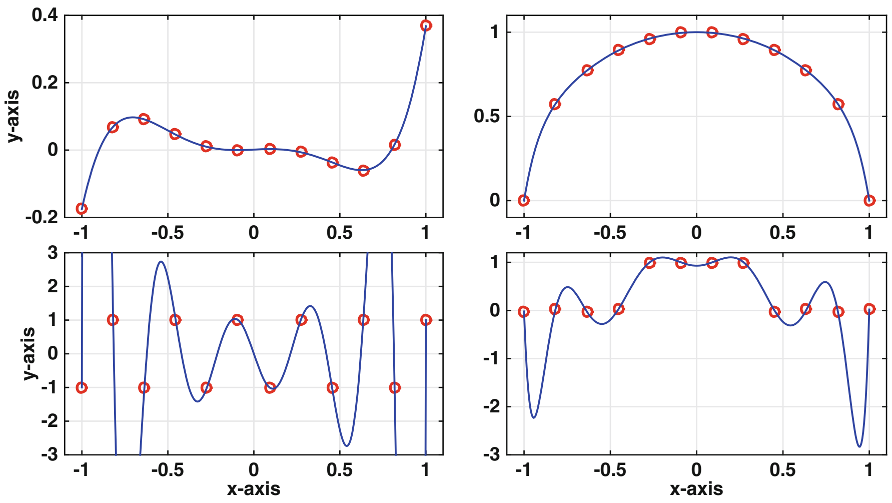


Figure 5.4 Using a global polynomial $p_{11}(x)$, as given in (5.2), for the data in Figure 5.3.

The pervious example indicates that there are concerns about using a global polynomial, particularly when you have a large number of data points. In fact, there are significant problems not evident in the example related to the condition number of the Vandermonde matrix. As shown in Section 3.4, $\mathbf{V}$ can be ill-conditioned for even small values of n. It is possible in some cases to rescale the data to improve the condition number, and an example of this is given in Exercise 5.28. However, it is possible to avoid this particular problem altogether, and this will be considered next. Even so, be warned that there is a second problem with a large number of equally spaced data points and this is discussed in Section 5.2.3.

5.2.2 Lagrange Approach

The easiest way to explain how to avoid using the Vandermonde matrix is to examine what happens with linear and quadratic functions. So, suppose the data points are (x_1, y_1) and (x_2, y_2), where $x_1 \neq x_2$. The global polynomial in this case is linear and it can be written as

$$\begin{aligned} p_1(x) &= y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) \\ &= y_1 \frac{x - x_2}{x_1 - x_2} + y_2 \frac{x - x_1}{x_2 - x_1} \\ &= y_1 \ell_1(x) + y_2 \ell_2(x), \end{aligned}$$

where

$$\ell_1(x) = \frac{x - x_2}{x_1 - x_2},$$

and

$$\ell_2(x) = \frac{x - x_1}{x_2 - x_1}.$$

The functions $\ell_1(x)$ and $\ell_2(x)$ are examples of *linear Lagrange interpolation functions* and they have the properties that $\ell_1(x_1) = \ell_2(x_2) = 1$, $\ell_1(x_2) = 0$, and $\ell_2(x_1) = 0$. In other words, each $\ell_i(x)$ is linear, equal to one when $x = x_i$ and equal to zero at the other x_j data point.

It is relatively easy to generalize this idea and write down the quadratic Lagrange interpolation functions. Namely, if the data points are (x_1, y_1), (x_2, y_2), and (x_3, y_3) then

$$\begin{aligned} \ell_1(x) &= \frac{(x - x_2)(x - x_3)}{(x_1 - x_2)(x_1 - x_3)}, \\ \ell_2(x) &= \frac{(x - x_1)(x - x_3)}{(x_2 - x_1)(x_2 - x_3)}, \\ \ell_3(x) &= \frac{(x - x_1)(x - x_2)}{(x_3 - x_1)(x_3 - x_2)}. \end{aligned}$$

By design, each $\ell_i(x)$ is quadratic, equal to one when $x = x_i$ and equal to zero at the other x_j data points. Using these functions the corresponding quadratic interpolation polynomial is

$$p_2(x) = y_1\ell_1(x) + y_2\ell_2(x) + y_3\ell_3(x). \tag{5.4}$$

For this to be well-defined it is required that the x_i's are distinct, which means that $x_1 \neq x_2$, $x_1 \neq x_3$, and $x_2 \neq x_3$. Also, although it looks different, (5.4) produces the same function as given in (5.2) in the case of when $n = 2$.

Generalizing the above results we have that given data points (x_1, y_1), $(x_2, y_2), \cdots, (x_{n+1}, y_{n+1})$, with the x_i's distinct, the interpolation polynomial can be written as

$$p_n(x) = \sum_{i=1}^{n+1} y_i\ell_i(x), \tag{5.5}$$

where the *Lagrange interpolation functions* are defined as

$$\begin{aligned}\ell_i(x) &= \frac{(x-x_1)(x-x_2)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_{n+1})}{(x_i-x_1)(x_i-x_2)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_{n+1})} && (5.6)\\ &= \prod_{\substack{j=1\\ j\neq i}}^{n+1} \frac{x-x_j}{x_i-x_j}\,. && (5.7)\end{aligned}$$

In a similar manner, as in the linear and quadratic examples, each $\ell_i(x)$ is an nth degree polynomial, it is equal to one when $x = x_i$, and it equals zero when $x = x_j$ for $j \neq i$.

Example

To find the global polynomial that interpolates the data in Table 5.1, first note that the data is $(x_1, y_1) = (0, 1)$, $(x_2, y_2) = (1/2, -1)$, and $(x_1, y_1) = (1, 2)$. Consequently, the polynomial is

$$\begin{aligned} p_2(x) &= y_1\ell_1(x) + y_2\ell_2(x) + y_3\ell_3(x)\\ &= \ell_1(x) - \ell_2(x) + 2\ell_3(x), \end{aligned}$$

where $\ell_1(x) = 2(x-1/2)(x-1)$, $\ell_2(x) = -4x(x-1)$, and $\ell_3(x) = 2x(x-1/2)$. ■

x	0	1/2	1
y	1	-1	2

Table 5.1 Data for example.

The Lagrange interpolation formulas in (5.5) and (5.7) have an advantage over the direct formula, given in (5.2), in that the Vandermonde matrix is avoided. There is still a potential computational problem related to overflow, particularly for (5.6). This is the same problem explored in Exercise 1.7. It can help to rescale the data, and this is explained in Exercise 5.28. However, writing the formula in factored form as in (5.7) significantly reduces the possibility of overflow.

The price paid for avoiding the Vandermonde matrix is the effort needed to evaluate the ℓ_i's, and this is often stated to be a drawback of the method. For a large number of data points, evaluating $\ell_i(x)$ can require about $2n^2$ flops. To translate this into computing time, if you use 20 data points and 2,000 evaluation points, the computing time is about 1 msec. Similarly, if you use 200 data points and 20,000 evaluation points, the computing time is about 1 sec. In other words, the computational time is not particularly significant unless you are working with a large data set. In such cases there are more efficient ways to write the interpolation formulas, using something called barycentric weights, and these are explored in Exercise 5.29. However, there is a more significant problem with this method and this is explained next.

5.2.3 Runge's Function

Now that the ill-conditioned matrix problem has been avoided it is time to explain the other problem with using a global interpolation polynomial. For this we can use the top two plots in Figure 5.4. It is seen that with the 10 data points we have obtained a fairly accurate approximation of the original functions. Always trying to improve things, one might think that by adding data points that the approximation will be even better. For many functions this does indeed happen but there are functions where it does not (and you would think it should). The example many use to demonstrate this is

$$f(x) = \frac{1}{1+25x^2}\,, \tag{5.8}$$

and this is known as *Runge's function.* This is plotted in Figure 5.5, along with $p_4(x)$ and $p_{12}(x)$. The interpolation polynomials are constructed using equally spaced data points. It is seen that in the center of the interval the approximation improves but it gets worse towards the endpoints. Increasing the number of data points makes the situation worse in the sense that the magnitude of $|p_n(x)|$ near the endpoints increases. For example, when $n = 40$ the maximum in $|p_n(x)|$ is about 10^4, while for $n = 100$ the maximum in $|p_n(x)|$ is about 10^{14}. Moreover, this behavior is not limited to equally spaced points. If you take the points randomly from the interval, the maximum in $|p_n(x)|$ is often even larger.

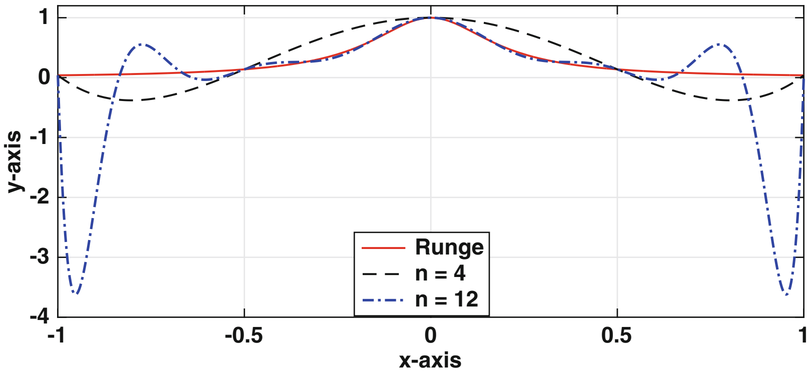


Figure 5.5 The Runge function (5.8) along with two interpolating polynomials.

The conclusion from the above discussion is that using a global interpolation polynomial works well for small data sets but has limited value as the number of data points increases. One solution for larger data sets is to break them into small groups, use interpolation on the subgroups, and then connect the information into a coherent whole. This is considered in the next section. If you are set on using a global polynomial then you need to consider where the x_i's are placed in the interval, and this is considered in Section 5.5.4.

5.3 Piecewise Linear Interpolation

We will consider using linear interpolation between adjacent data points. This is effectively what is done in a child's connect the dots puzzle. An example is shown in Figure 5.6 where the line between (x_1, y_1) and (x_{16}, y_{16}) is predrawn in the puzzle.

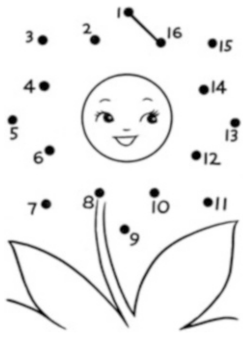


Figure 5.6 A typical child's puzzle of connect the dots.

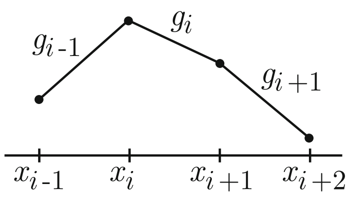


Figure 5.7 Intervals and functions used for piecewise linear interpolation.

To be able to write down the mathematical formula used for piecewise linear interpolation, assume that the data points are (x_1, y_1), (x_2, y_2), $\cdots$, (x_{n+1}, y_{n+1}), where $x_1 < x_2 < \cdots < x_{n+1}$. The linear function connecting adjacent data points is given as

$$g_i(x) = y_i + \frac{y_{i+1} - y_i}{x_{i+1} - x_i}(x - x_i), \quad \text{for } x_i \le x \le x_{i+1}. \tag{5.9}$$

Assembling these into a complete description of the interpolation function we get

$$g(x) = \begin{cases} g_1(x) & \text{if } x_1 \le x \le x_2 \\ g_2(x) & \text{if } x_2 \le x \le x_3 \\ \vdots & \qquad \vdots \\ g_n(x) & \text{if } x_n \le x \le x_{n+1}. \end{cases} \tag{5.10}$$

An illustration of this is given in Figure 5.7.

It is possible to write $g(x)$ is a form that can be easier to use, and looks a lot simpler than the expression in (5.10). To do this we introduce the piecewise linear function $G_i(x)$, with $G(x_i) = 1$ and $G(x_j) = 0$ if $j \neq i$. The formula for this function is

$$G_i(x) = \begin{cases} 0 & \text{if } \quad x \le x_{i-1}, \\ \dfrac{x - x_{i-1}}{x_i - x_{i-1}} & \text{if } \quad x_{i-1} \le x \le x_i, \\ \dfrac{x - x_{i+1}}{x_i - x_{i+1}} & \text{if } \quad x_i \le x \le x_{i+1}, \\ 0 & \text{if } \quad x_{i+1} \le x, \end{cases} \tag{5.11}$$

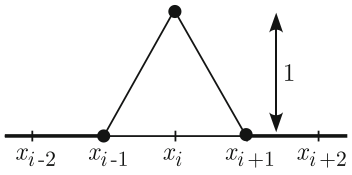


Figure 5.8 The piecewise linear function $G_i(x)$ defined in (5.11).

and a sketch of this is given in Figure 5.8. In the case of when the points are equally spaced, so $x_{i+1} - x_i = h$, this can be written in a more compact form as

$$G_i(x) = G\left(\frac{x - x_i}{h}\right), \tag{5.12}$$

where

$$G(x) = \begin{cases} 1 - |x| & \text{if} \quad |x| \le 1, \\ 0 & \text{if} \quad 1 \le |x|. \end{cases} \tag{5.13}$$

Note that the defining properties of $G_i(x)$ are very similar to the properties that were used to define the Lagrange interpolation functions $\ell_i(x)$ in Section 5.2.2.

With this, the piecewise linear interpolation function (5.10) can be written as

$$g(x) = \sum_{i=1}^{n+1} y_i G_i(x). \tag{5.14}$$

Just so it's clear, this expression produces the same interpolation function as the expanded version given in (5.10). Also, because of its shape, $G_i(x)$ has a variety of names, and they include the *hat function* and the *chapeau function*.

As a final comment, to define G_1 it is necessary to introduce x_0, with $x_0 < x_1$, and for G_{n+1} we need to add in x_{n+2}, with $x_{n+1} < x_{n+2}$. Exactly where these two points are located in not important because they have no affect on the interpolation function over the interval $x_1 \le x \le x_{n+1}$.

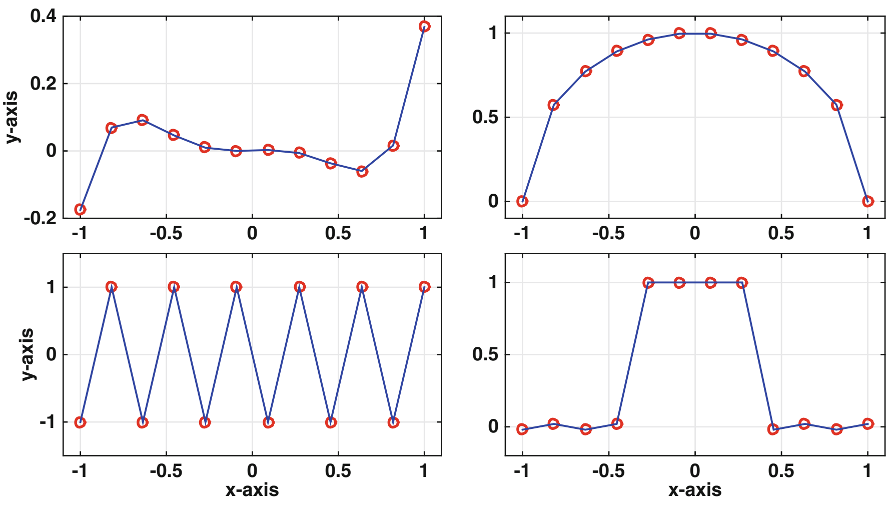


Figure 5.9 Using a piecewise linear function to fit the data in Figure 5.3.

Example

Using a piecewise linear function to fit the data in Figure 5.9 produces the curves shown in Figure 5.9. These curves are not bad fits but they are also not great. They are not bad because they do not contain the over- and undershoots seen in Figure 5.9. However, they are not great because they are jagged. Note that in some applications, such as the one in Figure 5.6, jagged is what is desired but in many applications this is something one wants to avoid. ■

Example

Find the piecewise linear function that interpolates the data in Table 5.2. Note that in this case, $x_1 = 0$, $x_2 = 1/2$, and $x_2 = 1$.

Method 1: Using (5.10),

$$g(x) = \begin{cases} g_1(x) & \text{if } 0 \le x \le 1/2 \\ g_2(x) & \text{if } 1/2 \le x \le 1, \end{cases}$$

where

$$\begin{aligned} g_1(x) &= y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) \\ &= 1 - 4x, \end{aligned}$$

and

$$\begin{aligned} g_2(x) &= y_2 + \frac{y_3 - y_2}{x_3 - x_2}(x - x_2) \\ &= -4 + 6x. \end{aligned}$$

Method 2: Using (5.14),

$$\begin{aligned} g(x) &= y_1 G_1(x) + y_2 G_2(x) + y_3 G_3(x) \\ &= G_1(x) - G_2(x) + 2G_3(x), \end{aligned}$$

x	0	1/2	1
y	1	-1	2

Table 5.2 Data for example.

where

$$G_1(x) = \begin{cases} 1+2x & \text{if} \quad -1/2 \le x \le 0, \\ 1-2x & \text{if} \quad 0 \le x \le 1/2, \\ 0 & \text{otherwise,} \end{cases}$$

$$G_2(x) = \begin{cases} 2x & \text{if} \quad 0 \le x \le 1/2, \\ 2-2x & \text{if} \quad 1/2 \le x \le 1, \\ 0 & \text{otherwise,} \end{cases}$$

and

$$G_3(x) = \begin{cases} -1+2x & \text{if} \quad 1/2 \le x \le 1, \\ 3-2x & \text{if} \quad 1 \le x \le 3/2, \\ 0 & \text{otherwise.} \quad \blacksquare \end{cases}$$

5.4 Piecewise Cubic Interpolation

The principal criticism of piecewise linear interpolation is that the approximation function has corners. One method that is often used to avoid this is to replace the linear functions with cubics. Instead of (5.10), we have a interpolation function of the form (see Figure 5.10)

$$s(x) = \begin{cases} s_1(x) & \text{if } x_1 \le x \le x_2 \\ s_2(x) & \text{if } x_2 \le x \le x_3 \\ \vdots & \vdots \\ s_n(x) & \text{if } x_n \le x \le x_{n+1} \,, \end{cases} \tag{5.15}$$

where in the ith interval the function is

$$s_i(x) = a_i + b_i(x-x_i) + c_i(x-x_i)^2 + d_i(x-x_i)^3, \quad \text{for } x_i \le x \le x_{i+1}. \tag{5.16}$$

To satisfy the interpolation conditions it is required that

$$s_i(x_i) = y_i, \quad s_i(x_{i+1}) = y_{i+1}, \quad \text{for } i = 1, 2, \cdots, n. \tag{5.17}$$

This will determine two of the four constants in s_i. We will also require that the transition between intervals is as smooth as possible. First, the slopes must match and this means that

$$s_i'(x_{i+1}) = s_{i+1}'(x_{i+1}), \quad \text{for } i = 1, 2, \cdots, n-1. \tag{5.18}$$

Second we will require that the second derivatives also match, and so

$$s_i''(x_{i+1}) = s_{i+1}''(x_{i+1}), \quad \text{for } i = 1, 2, \cdots, n-1. \tag{5.19}$$

Conditions (5.17)–(5.19) are the basic requirements for $s(x)$ to be a cubic spline. However, $s(x)$ has $4n$ coefficients that we need to determine, and these conditions produce $4n-2$ equations. In other words, we are short two conditions. What is usually done is to specify conditions at the left and right ends of the data interval. Some of the commonly made choices are as follows:

- *Natural Spline*: $s_1''(x_1) = 0$ and $s_n''(x_{n+1}) = 0$

  This produces a spline with an interesting property related to curvature, and this will be explained later.

- *Clamped Spline*: $s_1'(x_1) = y_1'$ and $s_n'(x_{n+1}) = y_{n+1}'$

  This requires knowing the value of the derivative at the endpoints, something that is not usually available.

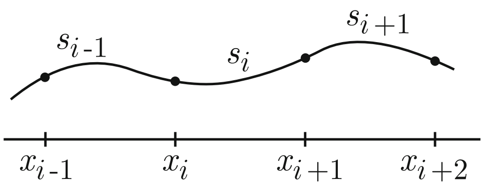


Figure 5.10 Intervals and functions used for piecewise cubic interpolation.

- *Not-a-Knot Spline*: $s_1'''(x_2) = s_2'''(x_2)$ and $s_{n-1}'''(x_n) = s_n'''(x_n)$

  This is the default choice in MATLAB.

Whichever choice is made, the resulting function $s(x)$ provides a smooth interpolation of the given data points.

Example

To find the natural cubic spline that interpolates the data in Table 5.3, we use (5.15) and write

$$s(x) = \begin{cases} s_1(x) & \text{if } 0 \le x \le 1/2 \\ s_2(x) & \text{if } 1/2 \le x \le 1, \end{cases}$$

where

$$s_1(x) = a_1 + b_1 x + c_1 x^2 + d_1 x^3,$$

and

$$s_2(x) = a_2 + b_2(x-1/2) + c_2(x-1/2)^2 + d_2(x-1/2)^3.$$

x	0	1/2	1
y	1	-1	2

Table 5.3 Data for example.

From the interpolation conditions (5.17),

$$\begin{aligned} s_1(0) = 1: &\quad & a_1 &= 1, \\ s_1(1/2) = -1: &\quad & a_1 + \tfrac{1}{2}b_1 + \tfrac{1}{4}c_1 + \tfrac{1}{8}d_1 &= -1, \\ s_2(1/2) = -1: &\quad & a_2 &= -1, \\ s_2(1) = 2: &\quad & a_2 + \tfrac{1}{2}b_2 + \tfrac{1}{4}c_2 + \tfrac{1}{8}d_2 &= 2. \end{aligned}$$

Also, from (5.18) and (5.19)

$$\begin{aligned} s_1'(1/2) = s_2'(1/2): &\quad & b_1 + c_1 + \tfrac{3}{4}d_1 &= b_2, \\ s_1''(1/2) = s_2''(1/2): &\quad & 2c_1 + 3d_1 &= 2c_2. \end{aligned}$$

Finally, to qualify to be a natural cubic spline it is required that

$$\begin{aligned} s_1''(0) = 0: &\quad & c_1 &= 0, \\ s_2''(1) = 0: &\quad & 2c_2 + 3d_2 &= 0. \end{aligned}$$

It is now a matter of solving the above equations, and after doing this one finds that

$$s_1(x) = 1 - \frac{13}{2}x + 10x^3,$$

and

$$s_2(x) = -1 + (x - 1/2) + 15(x - 1/2)^2 - 10(x - 1/2)^3. \qquad \blacksquare$$

To find $s(x)$ it remains to solve $4n$ equations with $4n$ unknowns. It is possible to just solve the resulting matrix equation for the unknowns, but there are better ways to find the coefficients. One possibility is to mathematically simplify the equations, and reduce the problem to solving a system with n unknowns. Another approach is to use cubic B-splines. This will also reduce the problem down to having to solve for (approximately) n unknowns. The advantage of B-splines is that they are easier to code. They are also very useful for least squares fitting of data (see Exercise 8.32), as well when solving differential equations numerically [Holmes, 2007].

5.4.1 Cubic B-Splines

The idea is to write the interpolation function in the form

$$s(x) = \sum a_i B_i(x). \tag{5.20}$$

The functions $B_i(x)$ are called *cubic B-splines*, and a sketch of a typical B-spline is given in Figure 5.11. The above expression has a passing similarity to the expressions used for Lagrange interpolation and piecewise linear interpolation. However, one important difference is that the coefficient a_i in the above sum is not necessarily equal to the data value y_i.

The derivation of (5.20) consists of two steps. The first is the construction of the cubic B-splines $B_i(x)$. This only has to be done once, which means that these functions do not need to be rederived if the data set is changed. Once this is complete, then the problem used to find the a_i's is determined. The values of the a_i's do depend on the data, and so this problem must be solved each time the data set is changed.

Finding the B_i's

Each $B_i(x)$ is a piecewise cubic function and has the form

$$B_i(x) = \begin{cases} 0 & \text{if} \quad x \le x_{i-2}, \\ q_{i-2}(x) & \text{if} \quad x_{i-2} \le x \le x_{i-1}, \\ q_{i-1}(x) & \text{if} \quad x_{i-1} \le x \le x_i, \\ q_{i+1}(x) & \text{if} \quad x_i \le x \le x_{i+1}, \\ q_{i+2}(x) & \text{if} \quad x_{i+1} \le x \le x_{i+2}, \\ 0 & \text{if} \quad x_{i+2} \le x, \end{cases} \tag{5.21}$$

where $q_j(x) = \overline{A}_j + \overline{B}_j(x-x_j) + \overline{C}_j(x-x_j)^2 + \overline{D}_j(x-x_j)^3$. We will assume that the x_i's are equally spaced, with $h = x_{i+1} - x_i$, so the function $B_i(x)$ is symmetric about $x = x_i$. Note that one consequence of the symmetry is that $B_i'(x_i) = 0$.

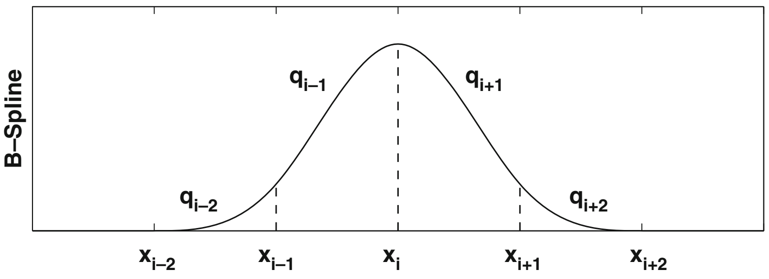


Figure 5.11 Sketch of the various components of a cubic B-spline.

The coefficients of the q_j's in (5.21) are determined from the requirement that $B_i \in C^2(-\infty, \infty)$. This is accomplished by requiring the following:

$x = x_{i-2}$: $q_{i-2}(x_{i-2}) = 0$, $q'_{i-2}(x_{i-2}) = 0$, and $q''_{i-2}(x_{i-2}) = 0$

$x = x_{i-1}$: $q_{i-2}(x_{i-1}) = q_{i-1}(x_{i-1})$, $q'_{i-2}(x_{i-1}) = q'_{i-1}(x_{i-1})$, and $q''_{i-2}(x_{i-1}) = q''_{i-1}(x_{i-1})$

$x = x_i$: $q_{i-1}(x_i) = q_{i+1}(x_i)$, $q'_{i-1}(x_i) = q'_{i+1}(x_i)$, and $q''_{i-1}(x_i) = q''_{i+1}(x_i)$

with similar conditions at $x = x_{i+1}$ and $x = x_{i+2}$. From the conditions at $x = x_{i-2}$ one easily concludes that $q_{i-2}(x) = \overline{D}_{i-2}(x - x_{i-2})^3$. In a similar way, it is found that $q_{i+2}(x) = \overline{D}_{i+2}(x - x_{i+2})^3$, where from the symmetry, $\overline{D}_{i+2} = -\overline{D}_{i-2}$. From the smoothness requirements at x_{i-1}, and the requirement that $B'(x_i) = 0$, one finds that

$$q_{i-1}(x) = D_{i-2}\left[h^3 + 3h^2(x - x_{i-1}) + 3h(x - x_{i-1})^2 - 3(x - x_{i-1})^3\right].$$

A similar equation can be derived for $q_{i+1}(x)$. This leaves one undetermined constant and the convention is to take $B_i(x_i) = 2/3$, which means that $\overline{D}_{i-2} = 1/(6h^3)$. With this, $B_i(x)$ is completely defined and the functions in (5.21) are

$$\begin{aligned}
q_{i-2}(x) &= \frac{1}{6h^3}(x - x_{i-2})^3,\\
q_{i-1}(x) &= \frac{1}{6} + \frac{1}{2h}(x - x_{i-1}) + \frac{1}{2h^2}(x - x_{i-1})^2 - \frac{1}{2h^3}(x - x_{i-1})^3,\\
q_{i+1}(x) &= \frac{1}{6} - \frac{1}{2h}(x - x_{i+1}) + \frac{1}{2h^2}(x - x_{i+1})^2 + \frac{1}{2h^3}(x - x_{i+1})^3,\\
q_{i+2}(x) &= -\frac{1}{6h^3}(x - x_{i+2})^3.
\end{aligned}$$

By factoring the above polynomials it is possible to show that

$$B_i(x) = B\left(\frac{x - x_i}{h}\right), \tag{5.22}$$

where

$$B(x) = \begin{cases} \frac{2}{3} - x^2\left(1 - \frac{1}{2}|x|\right) & \text{if} \quad |x| \le 1, \\ \frac{1}{6}(2 - |x|)^3 & \text{if} \quad 1 \le |x| \le 2, \\ 0 & \text{if} \quad 2 \le |x|. \end{cases}$$

A plot of the resulting function is shown in Figure 5.12. At x_{i-1} and x_{i+1} the curve makes such a smooth transition across the respective data point that you would not know that the cubics change there. The same is true at x_{i-2} and x_{i+2} where the function makes a smooth transition to zero.

Finding the a_i's

The cubic spline interpolation function can now be written as

$$s(x) = \sum_{i=0}^{n+2} a_i B_i(x). \tag{5.23}$$

Just so it is clear, this function satisfies the smoothness conditions in (5.18) and (5.19), but does not yet satisfy the interpolation conditions in (5.17) or the specific end conditions (clamped, natural, etc.). Also, the sum is over all possible B_i's that are nonzero on the interval $x_1 \le x \le x_{n+1}$. This has required us to include the $i = 0$ and $i = n + 2$ terms even though there is no x_{-1}, x_0, x_{n+2}, or x_{n+3} in the original data set. We will deal with this shortly. First note that at x_i only B_{i-1}, B_i, and B_{i+1} are nonzero. In particular, using the values given in Table 5.4,

$$\begin{aligned} s(x_i) &= a_{i-1}B_{i-1}(x_i) + a_iB_i(x_i) + a_{i+1}B_{i+1}(x_i) \\ &= \frac{1}{6}(a_{i-1} + 4a_i + a_{i+1}). \end{aligned}$$

Because of the interpolation requirement (5.17) we have that

$$a_{i-1} + 4a_i + a_{i+1} = 6y_i, \quad \text{for } i = 1, 2, \cdots, n+1. \tag{5.24}$$

To use this we need to know a_0 and a_{n+2}, and this is where the two additional conditions are used. We will use a natural spline, and for this note that

$$\begin{aligned} s''(x_i) &= a_{i-1}B''_{i-1}(x_i) + a_iB''_i(x_i) + a_{i+1}B''_{i+1}(x_i) \\ &= \frac{1}{h^2}(a_{i-1} - 2a_i + a_{i+1}). \end{aligned}$$

Solving $s''(x_1) = 0$ we get that $a_0 = 2a_1 - a_2$, and at the other end one finds that $a_{n+2} = 2a_{n+1} - a_n$. In (5.24), when $i = 1$ one finds that $a_1 = y_1$ and at the other end one gets that $a_{n+1} = y_{n+1}$. The rem-

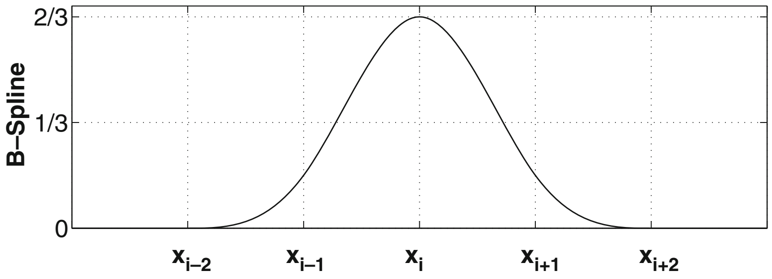


Figure 5.12 Plot of the cubic B-spline $B_i(x)$ defined in (5.22).

	x_{i-1}	x_i	x_{i+1}	x_j for $j \neq i, i \pm 1$
B_i	$\frac{1}{6}$	$\frac{2}{3}$	$\frac{1}{6}$	0
B_i'	$\frac{1}{2h}$	0	$-\frac{1}{2h}$	0
B_i''	$\frac{1}{h^2}$	$-\frac{2}{h^2}$	$\frac{1}{h^2}$	0

Table 5.4 Values of the B-spline $B_i(x)$, as defined in (5.21), at the grid points used in its construction.

aining a_i's are found by solving $\mathbf{Aa} = \mathbf{z}$ where $\mathbf{a} = (a_2, a_3, \cdots, a_n)^T$, $\mathbf{z} = (6y_2 - y_1, 6y_3, \cdots, 6y_{n-1}, 6y_n - y_{n+1})^T$, and $\mathbf{A}$ is the $(n-1) \times (n-1)$ tridiagonal matrix

$$\mathbf{A} = \begin{pmatrix} 4 & 1 & & & & \\ 1 & 4 & 1 & & 0 & \\ & 1 & 4 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & 0 & & & & 1 \\ & & & & 1 & 4 \end{pmatrix}.$$

This positive definite and tridiagonal matrix equation can be solved using the Thomas algorithm (see Section 3.8). This procedure is very fast, and requires minimal storage, which means finding the coefficients for a cubic spline is fairly easy even for a large number of interpolation points.

As a final comment, for a clamped spline it is also necessary to solve an equation of the form $\mathbf{Aa} = \mathbf{z}$, where $\mathbf{A}$ and $\mathbf{z}$ are given in Exercise 5.30.

Example

Using a natural cubic spline to fit the data in Figure 5.3 produces the solid (blue) curves shown in Figure 5.13. For comparison, the curves obtained using a not-a-knot spline are also shown. The interpolation of the top two data sets is as good as what was obtained using the global polynomial, and there are few minor differences between the two spline functions. Moreover, both splines give a better representation than a global polynomial for the lower two data sets. For the lower right data set some small over- and undershoots are present in the spline functions, but they are not as pronounced as those in Figure 5.4. It is also evident that there are also differences between the two spline functions, although these occur primarily in the regions close to the endpoints. ■

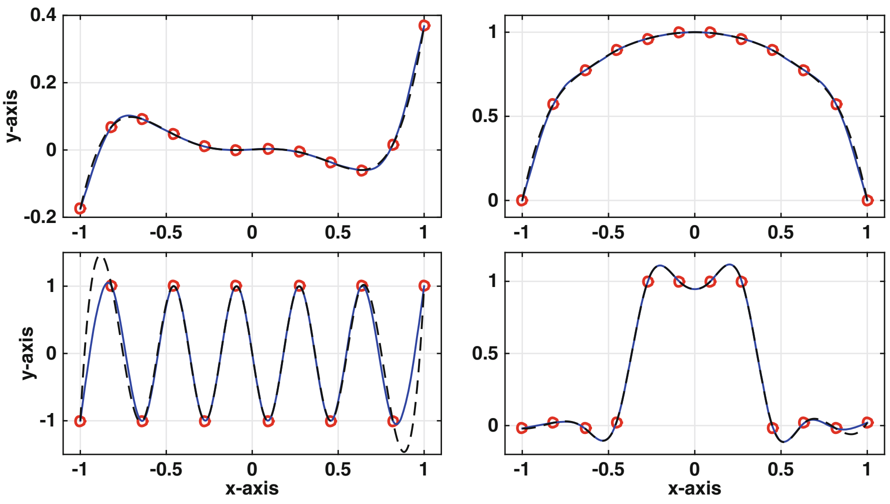


Figure 5.13 Using a natural cubic spline function, the solid (blue) curves, and a Not-a-Knot spline, the dashed (black) curves, to fit the data in Figure 5.3.

Example

To find the natural cubic spline that interpolates the data in Table 5.1, we use (5.23) and write

$$s(x) = a_0B_0(x) + a_1B_1(x) + a_2B_2(x) + a_3B_3(x) + a_4B_4(x),$$

where $x_0 = -1/2$, $x_4 = 3/2$, and $B_i(x) = B(2(x - x_i))$. The interpolation requirements are

$$\begin{aligned} s(0) = 1 : &\quad a_0 + 4a_1 + a_2 = 6, \\ s(1/2) = -1 : &\quad a_1 + 4a_2 + a_3 = -6, \\ s(1) = 2 : &\quad a_2 + 4a_3 + a_4 = 12, \end{aligned}$$

and the natural spline end conditions are

$$\begin{aligned} s''(0) = 0 : &\quad a_0 - 2a_1 + a_2 = 0, \\ s''(1) = 0 : &\quad a_2 - 2a_3 + a_4 = 0. \end{aligned}$$

Solving these equations it is found that

$$s(x) = \frac{17}{4}B_0(x) + B_1(x) - \frac{9}{4}B_2(x) + 2B_3(x) + \frac{15}{4}B_4(x). \quad \blacksquare$$

The question arises as to why the natural cubic spline works so well. There is a partial answer to this and it involves curvature. Recall that for a curve $y = f(x)$, the curvature at a point is defined as

$$\kappa = \frac{|f''(x)|}{[1+(f')^2]^{3/2}} .$$

Assuming the curve is not particularly steep, one can approximate the curvature as $\kappa \approx |f''(x)|$. This is brought up because of the next result due to Holladay [1957].

Theorem 5.1. *If $q \in C^2[a,b]$ interpolates the same data points that the natural cubic spline function (5.15) interpolates, then*

$$\int_a^b [s''(x)]^2 dx \le \int_a^b [q''(x)]^2 dx.$$

In these integrals, $a = x_1$ and $b = x_{n+1}$.

What this theorem states is that out of all smooth functions that interpolate the data, the *natural* cubic spline produces the interpolation function with the smallest total curvature (squared). This helps explain why the spline interpolations in Figure 5.13 do not suffer the significant under- and overshoots seen in Figure 5.4.

5.5 Function Interpolation

In using the data sets to test out the various interpolation methods we have been using a qualitative, or visual, determination of how well they do. We are now going to make the test more quantitative and this will restrict the applications. In particular, it is assumed that the data comes from the evaluation of a given function $f(x)$ and we are going to investigate how well the interpolation function approximates $f(x)$ between the data points.

In what follows the data points are $(x_1, y_1), (x_2, y_2), \cdots, (x_{n+1}, y_{n+1})$, where $y_i = f(x_i)$. Also, unless stated explicitly to the contrary, the step size $h = x_{i+1} - x_i$ is assumed constant (so the x_i's are equally spaced), with $a = x_1$ and $b = x_{n+1}$. What is of interest is whether the approximation gets better as the step size h gets smaller. In particular, does the error go to zero as h goes to zero?

5.5.1 Global Polynomial Interpolation

We begin with a global interpolation polynomial $p_n(x)$. As explained in Section 5.2.3, $p_n(x)$ can fail to provide a good approximation of a function if a large number of equally spaced points are used. However, it is effective for a small number of points, as long as they are not too far apart. In fact, such approximations are central to several of the methods considered later in the text. The critical result needed to determine the error when using $p_n(x)$ is given in the following theorem:

Theorem 5.2. *If $f \in C^{n+1}[a,b]$, then*

$$f(x) = p_n(x) + \frac{f^{(n+1)}(\eta)}{(n+1)!} q_{n+1}(x), \quad \text{for } a \le x \le b, \tag{5.25}$$

where

$$q_{n+1}(x) = (x-x_1)(x-x_2)\cdots(x-x_{n+1}), \tag{5.26}$$

and η is a point in (a,b).

Outline of Proof: To explain how this is proved, we will consider the case of when $n=1$. In this case, $q_2(x) = (x-a)(x-b)$. The formula in (5.25) holds when $x=a$ or $x=b$, so assume that $a<x<b$. The key step is a trick, which consists of introducing the function

$$F(z) = f(z) - p_1(z) + \frac{f(x)-p_1(x)}{q_2(x)} q_2(z).$$

Given the way it is defined, $F(a)=0$, $F(x)=0$, and $F(b)=0$. According to Rolle's theorem, there must be a point z_1, where $a<z_1<x$ and $F'(z_1)=0$, and there must be a point z_2, where $x<z_2<b$ and $F'(z_2)=0$. Using Rolle's theorem again, there must be a point η, where $z_1<\eta<z_2$ and $F''(\eta)=0$. From the above formula for $F(z)$, one finds that $F''(\eta)=0$ reduces to (5.25). The case of when $n>1$ is similar, except one uses Rolle's theorem $n+1$ times (instead of twice). $\square$

An immediate consequence of this theorem is the following:

Theorem 5.3. *If $f \in C^{n+1}[a,b]$, then the global interpolation polynomial $p_n(x)$ satisfies*

$$|f(x)-p_n(x)| \le \frac{1}{(n+1)!} ||f^{(n+1)}||_\infty ||q_{n+1}||_\infty, \quad \text{for } a \le x \le b,$$

where

$$||f^{(n+1)}||_\infty = \max_{a\le x\le b} |f^{(n+1)}(x)|,$$

and

$$||q_{n+1}||_\infty = \max_{a\le x\le b} |q_{n+1}(x)|.$$

It should be pointed out that the above two theorems hold in the case of when the x_i's are not equally spaced (this fact is used in Section 5.5.4).

To make use of Theorem 5.3, we need to determine $||q_{n+1}||_\infty$. This is not hard to do for small values of n, and to illustrate this, consider the case of when $n = 2$. For this, $q_2(x) = (x - x_1)(x - x_2)$, where $x_2 = x_1 + h$. The maximum, and minimum, of this function occur either at the endpoints or at a critical point inside the interval. First note that $q_2(x_1) = q_2(x_2) = 0$. As for the critical points, solving $q_1'(x) = 0$, one finds that $x = x_1 + \frac{1}{2}h$. From this it follows that $||q_2||_\infty = \frac{1}{4}h^2$. Similarly, if $n = 3$, then $q_3(x) = (x - x_1)(x - x_2)(x - x_3)$, where $x_2 = x_1 + h$ and $x_3 = x_1 + 2h$. Solving $q_3'(x) = 0$, one finds two solutions, $x = x_1 + h \pm \frac{1}{3}h\sqrt{3}$. From this it follows that $||q_3||_\infty = \frac{2}{9}h^3\sqrt{3}$. Continuing this, the values in Table 5.5 are obtained.

n	$\|\|q_{n+1}\|\|_\infty$
1	$\frac{1}{4}h^2$
2	$\frac{2}{9}\sqrt{3}\,h^3$
3	h^4
4	$\frac{1}{50}\left(\sqrt{145}-1\right)\sqrt{150+10\sqrt{145}}\,h^5$
5	$\frac{16}{27}(7\sqrt{7}+10)h^6$

Table 5.5 Value of $||q_{n+1}||_\infty$, which appears in the error formula in Theorem 5.3.

Example

Suppose that $f(x) = \cos(2\pi x)$ and we use the interpolation polynomial $p_2(x)$, with points x_1, $x_2 = x_1 + h$, and $x_3 = x_1 + 2h$. According to Theorem 5.3, how small does h need to be to guarantee an error of 10^{-6}, irrespective of the choice for x_1? To answer this, since $f''' = -(2\pi)^3\sin(2\pi x)$, then $||f'''||_\infty \leq (2\pi)^3$. With this we have that

$$\frac{1}{6}||f'''||_\infty||q_3||_\infty \leq \sqrt{3}\left(\frac{2\pi}{3}h\right)^3.$$

Consequently we will achieve the require error bound if $\sqrt{3}(2\pi h/3)^3 \leq 10^{-6}$, which means $h \leq 3^{5/6}/(200\pi) \approx 0.0040$. ■

Finding $||q_{n+1}||_\infty$ for larger values of n is difficult, and the usual approach is to find an upper bound on this number. One that is not hard to derive is (see Exercise 5.31)

$$||q_{n+1}||_\infty \leq \frac{1}{4}n!\,h^{n+1}. \tag{5.27}$$

Using this, the inequality in Theorem 5.3 can be written as

$$|f(x) - p_n(x)| \leq \frac{1}{4(n+1)}||f^{(n+1)}||_\infty h^{n+1}, \quad \text{for } a \leq x \leq b, \tag{5.28}$$

If this is used in the above example, the conclusion is that it is necessary to have $h \leq (3/2)^{1/3}/(100\pi) \approx 0.0036$. The fact that this is smaller than the requirement given in the example is a reflection of the inequality in (5.27).

5.5.2 Piecewise Linear Interpolation

The next easiest method to analyze is piecewise linear interpolation. An example is shown in Figure 5.14, where $f(x) = \cos(2\pi x)$ is approximated using 6 points over the interval $0 \leq x \leq 1$. How well the linear functions approximate $f(x)$ depends on the subinterval. This is why in the analysis for the general case in the next paragraph we first determine what happens over each subinterval $x_i \leq x \leq x_{i+1}$.

To determine how well $f(x)$ is approximated by $g_i(x)$, given in (5.9), over the subinterval $x_i \leq x \leq x_{i+1}$ we can use Theorem 5.3. In this case, $n = 1$ and $q_2(x) = (x - x_i)(x - x_{i+1})$. Consequently,

$$\begin{aligned} |f(x) - g_i(x)| &\leq \frac{1}{2} \max_{x_i \leq x \leq x_{i+1}} |f''(x)| \max_{x_i \leq x \leq x_{i+1}} |q_2(x)| \\ &= \frac{1}{8} h^2 \max_{x_i \leq x \leq x_{i+1}} |f''(x)| \\ &\leq \frac{1}{8} h^2 ||f''||_\infty . \end{aligned}$$

This applies to each subinterval, which gives the next result.

Theorem 5.4. *If $f \in C^2[a,b]$ then the piecewise linear interpolation function (5.9) satisfies*

$$|f(x) - g(x)| \leq \frac{1}{8} h^2 ||f''||_\infty, \quad \textit{for } a \leq x \leq b,$$

where $||f''||_\infty = \max_{a \leq x \leq b} |f''(x)|$.

This means that the piecewise linear interpolation function converges to the original function and the error is second order (because of the h^2). Consequently, if the number of interpolation points is doubled, the error in the approximation should decrease by about a factor of 1/4.

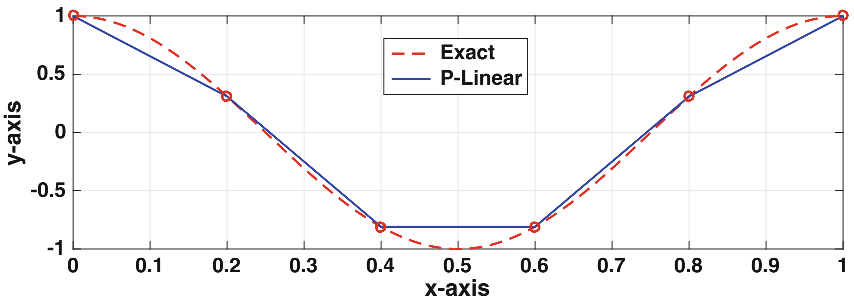


Figure 5.14 Piecewise linear interpolation of $f(x) = \cos(2\pi x)$.

Examples

1. Piecewise linear interpolation is used to approximate $f(x) = \cos(2\pi x)$ in Figure 5.14. According to Theorem 5.4, what is the error bound for this approximation?

   The interval in this case is $0 \le x \le 1$. Since $f(x) = \cos(2\pi x)$, then $f''(x) = -4\pi^2 \cos(2\pi x)$ and from this it follows that $||f''||_\infty = 4\pi^2$. Given that $h = 1/5$, the error bound is $|f(x)-g(x)| \le \pi^2/50$, where $\pi^2/50 \approx 0.2$. ■

2. For $f(x) = \cos(2\pi x)$, where $0 \le x \le 1$, how many interpolation points are needed to guarantee an error of 10^{-4}?

   Since $||f''||_\infty = 4\pi^2$, then we want $\pi^2 h^2/2 \le 10^{-4}$. This gives

   $$h \le \sqrt{2} \times 10^{-2}/\pi.$$

   If n is the number of interpolation points, then $h = 1/(n-1)$, and combining our results, $n \ge 1 + \sqrt{2}\pi^2 \times 10^2 \approx 1396.8$. Therefore, we need to take $n \ge 1397$. ■

3. According to Theorem 5.4, for what functions will there be zero error using piecewise linear interpolation, no matter what the interpolation interval?

   This requires $||f''||_\infty = 0$, which means that $f''(x) = 0$ for $a \le x \le b$. Therefore, to have zero error it must be that $f(x) = \alpha + \beta x$, i.e., it must be a linear function in this interval. ■

5.5.3 Cubic Splines

This brings us to the question of how well the cubic splines do when interpolating a function. The answer depends on what type of spline function is used (natural, clamped, or not-a-knot). To illustrate, in Figure 5.15 the function $f(x) = \cos(2\pi x)$ is approximated using both a natural and a clamped cubic spline. The clamped spline provides a somewhat better approximation but this is not surprising because it has the advantage of using the derivative information at the endpoints. It is possible to determine the error for the clamped spline, but because this requires some effort to derive only the result will be stated (see Hall and Meyer 1976 for the proof).

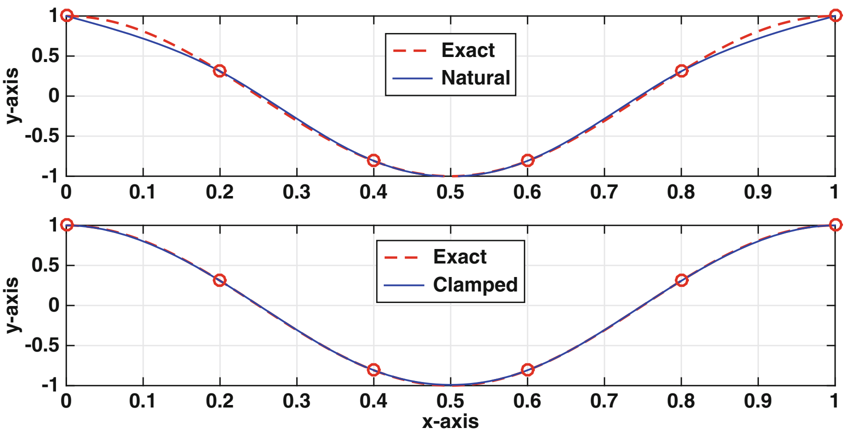


Figure 5.15 Natural and clamped cubic spline interpolation of $f(x) = \cos(2\pi x)$.

Theorem 5.5. *If $f \in C^4[a,b]$ then the clamped cubic spline interpolation function (5.15) satisfies*

$$|f(x) - s(x)| \le \frac{5}{384} h^4 ||f''''||_\infty, \quad \textit{for } a \le x \le b,$$

where $||f''''||_\infty = \max_{a \le x \le b} |f''''(x)|$. Moreover, for $a \le x \le b$,

$$|f'(x) - s'(x)| \le \frac{1}{24} h^3 ||f''''||_\infty,$$
$$|f''(x) - s''(x)| \le \frac{1}{3} h^2 ||f''''||_\infty.$$

This is an amazing result because it states that the clamped cubic spline can be used to approximate $f(x)$ and its first two derivatives. Moreover, the error in approximating $f(x)$ is fourth-order (because of the h^4). As an example, if the number of interpolation points is increased by a factor of 10, the error bound decreases by about a factor of 10^4. In comparison, according to Theorem 5.4, the error for piecewise linear interpolation decreases by a factor of 10^2.

Examples

1. A clamped cubic spline is used to approximate $f(x) = \cos(2\pi x)$ in Figure 5.15. According to Theorem 5.5, what is the error bound for this approximation?

   The interval is $0 \le x \le 1$. Since $f(x) = \cos(2\pi x)$, then $f''''(x) = (2\pi)^4 \cos(2\pi x)$ and $||f''''||_\infty = (2\pi)^4$. Given that $h = 1/5$, the error bound is $|f(x) - g(x)| \le \pi^4/3000$, where $\pi^4/3000 \approx 0.03$. ■

2. For $f(x) = \cos(2\pi x)$, where $0 \le x \le 1$, how many interpolation points are needed to guarantee an error of 10^{-4} when using a clamped spline?

   Since $||f''''||_\infty = (2\pi)^4$, then we want
$$\frac{5}{384} h^4 (2\pi)^4 \le 10^{-4}.$$
   This can be rewritten as $h \le (384/5)^{1/4}/(20\pi)$. If n is the number of interpolation points, then $h = 1/(n-1)$, and combining our results, $n \ge 1 + 20\pi(5/384)^{1/4} \approx 22.2$. Therefore, we need to take $n \ge 23$. ■

3. According to Theorem 5.5, for what functions will there be zero error using a clamped spline, no matter what the interpolation interval?

   This requires $||f''''||_\infty = 0$, which means that $f''''(x) = 0$ for $a \le x \le b$. Therefore, to have zero error it must be that $f(x)$ is a cubic function. ■

The major drawback in the above theorem is that it requires the clamped end conditions and for many problems $f'(x_1)$ and $f'(x_{n+1})$ are not known. It is possible to obtain the same level of accuracy with a natural cubic spline, except near the endpoints, as long as enough points are used. This is seen in Figure 5.15, where the natural spline does almost as well approximating the function except over the subintervals next to the boundary points. The exact statement of the result is given next (see [Kershaw, 1971] for the proof).

Theorem 5.6. *If $f \in C^4[a,b]$ then for the natural cubic spline interpolation function $s(x)$:*

1. *$|f(x) - s(x)| \le K_1 h^2$, for $a \le x \le b$, where K_1 is a positive constant.*

2. *For large n, there are points x_ℓ and x_r, with $a \le x_\ell < x_r \le b$, so that*
$$|f(x) - s(x)| \le K_2 h^4, \quad \textit{for } x_\ell \le x \le x_r,$$
where K_2 is a positive constant. Moreover, $x_\ell \to a$ and $x_r \to b$ as $n \to \infty$.

What this theorem states is that the error for a natural cubic spline is at least second-order. It also states that the error is actually fourth-order except near the endpoints. Moreover, the regions near the endpoints where it is not fourth-order shrink as n increases.

5.5.4 Chebyshev Interpolation

We saw earlier that the polynomial that interpolates all of the data points can be written as
$$p_n(x) = \sum_{i=1}^{n+1} y_i \ell_i(x), \tag{5.29}$$
where
$$\ell_i(x) = \prod_{\substack{j=1 \\ j \ne i}}^{n+1} \frac{x - x_j}{x_i - x_j}\,. \tag{5.30}$$
Given that we are now considering function interpolation, it is assumed that $y_i = f(x_i)$. One thing to note is that this does not require that the x_i's are equally spaced. Also, we know that when n is large, and the points are equally spaced, the above polynomial should not be used for interpolation. The question we examine now is, is it possible to pick the locations of the x_i's so $p_n(x)$ is capable of producing an accurate interpolation function. It is possible, and to explain how, we need Theorem 5.2. What is of interest here is the error term, which is
$$\frac{f^{(n+1)}(\eta)}{(n+1)!} q_{n+1}(x),$$
where $q_{n+1}(x)$ is given in (5.26). In Figure 5.5 we saw that the values of this can be huge. We want to prevent this from happening. For a given data set, so n is given, the assumption that $f \in C^{n+1}[a, b]$ means that there is a positive constant M_{n+1} so that $|f^{(n+1)}(x)| \le M_{n+1}$. So, the part of the error function that we need to concentrate on is $q_{n+1}(x)$, and what we are specifically interested in is the value of
$$Q = \max_{a \le x \le b} |q_{n+1}(x)|. \tag{5.31}$$
This brings us to the following question: given n, how do we position the x_i's in the interval $a \le x \le b$ to minimize Q.

The answer is easy to state, but deriving it requires some work. The result is given below, and afterwards the derivation is outlined.

Theorem 5.7. *The x_i's that produce the smallest value of Q are*

$$x_i = \frac{1}{2}[a + b + (b - a)z_i]\,, \ \textit{for } i = 1, 2, \cdots, n+1, \tag{5.32}$$

where

$$z_i = \cos\left(\frac{2i-1}{2(n+1)}\pi\right). \tag{5.33}$$

The x_i's in the above theorem are called the *Chebyshev points* because they correspond to the zeros of the $(n+1)$th Chebyshev polynomial. Note that the Chebyshev polynomials are usually defined for $-1 \le x \le 1$, and when using $a \le x \le b$ they are referred to as the Chebyshev polynomials for the general interval. This distinction is not made in what follows.

There is a simple geometric interpretation for how the x_i's are positioned in the interval that comes from (5.33). Placing a semi-circle over the interval, as in Figure 5.16, consider the $n+1$ points on the semi-circle that are a constant angle π/n apart, with the first one (on the far right) at an angle $\pi/(2n)$. Their x coordinates are given in (5.32) and they are the corresponding Chebyshev points. This figure also shows why the Chebyshev points are closer together at the endpoints of the interval, as compared to their placement towards the center.

In the proof of Theorem 5.7, the following result is also obtained:

Theorem 5.8. *With the x_i's given in (5.32), then the global interpolation polynomial $p_n(x)$ given in (5.5) satisfies*

$$|f(x) - p_n(x)| \le \frac{1}{2^n(n+1)!}\left(\frac{b-a}{2}\right)^{n+1} ||f^{(n+1)}||_\infty, \quad \textit{for } a \le x \le b,$$

where $||f^{(n+1)}||_\infty = \max_{a\le x\le b}|f^{(n+1)}(x)|$.

The fact that $2^n(n+1)!$ grows rapidly with n means that the error with Chebyshev interpolation can be quite small. To get an estimate of just how small, according to Stirling's formula, for large values of n,

$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n. \tag{5.34}$$

Using this approximation for the factorial, the inequality in Theorem 5.8 can be replaced with (see Exercise 5.31)

$$|f(x) - p_n(x)| \le \sqrt{\frac{2}{n\pi}}\,R^{n+1}||f^{(n+1)}||_\infty, \quad \text{for } a \le x \le b, \tag{5.35}$$

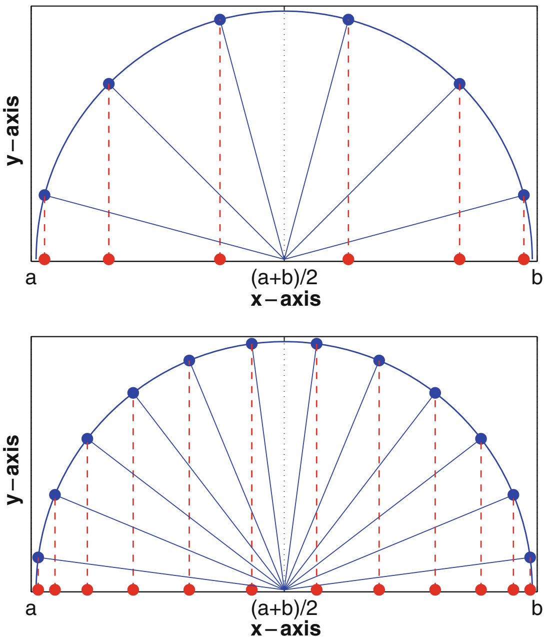


Figure 5.16 The Chebyshev points, which are the red dots along the x-axis, are determined by equally spaced points on the circumscribed semi-circle. In the top graph, $n = 5$, while in the bottom graph, $n = 11$.

where

$$R = \frac{(b-a)e}{4(n+1)} . \tag{5.36}$$

Consequently, if n is large enough that $R < 1$, then R^{n+1} approaches zero exponentially fast. Whether this means that the error for Chebyshev interpolation approaches zero exponentially fast, however, depends on how the

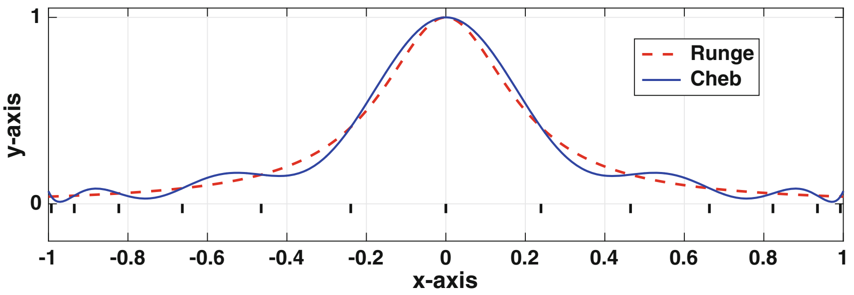


Figure 5.17 The Runge function (5.8) and the Chebyshev interpolation polynomial with $n = 12$.

$f^{(n+1)}$ term depends on n. This issue is considered in the examples to follow. Also, it's worth noting that the error using piecewise linear or cubic splines is not exponential, and the error in each case approaches zero as a fixed power of $h = (b-a)/(n-1)$.

Examples

1. Using the Chebyshev points with $n = 12$ to fit Runge's function in (5.8), the interpolation function shown in Figure 5.17 is obtained. The improvement over using equally spaced points, which is shown in Figure 5.5, is dramatic. Also shown in Figure 5.17, by small horizontal bars, are the locations of the x_i's. This shows the non-uniform spacing of the interpolation points, and they get closer together as you approach either of the endpoints of the interval. ■

2. For $f(x) = \cos(2\pi x)$, where $0 \leq x \leq 1$, according to Theorem 5.8, how many interpolation points are needed to guarantee an error of 10^{-4} when using Chebyshev interpolation?

   Since $||f^{(n+1)}||_\infty = (2\pi)^{n+1}$, then we want $\pi^{n+1}/(2^n(n+1)!) \leq 10^{-4}$. From this one finds that $n \geq 9$. In comparison, earlier we found that piecewise linear requires $n \geq 1397$, while a clamped cubic spline requires $n \geq 23$. It is also interesting to note that if the number of points is doubled to $n = 18$, that according to Theorem 5.8, the error bound is about 10^{-14}, and if doubled again to $n = 36$, the error bound is an astonishing 10^{-36}. In contrast, for a clamped cubic spline, doubling the number of points decreases the error bound by a factor of $2^{-4} \approx 6 \times 10^{-2}$. This is a clear demonstration of the benefits of an exponentially converging approximation, but as we will see shortly, exponential convergence is limited to certain types of functions. ■

Chebyshev Polynomials

To explain how the x_i's are determined, it is assumed that $a = -1$ and $b = 1$. We begin with a definition.

Definition 5.1. The Chebyshev polynomials $T_n(x)$ are defined using the following recursion formula:

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x), \text{ for } n = 1, 2, 3, \cdots, \tag{5.37}$$

where $T_0(x) = 1$ and $T_1(x) = x$.

Using this definition, the first few Chebyshev polynomials are

$$\begin{aligned}
T_2(x) &= 2x^2 - 1,\\
T_3(x) &= 4x^3 - 3x,\\
T_4(x) &= 8x^4 - 8x^2 + 1,\\
T_5(x) &= 16x^5 - 20x^3 + 5x.
\end{aligned}$$

As is seen in the above expressions, $T_n(x)$ is an nth degree polynomial and the leading coefficient is 2^{n-1}.

Two of the key results needed for finding the x_i's are contained in the following result:

Theorem 5.9.

1. If $P_n(x) = x^n + b_{n-1}x^{n-1} + \cdots + b_0$, where $n \geq 1$, then

$$\max_{-1\leq x\leq 1} |P_n(x)| \geq 2^{1-n}.$$

2. If $P_n(x) = 2^{1-n}T_n(x)$, then

$$\max_{-1\leq x\leq 1} |P_n(x)| = 2^{1-n}.$$

The usual proof of the first statement involves contradiction, and using the oscillatory properties of a polynomial. An illustration of how it is possible to prove it directly is given in Exercise 5.32.

Note that the $q_{n+1}(x)$ in (5.26) is an example of the function $P_{n+1}(x)$ appearing in the above theorem. The first result in the theorem states that no matter what we pick for the x_i's, the Q in (5.31) satisfies $Q \geq 2^{-n}$. What the second result states is that if we pick $q_{n+1}(x) = 2^{-n}T_{n+1}(x)$, then $Q = 2^{-n}$, i.e., it achieves the stated minimum value. Therefore, we should pick the x_i's so that

$$2^{-n}T_{n+1}(x) = (x - x_1)(x - x_2)\cdots(x - x_{n+1}).$$

In other words, the x_i's are the zeros of $T_{n+1}(x)$.

We need an easy way to find the zeros of $T_{n+1}(x)$, and this is given in the next result.

Theorem 5.10.

$$T_n(x) = \cos(n\cos^{-1} x), \ \textit{for } n = 0, 1, 2, 3, \cdots .$$

This is a strange looking equation because the right-hand side does not look to be a polynomial. Nevertheless, the proof is rather simple, and basically involves using trig identities to show that the right-hand side satisfies (5.37). With this it is easy to find the zeros of $T_{n+1}(x)$, and they are the values of x that satisfy

$$(n+1)\cos^{-1}x = \frac{\pi}{2}(2i-1), \text{ for } i = 1, 2, 3, \cdots, n+1.$$

The values given in (5.32) are the resulting positions when the above result is transformed from $-1 \le x \le 1$ to $a \le x \le b$.

5.5.5 Chebyshev Versus Cubic Splines

We saw that Chebyshev interpolation has the potential to have an error that approaches zero exponentially fast as n increases. It also has the distinction that it produces the smallest Q, as explained in Theorem 5.7. The natural cubic splines, on the other hand, produce the smallest total curvature squared, as defined in Theorem 5.1. What this means is that we have two interpolation methods that can claim to be optimal. Given this, it is of interest to compare Chebyshev and cubic spline interpolation on some more challenging examples.

Example 1

We begin with the Runge's function in (5.8), and assume 13 data points are used. The resulting Chebyshev interpolation function is shown in Figure 5.17. For comparison, the corresponding natural cubic spline is shown in Figure 5.18. In comparing the two figures, it is clear that the spline provides a better approximation function. To have a more quantitative comparison, suppose $g(x)$

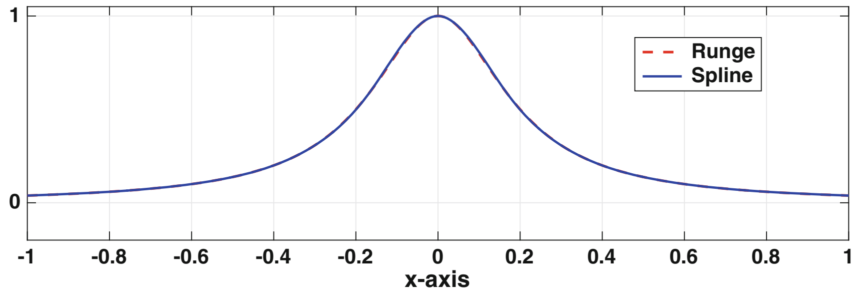


Figure 5.18 The function (5.8) and the natural cubic spline using 13 equally spaced points. The two curves are indistinguishable.

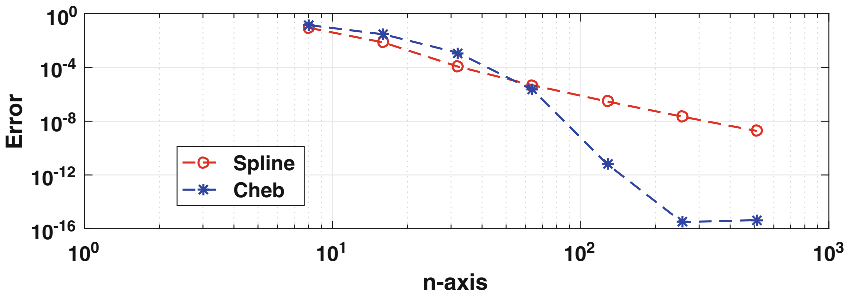


Figure 5.19 The error, as determined by the area integral in (5.38), when approximating Runge's function with a natural cubic spline, and with a Chebyshev interpolation function.

is an interpolation function of a given function $f(x)$. The error is using $g(x)$ to approximate $f(x)$ will be determined using the area between the two curves, which means

$$E = \int_a^b |f(x) - g(x)| dx. \tag{5.38}$$

The value of this integral is given in Figure 5.19, when $g(x)$ is the natural cubic spline, and when it is the Chebyshev interpolation function. It is seen that the spline produces a more accurate approximation when using up to about 60 interpolation points. The reason the spline does better is that $||f^{(n+1)}||_\infty$ grows rapidly with n, which effectively eliminates the exponential convergence for Chebyshev interpolation. For example, when $n = 13$, $||f^{(n+1)}||_\infty \approx 5 \times 10^{20}$, while $2^n(n+1)! \approx 3 \times 10^{13}$. It is not until n is rather large that the exponential convergence kicks in and Chebyshev begins to produce a better approximation than the spline. ■

Example 2

Suppose the function is $f(x) = \tanh(100x - 30)$, and the interval is $0 \le x \le 1$. Using 25 interpolation points, the resulting interpolation functions are shown in Figure 5.20. In this case, both have some difficulty with the rapid rise in the function. However, the under- and over-shoots in the spline function die out much faster than those for the Chebyshev function. The associated error for each interpolation function, as determined using (5.38), is shown in Figure 5.21. As in the last example, it is not until n is rather large that the exponential convergence enables Chebyshev to produce a better approximation than the spline. ■

It is evident from the examples that Chebyshev interpolation is capable of producing a more accurate approximation than cubic splines. However, this

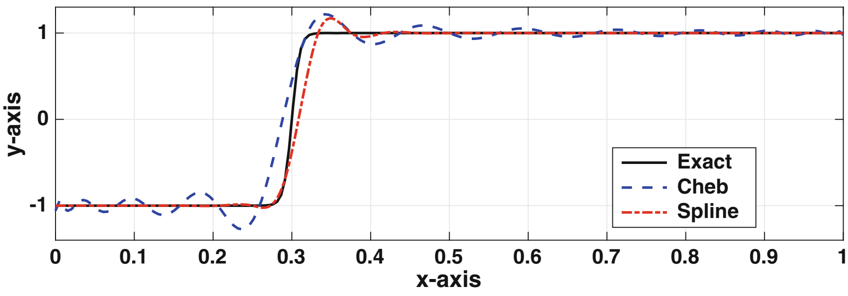


Figure 5.20 The function $f(x) = \tanh(100x - 30)$, along with Chebyshev and cubic spline interpolation functions using 25 data points.

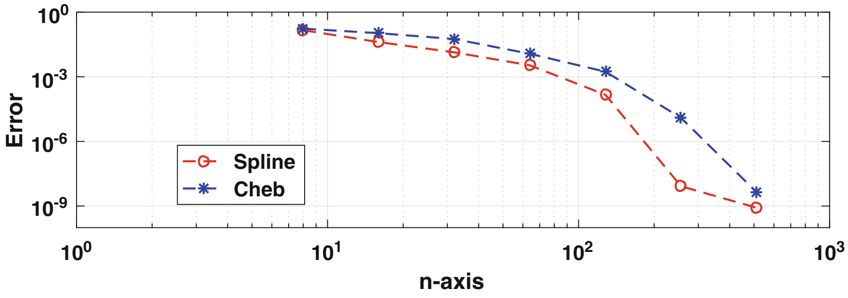


Figure 5.21 The error, as determined by the area integral in (5.38), when approximating $f(x) = \tanh(100x - 30)$ with a natural cubic spline, and with a Chebyshev interpolation function.

is based on the stipulation that the contribution of the $f^{(n+1)}(\eta)$ term in the error is not too large, and it can be difficult to know if this holds. This limits its usefulness, but it does not dampen the enthusiasm that some have for the method. To get insight into why they think this way, Trefethen [2012] should be consulted.

5.5.6 Other Ideas

There are a variety ways of modifying the interpolation procedure. For example, given a function $f(x)$ one can construct a piecewise cubic that interpolates $f(x_1), f(x_2), \cdots, f(x_{n+1})$ as well as the derivative values $f'(x_1)$, $f'(x_2)$, $\cdots$, $f'(x_{n+1})$. This is known as Hermite interpolation and it produces an approximation with an error that is $O(h^4)$. This puts it in the same category as the clamped cubic spline discussed earlier.

There is also the idea of being monotone. The objective here is that if the data appear to describe a monotonically increasing (or decreasing) function over an interval then the interpolation function should behave the same way over that interval. As seen in the lower right data set in Figures 5.4 and 5.13, the global and spline functions fail at this while the piecewise linear function in Figure 5.9 works. However, for the latter there is the usual problem with corners. So, the goal is to find a method that does well at preserving monotonicity and is also smooth. This comes under the more general heading of finding a smooth shape preserving interpolation function. A review of such methods, such as the Akima algorithm and the Fritsch-Butland procedure, can be found in Huynh [1993]. It is also worth noting that shape preserving interpolation is of particular interest in the mathematical finance community, and those interested in this application should consult Hagan and West [2006].

5.6 Questions and Additional Comments

Below are some random questions and comments about interpolation.

1. To fix the corner problem that arises with piecewise linear interpolation, we used piecewise cubic interpolation. What's wrong with piecewise quadratics?

   Answer: They have a couple of drawbacks. To explain, quadratics can interpolate the data and have a continuous first derivative (see Exercise 5.33). In comparison, cubic splines have continuous second derivatives, and so they are smoother. Another issue with quadratics is that they can produce what can best be described as bumpy curves, and this is illustrated in Exercise 5.33(d). However, it is possible to adjust the interpolation procedure to improve the situation and those interested might want to consult Marsden [1974] or Grasselli and Pelinovsky [2008].

2. The cubic B-spline $B_i(x)$ is nonzero for $x_{i-2} \leq x \leq x_{i+2}$. Why not use the smaller interval $x_{i-1} \leq x \leq x_{i+1}$?

   Answer: It simply won't work (try it).

3. Can the interpolation methods be used to solve the puzzle in Figure 5.6?

   Answer: Yes, but if you want to draw something that looks like a flower then you will need to rewrite the problem because our methods are based on interpolating a function. Possible solutions would be to break the data points into sections that can each be described as a function, or to use parametric coordinates. These ideas can be generalized, which leads naturally to something called Bezier curves and surfaces, and geometric modeling. Those interested might consider looking at Salomon [2006] and Mortenson [1997].

4. If just one of the y_i's is changed, what happens to the interpolation function?

   Answer: It depends on what method you are using. For piecewise linear, the

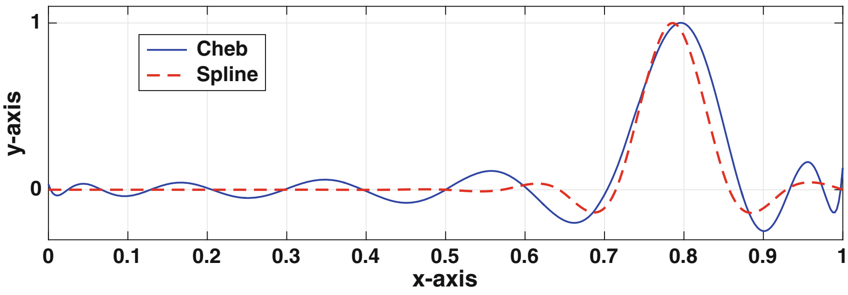


Figure 5.22 Interpolation using Chebyshev and cubic spline interpolation functions with 15 data points, with all values zero except for the one near 0.8.

   interpolation function is only affected over the interval $x_{i-1} < x < x_{i+1}$. If you are using cubic splines or Chebyshev, then the interpolation function over the entire interval $a < x < b$ is affected. To get an idea of what happens, if all of the y_i's are zero then the interpolation function is zero everywhere, irrespective of whether or not you are using a natural cubic spline or Chebyshev interpolation. Now, suppose one data point is changed and it is now nonzero. This situation is shown in Figure 5.22 using a natural cubic spline and Chebyshev interpolation, where the nonzero point is the one close to 0.8 (the exact point differs between the two methods due to how they position the points). In both cases, the changes in the interpolation function over the entire interval are less than the change at the given data point. In other words, if the data value is changed by a small amount, then the interpolation function over the interval is changed by no more than this value. However, it is also apparent that the changes in the Chebyshev function are more widespread than for the cubic spline.

5. In the early days of spline research, they used the cubic spline version of the piece linear hat functions $G_i(x)$. These functions were usually designated as $L_i(x)$ and they were defined as the cubic splines that satisfied $L_i(x_i) = 1$ and $L_i(x_j) = 0$ for $j \neq i$ (see, e.g., de Boor and Schoenberg 1976). These were called the fundamental functions, and one is shown in Figure 5.23. The reason for introducing them is that they have the nice property that the spline interpolation function is simply

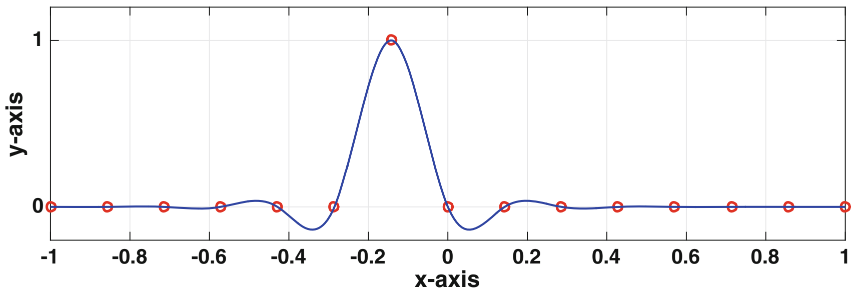


Figure 5.23 A fundamental function for (clamped) cubic splines. Shown is $L_7(x)$.

$$s(x) = \sum_{i=1}^{n+1} y_i L_i(x).$$

However, this hides all the work needed to determine the spline. In particular, finding the L_i's requires the solution of $n+1$ matrix equations.

Exercises

5.1. In this problem the data are: $(x_1, y_1) = (0, 0)$, $(x_2, y_2) = (1, 1)$, and $(x_3, y_3) = (2, 3)$.
(a) Find the global interpolation polynomial that fits these data.
(b) Find the piecewise linear interpolation function that fits these data.
(c) Find the natural cubic spline that fits these data.

5.2. This problem concerns the data in Table 5.6.
(a) Find the piecewise linear interpolation function $g(x)$ that fits these data.
(b) Find the global interpolation polynomial $p_3(x)$ that fits these data.

5.3. Use a Lagrange interpolating polynomial of degree 1 to find an approximate value for the following. Not all of the data points are needed, and you should explain which ones you use and why.
(a) $f(2.4)$ if $f(2.1) = 1$, $f(2.3) = 1.2$, $f(2.6) = 1.3$, $f(2.7) = 2$
(b) $f(-0.1)$ if $f(0.1) = 2$, $f(0) = 0.1$, $f(-0.2) = -0.1$, $f(0.4) = -0.5$
(c) $f(1)$ if $f(0.5) = -1$, $f(0.8) = -0.5$, $f(1.1) = 0.5$, $f(1.2) = 1$

5.4. Redo the previous problem but use a Lagrange interpolating polynomial of degree 2.

5.5. If $f(\theta) = \sin\theta$, then $f(0) = 0$, $f(\pi/4) = \sqrt{2}/2$, and $f(\pi/2) = 1$. Use these data points to answer the following questions. Note that the error that is asked for is the absolute value of the difference between the exact value $f(\pi/8) = \sqrt{2-\sqrt{2}}/2$ and the estimated value.

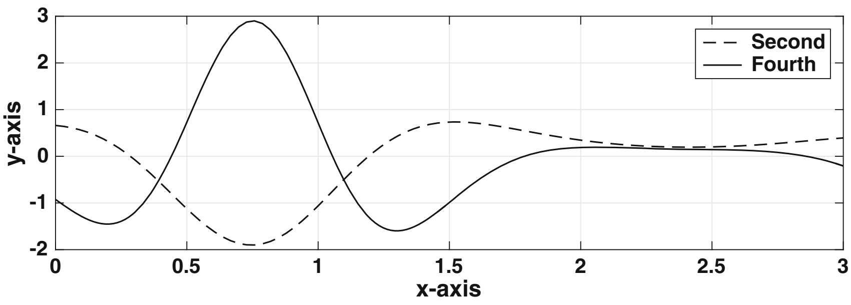


Figure 5.24 Graph used in Exercise 5.6.

x	-1	0	1	2
y	0	1	1	0

Table 5.6 Data for Exercise 5.2.

(a) Using piecewise linear interpolation, what is the estimated value of $f(\pi/8)$? What is the error in this estimate?
(b) Using a global interpolation polynomial, what is the estimated value of $f(\pi/8)$? What is the error in this estimate?
(c) Using natural cubic spline interpolation, what is the estimated value of $f(\pi/8)$? What is the error in this estimate?
(d) Using the additional information that $f'(0) = 1$ and $f'(\pi/2) = 0$, use clamped cubic spline interpolation to find an estimated value of $f(\pi/8)$. What is the error in this estimate?
(e) Suppose Chebyshev interpolation is used. Determine the three Chebyshev points in the interval, and evaluate $f(\theta)$ at these points. What is the resulting estimated value of $f(\pi/8)$? What is the error in this estimate?

5.6. A function $f(x)$ is going to be approximated using an interpolation function for $0 \leq x \leq 3$. The second, $f''(x)$, and fourth, $f''''(x)$, derivatives of the function are plotted in Figure 5.24.
(a) How many data points for piecewise linear interpolation are needed to guarantee the error is less than 10^{-8}?
(b) How many data points for a clamped cubic spline are needed to guarantee the error is less than 10^{-8}?

5.7. The function $y = \log_{10} x$ is going to be approximated using an interpolation function for $1 \leq x \leq 10$.
(a) How many data points for piecewise linear interpolation are needed to guarantee the error is less than 10^{-6}?

(b) How many data points for a clamped cubic spline are needed to guarantee the error is less than 10^{-6}?
(c) How many data points are needed when using Chebyshev interpolation to guarantee the error is less than 10^{-6}?

5.8. For the following functions, determine a step size h that will guarantee that the error is less than 10^{-6} using piecewise linear interpolation.
(a) $f(x) = x^{10}$, for $-1 \leq x \leq 1$.
(b) $f(x) = \ln(x)$, for $1 \leq x \leq 2$.
(c) $f(x) = 2\sin(3x) + 3\sin(2x)$, for $0 \leq x \leq \pi$.

5.9. Redo the previous problem but use a clamped cubic spline.

5.10. The Bessel function of order zero can be written as

$$J_0(x) = \frac{1}{\pi}\int_0^\pi \cos(x \sin s)ds.$$

(a) Show that $|J_0(x)| \leq 1$, $|J_0'(x)| \leq 1$, $|J_0''(x)| \leq 1$, and in fact, for any positive integer k,

$$\left|\frac{d^k}{dx^k}J_0(x)\right| \leq 1.$$

In what follows, determine how many interpolation points over the interval $0 \leq x \leq 10$ are needed so the error is no more than 10^{-6}.
(b) Using piecewise linear interpolation.
(c) Using a clamped cubic spline.
(d) Using Chebyshev interpolation.
(e) The Bessel function of order m can be defined as

$$J_m(x) = \frac{1}{\pi}\int_0^\pi \cos(x \sin s - ms)ds.$$

How do your answers in parts (b)–(d) change for this function?

5.11. This problem considers the function

$$g(x) = \begin{cases} 2 + 3x^2 + \alpha x^3 & \text{if} \quad -1 \leq x \leq 0, \\ 2 + \beta x^2 - x^3 & \text{if} \quad 0 \leq x \leq 1. \end{cases}$$

(a) For what values of α and β, if any, is $g(x)$ a cubic spline for $-1 \leq x \leq 1$? These values are to be used in the remainder of this problem.
(b) What were the data points that gave rise to this cubic spline?
(c) For what values of α and β is $g(x)$ a natural cubic spline?
(d) For what values of α and β is $g(x)$ a clamped cubic spline?

5.12. This problem considers the function

$$g(x) = \begin{cases} 2(x+1)^3 + 5(x+1) - 13x & \text{if} \quad -1 \le x \le 0, \\ 2(1-x)^3 + 9x + 5(1-x) & \text{if} \quad 0 \le x \le 1. \end{cases}$$

(a) Show that this is a cubic spline, and determine the data values used in its construction.
(b) Is this a natural cubic spline?
(c) Is this a clamped cubic spline?

5.13. Consider the function

$$g(x) = \begin{cases} x^3 - 1 & \text{if} \quad 0 \le x \le 1, \\ -x^3 + 6x^2 - 6x + 1 & \text{if} \quad 1 \le x \le 2. \end{cases}$$

Is $g(x)$ a cubic spline for $0 \le x \le 2$? If it is, is it natural, clamped, or neither? Make sure to justify your answers.

5.14. The data considered here are the population of a country for the years $x_1 = 1900$, $x_2 = 1910$, $x_3 = 1920$, $x_4 = 1930$, $\cdots$, $x_{12} = 2010$. The y_i's are the corresponding population values, and they should be given per million. For example, the population of the USA is given in Table 5.7, and this is from the Wikipedia page *Demographics of the United States.*
(a) Fit this data with: i) a global polynomial using Lagrange interpolation, and ii) a natural cubic spline. Plot these two curves and the data on the same axis. Make sure to include a legend in your plot.
(b) What do each of the two interpolation functions give as the population in 2005?
(c) What do each of the two interpolation functions predict the population will be in 2015?

5.15. The data considered here are the temperatures over a 24 hour period, in two hour increments. So, $x_1 = 0$, $x_2 = 2$, $x_3 = 4$, $x_4 = 6$, $\cdots$, $x_{13} = 24$. The y_i's are the corresponding temperatures. For example, the temperatures in Troy, NY on June 21 are given in Table 5.8 (in ^{o}F).
(a) Fit this data with: i) a global polynomial using Lagrange interpolation, and ii) a natural cubic spline. Plot these two curves and the data on the same axis. Make sure to include a legend in your plot.

x	1900	1910	1920	1930	1940	1950	1960	1970	1980	1990	2000	2010
y	76.21	92.23	106.0	123.2	132.2	151.3	179.3	203.3	226.5	248.8	281.4	308.7

Table 5.7 Sample population data for Exercises 5.14, 5.28, and 6.9.

(b) What do each of the two interpolation functions give as the temperature at 11 AM?

(c) What do the two interpolation functions predict the temperature will be at 1 AM the next day?

(d) What do the two interpolation functions predict the temperature will be at 9 AM the next day? Explain why the spline predicts the value it does.

5.16. The data below describe a cross-section of an airfoil where the points (X, Y_u) define the upper surface and the points (X, Y_ℓ) describe the lower surface.

X = [0, 0.005, 0.0075, 0.0125, 0.025, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]

Y_u = [0, 0.0102, 0.0134, 0.017, 0.025, 0.0376, 0.0563, 0.0812, 0.0962, 0.1035, 0.1033, 0.095, 0.0802, 0.0597, 0.034, 0]

Y_ℓ = [0, -0.0052, -0.0064, -0.0063, -0.0064, -0.006, -0.0045, -0.0016, 0.001, 0.0036, 0.007, 0.0121, 0.017, 0.0199, 0.0178, 0]

(a) Draw the airfoil by fitting cubic splines separately to the upper and lower surfaces, and then plotting the results as a single figure. To make it look like an airfoil you will probably need to resize the plot window.

(b) Redo (a) but use global polynomial interpolation instead of splines.

5.17. This problem considers some of the difficulties interpolating the function $f(x) = \sqrt{x}$.

(a) If the interpolation interval is $1 \leq x \leq 10$, how many data points are needed for piecewise linear interpolation to guarantee that the error is less than 10^{-6}?

(b) Explain why Theorem 5.4 is not so useful if the interval is $0 \leq x \leq 1$.

(c) One way to deal with the singularity at $x = 0$ is to break the interval into two segments, one is $0 \leq x \leq \delta$ and the other is $\delta \leq x \leq 1$, where δ is a small positive number. On the interval $0 \leq x \leq \delta$ the function is going to be interpolated with a single line. What is the equation for this line, and how small does δ need to be to guarantee that the approximation error is 10^{-6}?

(d) Assuming that δ is known, how many data points over the interval $\delta \leq x \leq 1$ are needed for piecewise linear interpolation to guarantee that the error is less than 10^{-6}?

5.18. The function $f(x) = 1/(1 + x^2)$ is to be approximated using a piecewise linear function $g(x)$ over the interval $0 \leq x < \infty$. The requirement is that $|f(x) - g(x)| \leq 10^{-4}$ for $0 \leq x < \infty$. Explain how to determine the spacing of the x_i's used in the construction of $g(x)$, and how you handle the approximation over the intervals $x_n \leq x \leq x_{n+1}$ and $x_{n+1} \leq x < \infty$, where x_{n+1} is the largest node you use.

5.19. This problem concerns interpolating the function $f(x) = \sin \pi x$ over the interval $1 \leq x \leq 3$, using three data points, with $x_1 = 1$, $x_2 = 2$, and $x_3 = 3$.

x	0	2	4	6	8	10	12	14	16	18	20	22	24
y	59	56	53	54	60	67	72	74	75	74	70	65	61

Table 5.8 Sample temperature data for Exercise 5.15. Note $x = 0$ and $x = 24$ correspond to midnight.

(a) Find the global interpolation polynomial that fits this data.
(b) Find the piecewise linear interpolation function that fits this data.
(c) Find the natural cubic spline that fits this data.
(d) Find the clamped cubic spline that fits this data.
(e) Chebyshev interpolation cannot use the stated x_i's. What are the three Chebyshev interpolation points for this interval, and what is the resulting interpolation function?

5.20. In this problem $x_i = i$, for $i = 1, 2, 3, 4$, and

$$s(x) = B_0(x) - B_1(x) + B_2(x) - B_3(x) + B_4(x) - B_5(x),$$

for $1 \leq x \leq 4$.
(a) What data points (x_i, y_i) were used to produce this cubic spline?
(b) Is this a natural cubic spline?
(c) Is it a clamped cubic spline?

5.21. In this problem $x_i = i$, for $i = 1, 2, 3, 4$, and

$$s(x) = B_2(x) + 5B_4(x),$$

for $1 \leq x \leq 4$.
(a) What data points (x_i, y_i) were used to produce this cubic spline?
(b) Is this a natural cubic spline?
(c) Is it a clamped cubic spline?

5.22. Given a data set (x_1, y_1), (x_2, y_2), $\cdots$, (x_{101}, y_{101}) suppose one of the following interpolation methods is to be used: Lagrange interpolation, piecewise linear interpolation, cubic spline interpolation using cubic B-splines.
(a) Order them by the number of flops needed to determine the interpolation function. Make sure to explain how you arrive at your answer. Assume the Thomas algorithm is used for the cubic spline (see Section 3.8).
(b) Order the methods by the number of flops needed to evaluate them at a given point (assume this point isn't in the data set). Make sure to explain how you arrive at your answer.

5.23. Suppose Chebyshev interpolation is applied to a function $f(x)$ using eight interpolation points. Also, suppose that no matter what interpolation interval is used, the error for the Chebyshev interpolation is zero. What conclusion can you make about the original function $f(x)$?

5.24. Given three points x_{i-1}, x_i, and x_{i+1}, this problem considers the quadratic interpolation formula

$$p_2(x) = y_{i-1}\ell_{i-1}(x) + y_i\ell_i(x) + y_{i+1}\ell_{i+1}(x).$$

It is assumed here that $x_i - x_{i-1} = h$ and $x_{i+1} - x_i = h$.

(a) Show that the above interpolation formula can be rewritten as

$$p_2(x) = y_i + \frac{1}{2h}(y_{i+1} - y_{i-1})(x - x_i) + \frac{1}{2h^2}(y_{i+1} - 2y_i + y_{i-1})(x - x_i)^2.$$

(b) Calculate $p_2'(x)$ and $p_2''(x)$.

(c) Suppose $p_2(x)$ interpolates $f(x)$ at x_{i-1}, x_i, and x_{i+1}, so $y_{i-1} = f(x_{i-1})$, $y_i = f(x_i)$, and $y_{i+1} = f(x_{i+1})$. Setting $x = x_i + \alpha h$, expand $f(x)$ and $p_2(x)$ about $h = 0$ and show that

$$f(x) = p_2(x) + \frac{1}{6}z(z^2 - h^2)f'''(x_i) + \frac{1}{24}z^2(z^2 - h^2)f''''(x_i) + \cdots,$$

where $z = x - x_i$.

(d) How does the result in part (c) compare to the result in Theorem 5.2 in the case of when $a = x_i - h$, $b = x_i + h$, and $n = 2$?

5.25. It is possible when solving for the coefficients for a cubic spline that one of the s_i's turns out to a linear function (versus a full cubic). This exercise explores this situation. Suppose there are three data points, with $x_1 = 0$, $x_2 = 1$, and $x_3 = 2$ and let

$$s(x) = \begin{cases} a + bx & \text{if } \ 0 \le x \le 1, \\ a_2 + b_2(x-1) + c_2(x-1)^2 + d_2(x-1)^3 & \text{if } \ 1 \le x \le 2. \end{cases}$$

(a) To be a cubic spline it is required that $s \in C^2(0, 2)$. What conditions must be imposed on the coefficients so this happens?

(b) Under what conditions, if any, is this a natural cubic spline?

5.26. This exercise explores some of the differences between a cubic polynomial and a cubic spline. In this problem the data are: $(x_1, y_1) = (0, 0)$, $(x_2, y_2) = (1, 1)$, $(x_3, y_3) = (2, 0)$, and $(x_4, y_4) = (3, 1)$.

(a) Find the global interpolation polynomial that fits this data, and then evaluate this function at $x = 1/2$.

(b) Find the natural cubic spline that fits this data, and then evaluate this function at $x = 1/2$.

(c) The cubic in part (a) satisfies the interpolation and smoothness conditions required of a spline, yet it produces a different result than the cubic spline in part (b). Why?

(d) What boundary conditions should be used so the cubic spline produces the cubic in part (a)?

5.27. The objective of this problem is to find a method that can evaluate $f(x) = \cos x$, for $0 \leq x \leq 2\pi$, with an error of no more than 10^{-6}. In doing this, the interpolation points are restricted to those x_i's for which the exact value of $\cos x_i$ is known. It is useful to know that, by considering the angles in a polygon, it is possible to determine the exact values of $\cos x$ and $\sin x$ for $x = \pi/10$, $\pi/12$, $\pi/15$, etc. (these are given on the Wikipedia page *Exact trigonometric constants*).

(a) Show that if the values of $\cos x$ and $\sin x$ are known for $x = \pi/k$, then they are known at $x = m\pi/k$, for $m = 2, 3, 4, \cdots$.

(b) For a given value of k, let $h = \pi/k$ and suppose that the interpolation points are $x_i = (i-1)h$, for $i = 1, 2, \cdots, n+1$. Find n in terms of k.

(c) According to Theorem 5.4, how small must h be so the error using piecewise linear interpolation with $f(x) = \cos x$ is no more than 10^{-6}? What is the smallest value of k so that $\pi/k \leq h$?

(d) According to Theorem 5.6, how small must h be so the error using a clamped cubic spline with $f(x) = \cos x$ is no more than 10^{-6}? What is the smallest value of k so that $\pi/k \leq h$?

(e) For a given value of x, describe a procedure that uses the exact values of $\cos x$ and/or $\sin x$ to evaluate $f(x) = \cos x$, for $0 \leq x \leq 2\pi$, with an error of less than 10^{-6}.

(f) Write a MATLAB program that implements your algorithm in part (e) and compares the computed values with MATLAB's built in cosine function, for $x = 1, 2, 5$.

5.28. This problem explores how to scale the data to help improve the computability of the interpolation polynomial. We consider the direct approach to determine $p_n(x)$, and as usual the data points are (x_1, y_1), (x_2, y_2), $\cdots$, (x_{n+1}, y_{n+1}), where $x_1 < x_2 < \cdots < x_{n+1}$. The x values are going to be scaled by letting

$$z = \frac{x - \alpha}{\beta},$$

where α and β are given numbers with $\beta > 0$. The data point (x_i, y_i) in this case changes to (z_i, y_i), where $z_i = (x_i - \alpha)/\beta$. Also, the interpolation polynomial also changes to

$$p_n(z) = a_0 + a_1 z + \cdots + a_n z^n.$$

(a) The original data interval is $x_1 \leq x \leq x_{n+1}$. What is the data interval when using z? What matrix equation must be solved to find the a_i's in the above formula for $p_n(z)$?

(b) The values for α and β are going to be selected so the z data interval is $-1 \leq z \leq 1$. What are α and β in this case?

(c) Using the population data from Exercise 5.14, plot the interpolation function using the direct approach on the original x_i data set. Also compute the condition number for $\mathbf{V}$.

(d) Using the population data from Exercise 5.14, scale the data based on the result from part (b), and then find the coefficients for $p_n(z)$. What is the condition number of the matrix in this case? Once the a_i's are computed then in terms of the original x variable,

$$p_n(x) = a_0 + a_1\left(\frac{x-\alpha}{\beta}\right) + a_2\left(\frac{x-\alpha}{\beta}\right)^2 + \cdots + a_n\left(\frac{x-\alpha}{\beta}\right)^n .$$

Plot this function and compare the result with what you found in part (c).

5.29. This problem concerns a method to reduce the computational effort to evaluate the Lagrange interpolation function given in (5.5).

(a) What is the flop count to evaluate (5.5) for a given value of x?

(b) Assuming that $x \neq x_i$, for any i, show that (5.5) can be written as

$$p_n(x) = \ell(x)\sum_{i=1}^{n+1} w_i y_i/(x-x_i)\,,$$

where $\ell(x) = \prod_{j=1}^{n+1}(x-x_j)$ and

$$w_i = 1/\prod_{\substack{j=1\\ j\neq i}}^{n+1}(x_i - x_j)\,.$$

This is known as the first form of the barycentric interpolation formula, and w_i's are called barycentric weights.

(c) Suppose the formula in part (b) is used to interpolate the constant function $f(x) = 1$. Use this to show that

$$\ell(x)\sum_{i=1}^{n+1} w_i/(x-x_i) = 1.$$

(d) Use the result from part (c) to show that

$$p_n(x) = \frac{\sum_{i=1}^{n+1} w_i y_i/(x-x_i)}{\sum_{i=1}^{n+1} w_i/(x-x_i)}\,,$$

This is called the second (true) form of the barycentric formula.

(e) What is the flop count to evaluate the formula for $p_n(x)$ given in part (d)?

5.30. The cubic spline interpolation function is

$$s(x) = \sum_{i=0}^{n+2} a_i B_i(x).$$

For a natural spline we found the a_i's by solving a matrix equation $\mathbf{Aa} = \mathbf{z}$. The purpose of this exercise is to find what this equation is for a clamped spline. Recall that for a clamped spline it is required that $s'(x_1) = y_1'$ and $s'(x_{n+1}) = y_{n+1}'$, where y_1' and y_{n+1}' are given. Show $\mathbf{a} = (a_1, a_2, \cdots, a_{n+1})^T$, $\mathbf{z} = (6y_1 + 2hy_1', 6y_2, \cdots, 6y_n, 6y_{n+1} - 2hy_{n+1}')^T$, and $\mathbf{A}$ is the $n \times n$ tridiagonal matrix

$$\mathbf{A} = \begin{pmatrix} 4 & 2 & & & & \\ 1 & 4 & 1 & & 0 & \\ & 1 & 4 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & 0 & & 1 & 4 & 1 \\ & & & & 2 & 4 \end{pmatrix}.$$

Also, once $\mathbf{a}$ is determined, then $a_0 = a_2 - 2hy_1'$ and $a_{n+2} = a_n + 2hy_{n+1}'$.

5.31. This problem concerns some of the inequalities arising for function interpolation.

(a) For q_{n+1}, given in (5.26), assume x is not one of the x_i's. So, there is an x_i so that $x_i < x < x_{i+1}$. With this, it is possible to write

$$q_{n+1}(x) = (x - x_i)(x - x_{i+1}) \prod_{j=1}^{i-1}(x - x_j) \prod_{j=i+2}^{n+1}(x - x_j).$$

Show that $|(x - x_i)(x - x_{i+1})| \leq \frac{1}{4}h^2$. Also show that $|\prod_{j=1}^{i-1}(x - x_j)| \leq i!h^{i-1}$ and $|\prod_{j=i+2}^{n+1}(x - x_j)| \leq (n + 1 - i)!h^{n-i}$. From this, derive (5.27). Make sure to comment about the case of when x equals one of the x_i's.

(b) It is possible to prove that for every positive integer n [Sandor and Debnath, 2000],

$$\sqrt{2\pi n}\left(\frac{n}{e}\right)^n \leq n!.$$

Use this, and Theorem 5.8, to derive (5.35).

5.32. This problem considers a direct proof of Theorem 5.9, at least for the case of when $n = 1$. This will help demonstrate how this result is independent of the coefficients of the polynomial.

(a) If $P_1(x) = x + b_0$, explain why

$$\max_{-1 \leq x \leq 1} |P_1(x)| = \max\{\, |1 + b_0|, |-1 + b_0| \,\}.$$

(b) Sketch the two absolute values in part (a) as a function of b_0. Use this to explain why $\max_{-1 \leq x \leq 1} |P_1(x)| = 1 + |b_0|$. From this derive the result stated in the theorem.

5.33. This problem derives the formulas for a piecewise quadratic interpolation function. This function is written as

$$w(x) = \begin{cases} w_1(x) & \text{if } x_1 \le x \le x_2 \\ w_2(x) & \text{if } x_2 \le x \le x_3 \\ \quad\vdots & \qquad\vdots \\ w_n(x) & \text{if } x_n \le x \le x_{n+1}, \end{cases}$$

where

$$w_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2, \quad \text{for } x_i \le x \le x_{i+1}.$$

The interpolation requirements are $w_i(x_i) = y_i$ and $w_i(x_{i+1}) = y_{i+1}$. Also, $w'(x)$ is required to be continuous, which means that $w_i'(x_{i+1}) = w_{i+1}'(x_{i+1})$ for $i = 1, 2, \cdots, n-1$.

(a) Explain why the stated requirements are not enough, and it is necessary to impose one additional condition. In this problem, it is assumed that b_1 is specified. Explain why this is the same as specifying the value of the slope $w'(x_1)$.

(b) From the stated requirements, deduce that $a_i = y_i$ for $i = 1, 2, \cdots, n+1$. Also, $b_i = -b_{i-1} + 2(y_i - y_{i-1})/h$ for $i = 2, 3, \cdots, n$, and $c_i = (y_{i+1} - y_i - hb_i)/h^2$ for $i = 1, 2, \cdots, n$.

(c) Explain why the results from parts (a) and (b) mean that one can determine the coefficients for w_1, then determine the coefficients for w_2, then for w_3, etc.

(d) Use $w(x)$ to interpolate $f(x) = \cos(6\pi x)$, over the interval $0 \le x \le 1$. Plot $w(x)$ and $f(x)$ for the following cases: i) $n = 5$, ii) $n = 10$, iii) $n = 15$, and iv) $n = 30$. Comment on how well the interpolation method works for this particular function.

Chapter 6
Numerical Integration

6.1 Introduction

The objective of this chapter is to derive and then test methods that can be used to evaluate the definite integral

$$\int_a^b f(x)dx.$$

In most calculus textbooks the examples and problems dedicated to integration are not particularly complicated, although some require a clever combination of methods to carry out the integration. In the real world the situation is much worse. As an example, to find the deformation of an elastic body when compressed by a rigid punch it is necessary to evaluate [Gladwell, 1980]

$$\int_0^\ell \left(\frac{2\alpha\sinh(x) - 2x}{1+\alpha^2+x^2+2\alpha\cosh(x)} - 1\right)\sin(\lambda x)dx. \tag{6.1}$$

Moreover, it is relatively easy to find integrals even worse than the one above. To illustrate, in the study of the emissions from a pulsar it is necessary to evaluate [Gwinn et al., 2012]

$$\int_0^1 \frac{1-x}{\alpha_0+\beta_0 x+\gamma_0 x^2+\delta_0 x^3}\, K_2\!\left(\frac{\sqrt{\alpha_1+\beta_1 x+\gamma_1 x^2}}{\alpha_2+\beta_2 x}\right)\exp\!\left(\frac{\alpha_3+\beta_3 x}{\alpha_4+\beta_4 x}\right)dx,$$

where K_2 is the modified Bessel function. The point here is that effective numerical methods for evaluating integrals are needed, and our objective is to determine what they are.

It is of interest to know that many of the integration methods derived in this chapter are summarized in Appendix C, Table C.2.

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_6

6.2 The Definition from Calculus

The definition of a definite integral introduced in calculus serves as the basis of most numerical integration methods. To review this definition, one first subdivides the interval as shown in Figure 6.1. In this case, $x_1 < x_2 < \cdots < x_{n+1}$, where $a = x_1$ and $b = x_{n+1}$. The reason for this is the additive property of areas, which for an integral can be written as

$$\begin{aligned}\int_a^b f(x)dx &= \int_{x_1}^{x_2} f(x)dx + \int_{x_2}^{x_3} f(x)dx + \cdots + \int_{x_n}^{x_{n+1}} f(x)dx \\ &= \sum_{i=1}^{n} \int_{x_i}^{x_{i+1}} f(x)dx. \qquad (6.2)\end{aligned}$$

Out of each subinterval $[x_i, x_{i+1}]$ one picks a point c_i and then approximates the area with the quantity $f(c_i)(x_{i+1} - x_i)$. This is illustrated in Figure 6.2. According to the definition, the value of the integral is approached as the number of subdivisions increases. In other words,

$$\int_a^b f(x)dx \equiv \lim_{n\to\infty} \sum_{i=1}^{n} f(c_i)(x_{i+1} - x_i).$$

This is useful because given any particular subdivision we have the approximation

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} f(c_i)(x_{i+1} - x_i).$$

Moreover, we are free to pick c_i from the subinterval $[x_i, x_{i+1}]$ any way we want, so we are able to produce different numerical methods depending on how we make this choice.

The examples usually considered in calculus often involve picking one of the endpoints, namely either $c_i = x_i$ or $c_i = x_{i+1}$. Another is the midpoint $c_i = \frac{1}{2}(x_i + x_{i+1})$. Given that we are on the verge of generating several different algorithms to calculate the integral we need some way to determine

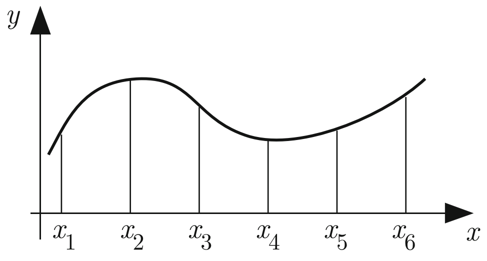


Figure 6.1 Subdivision of the interval of integration in the case of when $n = 5$.

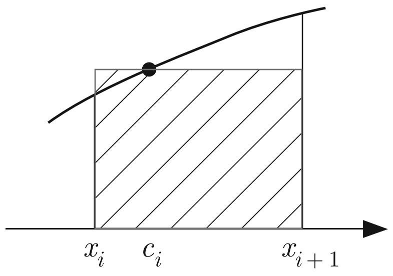


Figure 6.2 Rectangular region used as an approximation of the area over the subinterval.

how well they work. To be specific, whatever choice is made the resulting integration rule over each subinterval is

$$\int_{x_i}^{x_{i+1}} f(x)dx \approx f(c_i)(x_{i+1} - x_i). \tag{6.3}$$

In what follows it is assumed that the grid points are equally spaced, so $x_{i+1} - x_i = h$. Taylor's theorem, as usual, can be used to determine the accuracy of the approximation. First, expanding about $x = c_i$ we get

$$f(x) = f(c_i) + (x - c_i)f'(c_i) + \frac{1}{2}(x - c_i)^2 f''(c_i) + \cdots . \tag{6.4}$$

With this

$$\begin{aligned}\int_{x_i}^{x_{i+1}} f(x)dx &= \int_{x_i}^{x_{i+1}} [f(c_i) + (x - c_i)f'(c_i) + \frac{1}{2}(x - c_i)^2 f''(c_i) + \cdots]dx \\ &= f(c_i)h + 2hz_i f'(c_i) + \frac{1}{24}h(12z_i^2 + h^2)f''(c_i) + \cdots ,\end{aligned}$$

where $z_i = x_i + \frac{h}{2} - c_i$. Therefore, when using the approximation in (6.3), the error in the approximation is

$$\tau_r = 2hz_i f'(c_i) + \frac{1}{24}h(12z_i^2 + h^2)f''(c_i) + \cdots . \tag{6.5}$$

So, if we take $c_i = x_i$, then $z_i = h/2$ and $\tau_r = O(h^2)$. Similarly, if $c_i = x_{i+1}$, then $z_i = -h/2$ and $\tau_r = O(h^2)$. In other words, by picking one of the endpoints the error is $O(h^2)$. To get a better error we need to pick c_i so that $z_i = 0$, and this means we pick $c_i = x_i + \frac{h}{2}$. The error in this case is $O(h^3)$. In fact, the midpoint is the only choice in the above equation that guarantees an error that is $O(h^3)$ independently of the particular function $f(x)$.

6.2.1 Midpoint Rule

It is worth examining the $c_i = x_i + \frac{h}{2}$ choice in more detail. We have just shown that

$$\int_{x_i}^{x_{i+1}} f(x)dx = f(x_i + \frac{h}{2})h + O(h^3). \tag{6.6}$$

This gives rise to what is known as the *midpoint rule*. When this is put into the formula for the definite integral (6.2) we get

$$\begin{aligned}\int_a^b f(x)dx &= \left[f(x_1 + \frac{h}{2})h + O(h^3)\right] + \left[f(x_2 + \frac{h}{2})h + O(h^3)\right] \\ &\quad + \cdots + \left[f(x_n + \frac{h}{2})h + O(h^3)\right] \\ &= h(f_{1+1/2} + f_{2+1/2} + \cdots + f_{n+1/2}) + n\,O(h^3) \\ &= I_M + O(h^2), \end{aligned} \tag{6.7}$$

where

$$I_M = h(f_{1+1/2} + f_{2+1/2} + \cdots + f_{n+1/2}) \tag{6.8}$$

and $f_{i+1/2} = f(x_i + \frac{h}{2})$. The expression in (6.8) is known as the *composite midpoint rule*. Note that the error for the composite rule is a factor of h smaller than the integration rule (6.6). This happens with the other integration rules we study and to explain why, in (6.7) the term $n\,O(h^3)$ turned into $O(h^2)$. This is because $h = (b-a)/n$, so $n = O(1/h)$.

It is possible to modify the result in (6.5) to obtain a more explicit bound on the error and this is contained in the next result.

Theorem 6.1. *If $f \in C^2[a,b]$, then the composite midpoint rule (6.8) satisfies*

$$\left|\int_a^b f(x)dx - I_M\right| \le \frac{b-a}{24}h^2||f''||_\infty$$

where $||f''||_\infty = \max_{a\le x\le b}|f''(x)|$.

Examples

1. Suppose the composite midpoint rule is used to approximate the value of

$$\int_0^1 e^{3x}dx. \tag{6.9}$$

Using three subintervals, the approximation shown in Figure 6.3 is obtained. In this case, $h = 1/3$, and the midpoints are $x_{1+1/2} = 1/6$, $x_{2+1/2} = 1/2$, and $x_{3+1/2} = 5/6$. With this, (6.8) becomes

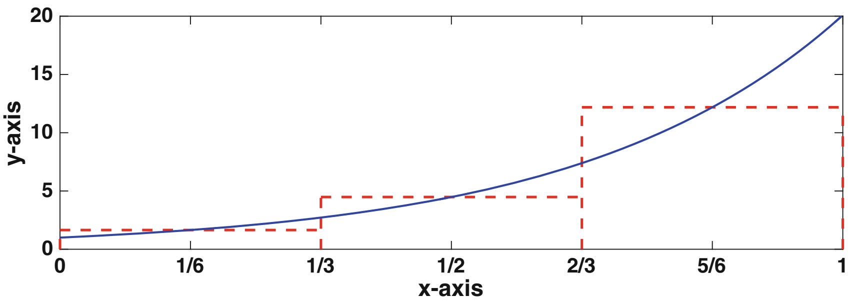


Figure 6.3 Composite midpoint rule used with $f(x) = e^{3x}$ for 3 subintervals. The area of the dashed (red) boxes is used as the approximate area for the area under the curve.

$$\begin{aligned} I_M &= \frac{1}{3}\left(e^{1/2} + e^{3/2} + e^{5/2}\right) \\ &\approx 6.1043. \quad \blacksquare \end{aligned}$$

2. According to Theorem 6.1, how many subintervals are necessary to guarantee an error of 10^{-8} if the composite midpoint rule is used to evaluate (6.9)?
   Answer: Since

$$\begin{aligned} ||f''||_\infty &= \max_{0\le x\le 1} 9e^{3x} \\ &= 9e^3, \end{aligned}$$

then we want $\frac{9}{24}h^2e^3 \le 10^{-8}$. This gives us $h \le 2 \times 10^{-4}\sqrt{2e^{-3}/3}$. Given that $h = (b-a)/n = 1/n$, then we want

$$\begin{aligned} n &\ge \frac{1}{2} \times 10^4\sqrt{3e^3/2} \\ &\approx 27{,}444.6. \end{aligned}$$

Therefore, according to Theorem 6.1, we should use at least 27,445 subintervals so the error is no more than 10^{-8}. To check on this, the computed values are given in Table 6.1, along with the value of the error

$$E_M = \left| \int_a^b f(x)dx - I_M \right|. \tag{6.10}$$

What is seen is that the desired error of 10^{-8} is obtained using a smaller number of subintervals than the predicted value of 27,445. This is not surprising, since the inequality in Theorem 6.1 is based on a worst-case

n	I_M	E_M
1000	6.36184325537107	2.39e−06
2000	6.36184504463955	5.96e−07
4000	6.36184549195681	1.49e−07
8000	6.36184560378610	3.73e−08
16000	6.36184563174344	9.32e−09
32000	6.36184563873278	2.33e−09

Table 6.1 Values of (6.9) when computed using the composite midpoint rule I_M, and the resulting error E_M.

assumption. As a final comment, although using 16,000 or 32,000 subintervals might sound large, the computing time is minimal, taking less than 10^{-2} sec. ■

The reason the midpoint rule works as well as it does is evident in Figure 6.3. Each box splits the curve in such a way that the (unsigned) area it misses on the right is about the same as the (unsigned) area it misses on the left. This balancing act is why the first term in the error, given in (6.5), is zero. Also, it is conventional to report computational results using tables, as is done in Table 6.1. However, it can be more informative to plot the results, particularly the error, and this is done in Figure 6.4. It is seen that if the number of subintervals increases by a factor of, say, 10, then the error drops by about a factor of 10^2. According to Theorem 6.1 this is exactly what should happen.

Examples

1. According to Theorem 6.1, what functions will the composite midpoint rule integrate exactly, no matter what the value of h?

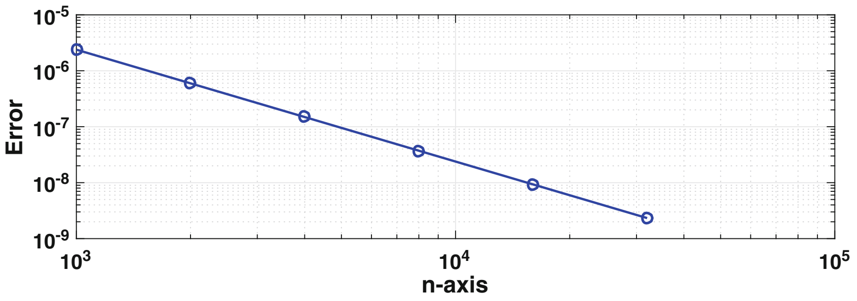


Figure 6.4 Values of the error E_M for the composite midpoint rule as given in Table 6.1.

This requires $||f''||_\infty = 0$, which means that $f''(x) = 0$ for $a \leq x \leq b$. Therefore, to have zero error it must be that $f(x) = \alpha + \beta x$, i.e., it must be a linear function in this interval. ■

2. Suppose that the error when using the composite midpoint rule is 10^{-4} if $n = 100$. What is the approximate error if $n = 200$?

   To answer this it is necessary to make an assumption about the error. Specifically, even though Theorem 6.1 cannot be used to determine the exact value of the error, it can be used to predict how the error depends on h. The assumption is that the error decreases as $O(h^2)$. This is true for the above example, and to verify this the values in Table 6.1 are plotted in Figure 6.4. In this plot it is seen that when doubling of the number of subintervals, the error decreases by a factor of 4. So, in response to what happens when using $n = 200$, the guess is that the error will be approximately $\frac{1}{4} \times 10^{-4}$. ■

As a final note, the explanation for the error formula in Theorem 6.1 used the Taylor series in (6.4). Using the series in this way gives rise to what is known as the asymptotic form of the error, and this is explained in Appendix C. A formal proof of the theorem can be obtained using the remainder form of the Taylor series and knowing certain inequalities involving integrals. This approach, using the series rather than the remainder version of a Taylor series to derive the formula for the error, is often used in this chapter. In each case, for those so inclined, a more rigorous proof can be obtained using similar modifications to the argument given.

6.3 Methods Based on Polynomial Interpolation

With the composite midpoint rule we have an $O(h^2)$ method that requires n function evaluations. This begs the question of whether we can do better and find methods that are, say, $O(h^4)$. The answer is definitely yes, and our approach to answering this question will use the ideas employed in the definition. As before, we will subdivide the interval. In the definition, the function is approximated with the constant $f(c_i)$, as shown in Figure 6.2, and this was then integrated exactly to produce the integration rule. To improve on this we simply use a better approximation for $f(x)$, one that can be integrated exactly, and for this we will use polynomial interpolation.

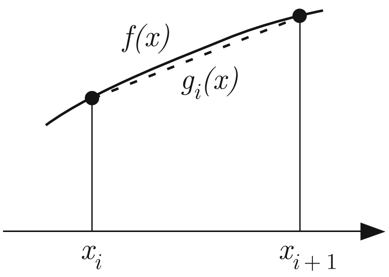


Figure 6.5 To derive the trapezoidal rule the function is approximated using linear interpolation.

6.3.1 Trapezoidal Rule

Instead of a constant we will use a linear approximation of $f(x)$ over $x_i \leq x \leq x_{i+1}$. In particular, we will use the linear function that interpolates the function at the endpoints (see Figure 6.5). From the point-slope formula, we have that

$$g_i(x) = f_i + \frac{1}{h}(f_{i+1} - f_i)(x - x_i).$$

By expanding $f(x_{i+1})$ using Taylor's Theorem, it is easy to show that

$$f(x) = g_i(x) + \frac{1}{2}(x - x_i)(x - x_{i+1})f''(x_i) + \cdots .$$

Therefore,

$$\begin{aligned} \int_{x_i}^{x_{i+1}} f(x)dx &= \int_{x_i}^{x_{i+1}} [g_i(x) + \frac{1}{2}(x - x_i)(x - x_{i+1})f''(x_i) + \cdots]dx \\ &= \frac{h}{2}(f_{i+1} + f_i) - \frac{1}{12}h^3 f''(x_i) + \cdots . \end{aligned} \tag{6.11}$$

This gives rise to what is known as the *trapezoidal rule*. When this is put into the formula for the definite integral (6.2) we get

$$\int_a^b f(x)dx = I_T + O(h^2),$$

where

$$I_T = h\left(\frac{1}{2}f_1 + f_2 + f_3 + \cdots + f_n + \frac{1}{2}f_{n+1}\right). \tag{6.12}$$

The expression in (6.12) is known as the *composite trapezoidal rule*.

Note that we have not done better than the composite midpoint rule in the sense that the error in both cases is $O(h^2)$. In fact, one might argue that the midpoint rule has a slight advantage over the trapezoidal rule because of the next result.

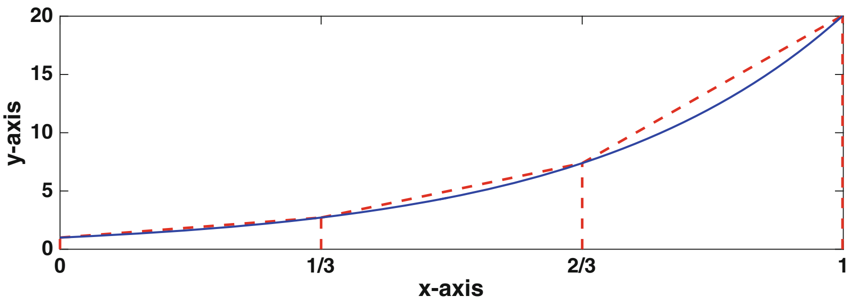


Figure 6.6 Composite trapezoidal rule used with $f(x) = e^{3x}$ for 3 subintervals. The area of the dashed (red) trapezoids is used as the approximate area for the area under the curve.

Theorem 6.2. *If $f \in C^2[a, b]$ then the composite trapezoidal rule (6.12) satisfies*

$$\left| \int_a^b f(x)dx - I_T \right| \leq \frac{b-a}{12} h^2 ||f''||_\infty$$

where $||f''||_\infty = \max_{a \leq x \leq b} |f''(x)|$.

The proof of this comes from using Taylor's theorem with remainder, in a manner similar to how (6.11) was derived.

Examples

1. Suppose the composite trapezoidal rule is used to approximate the value of

$$\int_0^1 e^{3x} dx. \tag{6.13}$$

Using three subintervals, the approximation shown in Figure 6.6 is obtained. In this case, $h = 1/3$, and (6.12) becomes

$$\begin{aligned} I_T &= \frac{1}{3}\left(\frac{1}{2} + e^1 + e^2 + \frac{1}{2}e^3\right) \\ &\approx 6.8834. \quad \blacksquare \end{aligned}$$

2. According to Theorem 6.2, how many subintervals are necessary to guarantee an error of 10^{-8} if the composite trapezoidal rule is used to evaluate (6.13)?
Answer: It was shown earlier that $||f''||_\infty = 9e^3$. So, to satisfy the inequality in the theorem, we want $\frac{3}{4}h^2 e^3 \leq 10^{-8}$. Since $h = 1/n$, it is required that $n \geq \frac{1}{2}\sqrt{3e^3} \times 10^4 \approx 38{,}812.6$. Therefore, according to Theorem 6.2,

n	I_T	E_T
1000	6.36185041244607	4.77e−06
2000	6.36184683390857	1.19e−06
4000	6.36184593927407	2.98e−07
8000	6.36184571561544	7.46e−08
16000	6.36184565970077	1.86e−08
32000	6.36184564572208	4.66e−09

Table 6.2 Values of (6.13) when computed using the composite midpoint rule I_T, as well as the error E_T.

we should use at least 38,813 subintervals so the error is no more than 10^{-8}. To check on this, the computed values are given in Table 6.2, along with the value of the error

$$E_T = \left| \int_a^b f(x)dx - I_T \right| . \tag{6.14}$$

What is seen is that the desired error of 10^{-8} is obtained using a somewhat smaller number of subintervals than the predicted value of 38,813. As pointed out for the midpoint rule, this is not surprising, since the inequality in Theorem 6.2 is based on a worst-case assumption. ■

It is worth comparing Figures 6.3 and 6.6. Although the piecewise linear approximation of the function is certainly better than the piecewise constant approximation, the midpoint rule produces a slightly better approximation of the integral. Although improving the approximation of $f(x)$ did not produce a better result, the idea is still worth pursuing as demonstrated next.

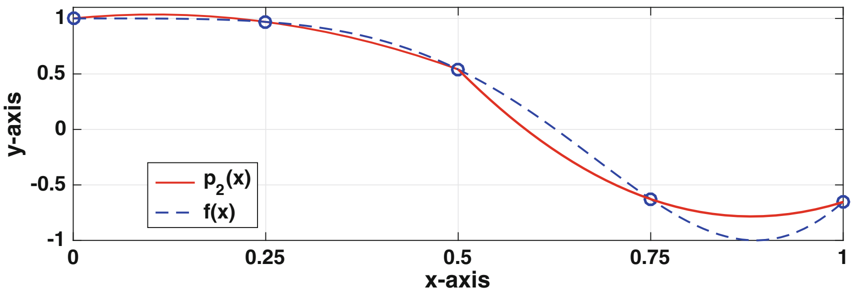


Figure 6.7 Example of the piecewise quadratic approximation $p_2(x)$ used in the derivation of Simpson's rule. One quadratic is used for $0 \leq x \leq \frac{1}{2}$ and another for $\frac{1}{2} \leq x \leq 1$.

6.3.2 Simpson's Rule

The next step is to try a quadratic approximation for $f(x)$, and for this we need three data points. One option is to use x_i, x_{i+1}, and some point within the subinterval. Another option is to pair up the subintervals and use an approximation over $x_1 \le x \le x_3$, another one over $x_3 \le x \le x_5$, etc. We will use the latter option although this will require n to be even. From (5.4), the quadratic that interpolates $f(x)$ over the interval $x_{i-1} \le x \le x_{i+1}$ is

$$p_2(x) = f_{i-1}\ell_{i-1}(x) + f_i\ell_i(x) + f_{i+1}\ell_{i+1}(x). \tag{6.15}$$

An example of the resulting approximation is shown in Figure 6.7 in the case when $n = 4$. There are two quadratics in this case, one used for $0 \le x \le \frac{1}{2}$ and another for $\frac{1}{2} \le x \le 1$. If you look closely, you will notice that over each subinterval the quadratic is above the function on one half, and below the function on the other half. This is also what happened for the midpoint rule, as seen in Figure 6.3, and it will result in the integration rule being more accurate than might be expected.

In deriving the resulting integration rule we are also interested in the error. Using Taylor's theorem (see Exercise 5.24(c)) one can show that

$$f(x) = p_2(x) + \frac{1}{6}z(z^2 - h^2)f''(x_i) + \frac{1}{24}z^2(z^2 - h^2)f''''(x_i) + \cdots ,$$

where $z = x - x_i$. Integrating both sides of this expression, one finds that

$$\begin{aligned}\int_{x_{i-1}}^{x_{i+1}} f(x)dx &= \int_{x_{i-1}}^{x_{i+1}} [p_2(x) + \frac{z}{6}(z^2 - h^2)f_i''' + \frac{z^2}{24}(z^2 - h^2)f_i'''' + \cdots]dx \\ &= \int_{x_{i-1}}^{x_{i+1}} p_2(x)dx \\ &\qquad + \int_{-h}^{h} \frac{z}{6}(z^2 - h^2)f_i''' + \frac{z^2}{24}(z^2 - h^2)f_i'''' + \cdots]dz \\ &= \frac{h}{3}(f_{i-1} + 4f_i + f_{i+1}) - \frac{1}{90}h^5 f_i'''' + \cdots . \end{aligned} \tag{6.16}$$

This gives rise to what is known as *Simpson's rule*, which is that

$$\int_{x_{i-1}}^{x_{i+1}} f(x)dx \approx \frac{h}{3}(f_{i-1} + 4f_i + f_{i+1}).$$

It remains to recombine the subintervals, and the result is

$$\int_a^b f(x)dx = I_S + O(h^4),$$

where

$$I_S = \frac{h}{3}(f_1 + 4f_2 + 2f_3 + 4f_4 + 2f_5 + \cdots + 4f_n + f_{n+1})\,. \tag{6.17}$$

The expression in (6.17) is known as the *composite Simpson's rule*. By keeping careful track of the approximation errors made with this rule one can prove the following.

Theorem 6.3. *If $f \in C^4[a,b]$, then the composite Simpson's rule (6.17) satisfies*

$$\left|\int_a^b f(x)dx - I_S\right| \le \frac{b-a}{90}h^4||f''''||_\infty$$

where $||f''''||_\infty = \max_{a\le x\le b}|f''''(x)|$.

This is an impressive result, and it is clearly better than what was achieved with the midpoint or trapezoidal rule. In fact, it is the "gold standard" in the sense that when new integration rules are derived they are almost invariably compared to Simpson's rule.

Example

According to Theorem 6.3, how many subintervals are necessary to guarantee an error of 10^{-8} if the composite Simpson's rule is used to evaluate

$$\int_0^1 e^{3x}dx? \tag{6.18}$$

Since $||f''''||_\infty = 81e^3$, then we want $\frac{9}{10}h^4e^3 \le 10^{-8}$. This gives us $h \le (10e^{-3}/9)^{1/4}\times 10^{-2}$. Recall that $h = (b-a)/n = 1/n$. Combining our results, we require $n \ge (9e^3/10)^{1/4}\times 10^2 \approx 206.0$. So, according to Theorem 6.3, we

n	I_S	E_S
10	6.36212888551990	2.83e−04
20	6.36186348593954	1.78e−05
40	6.36184675860732	1.12e−06
80	6.36184571094418	6.99e−08
160	6.36184564543071	4.37e−09

Table 6.3 Values of (6.18) when computed using the composite Simpson's rule I_S, as well as the error E_S.

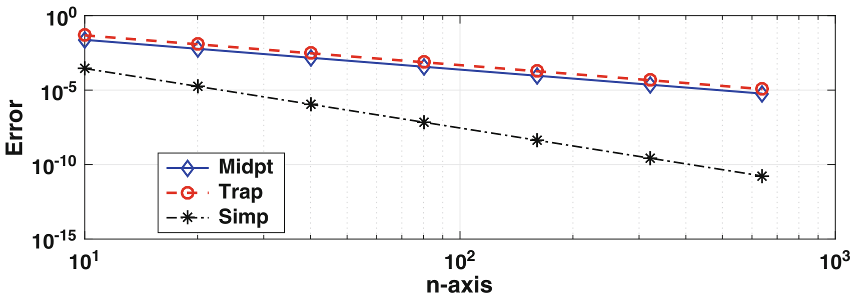


Figure 6.8 The errors E_M, E_T, and E_S for the composite midpoint, trapezoidal, and Simpson's rules, respectively, as a function of the number n of subintervals used to evaluate (6.18).

should use at least 207 subintervals so the error is no more than 10^{-8}. To check on this, the computed values are given in Table 6.3, along with the value of the error

$$E_S = \left| \int_a^b f(x)dx - I_S \right| . \tag{6.19}$$

What is seen is that the desired error of 10^{-8} is obtained using a somewhat smaller number of subintervals than the predicted value of 207. It is also informative to plot the error E_S as a function of the number of subintervals, and to compare this with the midpoint error E_M, and the trapezoidal error E_T. This is done in Figure 6.8. As expected, E_M and E_T are very similar, decreasing in the predicted $O(h^2)$ manner. It is also evident how much better Simpson's rule does compared to the other two methods. ■

Examples

1. According to Theorem 6.3, what functions will the composite Simpson rule integrate exactly, no matter what the value of h?

   This requires $||f''''||_\infty = 0$, which means that $f''''(x) = 0$ for $a \le x \le b$. Therefore, the composite Simpson rule will integrate cubics exactly. ■

2. Suppose that the error when using the composite midpoint rule is 10^{-4} if $n = 100$. What is the approximate error if $n = 200$?

   Assuming the error decreases as $O(h^4)$, then for $n = 200$ the error will be approximately $2^{-4} \times 10^{-4} \approx 6 \times 10^{-6}$. ■

6.3.3 Cubic Splines

One of the best interpolation methods considered in Chapter 5 used cubic splines, and this makes it a intriguing method for numerical integration. One difference with the other interpolation based methods we have considered, the cubic spline interpolation formula applies to the whole interval. A consequence of this is that we will obtain the composite rule directly, without first deriving the formula over the subintervals which are then added to produce a composite rule.

As explained in Section 5.4.1, the cubic spline interpolation formula can be written as

$$s(x) = \sum_{i=0}^{n+2} a_i B_i(x),$$

where the B_i's are cubic B-splines and are defined in (5.22). Using the integration formulas given in Appendix B, one finds that

$$\begin{aligned}\int_a^b s(x)dx &= \sum_{i=0}^{n+2} a_i \int_a^b B_i(x)dx \\ &= h\left(\sum_{i=1}^{n+1} f_i + \frac{1}{24}(a_2 - a_0) - \frac{1}{2}f_1 - \frac{1}{24}(a_{n+2} - a_n) - \frac{1}{2}f_{n+1}\right). \qquad (6.20)\end{aligned}$$

To go any further we need to specify which type of spline is going to be used, and we will use a clamped spline. This means that $s(x)$ satisfies

$$s'(x_1) = f'(a) \qquad \text{and} \qquad s'(x_{n+1}) = f'(b).$$

Given that

$$\begin{aligned}s'(x_i) &= a_{i-1}B'_{i-1}(x_i) + a_i B'_i(x_i) + a_{i+1}B'_{i+1}(x_i) \\ &= -\frac{1}{2h}a_{i-1} + \frac{1}{2h}a_{i+1},\end{aligned}$$

then $a_0 = a_2 - 2hf'(a)$ and $a_{n+2} = a_n + 2hf'(b)$. Substituting these into (6.20) yields

$$\int_a^b s(x)dx = h\left(\frac{1}{2}(f_1 + f_{n+1}) + \sum_{i=2}^{n} f_i + \frac{h}{12}(f'_1 - f'_{n+1})\right),$$

where $f'_1 = f'(a)$ and $f'_{n+1} = f'(b)$. From Theorem 5.5 we know that $f(x) = s(x) + O(h^4)$, and we therefore have that

$$\int_a^b f(x)dx = I_H + O(h^4), \qquad (6.21)$$

where

$$I_H = h\left(\frac{1}{2}f_1 + f_2 + f_3 + \cdots + f_n + \frac{1}{2}f_{n+1}\right) + \frac{1}{12}h^2\left(f_1' - f_{n+1}'\right). \tag{6.22}$$

This is known as the *composite Hermite rule* or the *corrected trapezoidal rule.* It gets the latter name because

$$I_H = I_T + \frac{1}{12}h^2\left(f_1' - f_{n+1}'\right). \tag{6.23}$$

This is interesting because it shows that the composite trapezoidal rule I_T, given in (6.12), can be adjusted so it has the same order of error as Simpson's rule. As for calling it the Hermite rule, it gets this name because you also obtain (6.22) by integrating something called the cubic Hermite interpolation function. A more extended discussion of this can be found in Holmes [2014].

The following theorem states the error for this method.

Theorem 6.4. *If $f \in C^4[a,b]$, then the composite Hermite rule (6.22) satisfies*

$$\left|\int_a^b f(x)dx - I_H\right| \le \frac{b-a}{720}h^4||f''''||_\infty$$

where $||f''''||_\infty = \max_{a\le x\le b}|f''''(x)|$.

This result is not particularly surprising given the error using a clamped cubic spline, as given in Theorem 5.5.

Example

According to Theorem 6.4, how many subintervals are necessary to guarantee an error of 10^{-8} if the composite Hermite rule is used to evaluate

$$\int_0^1 e^{3x}dx? \tag{6.24}$$

Since $||f''''||_\infty = 81e^3$, then we want $\frac{9}{80}h^4e^3 \le 10^{-8}$. This gives us $h \le (80e^{-3}/9)^{1/4} \times 10^{-2}$. Since $h = (b-a)/n = 1/n$, then we require $n \ge (9e^3/80)^{1/4} \times 10^2 \approx 122.6$. So, according to Theorem 6.4, we should use at least 123 subintervals so the error is no more than 10^{-8}. To check on this, the computed values are given in Table 6.4, along with the value of the error

$$E_H = \left|\int_a^b f(x)dx - I_H\right|. \tag{6.25}$$

n	I_H	E_H
10	6.36177422332073	7.14e−05
20	6.36184117028484	4.47e−06
40	6.36184536152670	2.80e−07
80	6.36184562358981	1.75e−08
160	6.36184563997048	1.09e−09

Table 6.4 Values of (6.24) when computed using the composite Hermite rule I_H, as well as the error E_H.

What is seen is that the desired error of 10^{-8} is obtained using a somewhat smaller number of subintervals than the predicted value of 26. ■

It is interesting that Hermite has a slightly better error than Simpson, although it requires a bit more information about the function. Also, because of (6.23), it shows that the trapezoidal rule can be more accurate than Simpson for functions which satisfy $f'(a) = f'(b)$. It needs to be stated however that most functions do not satisfy this condition, and for this reason Simpson is expected to produce a more accurate approximation than the trapezoidal method. The most noteworthy exception to this arises with periodic functions, where having $f'(a) = f'(b)$ is not uncommon.

6.3.4 Other Interpolation Ideas

As pointed out earlier, (6.22) is known as the corrected trapezoidal rule. It turns out that there is also a corrected midpoint rule, and it states that

$$\int_a^b f(x)dx = I_M - \frac{1}{24}h^2\left(f_1' - f_{n+1}'\right) + O(h^4).$$

The error term in this case is $(b-a)Kh^4||f''''||_\infty$, where $K = 7/5760$. This makes it competitive with both Simpson's rule and the corrected trapezoidal rule. More information about this idea of correcting, or improving, an integration rule using information at the endpoints can be found in Nenad and Roberts [2008].

One might think that the next step is to use higher order polynomial approximations and see what sort of integration rule results. Although there are people with a lot of time on their hands that do this sort of thing, it is not worth the effort. Instead, it is worth comparing the formulas for the composite trapezoidal and Simpson rules. If you add up the f_i's with the coefficients in (6.12), you get an approximation for the integral, but if you add up the f_i's with the coefficients is (6.17) you get a better approximation.

This raises the question as to whether there are coefficients that can be used that will produce an even better approximation. This idea is pursued in the next section.

6.4 Methods Based on Precision

The next family of methods we are going to consider are based on two observations that come from the rules we derived using interpolation. The first is the observation that most of the integration rules have the form

$$\int_{x_i}^{x_{i+1}} f(x)dx \approx w_1 f(z_1) + w_2 f(z_2) + \cdots + w_\ell f(z_\ell), \tag{6.26}$$

where the z_j's are from the interval $x_i \le x \le x_{i+1}$. The z_j's are called the *nodes*, and the w_j's are the *weights*. So, the midpoint rule in (6.6) has one node $z_1 = x_i + \frac{h}{2}$, with corresponding weight $w_1 = h$. Similarly, the trapezoidal rule given in (6.11) has two nodes, $z_1 = x_i$ and $z_2 = x_{i+1}$, with weights $w_1 = w_2 = h/2$. It should also be noted that (6.26) does not include derivative terms that arose with the Hermite rule, and how these can be incorporated into the procedure is explored in Exercise 6.23.

The second observation concerns the error. To explain, in deriving the midpoint rule it is found that

$$\int_{x_i}^{x_{i+1}} f(x)dx = f(x_i + \frac{h}{2})h + \frac{1}{24}h^3 f''(\eta), \tag{6.27}$$

where η is located somewhere in the interval (x_i, x_{i+1}). The proof of this statement is similar to the derivation used to obtain (6.5). Because of (6.27) the midpoint rule has zero error (i.e., it is exact) if $f(x) = 1$ or $f(x) = x$ and the reason is that for both of these functions $f''(x) = 0$. However, it is not necessarily exact when $f(x) = x^2$ and because of this the midpoint rule is said to have precision one. Similarly, as stated in Theorem 6.3, the error of the composite Simpson's rule is bounded by $\frac{b-a}{90}h^4||f''''||_\infty$. Consequently, it has zero error if $f(x) = 1$, $f(x) = x$, $f(x) = x^2$, or $f(x) = x^3$, but it is not necessarily zero if $f(x) = x^4$. For this reason it has precision three.

Before continuing to use this word it is best to define what it means.

Definition 6.1. The *precision* of an integration rule is the largest value of m for which the rule is exact for the functions $f(x) = x^k$, for $k = 0, 1, 2, \cdots, m$.

This definition is used to derive integration rules, and to do this, it is necessary to know the exact value of integrals of the form

$$\int_{x_i}^{x_{i+1}} x^k dx. \tag{6.28}$$

The values, up to $k = 5$, are given in Table 6.5.

k	$f(x)$	$\int_{x_i}^{x_{i+1}} f(x)dx$
0	1	h
1	x	$h\left(x_i + \frac{1}{2}h\right)$
2	x^2	$h\left(x_i^2 + hx_i + \frac{1}{3}h^2\right)$
3	x^3	$h\left(x_i^3 + \frac{3}{2}hx_i^2 + h^2x_i + \frac{1}{4}h^3\right)$
4	x^4	$h\left(x_i^4 + 2hx_i^3 + 2h^2x_i^2 + h^3x_i + \frac{1}{5}h^4\right)$
5	x^5	$h\left(x_i^5 + \frac{5}{2}x_i^4h + \frac{10}{3}x_i^3h^2 + \frac{5}{2}x_i^2h^3 + x_ih^4 + \frac{1}{6}h^5\right)$

Table 6.5 Values of (6.28) for various values of k.

We are now going to derive integration rules that have maximum precision, and these fall under the general category of Gaussian rules.

6.4.1 1-Point Gaussian Rule

We start with a question: what integration rule gives the maximum precision using one point from the subinterval? The assumption is that the general form of the rule is

$$\int_{x_i}^{x_{i+1}} f(x)dx \approx w_1 f(z_1),$$

where $x_i \leq z_1 \leq x_{i+1}$. To find the value of w_1 and z_1 that maximize the precision, we want to find the largest value of k so that

$$\int_{x_i}^{x_{i+1}} x^k dx = w_1 z_1^k, \tag{6.29}$$

and this generates the following list.

1. $k = 0$: From (6.29) we require that $\int_{x_i}^{x_{i+1}} dx = w_1$, and from this it follows that
$$h = w_1.$$

2. $k = 1$: Putting $f = x$ in (6.29), and using Table 6.5, it is required that
$$h\left(x_i + \frac{1}{2}h\right) = w_1 z_1.$$
Given that $w_1 = h$, we obtain $z_1 = x_i + \frac{1}{2}h$.

3. $k = 2$: Using Table 6.5, it is required that
$$h\left(x_i^2 + hx_i + \frac{1}{3}h^2\right) = h(x_i + \frac{1}{2}h)^2. \tag{6.30}$$
It is not possible for this to hold for nonzero h, and so this rule does not integrate this function exactly.

Therefore, the 1-point Gaussian quadrature rule is the midpoint rule. This is worth knowing but it is disappointing because the new idea of maximizing the precision has not produced a new method. As demonstrated next, this changes when we use more points.

6.4.2 2-Point Gaussian Rule

The general form of the rule in this case is
$$\int_{x_i}^{x_{i+1}} f(x)dx \approx w_1 f(z_1) + w_2 f(z_2).$$
Our goal is to find the values of the w_i's and z_i's that maximize the precision. So, we want to find the largest value of k so that
$$\int_{x_i}^{x_{i+1}} x^k dx = w_1 z_1^k + w_2 z_2^k.$$
As was done for the 1-point rule, we make a list using Table 6.5.

1. $k = 0$:
$$h = w_1 + w_2$$
2. $k = 1$:
$$h\left(x_i + \frac{1}{2}h\right) = w_1 z_1 + w_2 z_2.$$
3. $k = 2$:
$$h\left(x_i^2 + hx_i + \frac{1}{3}h^2\right) = w_1 z_1^2 + w_2 z_2^2$$
4. $k = 3$:
$$h\left(x_i^3 + \frac{3}{2}hx_i^2 + h^2 x_i + \frac{1}{4}h^3\right) = w_1 z_1^3 + w_2 z_2^3$$
5. $k = 4$:
$$h\left(x_i^5 + \frac{5}{2}x_i^4 h + \frac{10}{3}x_i^3 h^2 + \frac{5}{2}x_i^2 h^3 + x_i h^4 + \frac{1}{6}h^5\right) = w_1 z_1^4 + w_2 z_2^4 \tag{6.31}$$

We have four unknowns so we will consider the first four equations from the above list. The fact that three of them are nonlinear makes the problem of solving them a bit challenging. One approach to help simplify things is to anticipate where the points z_1 and z_2 are located relative to each other in the interval. For the midpoint rule, z_1 ended up being symmetrically located in the interval (i.e., at the midpoint). Given that z_1 and z_2 appear in a symmetric manner in the integration rule, it is not unreasonable to expect them to also end up being placed symmetrically in the interval. In other words, they have the form $z_1 = \frac{1}{2}(x_{i+1} + x_i) + q$ and $z_2 = \frac{1}{2}(x_{i+1} + x_i) - q$, where we need to find the value of q. Assuming this, then from the $f(x) = 1$ and $f(x) = x$ conditions one finds that $w_1 = w_2 = \frac{1}{2}h$. With this and the $f(x) = x^2$ condition one finds that $q^2 = \frac{1}{12}h^2$. The remaining tasks are to show that this solution satisfies the $f(x) = x^3$ condition but not the one for $f(x) = x^4$. Both of these are left as exercises.

The resulting integration rule is

$$\int_{x_i}^{x_{i+1}} f(x)dx \approx \frac{1}{2}h\big[f(z_i^+) + f(z_i^-)\big]\,, \tag{6.32}$$

where

$$z_i^{\pm} = x_i + \frac{1}{2}h \pm \frac{1}{2\sqrt{3}}h. \tag{6.33}$$

This is the *2-point Gaussian quadrature rule*, and it has precision 3. As will be explained in the next section, a Gaussian rule with precision m has an error that is $O(h^{m+2})$. Consequently, the error for the above rule has an error that is $O(h^5)$, which is the same order as obtained for Simpson's rule. Also note that the Gaussian points z_i^- and z_i^+ are symmetrically placed in the subinterval but they do not split the interval into thirds (see Figure 6.9).

6.4.3 Error Formulas

Now that we have derived a new method for numerical integration we turn our attention to the error. We begin with the theorem which provides the needed formula.

Theorem 6.5. *If $f \in C^{2\ell}[x_i, x_{i+1}]$, then the error E_G using the ℓ-point rule in (6.26) is*

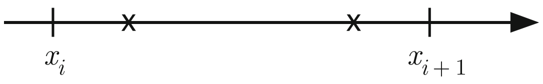


Figure 6.9 Location of the Gaussian points z_i^- and z_i^+ in the subinterval.

$$E_G = Kh^{2\ell+1} f^{(2\ell)}(\eta),$$

where

$$K = \frac{(\ell!)^4}{(2\ell+1)[(2\ell)!]^3}, \tag{6.34}$$

and $x_i \le \eta \le x_{i+1}$.

The proof of this theorem can be found in Isaacson and Keller [1994]. The significance of this result is that it shows that a Gaussian rule with precision m has an error that is $O(h^{m+2})$. We will make use of this in a couple of ways. The first will follow what we did with the other rules, where we combine the subintervals into a composite formula. For the second way, subintervals will not be used, and the formula will be applied to the entire interval.

Applying the above theorem to the 2-point rule (6.32),

$$\int_{x_i}^{x_{i+1}} f(x)dx = \frac{1}{2}h\big[f(z_i^+) + f(z_i^-)\big] + Kh^5 f''''(\eta).$$

where $K = 1/4320$. The resulting composite rule is

$$\int_a^b f(x)dx = I_{2G} + O(h^4),$$

where

$$I_{2G} = \frac{h}{2}\left(f_1^- + f_1^+ + f_2^- + f_2^+ + \cdots + f_n^- + f_n^+\right). \tag{6.35}$$

In the above expression, $f_i^\pm = f(z_i^\pm)$ denotes the values of $f(x)$ at the two Gaussian points within the ith subinterval as given in (6.33). The corresponding error using this composite rule is given in the following theorem.

Theorem 6.6. *If $f \in C^4[a,b]$, then the composite 2-point rule satisfies*

$$\left| \int_a^b f(x)dx - I_{2G} \right| \le \frac{b-a}{4320} h^4 ||f''''||_\infty,$$

where $||f''''||_\infty = \max_{a\le x\le b} |f''''(x)|$.

The denominator in the above expression is much larger than what we have obtained with the other composite rules. However, this is not as impressive as might be assumed at first glance. When using n subintervals, Simpson's rule uses $n+1$ function evaluations while the above Gaussian rule uses $2n$. To make a fair comparison one should use approximately the same number of function evaluations. So, for the Gaussian rule one should use approximately $n/2$ subintervals, which means the grid spacing should be $2h$. If this

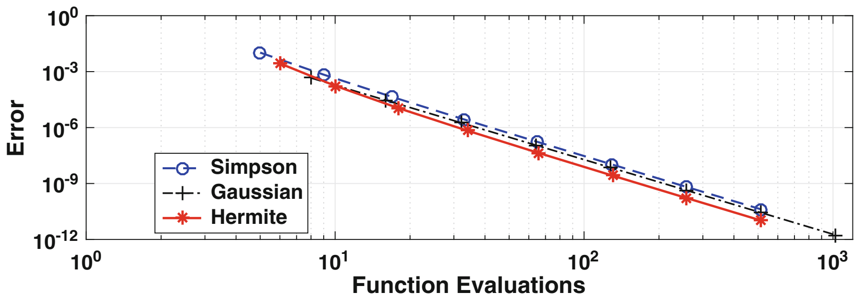


Figure 6.10 Error in using the composite Simpson's rule, the Hermite rule, and the composite 2-point Gaussian rule to evaluate (6.36).

is done, then the multiplicative factor changes from $\frac{1}{4320}$ to $\frac{1}{270}$. Given that the corresponding factor for Simpson is $\frac{1}{90}$ then the 2-point Gaussian rule does produce a better error but the improvement is approximately $\frac{1}{3}$ and not a factor of $\frac{1}{48}$ as might be inferred from Theorem 6.6.

Examples

1. Suppose the composite 2-point rule is used to approximate the value of

$$\int_0^1 e^{3x}dx. \tag{6.36}$$

Using two subintervals, then the Gaussian points in the first interval are $z_1^{\pm} = \frac{1}{4}(1 \pm 1/\sqrt{3})$, and for the second interval $z_2^{\pm} = \frac{1}{4}(3 \pm 1/\sqrt{3})$. In this case, (6.35) becomes

$$\begin{aligned} I_{2G} &= \frac{1}{4}\Big(e^{3z_1^-} + e^{3z_1^+} + e^{3z_2^-} + e^{3z_2^+}\Big) \\ &\approx 6.3549. \quad \blacksquare \end{aligned}$$

2. The error using the composite versions of the Simpson, Hermite, and 2-point Gaussian rules are shown Figure 6.10 for the integral (6.36). The error is shown as a function of the number of function evaluations needed to calculate the integration rule. Assuming there are n subintervals, and n is even, then the number of function evaluations for Simpson is $n+1$, for Hermite it is $n+3$, and for 2-point Gaussian it is $2n$. The curves all show the expected $O(h^4)$ rate of convergence. Also, as expected the Hermite rule produces the best approximation although the differences between these three methods are relatively minor. $\blacksquare$

6.4.4 General Case

The general version of the ℓ-point Gaussian quadrature rule given in (6.26) results in an integration rule with precision $2\ell-1$. This is interesting because it allows for the possibility of reducing the number of subintervals, but increasing the number of Gaussian points in each subinterval to achieve the required accuracy. This can be very effective, and to illustrate, consider the integral in (6.36). Not using any subintervals, then the Gaussian points are positioned over the interval of integration $a \le x \le b$. To demonstrate just how effective this can be, the error involved with computing (6.36) is given in Table 6.6 as a function of the number of Gaussian points used. For comparison, the value obtained using the composite Simpson's rule is given, using the corresponding number of interpolation points. The improvement of the Gaussian value is impressive, easily besting the results obtained using the composite Simpson's rule. Moreover, given the results in Figure 6.10, it is also significantly better than Hermite or the composite 2-point Gaussian rule.

An explanation of why it does so well can be found in the error formula given in Theorem 6.5. Before stating the result, note that we are not using subintervals, and so the integration rule is

$$\int_a^b f(x)dx \approx w_1 f(z_1) + w_2 f(z_2) + \cdots + w_\ell f(z_\ell), \tag{6.37}$$

with error

$$E_G = K(b-a)^{m+2} f^{(m+1)}(\eta), \tag{6.38}$$

and precision $m = 2\ell - 1$. The rather unwieldy expression given in Theorem 6.5 for the factor K can be simplified using the Stirling approximation (5.34). After doing this, it is found that

$$|E_G| \le \frac{\alpha}{\sqrt{\ell}} R^{2\ell} ||f^{(2\ell)}||_\infty, \tag{6.39}$$

where $||f^{(2\ell)}||_\infty = \max_{a\le x\le b} |f^{(2\ell)}(x)|$, $\alpha = (b-a)\sqrt{\pi}/4$, and

$$R = \frac{(b-a)e}{8\ell}\,. \tag{6.40}$$

Consequently, if ℓ is large enough that $R < 1$, then $R^{2\ell}$ approaches zero exponentially fast as ℓ increases. Whether this means that the error using Gaussian quadrature approaches zero exponentially fast depends on the contribution of the $f^{(2\ell)}$ term in the error formula. This is the same situation we encountered with Chebyshev interpolation in Section 5.5.4. In fact, the examples used in Section 5.5.5 can be used to demonstrate that both the composite Simpson's and Hermite rules are competitive with, if not better than, Gaussian quadrature (on those examples).

ℓ	E_G	E_S
1	1.88	
2	9.18e−02	
3	1.74e−03	1.40e−01
4	1.75e−05	
5	1.09e−07	1.05e−02
6	4.66e−10	
7	1.45e−12	2.14e−03
8	8.88e−16	
9	7.11e−15	6.87e−04

Table 6.6 Error E_G evaluating (6.36) using ℓ-point Gaussian quadrature, and the error E_S using composite Simpson's rule. For the latter, ℓ designates the number of function evaluations.

Something not mentioned in the above discussion is how the nodes and weights in (6.37) are determined. As it turns out, the locations of the nodes correspond to the roots of the ℓth order Legendre polynomial. Assuming that $\ell \geq 2$, and using the recursive properties of these polynomials, finding the nodes and weights can be reduced to an eigenvalue problem involving an $\ell \times \ell$ tridiagonal matrix. This problem is given in Exercise 4.22. Letting λ_j denote the jth eigenvalue, then in (6.37) the corresponding node is given as

$$z_j = \frac{b+a}{2} + \frac{b-a}{2}\lambda_j,$$

and the weight is $w_j = \frac{1}{2}(b-a)\overline{w}_j$, where $\overline{w}_j$ is defined in Exercise 4.22. This is the basis for what is known as the *Golub-Welsch algorithm*, and more about this can be found in Golub and Welsch [1969] and Davis and Rabinowitz [2007].

The values for the Gaussian quadrature rules are listed in most handbooks or compilations of mathematical formulas (e.g., the values for the 80-point rule are listed in Olver et al. 2010). For those with a more extreme interest in this should also consult Love [1966], which considers up to 200 nodes, or Bogaert [2014], who discusses cases with millions of nodes. For the latter, the nodes near the endpoints of the interval are so close that they are not resolvable using double precision. The need of such a large number of nodes is not clear, and for the moment the result is mostly of theoretical interest.

There are numerous variations of the Gaussian rule given in (6.37). To describe one of particular note, recall that in Section 5.5.4 we found that to obtain the best interpolation polynomial, the x_i's should be chosen to be the Chebyshev points. It would seem that these would be a good choice for the nodes in (6.37). However, there is evidence that it is actually better to use the extrema points of the Chebyshev polynomial from the interval $a \leq x \leq b$ rather than the points where it is zero. The respective weights are determined, as usual, by maximizing the precision. This gives rise to what is known as Clenshaw-Curtis quadrature. Assuming there are ℓ nodes, this method has precision $\ell - 1$, while the Gaussian rule has precision $2\ell - 1$. Even so, there are situations where Clenshaw-Curtis might be preferable to Gaussian quadrature, and this is discussed in Trefethen [2008].

6.5 Romberg Integration

An interesting idea on how to improve the accuracy of numerical integration is based on an observation for the error. To illustrate, the composite midpoint rule (6.7) states that

$$\int_a^b f(x)dx = I_M + O(h^2).$$

This can be written as

$$\int_a^b f(x)dx = I_M(n) + \alpha h^2 + \beta h^3 + \gamma h^4 + \cdots . \tag{6.41}$$

In the above expression, $I_M(n)$ is written to make it explicit how many subintervals are used, and the various terms making up the error are written out. It is also worth recalling that $h = (b-a)/n$. Because of this, if the number of subintervals is increased from n to $2n$, then the width of the subintervals changes from h to $h/2$. Consequently,

$$\int_a^b f(x)dx = I_M(2n) + \frac{1}{4}\alpha h^2 + \frac{1}{8}\beta h^3 + \frac{1}{16}\gamma h^4 + \cdots ,$$

where h is the value of the width when using n subintervals. It is possible to combine the two approximations and in the process eliminate the $O(h^2)$ term. Multiplying the second equation by 4 and subtracting the first equation we obtain

$$3\int_a^b f(x)dx = 4I_M(2n) - I_M(n) - \frac{1}{2}\beta h^3 + \cdots .$$

In other words,

$$\int_a^b f(x)dx = \frac{4}{3}I_M(2n) - \frac{1}{3}I_M(n) + O(h^3). \tag{6.42}$$

This idea of increasing the number of subintervals and then combining the two results to get a better approximation is known as *Romberg integration.*

The formula in (6.42) is due entirely to the form of the error in (6.41), and it can be applied to any composite rule with this type of error. For example, the composite trapezoidal rule has this form, and so the Romberg formula for it is

$$\int_a^b f(x)dx = \frac{4}{3}I_T(2n) - \frac{1}{3}I_T(n) + O(h^3). \tag{6.43}$$

It can also be easily extended to composite rules with other forms for the error. An example is Simpson's rule, and the Romberg procedure applied to it produces

$$\int_a^b f(x)dx = \frac{15}{16}I_S(2n) - \frac{1}{15}I_S(n) + O(h^6). \tag{6.44}$$

The derivation of this result, as well as other Romberg formulas, can be found in Exercise 6.20.

6.5.1 Computing Using Romberg

The question arises when computing an integral of how many subdivisions to use. For most of the methods considered in this chapter, theorems were given that provided bounds on the error, and these were used to predict how many subintervals are need to guarantee a certain error. In real world situations, these often are not useful because calculating $||f''||_\infty$ or $||f''''||_\infty$ is either not possible or too difficult to be practical. For this reason, integration methods are used in a similar manner as Newton's method or the secant method. Namely, you use the method to generate a sequence of approximate

```
Pick:  $n$
       $tol > 0$
       $A(1) = I(n)$
       $R(1) = A(1)$
Loop:  For $k = 2, 3, 4, \cdots$
           $n = 2n$
           $A(k) = I(n)$
           $R(k) = (16A(k) - A(k-1))/15$
           if $|R(k) - R(k-1)| < tol$, then stop
       end
```

Table 6.7 Algorithm for evaluating an integral numerically using Romberg integration applied to a composite integration rule that has error $O(h^4)$.

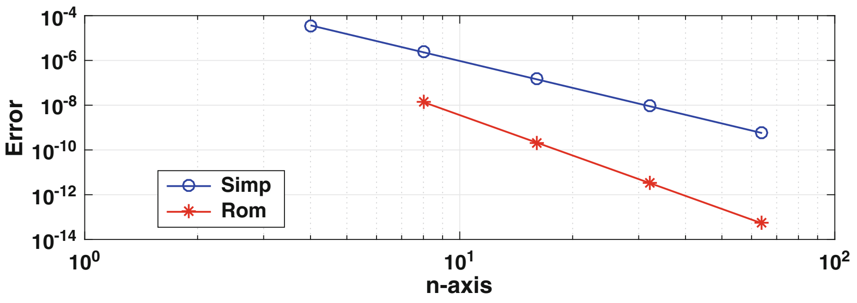


Figure 6.11 The error when evaluating (6.45). Shown is the error E_S using the composite Simpson's rule, and the error E_R obtained using Romberg integration (the latter are from Table 6.8).

values for the integral. Unlike Newton's method where you are not sure it will converge, the theorems for the numerical integration methods guarantee they will work as long as the function you are computing with is smooth enough.

The basic procedure is to keep doubling the number of subintervals, using one of the Romberg rules to compute the integral, and then stopping when the desired accuracy is achieved. Such an algorithm using Simpson's rule is given in Table 6.7. The formula for $R(k)$ in this case comes from (6.44).

Example

To demonstrate the usefulness of Romberg integration we consider evaluating the integral

$$\int_0^1 e^{3x}dx. \tag{6.45}$$

k	n	I_S	I_R	E_R
1	4	1.71831884192175		
2	8	1.71828415469990	1.71828184221844	1.376e−08
3	16	1.71828197405189	1.71828182867536	2.163e−10
4	32	1.71828183756177	1.71828182846243	3.385e−12
5	64	1.71828182902802	1.71828182845910	5.240e−14

Table 6.8 Values of (6.45) when using Romberg integration with Simpson's rule as given in the algorithm in Table 6.7. Also given is the error $E_R = |I_R(n) - I|$, where I_R is the value using Romberg integration and I is the exact value. The stopping error used is $tol = 10^{-10}$.

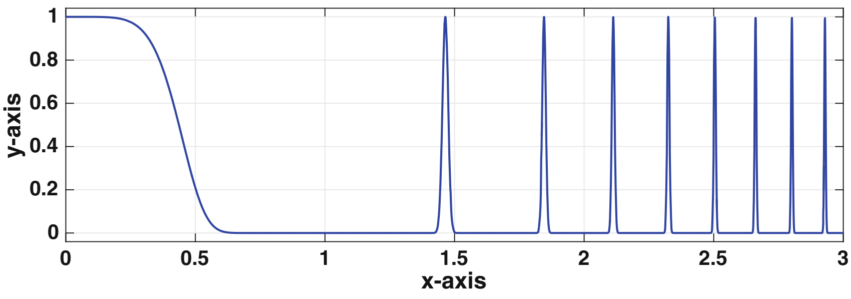


Figure 6.12 Example of a function that shows significant variation over the interval of integration.

Using the procedure given in Table 6.7, the values given in Table 6.8 are obtained. To make it more evident that the error for Romberg has the predicted $O(h^6)$ decrease, the errors are plotted in Figure 6.11. ■

6.6 Adaptive Quadrature

Consider the problem of integrating the function $f(x) = \cos(x^3)^{200}$ over the interval $0 \leq x \leq 3$, which is shown in Figure 6.12. If Simpson's rule is used, and an error of less than 10^{-6} is desired, then according to Theorem 6.3 it is necessary to use about 19,200 equally spaced subintervals. This large number is necessary to accurately calculate the area under the sharp peaks near $x = 3$. However, such small subintervals are not necessary to accurately compute the area over the interval $0 \leq x \leq 1$. The way to deal with this is simple, namely you just use smaller subintervals in the regions with peaks, and larger subintervals in the flatter regions. What you do not want to do is to manually decide how to break up the interval, but instead design a procedure that progressively refines the subdivisions in the peak regions until the required accuracy is attained. This is the essence of what is called *adaptive quadrature*.

To explain the basic idea underlying adaptive quadrature, consider evaluating the integral

$$\int_0^4 f(x)dx, \tag{6.46}$$

where $f(x)$ is the shifted Lorentzian function

$$f(x) = 1 + \frac{1}{\pi}\frac{\omega}{(x-x_0)^2+\omega^2} ,$$

where $x_0 = 3$ and $\omega = 1/(2\pi)$. The graph of $f(x)$ is shown in Figure 6.13. The version of adaptive quadrature described here uses Simpson's rule. According to Theorem 6.3 it is necessary to use about 960 equally spaced subintervals to have an error of no more than 10^{-6}, while if the Simpson-Romberg integration rule is used then the number drops to about 200 subintervals. To derive an adaptive method it is necessary to know how the error is affected as the number of subintervals increases, something similar to what was used to derive for Romberg integration. The needed formulas are derived in the next paragraph for the general case using Simpson's rule, and after that the method will be applied to the above example.

To determine the improvement as the number of subintervals is increased, we first rewrite (6.16) as

$$\int_{x_{i-1}}^{x_{i+1}} f(x)dx = S_2(x_{i-1}, x_{i+1}) + E_2, \tag{6.47}$$

where

$$S_2(x_{i-1}, x_{i+1}) = \frac{h}{3}(f_{i-1} + 4f_i + f_{i+1}),$$

and

$$E_2 = -\frac{1}{90}h^5 f_i'''' + \cdots .$$

The subscript 2 is used to indicate how many subintervals are used. If we use four subintervals, instead of two, then

$$\begin{aligned}\int_{x_{i-1}}^{x_{i+1}} f(x)dx &= \int_{x_{i-1}}^{x_i} f(x)dx + \int_{x_i}^{x_{i+1}} f(x)dx \\ &= S_2(x_{i-1}, x_i) - \frac{1}{90}(h/2)^5 f''''(x_i - h/2) + \cdots \\ &\quad + S_2(x_i, x_{i+1}) - \frac{1}{90}(h/2)^5 f''''(x_i + h/2) + \cdots .\end{aligned}$$

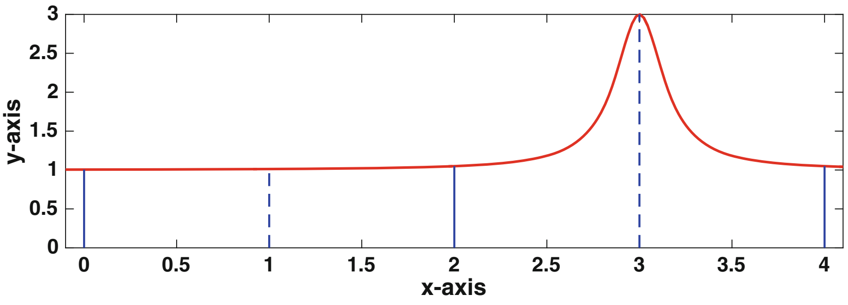


Figure 6.13 Function used in the integral (6.46), and the initial subdivisions used for Simpson's rule.

From Taylor's theorem, $f''''(x_i \pm h/2) = f''''(x_i) \pm (h/2)f^{(5)}(x_i) + \cdots$. With this, we have that

$$\int_{x_{i-1}}^{x_{i+1}} f(x)dx = S_4(x_{i-1}, x_{i+1}) + E_4, \tag{6.48}$$

where $S_4(x_{i-1}, x_{i+1}) = S_2(x_{i-1}, x_i) + S_2(x_i, x_{i+1})$ and $E_4 = \frac{1}{16}E_2 + \cdots$. Since (6.47) and (6.48) are supposed to produce the same result, we equate the right-hand sides and conclude that

$$E_4 \approx \frac{1}{15}\left[S_4(x_{i-1}, x_{i+1}) - S_2(x_{i-1}, x_{i+1})\right]. \tag{6.49}$$

What this result states is that the error in approximating the integral with $S_4(x_{i-1}, x_{i+1})$ can be estimated using the computed values for $S_4(x_{i-1}, x_{i+1})$ and $S_2(x_{i-1}, x_{i+1})$. The remaining question is, how small do we want $|E_4|$? Assuming the requirement is that the error in the computed value for the entire integral

$$\int_a^b f(x)dx$$

is no more than *tol*, then we will require

$$|E_4| \le \frac{x_{i+1} - x_{i-1}}{b - a}\, tol\,. \tag{6.50}$$

The coefficient of *tol* in the above expression is so the errors from the subintervals add up to *tol*.

We return to the original problem of evaluating (6.46). The method will produce levels of subdivision, and it is worth labeling them as the refinement proceeds. It is also assumed that *tol* is a prescribed error tolerance.

`Level 1:` Taking two subintervals, so $h = (b - a)/2 = 2$ (which are the solid vertical blue lines in Figure 6.13), the first approximation is

$$\begin{aligned} S_2(0,4) &= \frac{2}{3}[f(0) + 4f(2) + f(4)] \\ &\approx 4.1684\,. \end{aligned}$$

Doubling the number of subintervals, then $h = 1$ and the subintervals now include the dashed blue lines in Figure 6.13. In this case,

$$\begin{aligned} S_4(0,4) &= \frac{1}{3}[f(0) + 4f(1) + f(2)] + \frac{1}{3}[f(2) + 4f(3) + f(4)] \\ &\approx 6.7347\,. \end{aligned}$$

Using (6.49), this means that the error in approximating the integral using S_4 is

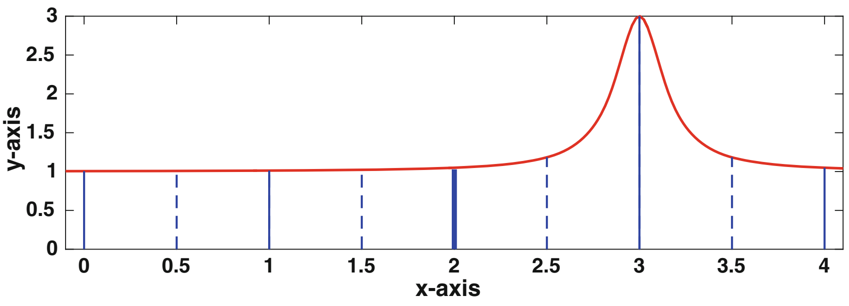


Figure 6.14 Function used in the integral (6.46), and the subdivisions used in Level 2 of the adaptive procedure.

$$E_4 \approx \frac{1}{15}\left(S_4 - S_2\right) \approx 0.1711\,.$$

If this satisfies (6.50), which means that $|E_4| \leq tol$, then you are finished and the computed value is S_4. If not, then more subdivisions are needed and you proceed to Level 2.

`Level 2:` If the error condition is not satisfied, then the method splits the interval in half and writes

$$\int_0^4 f(x)dx = \int_0^2 f(x)dx + \int_2^4 f(x)dx, \tag{6.51}$$

and then applies the approximations used in Level 1 to each integral. The resulting subintervals are shown in Figure 6.14. For example, for $\int_0^2 f(x)dx$ it calculates

$$\begin{aligned} S_2(0,2) &= \frac{1}{3}[f(0) + 4f(1) + f(2)] \\ &\approx 2.0351\,, \end{aligned}$$

and

$$\begin{aligned} S_4(0,2) &= \frac{1}{6}[f(0) + 4f(0.5) + f(1)] + \frac{1}{6}[f(1) + 4f(1.5) + f(2)] \\ &\approx 2.0336\,. \end{aligned}$$

The error in this case is

$$E_4 = \frac{1}{15}\left(S_4 - S_2\right) \approx -10^{-4}\,.$$

If this satisfies (6.50), which means that $|E_4| \leq tol/2$, then computing the integral over the interval $0 \leq x \leq 2$ is done. If not, then this integral is passed to Level 3. A similar calculation is done for $\int_2^4 f(x)dx$.

`Level 3, 4, ···:` The method keeps subdividing the subintervals that are transferred from the previous level, until it finally obtains a value for each that satisfies the error condition.

In the above description, the value of S_4 is used as the computed value of the integral over the subinterval. Given the way the number of subintervals are doubled in this procedure, it is possible to compute the final value using Romberg integration. So, instead of S_4, a more accurate value for the subinterval is obtained by using

$$R_4 = \frac{1}{15}(16S_4 - S_2).$$

Letting the procedure run, the final subdivision is shown in Figure 6.15. The error condition used to decide when to stop subdividing is that the error for the integral is no more than $tol = 10^{-6}$. This resulted in the need for 8 levels, and a total of 89 evaluations of the function $f(x)$. In comparison, it takes approximately 960 equally spaced subdivisions to achieve the same error, and this is also the approximate number of function evaluations needed.

As another example, for the function shown in Figure 6.13, 19,200 equally spaced subintervals are needed to achieve an error of 10^{-6}, and this is also the approximate number of function evaluations required. In contrast, using the adaptive Simpson method, this error is achieved using 849 function evaluations (and 14 levels).

There are numerous variations on the idea of adaptive integration. For example, instead of using Simpson's rule, the 2-point Gaussian rule is a possibility and it has an error comparable with Simpson. The drawback is that when subdividing the intervals, the nodes $z_i^{\pm}$ for the larger interval are not shared with the smaller subintervals. This means that a completely new set of function evaluations are required for each level, with a corresponding increase in computing time. There are some clever ways to avoid this difficulty, and

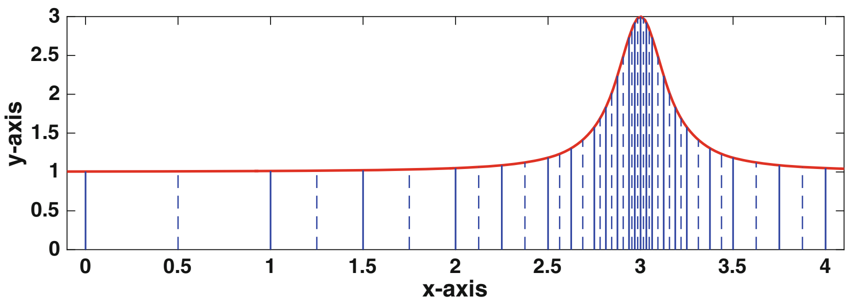


Figure 6.15 Function used in the integral (6.46), and the final subdivisions used for adaptive Simpson's rule.

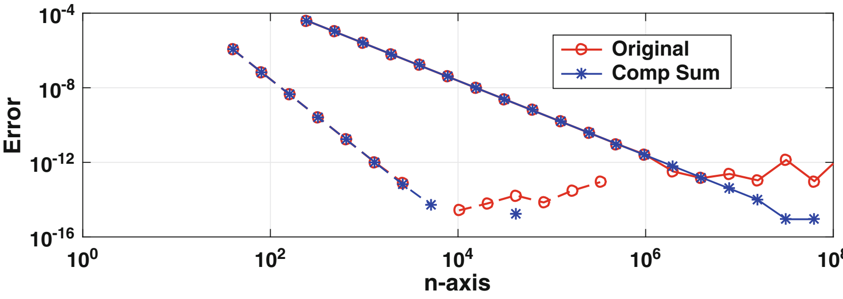


Figure 6.16 Upper right curves: Composite midpoint rule with and without compensated summation. Lower left curves: Composite Simpson's rule with and without compensated summation.

one of the more well-known gives rise to what are called Gauss-Kronrod rules. Recent reviews, or discussions, of this can be found in Gander and Gautschi [2000] and Gonnet [2012].

6.7 Other Ideas

In Section 1.4 the idea of compensated summation was introduced and its benefits are illustrated in Table 1.4. Given the potentially large number of additions that can arise with numerical integration, the question arises as to whether compensated summation is worth using. To investigate this, we return to evaluating the integral

$$\int_0^1 e^{3x} dx.$$

The error in computing this integral using the composite midpoint and Simpson's rules, with and without compensated summation, is shown in Figure 6.16. What it shows is that for Simpson's rule compensated summation does help once the error gets close to machine epsilon. In fact, the missing data values when using compensated summation, like the value when $n = 10^4$, means the error is zero. In comparison, because the midpoint rule requires a significantly larger number of subintervals, compensated summation starts to make a noticeable improvement at a larger error level.

6.8 Epilogue

This chapter is guilty of overkill in the sense that several methods were derived that basically do the same thing. This is done, partly, because there is no "best" method. It is certainly true that the Gaussian rule has the distinct advantage of having exponential convergence, assuming the function being integrated is smooth enough. The reason for wanting exponential convergence is evident in Table 6.6. There are limitations to using Gaussian quadrature, and one is that it requires access to an efficient eigenvalue problem solver. Also, as explained in Section 6.5.1, integration is often an iterative process, and unlike most of the other methods discussed in this chapter, the Gaussian rule cannot make use, in an obvious way, of the function evaluations made in earlier steps of the iteration. Finally, there is the actual computing time needed to evaluate the integral. Even a function as complicated as the one in (6.1) takes only about 10^{-7} seconds to evaluate. So, evaluating the integral using Simpson-Romberg or adaptive Simpson requires about 10^{-3} seconds, while using Gaussian quadrature requires about 3×10^{-4} seconds. In other words, for such an integral, the differences between the three methods are effectively imperceptible.

Which rule to consider does depend on the application. For example, if the function is experimentally determined, and you only have data points, then the Hermite and Gaussian rules are difficult to use and the default would be Simpson. This assumes of course that the grid points are equally spaced. If not, then one might consider either the midpoint or trapezoidal rules, which are easily adapted to uneven grid points. Another situation worth mentioning is when $f(x)$ is periodic. In such cases the composite midpoint and trapezoidal methods can be very effective, giving rise to exponential convergence. An explanation of why, and what this is, can be found in Weideman [2002], Waldvogel [2011], and Trefethen and Weideman [2014]. A related, but more difficult situation, arises when $f(x)$ is an oscillatory function and it oscillates rapidly. These can arise, for example, when you use a Fourier transform to solve a differential equation. Some of the ideas on how to integrate such functions can be found in Evans and Webster [1999] and Iserles et al. [2006].

Exercises

6.1. This problem concerns using numerical methods to calculate the integral

$$I = \int_1^2 \ln(x)dx.$$

Note that the exact value is, $I = 2\sqrt{2} - 1$.

(a) Using the composite trapezoidal rule, and 4 subintervals, find an approximate value for the integral. What is the error?
(b) Using the composite Simpson's rule, and 4 subintervals, find an approximate value for the integral. What is the error?
(c) Using the composite Hermite rule, and 4 subintervals, find an approximate value for the integral. What is the error?

x	0	2	4	6	8
F	0	1	5	17	37

Table 6.9 Values for Exercise 6.3.

(d) Using the composite trapezoidal rule, how small does the step size h have to be to guarantee that the numerical error is less than 10^{-6}?
(e) Using the composite Simpson's rule, how small does the step size h have to be to guarantee that the numerical error is less than 10^{-6}?
(f) Using the composite Hermite rule, how small does the step size h have to be to guarantee that the numerical error is less than 10^{-6}?

6.2. This problem concerns using numerical methods to calculate the integral

$$I = \int_{-1}^{1} e^{-2x} dx.$$

Note that the exact value is, $I = (e^2 - e^{-2})/2$.
(a) Using the composite trapezoidal rule, and 4 subintervals, find an approximate value for the integral. What is the error?
(b) Using the composite Simpson's rule, and 4 subintervals, find an approximate value for the integral. What is the error?
(c) Using the composite Hermite rule, and 4 subintervals, find an approximate value for the integral. What is the error?
(d) Using the composite trapezoidal rule, how small does the step size h have to be to guarantee that the numerical error is less than 10^{-6}?
(e) Using the composite Simpson's rule, how small does the step size h have to be to guarantee that the numerical error is less than 10^{-6}?
(f) Using the composite Hermite rule, how small does the step size h have to be to guarantee that the numerical error is less than 10^{-6}?

6.3. The measured values of a force $F(x)$ as a function of the displacement x are given in Table 6.9. The object of this exercise is to calculate the work W.

(a) Use the trapezoidal rule to find the value of $W(x)$ at $x = 2, 4, 6, 8$. You can assume that $W(0) = 0$.
(b) Use the composite midpoint rule to calculate $W(8)$.
(c) Use the composite Simpson rule to calculate $W(8)$.
(d) Use Romberg integration with the composite trapezoidal rule to calculate $W(8)$.

6.4. For a linearly elastic material, the stress $T(x)$ is given as

$$T = E\frac{du}{dx},$$

where $u(x)$ is the displacement of the material and E is a positive constant known as the Young's modulus. The question considered here is, how to determine u from measurements of T, as given in Table 6.10.

(a) Show that

$$u(x) = u(0) + \frac{1}{E}\int_0^x T(s)ds.$$

In the following you can assume that $u(0) = 0$ and $E = 4$.

(b) Use the trapezoidal rule to find the value of $u(x)$ at $x = 1/4, 1/2, 3/4, 1$.

(c) Use the composite midpoint rule to calculate $u(1)$.

(d) Use the composite Simpson rule to calculate $u(1)$.

(e) Use Romberg integration with the composite trapezoidal rule to calculate $u(1)$.

6.5. True or False. In the following a claim is made about the value of the integral. Using one of the integration rules considered in this chapter provide a compelling reason why you believe the value is correct or why it is an error. Make sure to explain which integration method you used, including the error condition(s).

(a)
$$\int_0^{\pi/2} \frac{\sin^2 x}{\sin x + \cos x} dx = \frac{1}{2\sqrt{2}}\ln(3 + 2\sqrt{2})$$

(b)
$$\int_0^{\pi/2} \frac{1}{(\cos^2 x + 3\sin^2 x)^3} dx = \frac{\pi}{12}\sqrt{3}$$

(c)
$$\int_0^1 \frac{\tan^{-1}(\sqrt{2+x^2})}{(1+x^2)\sqrt{2+x^2}} dx = \frac{5\pi^5}{96}$$

(d)
$$\int_0^{\pi/2} \cos^{-1}\left(\frac{\cos x}{1 + 2\cos x}\right) dx = \frac{5\pi^2}{24}$$

(e)
$$\int_0^1 x^{x^3} dx = 1 - \frac{1}{4^2} + \frac{1}{7^3} - \frac{1}{10^4} + \frac{1}{13^5} - \cdots$$

6.6. The second, $f''(x)$, and fourth, $f''''(x)$, derivatives of a function $f(x)$ are plotted in Figure 6.17. This question concerns evaluating $\int_0^3 f(x)dx$.

x	0	1/4	1/2	3/4	1
T	1	−1	2	3	4

Table 6.10 Values for Exercise 6.4.

(a) How large must n be to guarantee the error using the composite trapezoidal rule is less than 10^{-8}?
(b) How large must n be to guarantee the error using the composite Simpson rule is less than 10^{-8}?

6.7. This problem considers how well the integration methods do when applied to a periodic function. The particular integral is

$$\int_0^1 \sin(2\pi x)dx.$$

(a) What is the exact value?
(b) Determine (by hand) the value of I_T when $n = 3, 4, 5, 6$. For each n, sketch $\sin(2\pi x)$, and the corresponding piecewise linear approximation used in the trapezoidal method, and use this to explain why the method works as well as it does.
(c) Determine (by hand) the value of I_M when $n = 3, 4, 5, 6$. For each n, sketch $\sin(2\pi x)$, and the corresponding piecewise constant approximation used in the midpoint method, and use this to explain why the method works as well as it does.

6.8. The error function is defined as

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-s^2} ds.$$

(a) Suppose the composite trapezoidal rule is used to evaluate erf(2). According to Theorem 6.2, what value for h is needed so that the error is no more than 10^{-6}.
(b) Using MATLAB and the composite trapezoidal rule, find a value of h so the computed error is approximately 10^{-6}, and explain the difference between this value and the one found in part (a). Also, note that $\text{erf}(2) = 0.995322265\cdots$

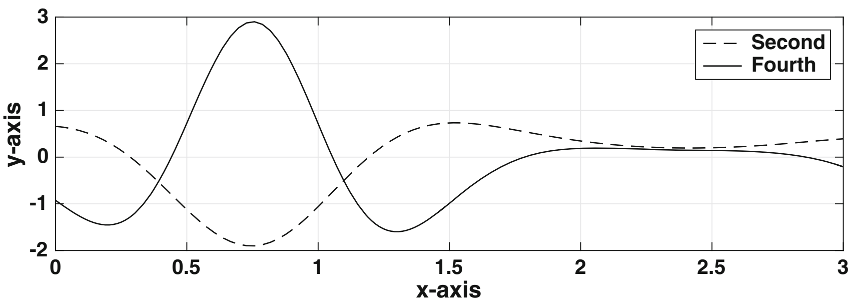


Figure 6.17 Graph used in Exercise 6.6.

n	Method 1	Method 2	Method 3	Method 4
5	1.002780193	0.650243328	1.016909995	1.168601449
10	1.000180902	0.909422388	1.001089055	1.044881435
20	1.000011420	0.977151912	1.000068578	1.011398343
40	1.000000716	0.994275128	1.000004294	1.002860826
80	1.000000045	0.998567977	1.000000269	1.000715911
160	1.000000003	0.999641944	1.000000017	1.000179022

Table 6.11 Values for Exercise 6.10.

(c) Redo part (a) but use the composite Simpson rule. Note that it will help if you plot $f''''(s)$.
(d) Redo part (b) but use the composite Simpson rule (and compare with the answer from part (c)).

6.9. Letting $P(t)$ be the population of a country at time t (measured in years), consider the integral

$$\int_{1930}^{2010} P(t)dt.$$

It is assumed that $P(t)$ is known at 1930, 1940, $\cdots$, 2010 (as an example, the values for the USA are given in Table 5.7).
(a) Use the composite midpoint rule to evaluate the integral.
(b) Use the composite trapezoidal rule to evaluate the integral.
(c) Use the composite Simpson rule to evaluate the integral.
(d) Use Romberg integration with the composite midpoint rule to evaluate the integral.

6.10. For a given function $f(x)$, four of the composite integration rules listed in Table C.2, in Appendix C, were used to compute

$$\int_0^1 f(x)dx.$$

The computed values are given in Table 6.11 as a function of the number n of subintervals used. Determine the name of each of the four methods, making sure to explain your reasoning.

6.11. According to Planck's law of blackbody radiation, the spectral energy density is

$$E_d(\lambda) = \frac{8\pi hc}{\lambda^5(e^{\alpha/\lambda} - 1)},$$

where λ is the wavelength and $\alpha = hc/(k_B T)$. Also, T is absolute temperature, k_B is the Boltzmann constant, h is Planck's constant, and c is the speed of light. The energy emitted in the wavelength band $\lambda_1 \leq \lambda \leq \lambda_2$ is

$$E(\lambda_1, \lambda_2) = \int_{\lambda_1}^{\lambda_2} E_d(\lambda) d\lambda.$$

In this problem assume $T = 7000\,\mathrm{K}$, so that $\alpha = 2\,\mu\mathrm{m}$. Note that in this case the wavelength λ is measured in μm.

(a) Plot $E_d(\lambda)$ for $0.01 \leq \lambda \leq 4$.

(b) Using the composite Simpson's rule, calculate $E(0.01, 0.4)$. Your answer should be correct to at least six significant digits.

6.12. Find the area of the region enclosed by the curve $x^4 + 2y^4 = 1$. Make sure to state which integration method you used, what integral it was used to evaluate, and how you selected your error tolerance.

6.13. Find the arc length of the curve $(x, y) = (t^5, t^3)$, for $0 \leq t \leq 4$. Make sure to state which integration method you used, what integral it was used to evaluate, and how you selected your error tolerance.

6.14. The Mooney-Rivlin law, often used for elastic polymers, states that the stress $T(x)$ in a material is given as

$$T = \left(\alpha + \frac{\beta}{\lambda}\right)\left(\lambda^2 - \frac{1}{\lambda}\right),$$

where λ, known as the stretch, is given as

$$\lambda = 1 + \frac{du}{dx}.$$

The function $u(x)$ is the displacement of the material, and α and β are positive constants. The objective of this exercise is, for a material which occupies the interval $0 \leq x \leq \ell$, to determine u from measured (known) values of T. This will be done by first finding λ from T, and then determining u from λ. Also, note that the stretch is positive.

(a) Write down an algorithm that uses either Newton's method or the secant method, for finding the value of λ for a given value of T.

(b) Show that

$$u(x) = u(0) - x + \int_0^x \lambda(s) ds.$$

(c) Suppose the interval $0 \leq x \leq \ell$ contains n equally spaced points x_i, where $x_1 = 0$ and $x_n = \ell$. Also, suppose that the value of λ is known at each of these points, and designate these values as λ_i, for $i = 1, 2, \cdots, n$. Write down an algorithm that uses the trapezoidal rule, which can be used to determine the u_i values from the λ_i values.

(d) Suppose that the values for the T_i's are given as

$$T_i = x_i^2[\alpha(1+x_i^2)+\beta](2+x_i^2)/(1+x_i^2)^2, \quad \text{for } i = 1,2,\cdots,n.$$

Using your algorithms from parts (a) and (c), compute the u_i's in the case of when $n = 20$ and $\ell = 1$, and then plot the values (x_i, u_i). In this calculation assume that $u(0) = 0$, and take $\alpha = 20$ and $\beta = 10$, which are typical values for an elastomer.

(e) Note that the exact solution at $x = 1$ is $u = 1/3$. For your algorithm in part (d), what value for n do you need to take so the error in your computed answer at $x = 1$ is no more than 10^{-6}?

6.15. The position $y(t)$, velocity $v(t)$, and acceleration $a(t)$ are related through the equations: $a(t) = v'(t)$ and $v(t) = y'(t)$. In this problem it is assumed that $v(0) = 0$ and $y(0) = 0$. In this case,

$$v(t) = \int_0^t a(r)dr \quad \text{and} \quad y(t) = \int_0^t v(r)dr.$$

It is also assumed that $a(t)$ is known, and the objective of this exercise is to compute the velocity and position from this information.

(a) Given a subinterval $t_i \le t \le t_{i+1}$, then $a_i = a(t_i)$ and $a_{i+1} = a(t_{i+1})$ are known. Assuming v_i and y_i have already been computed, use the trapezoidal rule to obtain the following expressions

$$v_{i+1} = v_i + \frac{1}{2}h(a_i + a_{i+1}),$$

and

$$y_{i+1} = y_i + \frac{1}{2}h(v_i + v_{i+1}).$$

(b) Suppose the interval $0 \le t \le 3$ is subdivided into n equally spaced subintervals. So, $t_i = (i-1)h$, where $i = 1,2,3,\cdots,n+1$ and $h = 3/n$. Assuming that $a(t) = \sin(t^4)$, plot y as a function of t, for $n = 10, 20, 40$. The three curves should be on the same axis.

(c) An accurately computed value for the position at $t = 3$ is $y(3) = 0.72732289075\ldots$.. What is the difference between this value and what you compute for $y(3)$ at $n = 10, 20, 40$? How large does n need to be so that this value and what you compute for $y(3)$ is less than 10^{-8} in absolute value?

6.16. This problem considers a way to compute velocity and position that differs from the one considered in Exercise 6.15. You will find that the value of n in part (c) is a factor of about 0.07 smaller than the corresponding value from Exercise 6.15(c).

(a) Suppose the interval $0 \le t \le 3$ is subdivided into n equally spaced subintervals. So, $t_i = (i-1)h$, where $i = 1,2,3,\cdots,n+1$ and $h = 3/n$.

Explain how the Hermite rule can be used to obtain the following expressions

$$v_{i+1} = v_i + \frac{1}{2}h(a_i + a_{i+1}) + \frac{1}{12}h^2(a_i' - a_{i+1}')$$

and

$$y_{i+1} = y_i + \frac{1}{2}h(v_i + v_{i+1}) + \frac{1}{12}h^2(a_i - a_{i+1})$$

(b) Assuming that $a(t) = \sin(t^4)$, plot, on the same axis, y as a function of t, for $n = 10, 20, 40$.

(c) An accurately computed value for the position at $t = 3$ is $y(3) = 0.72732289075\ldots$.. What is the difference between this value and what you compute for $y(3)$ at $n = 10, 20, 40$? How large does n need to be so that this value and what you compute for $y(3)$ is less than 10^{-8} in absolute value?

6.17. In this exercise you are to evaluate

$$\int_0^1 y(t)dt,$$

where $y(t)$ is determined by solving $y + t = e^{-y}$.

(a) Pick one of the numerical integration methods in Table C.2 and explain how it can be used to evaluate the integral. Also explain why you picked the particular integration rule.

(b) Evaluate the integral and also explain how you determined the number of subintervals to use. Make sure to turn in your m-file for this.

6.18. This exercise explores some connections between Simpson's rule and some of the other methods that were derived. In this exercise, $I(n)$ designates a composite rule that uses n subintervals. Assume here that n is even.

(a) Show that $I_S(n) = \frac{2}{3}I_T(n) + \frac{1}{3}I_M(n/2)$.

(b) Show that $I_S(n) = \frac{4}{3}I_T(n) - \frac{1}{3}I_T(n/2)$.

6.19. Occasionally you will see someone try to adjust their data in an attempt to use Simpson's rule. To explain, the goal is to evaluate the integral

$$\int_{x_{i-1}}^{x_{i+1}} f(x)dx.$$

Assume that the value of $f(x)$ is known at x_{i-1} and x_{i+1} but not at x_i. The question is, can you use the data to find an approximation for $f(x_i)$ that will enable you to use Simpson's rule, and in the process get a better result than you would get using the trapezoidal rule?

(a) What approximation of the integral is obtained using the trapezoidal rule?

(b) One possibility for approximating $f(x_i)$ is to use piecewise linear interpolation using the two data points (x_{i-1}, f_{i-1}) and (x_{i+1}, f_{i+1}). Doing this, and inserting the resulting approximation for f_i into Simpson's rule, what results? How does this differ from your answer in part (a)?

(c) Suppose one just assumes that there are constants A and B so that $f_i = Af_{i-1} + Bf_{i+1}$. With this, Simpson's rule reduces to an integration rule of the form

$$\int_{x_{i-1}}^{x_{i+1}} f(x)dx = w_1 f_{i-1} + w_2 f_{i+1}.$$

What do w_1 and w_2 have to be to maximize the precision? How does this differ from your answer in part (a)?

6.20. In this exercise Romberg rules are derived. Assume that $I(n)$ is an integration rule that uses n subintervals and h is the corresponding width of each subinterval.

(a) It is known that the error term for the composite Simpson's rule involves even powers of h. In particular,

$$\int_a^b f(x)dx = I_S(n) + \alpha h^4 + \beta h^6 + \gamma h^8 + \cdots .$$

Show that the integration rule

$$I_R = \frac{1}{15}\left[16 I_S(2n) - I_S(n)\right]$$

has an error that is $O(h^6)$.

(b) Suppose

$$\int_a^b f(x)dx = I(n) + \alpha h^2 + \beta h^3 + \gamma h^4 + \cdots .$$

Show that the integration rule

$$I_R = \frac{1}{21}\left[32 I(4n) - 12 I(2n) + I(n)\right]$$

has an error that is $O(h^4)$.

(c) Suppose

$$\int_a^b f(x)dx = I(n) + \alpha h^2 + \beta h^3 + \gamma h^4 + \cdots .$$

Show that the integration rule

$$I_R = \frac{1}{12}\left[27 I(3n) - 16 I(2n) + I(n)\right]$$

has an error that is $O(h^4)$.

6.21. Suppose the integration rule has the form

$$\int_{x_i}^{x_{i+1}} f(x)dx \approx w_1 f(x_i) + w_2 f(z)$$

This is an example of what is called Radau quadrature, or Gauss-Radau quadrature, which means that one, and only one, of the points used is an endpoint.

(a) Find the values of w_1, w_2, and z that maximize the precision. [Hint: let $z = x_i + \alpha h$ and find α]

(b) The error is known to have the form

$$\int_{x_i}^{x_{i+1}} f(x)dx = w_1 f(x_i) + w_2 f(z) + Kh^4 f'''(\eta)$$

where, as usual, η is a point somewhere in the interval $[x_i, x_{i+1}]$. Find K.

6.22. The purpose of this exercise is to use the idea of precision to derive the error formula for Simpson's rule.

(a) Using the formulas in Table 6.5, show that Simpson's rule has precision 3.

(b) Use $f(x) = x^4$ to derive the formula for the error for Simpson's rule.

6.23. Suppose the integration rule has the form

$$\int_{x_i}^{x_{i+1}} f(x)dx \approx w_1 f(z_1) + b_1 f'(w_1)$$

(a) Find the values of w_1, z_1, b_1, and w_1 that maximize the precision. [Hint: let $z = x_i + \alpha h$ and find α]

(b) The error is known to have the form

$$\int_{x_i}^{x_{i+1}} f(x)dx = w_1 f(x_i) + b_1 f'(w_1) + Kh^4 f'''(\eta)$$

where, as usual, η is a point somewhere in the interval $[x_i, x_{i+1}]$. Find K.

6.24. This problem considers what is known as Lobatto quadrature, or Gauss-Lobatto quadrature. It differs from Gaussian quadrature in that it assumes that the integration rule includes both endpoints, and possibly other points within the interval.

(a) The assumed form using two points is

$$\int_{x_i}^{x_{i+1}} f(x)dx \approx w_1 f(x_i) + w_2 f(x_{i+1}).$$

Find the values of w_1 and w_2 that maximize the precision.

(b) The assumed form using three points is

$$\int_{x_i}^{x_{i+1}} f(x)dx \approx w_1 f(x_i) + w_2 f(z) + w_3 f(x_{i+1}).$$

Find the values of w_1, w_2, w_3, and z that maximize the precision.

(c) The assumed form using four points is

$$\int_{x_i}^{x_{i+1}} f(x)dx \approx w_1 f(x_i) + w_2 f(z_1) + w_3 f(z_2) + w_4 f(x_{i+1}).$$

Show that the values of w_1, w_2, w_3, w_4, z_1, and z_2 that maximize the precision are: $w_1 = w_4 = h/12$, $w_2 = w_3 = 5h/12$, and $q = h/(2\sqrt{5})$. In deriving this result you can assume that $z_1 = x_i + \frac{1}{2}h - q$ and $z_2 = x_i + \frac{1}{2}h + q$. Also note that the error has the same form as given in Theorem 6.5, but $K = -1/1{,}512{,}000$ and $m = 5$ (you do not need to show this).

6.25. Suppose you want to determine how well a function $g(x)$ approximates another function $f(x)$, over an interval $a \leq x \leq b$. One way to do this is to calculate the area of the region between them, which is determined by the value of the integral

$$A = \int_a^b |f(x) - g(x)|dx.$$

To compute an approximate value of this, you can use $n+1$ equally spaced points over the interval, with $x_1 = a$ and $x_{n+1} = b$. Letting $\mathbf{f}$ and $\mathbf{g}$ be the vectors determined from the $n+1$ values of $f(x)$ and $g(x)$ at these points, show that

$$A \approx \frac{b-a}{n}||\mathbf{f} - \mathbf{g}||_1 - \frac{b-a}{2n}\left[\,|f(a) - g(a)| + |f(b) - g(b)|\,\right].$$

Chapter 7
Initial Value Problems

In this chapter we derive numerical methods to solve the first-order differential equation

$$\frac{dy}{dt} = f(t, y), \quad \text{for } \ 0 < t, \tag{7.1}$$

where

$$y(0) = \alpha. \tag{7.2}$$

This is known as an initial value problem (IVP), and it consists of the differential equation (7.1) along with the initial condition in (7.2). Numerical methods for solving this problem are first derived for the case of when there is one differential equation. Afterwards, the methods are extended to problems involving multiple equations.

It is of interest to know that several of the methods derived in this chapter are summarized in Appendix C, Table C.4.

7.1 Examples of IVPs

7.1.1 Radioactive Decay

According to the law of radioactive decay, the mass of a radioactive substance decays at a rate that is proportional to the amount present. To express this in mathematical terms, let $y(t)$ designate the amount present at time t. In this case the decay law can be expressed as

$$\frac{dy}{dt} = -ry, \quad \text{for } \ 0 < t. \tag{7.3}$$

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_7

If we start out with an amount α of the substance then the corresponding initial condition is

$$y(0) = \alpha. \tag{7.4}$$

In the decay law (7.3), r is the proportionally constant and it is assumed to be positive. Because $f(t, y) = -ry$ is a linear function of y the IVP is said to be linear. One consequence of this is that it is possible to find the solution using an integrating factor or using the method of separation of variables. What one finds is

$$y(t) = \alpha e^{-rt}. \tag{7.5}$$

Consequently, the solution starts at α and decays exponentially to zero as time increases.

To put a slightly different spin on this, recall that $y = Y$ is a *steady-state* solution of (7.1) if it is constant and satisfies $f(t, Y) = 0$. Also, a steady-state Y is stable if any solution that starts near Y stays near it. If, in addition, initial conditions starting near Y actually result in the solution converging to Y as $t \to \infty$, then $y = Y$ is said to be *asymptotically stable*. With the solution in (7.5) we conclude that $y = 0$ is an asymptotically stable steady-state solution for (7.3).

7.1.2 Logistic Equation

In the study of populations limited by the supply of food, one obtains the logistic equation, which is

$$\frac{dy}{dt} = ry(1 - y), \quad \text{for} \ \ 0 < t, \tag{7.6}$$

where

$$y(0) = \alpha. \tag{7.7}$$

It is assumed that r and α are positive. For this problem, $f(t, y) = ry - ry^2$ is a nonlinear function of y and therefore the IVP is nonlinear. It is possible to find the solution using separation of variables, and the result is

$$y(t) = \frac{\alpha}{\alpha + (1 - \alpha)e^{-rt}} \,. \tag{7.8}$$

Also, the steady-state solutions for this equation are the constants that satisfy $ry(1 - y) = 0$, which means that $y = 1$ or $y = 0$. Because $r > 0$, the solution in (7.8) approaches $y = 1$ as t increases. Consequently, $y = 1$ is an asymptotically stable steady-state solution, whereas $y = 0$ is not.

7.2 Numerical Differentiation

There are many ways to solve an IVP like the one in (7.1), (7.2) and we will begin with the direct approach. This means we will need to be able to compute the derivative. So, before attempting to solve the IVP we need to derive formulas for calculating derivatives.

Just as with integration, the definition of a derivative from calculus is the starting point for our approximations. Although there are different ways to state the definition, the one used here is the following:

$$y'(t) = \lim_{k\to 0} \frac{y(t+k)-y(t)}{k} .$$

Consequently, for small values of k, we have the approximation

$$y'(t) \approx \frac{y(t+k)-y(t)}{k} .$$

As will be explained in more detail later, we will compute the solution of the IVP at equally spaced time points $t_0, t_1, t_2, \ldots, t_M$, where $t_j = jk$, for $j = 0, 1, 2, \ldots, M$. With this, the above approximation can be written as

$$y'(t_j) \approx \frac{y(t_j+k)-y(t_j)}{k} .$$

To be useful as a computing tool, it is essential to know the error for this approximation. For this we rely, as usual, on Taylor's theorem. Given that

$$y(t_j+k) = y(t_j) + ky'(t_j) + \frac{1}{2}k^2y''(t_j) + \cdots ,$$

then

$$\begin{aligned}\frac{y(t_j+k)-y(t_j)}{k} &= \frac{[y(t_j) + ky'(t_j) + \frac{1}{2}k^2y''(t_j) + \cdots] - y(t_j)}{k} \\ &= \frac{ky'(t_j) + \frac{1}{2}k^2y''(t_j) + \cdots}{k} \\ &= y'(t_j) + \frac{1}{2}ky''(t_j) + \cdots .\end{aligned}$$

In other words,

$$y'(t_j) = \frac{y(t_{j+1})-y(t_j)}{k} + \tau_j, \tag{7.9}$$

where

$$\tau_j = -\frac{1}{2}ky''(t_j) + \cdots . \tag{7.10}$$

Note that τ_j is the called the *truncation error*, and because it is $O(k)$, the resulting approximation is first-order. The expression in (7.9) is listed in

Table 7.1 as a forward-difference formula. It is forward because it uses information at a future time, t_{j+1}, to construct the approximation. Also the formula for τ_j in the table looks different than the one given in (7.10). They are equivalent in the sense that the one in the table comes from the remainder term in Taylor's theorem, while (7.10) is the series expansion for τ_j.

A criticism of (7.9) is that it is only first-order, and we are interested in having higher-order approximations. To do this, note that (7.9) uses t_j and t_{j+1} to obtain the approximation. In what follows, we will investigate how it is possible to find approximations that use other time points. For example, we will look to see if it is possible to find an approximation that uses t_{j-1} and t_{j+1}. One outcome of this is that we will find higher-order approximations for the derivative, as well as approximations for the other derivatives of $y(t)$.

7.2.1 Using t_{j+2}, t_{j+1}, and t_j

The assumption is that we can find A, B, and C so that

$$y'(t_j) \approx Ay(t_{j+2}) + By(t_{j+1}) + Cy(t_j).$$

Using Taylor's theorem,

$$\begin{aligned} y(t_{j+2}) &= y(t_j + 2k) \\ &= y(t_j) + 2ky'(t_j) + 2k^2y''(t_j) + \frac{4}{3}k^3y'''(t_j) + \cdots . \end{aligned} \tag{7.11}$$

Also, we know that

$$y(t_{j+1}) = y(t_j) + ky'(t_j) + \frac{1}{2}k^2y''(t_j) + \frac{1}{6}k^3y'''(t_j) + \cdots . \tag{7.12}$$

This means, we want

$$\begin{aligned} y'(t_j) &\approx Ay(t_{j+2}) + By(t_{j+1}) + Cy(t_j) \\ &= A\Big[y(t_j) + 2ky'(t_j) + 2k^2y''(t_j) + \frac{4}{3}k^3y'''(t_j) + \cdots\Big] \\ &\quad + B\Big[y(t_j) + ky'(t_j) + \frac{1}{2}k^2y''(t_j) + \frac{1}{6}k^3y'''(t_j) + \cdots\Big] + Cy(t_j) \\ &= (A + B + C)y(t_j) + (2A + B)ky'(t_j) \\ &\quad + \frac{1}{2}(4A + B)k^2y''(t_j) + \frac{1}{6}(8A + B)k^3y'''(t_j) + \cdots . \end{aligned} \tag{7.13}$$

This is suppose to hold for any (smooth) function $y(t)$, which means that A, B, and C do not depend on y. Consequently, equating the left and right sides we conclude that

$$A + B + C = 0,$$
$$(2A + B)k = 1.$$

The error in the resulting approximation is given in (7.13). Given that we have three unknowns, we are able to impose one more condition. With the goal of achieving the best error possible we will remove the $O(k^2)$ term, and this means that

$$4A + B = 0.$$

Solving the resulting three equations one finds that $A = -1/(2k)$, $B = 2/k$, and $C = -3/(2k)$. Therefore, our approximation is

$$y'(t_j) = \frac{-y(t_{j+2}) + 4y(t_{j+1}) - 3y(t_j)}{2k} + \tau_j, \tag{7.14}$$

where

$$\tau_j = -\frac{1}{6}k^2 y'''(t_j) + \cdots .$$

This is listed in Table 7.1 as a one-sided difference formula, because of where the time points are located. It is also an example of a forward difference formula.

7.2.2 Using $\mathbf{t_{j+1}}$ and $\mathbf{t_{j-1}}$

This means, we want

$$\begin{aligned}
y'(t_j) &\approx Ay(t_{j+1}) + By(t_{j-1}) \\
&= A\left[y(t_j) + ky'(t_j) + \frac{1}{2}k^2y''(t_j) + \frac{1}{6}k^3y'''(t_j) + \cdots\right] \\
&\qquad + B\left[y(t_j) - ky'(t_j) + \frac{1}{2}k^2y''(t_j) - \frac{1}{6}k^3y'''(t_j) + \cdots\right] \\
&= (A + B)y(t_j) + (A - B)ky'(t_j) \\
&\qquad + \frac{1}{2}(A + B)k^2y''(t_j) + \frac{1}{6}(A - B)k^3y'''(t_j) + \cdots .
\end{aligned} \tag{7.15}$$

Equating the left and right sides we conclude that

$$A + B = 0,$$
$$(A - B)k = 1.$$

Solving the resulting two equations one finds that $A = 1/(2k)$ and $B = -2/k$. Therefore, our approximation is

$$y'(t_j) = \frac{y(t_{j+1}) - y(t_{j-1})}{2k} + \tau_j, \tag{7.16}$$

where

$$\tau_j = -\frac{1}{6}k^2 y'''(t_j) + \cdots . \tag{7.17}$$

This is listed in Table 7.1 as a centered difference formula. It is centered because it uses time points that are symmetrically placed around t_j. Also, it produces a second-order approximation because the error is $O(k^2)$.

Example

Suppose $y(t) = \sqrt{t}$, and we use the above formulas to calculate $y'(1)$. Taking $t_j = 1$ and $t_{j\pm1} = 1 \pm k$ then the forward approximation is

$$y'(1) \approx \frac{y(t_{j+1}) - y(t_j)}{k} = \frac{\sqrt{1+k} - 1}{k}, \tag{7.18}$$

and the centered approximation is

$$y'(1) \approx \frac{y(t_{j+1}) - y(t_{j-1})}{2k} = \frac{\sqrt{1+k} - \sqrt{1-k}}{2k}. \tag{7.19}$$

The exact value is $y'(1) = \frac{1}{2}$, and the computed values obtained from the above two approximations are shown in Figure 7.1 for decreasing values of k.

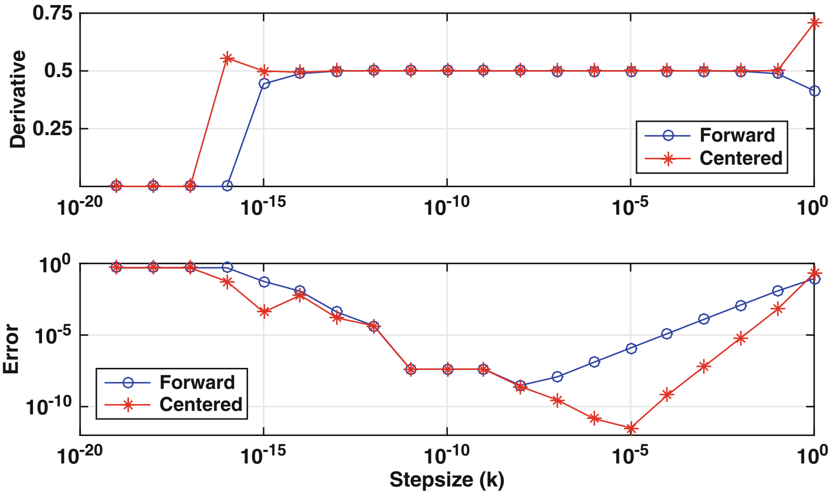


Figure 7.1 Upper graph: Values obtained from (7.18) and (7.19) for $y'(1) = \frac{1}{2}$ when $y(t) = \sqrt{t}$. Lower graph: Error in each approximation.

In the upper graph, both look to be doing well down to about $k = 10^{-14}$ but show problems at $k = 10^{-15}$. To examine this, the error for (7.18) is

$$\left| y'(1) - \frac{\sqrt{1+k} - 1}{k} \right|,$$

with a similar error formula for (7.19). The values of these errors are shown in the lower graph in Figure 7.1. According to (7.10), the error for (7.18) should decrease as $O(k)$, while from (7.17) the error for (7.19) should decrease as $O(k^2)$. Starting at $k = 1$, they both behave as expected. Namely, the error in the forward approximation (7.18) decreases as a first-order method should (i.e., decreasing k by a factor of 10 decreases the error by the same factor) and the centered approximation decreases as a second-order method should. However, for both approximations, there is a value for k where the error starts to increase, and continues to increase (mostly) for smaller values of k. As explained in Section 7.7, this is due to round-off. This is a common problem with numerical differentiation, and it limits the usefulness of these formulas. There are ways to avoid this, and one possibility is to rewrite the formula. As an example, one can rewrite (7.19) as

$$y'(1) \approx \frac{1}{\sqrt{1+k} + \sqrt{1-k}}.$$

Unfortunately, it is not possible to do this in the general case. Another possibility is to use a complex valued time step, something called a complex Taylor series expansion, and this will be explained in Section 7.7. However, in the end, with numerical differentiation and solving IVPs, we are stuck with the situation shown in Figure 7.1. This gives rise to what is known as the optimal step size, which is the step size where the error is minimized. It is possible, using the ideas discussed in Section 1.4, to derive formulas for the optimal step size, but they are not really needed when solving IVPs. The reason is that the better IVP solvers do not usually require very small values of k to produce accurate solutions of an IVP. ■

7.2.3 Higher Derivatives

It is relatively easy to use the procedure to derive approximations for higher derivatives. For example, if you intend on using t_{j+1}, t_j, and t_{j-1} to obtain an approximation for $y''(t_j)$, then

$$\begin{aligned} y''(t_j) &\approx Ay(t_{j+1}) + By(t_j) + Cy(t_{j-1}) \\ &= (A+B+C)y(t_j) + (A-C)ky'(t_j) + \frac{1}{2}(A+C)k^2y''(t_j) \\ &\qquad + \frac{1}{6}(A-C)k^3y'''(t_j) + \frac{1}{24}(A+C)k^4y''''(t_j) + \cdots . \end{aligned}$$

Equating the left and right sides we conclude that

$$\begin{aligned} A+B+C &= 0, \\ A-C &= 0, \\ \frac{1}{2}(A+C)k^2 &= 1. \end{aligned}$$

Solving these, one obtains

$$y''(t_j) = \frac{y(t_{j+1}) - 2y(t_j) + y(t_{j-1})}{k^2} + \tau_j, \tag{7.20}$$

where

$$\tau_j = -\frac{1}{12}k^2y''''(t_j) + \cdots .$$

This is listed in Table 7.1 as a centered difference approximation.

Type	**Difference Approximation**	**Truncation Term**
Forward	$y'(t_j) \approx \frac{y(t_{j+1})-y(t_j)}{k}$	$\tau_j = -\frac{1}{2}ky''(\eta_j)$
Backward	$y'(t_j) \approx \frac{y(t_j)-y(t_{j-1})}{k}$	$\tau_j = \frac{1}{2}ky''(\eta_j)$
Centered	$y'(t_j) \approx \frac{y(t_{j+1})-y(t_{j-1})}{2k}$	$\tau_j = -\frac{1}{6}k^2y'''(\eta_j)$
One-sided	$y'(t_j) \approx \frac{-y(t_{j+2})+4y(t_{j+1})-3y(t_j)}{2k}$	$\tau_j = \frac{1}{3}k^2y'''(\eta_j)$
One-sided	$y'(t_j) \approx \frac{3y(t_j)-4y(t_{j-1})+y(t_{j-2})}{2k}$	$\tau_j = \frac{1}{3}k^2y'''(\eta_j)$
Centered	$y''(t_j) \approx \frac{y(t_{j+1})-2y(t_j)+y(t_{j-1})}{k^2}$	$\tau_j = -\frac{1}{12}k^2y''''(\eta_j)$

Table 7.1 Numerical differentiation formulas. The exact relationships between the approximation and the derivative are illustrated in (7.9), (7.14), (7.16), and (7.20). Also, these formulas assume equally spaced points with step size $k = t_{j+1} - t_j$, and the point η_j is located between the left- and rightmost points used in the formula.

7.2.4 Interpolation

It is possible to use some of the interpolation methods from Chapter 5 to produce approximations for the derivative of a function. For example, the forward and backward difference approximations can be obtained by differentiating the formula for piecewise linear interpolation (5.9). Similarly, the centered difference approximation as well as the one for $y''(t_j)$ can be derived from the piecewise quadratic interpolation formula (see Exercise 5.24). The drawback with this approach is that the corresponding error formulas do not, necessarily, come from differentiating the interpolation error given in Theorem 5.2. Why this is the case, and how to determine the appropriate error formulas, is discussed in Süli and Mayers [2003]. The exception to this occurs with cubic splines, and this is explained in Section 5.5.3.

7.3 IVP Methods Using Numerical Differentiation

The task we now undertake is to approximate the differential equation, and its accompanying initial condition, with a problem we can solve using a computer. To explain how this is done, consider the problem of solving

$$\frac{dy}{dt} = f(t,y), \quad \text{for } \ 0 < t, \tag{7.21}$$

where

$$y(0) = \alpha. \tag{7.22}$$

The function $f(t,y)$ is assumed to be given. For example, with radioactive decay $f(t,y) = -ry$ and for the logistic problem $f(t,y) = ry(1-y)$. The question is, can we accurately compute the solution directly from the problem without first finding an analytical solution? As it turns out, most realistic mathematical models of physical and biological systems cannot be solved by hand, so having the ability to find accurate numerical solutions directly from the original equations is an invaluable tool.

7.3.1 The Five Steps

To explain how we will construct a numerical algorithm that can be used to solve (7.21) it should be noted that the variables in this problem, t and y, are continuous. Our objective is to replace these with discrete variables so that the resulting problem is algebraic and therefore solvable using standard numerical methods. Great care must be taken in making this replacement,

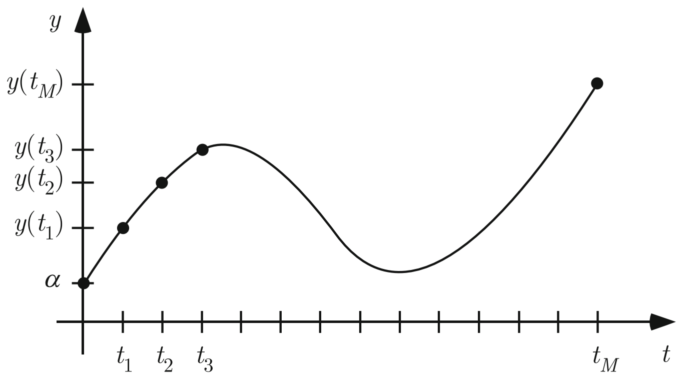


Figure 7.2 Grid system used to derive a finite difference approximation of the initial value problem. The points are equally spaced and $t_M = T$.

because the computed solution must accurately approximate the solution of the original IVP. The approach we take proceeds in a sequence of five steps.

One point to make before beginning is that the computer cannot run forever. Therefore, it is necessary to specify the time interval $0 \leq t \leq T$ over which the solution will be computed.

Step 1: Grid

We first introduce the time points at which we will compute the solution. These points are labeled sequentially as $t_0, t_1, t_2, \dots, t_M$ and a schematic drawing indicating their location along the time axis is shown in Figure 7.2. We confine our attention to a uniform grid with step size k, so, the formula for the time points is

$$t_j = jk, \quad \text{for } \ j = 0, 1, 2, \dots, M. \tag{7.23}$$

Because the time interval is $0 \leq t \leq T$, we require $t_M = T$. Therefore, k and M are connected through the equation

$$k = \frac{T}{M}\,. \tag{7.24}$$

Step 2: Evaluation

Evaluate the differential equation at the time point $t = t_j$ to obtain

$$y'(t_j) = f(t_j, y(t_j)). \tag{7.25}$$

Step 3: Finite Difference Formula

Replace the derivative term in Step 2 with a finite difference formula using the values of y at one or more of the grid points in a neighborhood of t_j. This is where things get a bit interesting, because numerous choices can be made, a few of which are listed in Table 7.1. Different choices result in different numerical procedures, and as it turns out, not all choices will work. To start, we take the first entry listed in the table, which means we use the following expression for the first derivative:

$$y'(t_j) = \frac{y(t_{j+1}) - y(t_j)}{k} + \tau_j, \tag{7.26}$$

where

$$\tau_j = -\frac{k}{2} y''(\eta_j) \tag{7.27}$$

and η_j is a point between t_j and t_{j+1}. Introducing this into (7.25) we obtain

$$\frac{y(t_{j+1}) - y(t_j)}{k} + \tau_j = f(t_j, y(t_j)), \tag{7.28}$$

or equivalently,

$$y(t_{j+1}) - y(t_j) + k\tau_j = kf(t_j, y(t_j)). \tag{7.29}$$

An important point to make here concerns the term τ_j. As it appears in (7.28), τ_j represents how well we have approximated the differential equation. For this reason it is the *truncation error* for the method, and from (7.27) it is seen that it is $O(k)$. It is essential that whatever approximations we use, the truncation error goes to zero as k goes to zero. This means that, at least in theory, we can approximate the original problem as accurately as we wish by making the time step k small enough. It is said in this case that the approximation is *consistent*. Unfortunately, as we demonstrate shortly, consistency is not enough to guarantee an accurate numerical solution.

Step 4: Finite Difference Approximation

Drop the term containing the truncation error. This is the step where we go from an exact problem to one that is, hopefully, an accurate approximation of the original. After dropping τ_j in (7.29) the resulting equation is

$$y_{j+1} - y_j = kf(t_j, y_j), \tag{7.30}$$

or equivalently,

$$y_{j+1} = y_j + kf(t_j, y_j), \quad \text{for} \ \ j = 0, 1, 2, \ldots, M-1. \tag{7.31}$$

From the initial condition (7.22) we have that the starting value is

$$y_0 = \alpha. \tag{7.32}$$

The finite difference equation (7.31) is known as *Euler's method* for solving (7.21). It is a recursive algorithm in which one starts with $j = 0$ and then uses (7.31) to determine the solution at $j = 1$, then $j = 2$, then $j = 3$, etc. Because (7.31) gives the unknown y_{j+1} explicitly in terms of known quantities, it is an explicit method.

Example

Let's see how well Euler's method does with the logistic equation (7.6). Specifically, suppose the IVP is

$$\frac{dy}{dt} = 10y(1-y), \quad \text{for} \ \ 0 < t, \tag{7.33}$$

where

$$y(0) = 0.01. \tag{7.34}$$

We will use the Euler method to calculate the solution for $0 \le t \le 1$. In this case, using (7.24), k and M are connected through the equation

$$k = \frac{1}{M}. \tag{7.35}$$

For this example, the finite difference equation in (7.31) takes the form

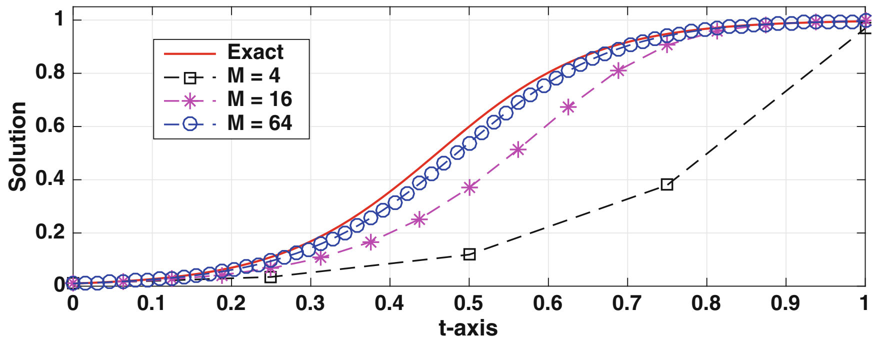


Figure 7.3 Solution of the logistic equation (7.33) using the Euler method (7.36) for three values of M. Also shown is the exact solution. The symbols are the computed values, and the dashed lines are drawn by the plotting program simply to connect the values.

j	t_j	$y(t_j)$	y_j	$\overline{y}_j$	$y(t_j)-y_j$	$y_j-\overline{y}_j$
0	0	$\frac{1}{100}$	$\frac{1}{100}$	0.01	0	0
1	$\frac{1}{6}$	$\left(1+99e^{-5/3}\right)^{-1}$	$\frac{53}{2000}$	2.6500e−02	2.43e−02	−3.47e−18
2	$\frac{1}{3}$	$\left(1+99e^{-10/3}\right)^{-1}$	$\frac{55597}{800000}$	6.9496e−02	1.51e−01	−1.39e−17
3	$\frac{1}{2}$	$\left(1+99e^{-5}\right)^{-1}$	$\frac{68073133591}{384000000000}$	1.7727e−01	4.23e−01	−2.78e−17
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

Table 7.2 The first few time steps in solving the logistic equation (7.33) using the Euler method. Note that $y(t_j)$ is the exact solution of logistic equation, y_j is the exact value from (7.36), and $\overline{y}_j$ is the computed value from (7.36).

$$y_{j+1} = y_j + 10ky_j(1-y_j), \quad \text{for } \; j = 0,1,2,\ldots,M-1. \tag{7.36}$$

Taking $M = 6$, so $k = \frac{1}{6}$, the first few steps using the Euler method are shown in Table 7.2. For a more graphical picture of the situation, the exact solution, given in (7.8), and the computed solutions, are shown in Figure 7.3 using successively smaller values of the time step k or, equivalently, larger values of M. It is seen that the numerical solution with $M = 4$ is not so good, but the situation improves considerably as more time points are used. In fact, it would appear that if we keep increasing the number of time points that the numerical solution converges to the exact solution. ■

Step 5: Stability

For a finite difference method to work, two requirements must be satisfied. One is that the approximation is consistent. This was mentioned in Step 3 and will be discussed again later. The second requirement is that it is stable, which means, roughly, that the approximation produces a solution with properties similar to those of the exact solution. As an example, the simplest IVP is $y' = 0$, where $y(0) = 1$, and the exact solution is $y = 1$. The requirement for what is known as 0-stability is that the numerical method applied to $y' = 0$ produces a solution that is at least bounded. There is a stronger form of stability, which is more useful for many of the problems which arise in applications, and it is known as A-stability. This is determined by using the method to solve the radioactive decay equation

$$\frac{dy}{dt} = -ry, \tag{7.37}$$

where

$$y(0) = 1. \tag{7.38}$$

The exact solution is $y(t) = e^{-rt}$, and, assuming that $r > 0$, this function approaches zero as t increases. It is required that the solution y_j of the finite difference problem also approaches zero, and this is the basis for the following definition:

Definition 7.1. If a numerical method, when applied to (7.37) and (7.38), produces a solution with the property that $y_j \to 0$ as $j \to \infty$, irrespective of the (positive) value of r and k, then the method is said to be *A-stable.* If the zero limit occurs only when k is small (with k and r positive), then the method is *conditionally A-stable.* Otherwise, the method is *unstable.*

Let's apply this definition to Euler's method. For the equation in (7.37), Euler's method (7.31) reduces to $y_{j+1} = (1 - rk)y_j$. This can be written as

$$y_{j+1} = \kappa y_j,$$

where $\kappa = 1 - rk$ is called the *amplification factor* for Euler's method. The above equation arises for many of the methods we will consider, and this means that the following result will be useful:

Theorem 7.1. *If $y_0 = 1$ then the solution of*

$$y_{j+1} = \kappa y_j, \text{ for } j = 0, 1, 2, \cdots, \tag{7.39}$$

is $y_j = \kappa^j$. Therefore, $y_j \to 0$ as $j \to \infty$ if and only if the amplification factor κ satisfies $|\kappa| < 1$.

The proof of this is straightforward, and is left as an exercise. Since the amplification factor for Euler's method is $\kappa = 1 - rk$, then according to the theorem, Euler's method is A-stable only as long as

$$|1 - rk| < 1.$$

From this we conclude that the step size must satisfy the condition $k < 2/r$. Therefore, the Euler method is conditionally A-stable.

Wonder what an unstable solution might look like? In Figure 7.4, four solutions of the logistic equation obtained using the Euler method are shown. The top two graphs show an unstable situation, with the solution growing with t. For example, when $kr = 4$, the method finds that $y(5) \approx -10^{18}$. The graph at the bottom, where $rk = 1$, corresponds to a stable case. Comparing the curves in this lower graph for $0 < t < 1$ it is evident that satisfying the stability condition does not necessarily mean that the error is small.

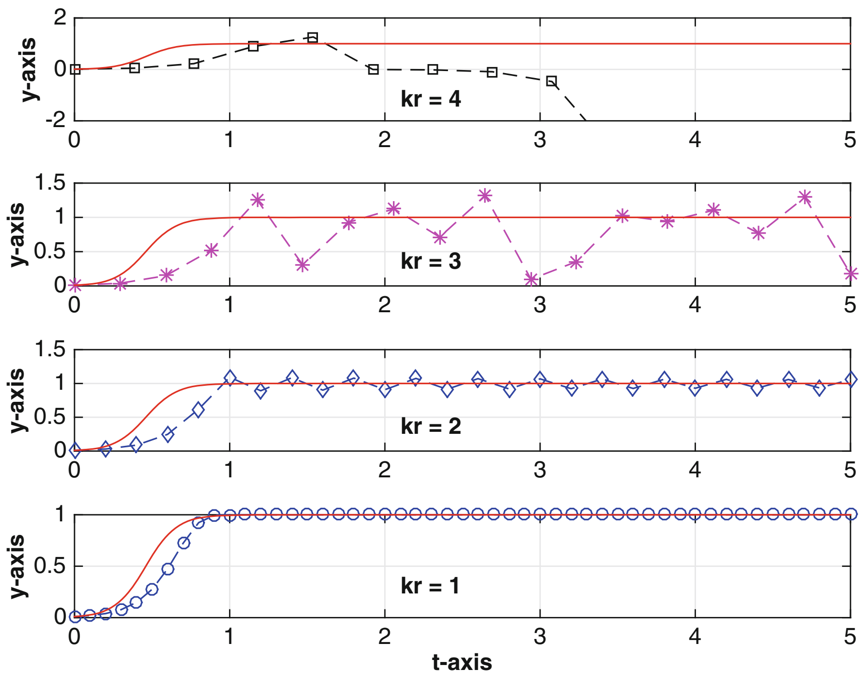


Figure 7.4 Solution of the logistic equation (7.33) using the Euler method (7.36) for four values of kr, where $r = 10$. Also shown in each graph, with the solid (red) curve, is the exact solution.

As a final comment, it might be puzzling why the radioactive decay equation is the arbiter for stability. The reason comes from the desire that if the differential equation has an asymptotically stable steady-state $y = a$, then $y = a$ is also an asymptotically stable steady-state for the approximating difference equation. Finding a condition that ensures this for the general problem is not particularly easy, but it is possible to derive a condition for differential equations of the form $y' = g(y)$. Writing $y(t) = a + Y(t)$, and using Taylor's theorem, one can show that the equation for Y is approximately $Y' = -rY$, where $r = g'(a)$. In other words, near $y = a$, and assuming that $r \neq 0$, the original differential equation can be approximated with the radioactive decay equation. The requirement of A-stability ensures that the numerical method produces a solution that, like the actual solution, approaches the steady-state. However, there are situations where A-stability is either not enough or not relevant. This is why other types of stability have been defined, and these include L-stability, B-stability, and BN-stability. Those interested in this should consult Butcher [2008].

7.3.2 Error

It is essential to examine the requirements needed to guarantee a numerical IVP solver will work. This, as usual, will require us to consider the error. As illustrated in Table 7.2, at each time point we have three different solutions, and they are

$$y(t_j) \equiv \text{exact solution of the IVP at } t = t_j; \tag{7.40}$$

$$y_j \equiv \text{exact solution of finite difference equation at } t = t_j; \tag{7.41}$$

$$\overline{y}_j \equiv \text{solution of difference equation at } t = t_j \text{ calculated by the computer.} \tag{7.42}$$

We are interested in the difference between the exact solution of the IVP and the values we actually end up computing using our algorithm. Therefore, we are interested in the error $e_j = |y(t_j) - \overline{y}_j|$. To help make it more apparent what is contributing to the error we rewrite it as follows:

$$e_j = |y(t_j) - y_j + y_j - \overline{y}_j|. \tag{7.43}$$

From this, the error can be considered as coming from the following two sources:

$y(t_j) - y_j$: This is the *approximation error* at $t = t_j$. This corresponds to the difference between the exact solution of the IVP and the exact solution of the problem we use as its approximation. As occurs in Table 7.2, this should be the major contributor to the error until k is small enough that this difference gets down to approximately that of the round-off.

$y_j - \overline{y}_j$: This is the *computational error* at $t = t_j$. This originates from round-off when using floating-point calculations to compute the solution, and if the method is implicit then this also includes the possible iteration error. The last column of Table 7.2 gives the values of this error for the first few time points. Getting values of 10^{-15} or smaller, as occur in this calculation, is about as good as can be expected using double precision.

The question we are going to ask is, if we increase the number of time steps in the time interval $0 \leq t \leq T$, will the error decrease to zero or at least decrease down to the level of the round-off? We want the answer to this question to be yes and, moreover, that it is true no matter what choice we make for T. If this holds, then the method is *convergent*. It is possible to prove that if a method is consistent, and A-stable or conditionally A-stable, then the method is convergent.

In terms of computing the solution, knowing the method will work is important, but it is just as important to know how well it works. More

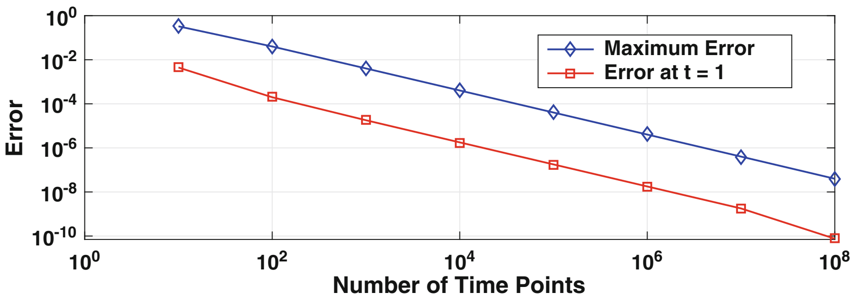


Figure 7.5 The difference between the exact and computed solutions, as a function of the number of time steps, M, used in solving the logistic equation (7.33) with the Euler method (7.36). Shown is the error $|y(T) - \overline{y}_M|$ at $t = 1$ as well as the maximum error as determined using (7.44).

specifically, how does the error decrease as we reduce the time step? This is illustrated in the next example.

Example

The error $e_M = |y(T) - \overline{y}_M|$ from the Euler method is plotted in Figure 7.5 as a function of the number of time points used to reach $T = 1$. It is seen that the error decreases linearly in the log-log plot in such a way that increasing M by a factor of 10 decreases the error by the same factor. In other words, the error decreases as k^n, with $n = 1$. It is not a coincidence that this is the same order as for the truncation error (7.27). At first glance, because the term that is neglected in (7.29) is $k\tau_j = O(k^2)$, one might expect that the error in Figure 7.5 would decrease as k^2. However, $k\tau_j$ is the error we generate at each time step. To get to T we take $M = 1/k$ time steps so the accumulated error we generate in getting to T is reduced by a factor of k. Therefore, with a convergent method the order of the truncation error determines the order of the error. ■

We are using the error at $t = T$ to help determine how the approximation improves as the number of time steps increases. In many applications, however, one is interested in how well the numerical solution approximates the solution throughout the entire interval $0 \le t \le T$. For this it is more appropriate to consider using a vector norm to define the error. For example, using the maximum norm the error function takes the form

$$\begin{aligned} e_\infty &= \max_{j=0,1,\ldots,M} |y(t_j) - \overline{y}_j| \\ &= ||\mathbf{y} - \overline{\mathbf{y}}||_\infty, \end{aligned} \tag{7.44}$$

where $\mathbf{y} = (y(t_0), y(t_1), \cdots, y(t_M))^T$ and $\overline{\mathbf{y}} = (\overline{y}_0, \overline{y}_1, \cdots, \overline{y}_M)^T$. To indicate how this differs from the error at $t = T$, (7.44) is plotted in Figure 7.5 for the logistic equation example. As expected, e_∞ is larger but its dependence on M is still $O(k)$.

7.3.3 Additional Difference Methods

The steps used to derive the Euler method can be employed to obtain a host of other finite difference approximations. The point in the derivation that separates one method from another is Step 3, where one makes a choice for the difference formula. A few possibilities are given in Table 7.1. It is interesting to see what sort of numerical methods can be derived using these expressions, and a couple of the possibilities are discussed below.

Backward Euler

If one uses the backward difference formula in Table 7.1, then in place of (7.26), we get

$$y'(t_j) = \frac{y(t_j) - y(t_{j-1})}{k} + \tau_j, \tag{7.45}$$

where

$$\tau_j = \frac{k}{2} y''(\eta_j). \tag{7.46}$$

Introducing this into (7.29), we obtain

$$y(t_j) - y(t_{j-1}) + k\tau_j = kf(t_j, y(t_j)). \tag{7.47}$$

Dropping the truncation error τ_j, the resulting finite difference approximation is

$$y_j = y_{j-1} + kf(t_j, y_j), \quad \text{for} \ \ j = 1, 2, \ldots, M. \tag{7.48}$$

From the initial condition (7.22) we have that the starting value is

$$y_0 = \alpha. \tag{7.49}$$

The difference equation in (7.48) is the *backward Euler method*. It has the same order of truncation error as the Euler method. However, because of the $f(t_j, y_j)$ term this method is implicit. This is both good and bad. It is good because it helps make the method A-stable (see below). However, it is bad because it can make finding y_j computationally difficult. Unless the problem is simple enough that the difference equation can be solved by hand, it is necessary to use something like Newton's method to solve (7.48), and this must be done at each time step.

As for stability (Step 5), for the radioactive decay equation (7.37) one finds that (7.48) reduces to (7.39), with the amplification factor

$$\kappa = \frac{1}{1 + rk}\,.$$

Since $|\kappa| < 1$, then from Theorem 7.1, this method is A-stable.

Example

A numerical comparison between backward Euler and some of the other methods we consider is made in Section 7.4 (see Figure 7.6). The objective of this example is to explain the difference between explicit and implicit methods, and this is done by solving

$$\frac{dy}{dt} = 6y(1 - y^3),$$

where $y(0) = 2$. Using Euler's method, which is explicit, the finite difference equation is

$$y_{j+1} = y_j + 6ky_j(1 - y_j^3), \quad \text{for } \ j = 0, 1, 2, \ldots,$$

where $y_0 = 2$. Assuming $k = \frac{1}{3}$, then it is a simple matter to evaluate the above formula to find that $y_1 = -26$. In comparison, using the backward Euler method (7.48), the finite difference equation is

$$y_{j+1} = y_j + 6ky_{j+1}(1 - y_{j+1}^3), \quad \text{for } \ j = 0, 1, 2, \ldots, \tag{7.50}$$

where $y_0 = 2$. Assuming $k = \frac{1}{3}$, then from the above formula we have that

$$y_1 = 2 + 2y_1(1 - y_1^3).$$

It is necessary to solve this nonlinear equation to determine y_1. Consequently, a computer program that uses the backward Euler method will have to include a nonlinear equation solver, like Newton's method, to compute y_{j+1} from (7.50). This is true for all implicit methods. As will be explored in some of the exercises, the stopping error used for the solver has the potential to affect the accuracy of the computed solution. ■

The difference between an explicit and implicit method made in the above example is important enough that it should be discussed in more detail. In many applications it is necessary to solve large systems of nonlinear differential equations. In such cases, implicit methods are avoided if possible because using a solver like Newton's method is computationally expensive. The one advantage implicit methods have is that they usually have better stability properties, which means you can use larger step sizes. This is only partially true because Newton's method requires an initial guess that is close to the solution, and this can limit the step size used by an implicit method. To use an explicit method for these types of problems, the usual choice is to use an adaptive time step. What this means is that the size of the time step is adjusted based on the properties of the solution. This allows for small steps if the solution is undergoing a rapid change, and larger step sizes in regions where the solution is slowing varying. The trick is to have ways to determine when this is happening, and those interested in learning about this should consult Lambert [1991] and Griffiths and Higham [2010]. Nevertheless, there are problems where implicit methods do play an important role, and they are referred to as being "stiff." The key tool for these types of problems is what are called BDF (backward difference formula) methods, and backward Euler is an example of such a method. More information about stiff problems can be found in Hairer and Wanner [2002]. A key component of most stiff solvers is a way to avoid the direct use of Newton's method. One way to do this is to use what are called Newton-Krylov methods, and a review of the recent work on this can be found in Knoll and Keyes [2004] and Loffeld and Tokman [2013].

Leapfrog Method

It is natural to expect that a more accurate approximation of the derivative will improve the accuracy of the computed solution. In looking over Table 7.1, the centered difference formula would appear to be a good choice for such an improvement because it has quadratic error (versus linear for the first two formulas listed). Introducing this into (7.21) we obtain

$$y(t_{j+1}) - y(t_{j-1}) + 2k\tau_j = 2kf(t_j, y(t_j)), \tag{7.51}$$

where $\tau_j = O(k^2)$. Dropping the $2k\tau_j$ term, the resulting finite difference approximation is

$$y_{j+1} = y_{j-1} + 2kf(t_j, y_j), \quad \text{for} \;\; j = 1, 2, \ldots, M-1. \tag{7.52}$$

This is known as the leapfrog, or explicit midpoint, method. Because this equation uses information from two previous time steps it is an example of a two-step method. In contrast, both Euler methods use information from a single time step back, so they are one-step methods. What this means is that the initial condition (7.22) is not enough information to get leapfrog started, because we also need y_1. This is a relatively minor inconvenience compared to the problem this method has with stability. To explain, applying (7.52) to the radioactive decay equation (7.37) yields $y_{j+1} = y_{j-1} - 2rky_j$. This second-order difference equation can be solved by assuming a solution of the form $y_j = s^j$. By doing this, it is found that the general solution has the form

$y_j = \alpha_0 s_+^j + \alpha_1 s_-^j$, where $s_\pm = -kr \pm \sqrt{1+k^2r^2}$ and α_0, α_1 are arbitrary constants. Because $|s_-| > 1$, it is impossible to find a step size k to satisfy the stability condition. Therefore, the leapfrog method is unstable.

7.3.4 Extensions

The method developed here is easily applied to almost any differential equation. Basically, the procedure involves evaluating the differential equation at a generic grid point t_j, picking one or more formulas from Table 7.1, and then writing down the resulting finite difference equation. An illustration of the procedure is given in the next example.

Example

The Duffing equation is the nonlinear second-order differential equation

$$y'' + y + y^3 = \cos(\omega t).$$

Evaluating this at t_j gives us $y''(t_j) + y(t_j) + y^3(t_j) = \cos(\omega t_j)$. There is only one choice for the second derivative in Table 7.1, and using it we obtain

$$\frac{y_{j+1} - 2y_j + y_{j-1}}{k^2} + y_j + y_j^3 = \cos(\omega t_j).$$

The truncation error is $O(k^2)$, and so this is a consistent approximation of the original differential equation. ■

So the method provides an easy way to obtain a *consistent approximation* of an IVP. By consistent it is meant that the error in the approximation goes to zero as the step size k approaches zero. However, as we saw with the leapfrog method, consistently is not enough to guarantee that the method will work. There is an additional requirement, which is that the method is also *stable.* Stability theory for approximating a problem involving something like the Duffing equation is beyond the scope of this text, and those interested in learning about this should consult Stuart and Humphries [1998] or Ascher and Petzold [1998].

7.4 IVP Methods Using Numerical Integration

Another approach to deriving a finite difference approximation of an IVP is to integrate the differential equation and then use a numerical integration rule. This is a very useful idea that is best explained by working through an example. To get started, a time grid must be introduced, and so Step 1 is the same as before. However, Step 2 and Step 3 differ from what we did earlier.

Step 2. Integrate the differential equation between two time points. We will take t_j and t_{j+1}, and so from (7.21) we have

$$\int_{t_j}^{t_{j+1}} \frac{dy}{dt} dt = \int_{t_j}^{t_{j+1}} f(t, y(t))dt. \tag{7.53}$$

Using the Fundamental Theorem of Calculus we obtain

$$y(t_{j+1}) - y(t_j) = \int_{t_j}^{t_{j+1}} f(t, y(t))dt. \tag{7.54}$$

Step 3. Replace the integral in Step 2 with a finite difference approximation. There are numerous choices, and they produce different numerical procedures. A few possibilities are listed in Table 7.3. We will use the trapezoidal rule, and introducing this into (7.54) yields

$$y(t_{j+1}) - y(t_j) = \frac{k}{2}\left[f(t_{j+1}, y(t_{j+1})) + f(t_j, y(t_j))\right] + O(k^3). \tag{7.55}$$

Step 4. Drop the big-O term. After dropping the $O(k^3)$ term in (7.55) the resulting equation is

$$y_{j+1} = y_j + \frac{k}{2}(f_{j+1} + f_j), \quad \text{for } \ j = 0, 1, 2, \ldots, M-1, \tag{7.56}$$

Rule	**Integration Formula**
Right Box	$\int_{t_j}^{t_{j+1}} f(x)dx = kf(t_{j+1}) + O(k^2)$
Left Box	$\int_{t_j}^{t_{i+1}} f(x)dx = kf(t_j) + O(k^2)$
Midpoint	$\int_{t_{j-1}}^{t_{j+1}} f(x)dx = 2kf(t_j) + \frac{k^3}{3}f''(\eta_i)$
Trapezoidal	$\int_{t_j}^{t_{j+1}} f(x)dx = \frac{k}{2}(f(t_j) + f(t_{j+1})) - \frac{k^3}{12}f''(\eta_j)$
Simpson	$\int_{t_{j-1}}^{t_{j+1}} f(x)dx = \frac{k}{3}(f(t_{j+1}) + 4f(t_j) + f(t_{j-1})) - \frac{k^5}{90}f''''(\eta_j)$

Table 7.3 Numerical integration formulas. The points $t_1, t_2, t_3, \ldots$ are equally spaced with step size $k = t_{j+1} - t_j$. The point η_j is located within the interval of integration.

where $f_j = f(t_j, y_j)$. From the initial condition (7.22) we have that the starting value is

$$y_0 = \alpha. \tag{7.57}$$

The finite difference equation (7.56) is known as the *trapezoidal method* for solving (7.21). Because of the f_{j+1} term this method is implicit, and it is not hard to show that its amplification factor is

$$\kappa = \frac{1 - rk/2}{1 + rk/2}.$$

Since $|\kappa| < 1$, it follows from Theorem 7.1 that the method is A-stable. To determine the truncation error for the method note that in (7.55) the error at each time step is $O(k^3)$. In taking M time steps to reach $t = T$ the resulting error is therefore $M \times O(k^3) = O(k^2)$. In other words, the truncation error is $\tau_j = O(k^2)$.

One of the attractive features of the quadrature approach is that it involves multiple decision points that can be varied to produce different numerical methods. For example, the integration interval can be changed to, say, $t_{j-1} \leq t \leq t_{j+1}$ and then Simpson's rule used on the resulting integral. Another option is to not use a quadrature rule but instead replace the function f in the integral in (7.54) with an approximation that can be integrated exactly. Ideas such as this are explored more fully in Holmes [2007].

Example

We have derived several methods for solving IVPs, including the Euler, backward Euler, leapfrog, and trapezoidal methods. It is worth taking them out for a test drive to see how they compare, and the logistic equation (7.6) is a good candidate for this. The equation that is solved is

$$\frac{dy}{dt} = 10y(1 - y), \quad \text{for } \; 0 < t, \tag{7.58}$$

where $y(0) = 0.1$. As before, we take $T = 1$, so the time points are determined from the expression $t_j = jk$, for $j = 0, 1, 2, \ldots, M$ and $k = 1/M$. Because $f(t, y) = 10y(1 - y)$ our methods reduce to the finite difference equations listed below:

$$\begin{aligned}
\text{Euler:} &\quad y_{j+1} = y_j + 10ky_j(1 - y_j), \\
\text{Backward Euler:} &\quad y_{j+1} = y_j + 10ky_{j+1}(1 - y_{j+1}), \\
\text{Leapfrog:} &\quad y_{j+1} = y_{j-1} + 20ky_j(1 - y_j), \\
\text{Trapezoidal:} &\quad y_{j+1} = y_j + 5k\left[y_j(1 - y_j) + y_{j+1}(1 - y_{j+1})\right].
\end{aligned}$$

The initial condition is $y_0 = 0.1$, and for the leapfrog method it is assumed that $y_1 = y(k)$ (i.e., the exact value at $t = t_1$ is used). Just how well these four expressions do is shown in Figure 7.6 for the case $M = 10$. The first thing one notices is just how badly the leapfrog method does (it had to be given its own graph because it behaves so badly). This is not unexpected, because we know that the method is not A-stable. The other three solution curves also behave as expected. In particular, the two Euler methods are not as accurate as the trapezoidal method and are approximately equal in how far each differs from the exact solution. To quantify just how accurately each method does in solving the problem, in Figure 7.7 the error is plotted as a function of the number of grid points used to reach T. As predicted, all decrease according to their respective truncation errors. Specifically, the trapezoidal method decreases as $O(k^2)$ and the two Euler methods as $O(k)$. ■

7.5 Runge–Kutta Methods

An extraordinarily successful family of numerical approximations for IVPs comes under the general classification of Runge–Kutta (RK) methods. The derivation is based on the following question: is it possible to determine an

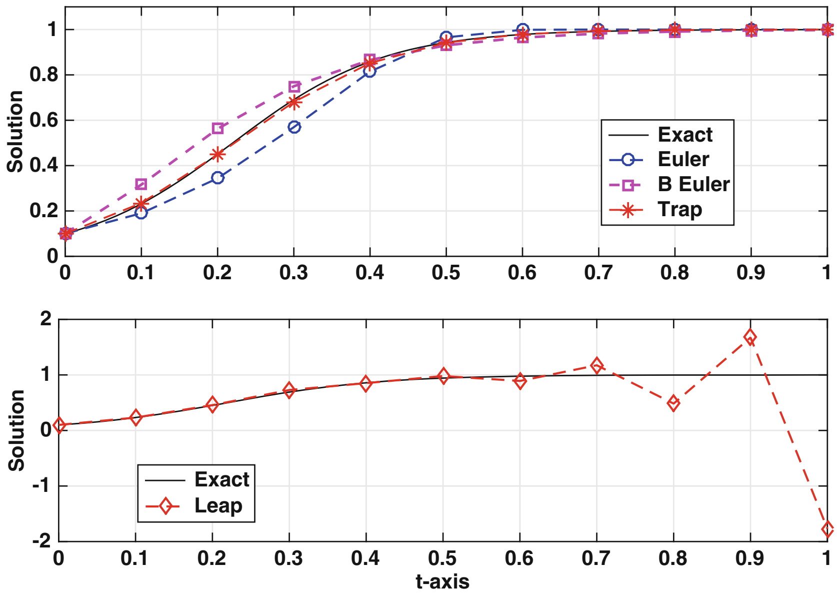


Figure 7.6 Solution of the logistic equation (7.58) using different numerical schemes. The leapfrog method is shown in the lower plot, and the two Euler schemes and the trapezoidal method are in the upper graph.

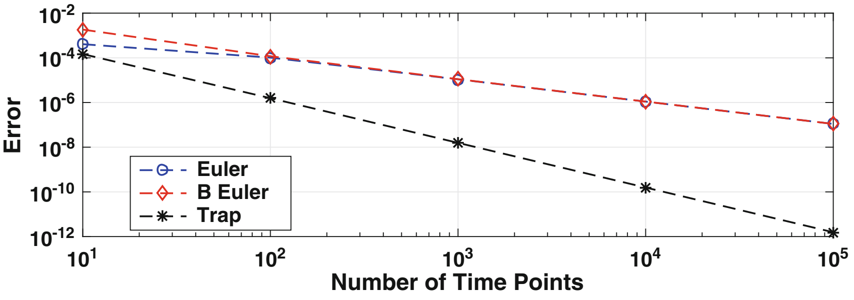


Figure 7.7 Error at $t = 1$ as a function of the number of time steps used to solve the logistic equation (7.58). Each curve decreases as $O(k^n)$, where n is determined from the truncation error for the method.

explicit method for finding y_{j+1} that only uses the value of the solution at t_j and has a predetermined truncation error. The secret in getting this to work is making a good guess as to what such a formula might look like.

7.5.1 RK2

To demonstrate how RK methods are derived, the best one-step explicit method we have so far is Euler's method, and this has a truncation error of $O(k)$. The RK question is, can we find an explicit one-step method that is $O(k^2)$? We have been able to derive an implicit scheme with this error, and this is the trapezoidal method (7.56). The reason it is implicit is the term $f(t_{j+1}, y_{j+1})$. Suppose we experiment a little and use Euler's method to approximate y_{j+1} in this term with $y_j + kf_j$. The resulting finite difference approximation is

$$y_{j+1} = y_j + \frac{k}{2}[f(t_j, y_j) + f(t_j + k, y_j + kf_j)]. \tag{7.59}$$

This is explicit but it is not clear whether it has the desired truncation error. However, it is useful because it provides insight into what a $O(k^2)$ explicit might look like.

Based on (7.59), the Runge–Kutta assumption for a $O(k^2)$ method is that the difference equation has the form

$$y_{j+1} = y_j + k[af(t_j, y_j) + bf(t_j + \alpha k, y_j + \beta kf_j)]. \tag{7.60}$$

The constants a, b, α, and β are determined from the requirement that the method has a $O(k^2)$ truncation error. Finding their values is fairly straightforward but for most RK methods this can be messy and rather tedious. To explain, the truncation error is determined by how well the difference equation approximates the original differential equation. To determine this, the exact solution $y(t)$ is substituted into (7.60) and then everything is expanded based on the assumption that k is small. For example, y_{j+1} is replace with $y(t_j + k)$, and then one expands to find that

$$\begin{aligned} y(t_j + k) &= y(t_j) + ky'(t_j) + \frac{1}{2}k^2 y''(t_j) + \cdots \\ &= y + kf + \frac{1}{2}k^2(f_t + f_y f) + O(k^3). \end{aligned} \tag{7.61}$$

The functions y and f in the last expression are evaluated at t_j. Also note that the last step used the fact that $y' = f$ and $y'' = f_t + f_y f$. In a similar manner one finds that the right-hand side of (7.60) becomes

$$\begin{aligned} & y + k[af + bf(t_j + \alpha k, y + \beta kf)] \\ &\qquad = y + kaf + kb\,(f + \alpha k f_t + \beta k f f_y) + \cdots \\ &\qquad = y + k(a+b)f + k^2 b(\alpha f_t + \beta f f_y) + O(k^3). \end{aligned} \tag{7.62}$$

Equating the last expression with (7.61) one concludes that

$$\begin{aligned} a + b &= 1, \\ 2\alpha b &= 1, \\ 2\beta b &= 1. \end{aligned} \tag{7.63}$$

These three equations are called the *order conditions* for the RK2 methods, and interestingly, the solution is not unique. Some example solutions for the order conditions are listed below.

1. $a = b = \frac{1}{2}$, $\alpha = \beta = 1$:

$$y_{j+1} = y_j + \frac{k}{2}[f(t_j, y_j) + f(t_{j+1}, y_j + kf_j)]. \tag{7.64}$$

This is known as *Heun's method.* It is also the method, given in (7.59), that we derived by combining the trapezoidal and Euler methods.

2. $a = 0$, $b = 1$, $\alpha = \beta = \frac{1}{2}$:

$$y_{j+1} = y_j + kf\left(t_j + \frac{k}{2}, y_j + \frac{k}{2}f_j\right). \tag{7.65}$$

This is known as the midpoint method.

3. $a = \frac{1}{4}$, $b = \frac{3}{4}$, $\alpha = \beta = \frac{2}{3}$:

$$y_{j+1} = y_j + \frac{1}{4}k\left[f(t_j, y_j) + 3f\left(t_j + \frac{2}{3}k, y_j + \frac{2}{3}kf_j\right)\right]. \tag{7.66}$$

This choice has a slightly better error coefficient for certain equations than the other RK2 methods. Why this happens is explained in Exercise 7.11.

Whichever solution of the order conditions is selected, the truncation error is $O(k^2)$. The reason is because (7.61) and (7.62) both hold to terms that are $O(k^3)$, and as explained in Section 7.4, this means that the truncation error for the method is $O(k^2)$.

Example

As usual, we will test our new methods using the logistic equation. Specifically, we solve

$$\frac{dy}{dt} = 10y(1-y), \quad \text{for } \; 0 < t, \tag{7.67}$$

where $y(0) = 0.1$. The error in the computed solution at $t = 1$ is shown in Figure 7.8 as a function of the number of grid points used to reach $t = 1$. The curves are for Heun's method given in (7.64), the method in (7.66), and the trapezoidal method. These curves are parallel because all three methods are $O(k^2)$. Also, even though the trapezoidal method does slightly better than the two RK2 methods, it requires more work to compute. ■

7.5.2 RK4

The one method from the Runge–Kutta family that deserves special attention is RK4. This is used in so many computer codes that it has become the

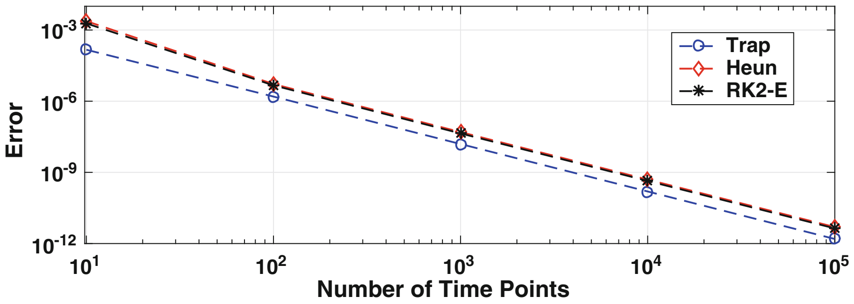


Figure 7.8 Error at $t = 1$ as a function of the number of time steps used to solve the logistic equation (7.67). The RK2-E curve comes from (7.66), and Heun comes from (7.64).

workhorse of IVP solvers. The derivation of RK4 uses a generalization of the assumption in (7.60) and it is that

$$y_{j+1} = y_j + c_1k_1 + c_2k_2 + c_3k_3 + c_4k_4, \tag{7.68}$$

where

$$k_1 = kf(t_j, y_j), \tag{7.69}$$
$$k_2 = kf(t_j + \alpha_2 k, y_j + \beta_{21}k_1), \tag{7.70}$$
$$k_3 = kf(t_j + \alpha_3 k, y_j + \beta_{31}k_1 + \beta_{32}k_2), \tag{7.71}$$
$$k_4 = kf(t_j + k, y_j + \beta_{41}k_1 + \beta_{42}k_2 + \beta_{43}k_3). \tag{7.72}$$

There are 12 constants in the above assumption, and after plugging the exact solution into (7.68) one finds 11 order conditions. For the mildly curious, these are listed in Section 7.7.1. Any choice of the constants that satisfies the order conditions will produce a method with a truncation error that is $O(k^4)$.

Based on the above discussion, one of the parameters in (7.68)–(7.72) is arbitrary. There is a standard choice for this constant, and to explain where it comes from suppose the differential equation is $y' = f(t)$. Integrating this as in (7.54) and then using Simpson's rule yields

$$\begin{aligned} y(t_{j+1}) - y(t_j) &= \frac{k}{6}\left[f(t_j) + 4f(t_j + 0.5k) + f(t_{j+1})\right] + O(k^5) \\ &= \frac{k}{6}\left[f(t_j) + 2f\left(t_j + \frac{k}{2}\right) + 2f\left(t_j + \frac{k}{2}\right) + f(t_{j+1})\right] + O(k^5). \end{aligned}$$

If we want our RK method to reproduce this result then we should take $c_1 = \frac{1}{6}$. The resulting RK method is

$$y_{j+1} = y_j + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4), \tag{7.73}$$

where

$$\begin{aligned} k_1 &= kf(t_j, y_j), \\ k_2 &= kf(t_j + \frac{k}{2}, y_j + \frac{1}{2}k_1), \\ k_3 &= kf(t_j + \frac{k}{2}, y_j + \frac{1}{2}k_2), \\ k_4 &= kf(t_{j+1}, y_j + k_3). \end{aligned} \tag{7.74}$$

This is often referred to as *the* RK4 method, or as the classic RK4 method. However, as is evident in the above discussion, there are other RK methods with a $O(k^4)$ truncation error. It is worth noting that the one in (7.73) does not have the best error out of all the possible RK4 methods, but it does have the significant property of providing a relative simple formula. A comparison between the classic RK4 and a minimum error version is given in Section 7.7.1.

Example

For the logistic example considered earlier, the RK4 formulas given above are

$$\begin{aligned} k_1 &= 10ky_j(1-y_j), \\ k_2 &= 10k\left(y_j + \frac{1}{2}k_1\right)\left(1 - y_j - \frac{1}{2}k_1\right), \\ k_3 &= 10k\left(y_j + \frac{1}{2}k_2\right)\left(1 - y_j - \frac{1}{2}k_2\right), \\ k_4 &= 10k(y_j + k_3)(1 - y_j - k_3). \end{aligned}$$

The resulting numerical accuracy of the method is shown in Figure 7.9. RK4 is clearly superior to the trapezoidal method, to the point that it achieves an error on the order of machine ε far ahead of the others we have considered. Given this, you might wonder why the other methods are even discussed, much less used by anyone. Well, there are several reasons for considering other methods, and one relates to stability (which is discussed shortly). Another reason is that RK4 does not do well in preserving certain properties of the solution, and an example is energy. In problems where energy conservation is important, other methods, called symplectic methods, are usually used. An example of such a method is introduced in Section 7.6.3. ■

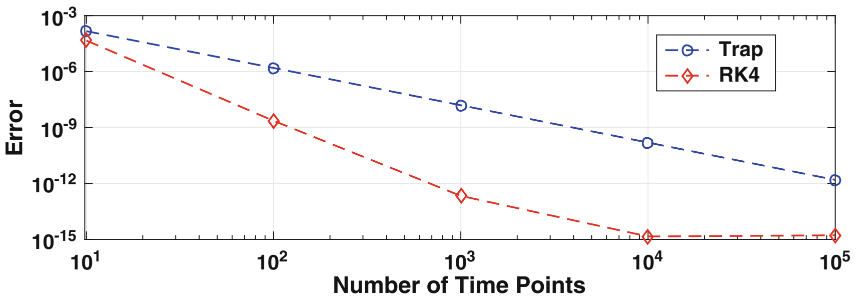


Figure 7.9 Error at $t = 1$ as a function of the number of time steps used to solve the logistic equation (7.67) using the RK4 and the trapezoidal methods.

7.5.3 Stability

The entire derivation of the RK methods centered on the truncation error. What is missing is the Step 5 question, which is whether the methods are A-stable. As it turns out, all explicit Runge–Kutta methods are conditionally A-stable. As an example, given the order conditions in (7.63), the RK2 method in (7.60) is conditionally A-stable and the requirement is $k < 2/r$. This is the same inequality we obtained for Euler's method, and so RK2 and Euler have the same stability requirement. As for RK4, the requirement is $k < \lambda/r$ where $\lambda \approx 2.785$. This means that the stability region for RK4 is slightly larger that it is for RK2 or Euler.

7.5.4 RK-n

The ideas developed here can be generalized to produce higher-order RK methods, although the complexity of the derivation can be enormous. For example, in celestial mechanics you occasionally see people use twelfth-order RK methods. Such a scheme is not easy to derive, because it results in 5972 order conditions, and, as occurred earlier, these form an underdetermined nonlinear system. This situation is further complicated by the somewhat unexpected problem that Runge–Kutta methods that are $O(k^p)$ for the scalar equation $y' = f(t, y)$ are not necessarily $O(k^p)$ for the system $\mathbf{y}'(t) = \mathbf{f}(t, \mathbf{y})$ if $p \geq 5$. In other words, to derive a higher-order Runge–Kutta method for systems you are not able to simply use a scalar equation and then convert the variables to vectors when you are done (this idea is used in the next section). Those interested in deriving higher-order methods, or in a more systematic derivation of RK4, should consult the texts by Butcher [2008] and Lambert [1991].

7.6 Solving Systems of IVPs

Most applications involving IVPs have multiple equations. In such cases the mathematical problem then has the form

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(t, \mathbf{y}), \quad \text{for} \;\; 0 < t, \tag{7.75}$$

where

$$\mathbf{y}(0) = \boldsymbol{\alpha}. \tag{7.76}$$

In the above, $\mathbf{y} = (y_1(t), y_2(t), \cdots, y_n(t))^T$, $\mathbf{f} = (f_1, f_2, \cdots, f_n)^T$, and $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \cdots, \alpha_n)^T$ are n-vectors. In component form, the IVP can be written as

$$y_i' = f_i(t, y_1, y_2, \cdots, y_n), \text{ for } i = 1, 2, \cdots, n,$$

where $y_i(0) = \alpha_i$.

7.6.1 Examples

Before discussing numerical solutions we begin with a couple of typical examples.

7.6.1.1 Law of Mass Action

An often occurring situation involves one or more species combining, or transforming, to form new or additional species. This is what happens when hydrogen and oxygen combine to form water. It also can be applied to a disease moving through a population. One example is the Kermack-McKendrick model for epidemics. This assumes the population can be separated into three groups. One is the population $S(t)$ of those susceptible to the disease, another is the population $I(t)$ that is ill (as well as infectious), and the third is the population $R(t)$ of individuals that have recovered. Using the law of mass action one can derive a model that accounts for the susceptible group getting sick, the subsequent increase in the ill population, and the eventual increase in the recovered population [Holmes, 2009]. The result is the following set of equations:

$$\begin{aligned}\frac{dS}{dt} &= -aSI,\\ \frac{dI}{dt} &= -bI + aSI,\\ \frac{dR}{dt} &= bI,\end{aligned}$$

where $S(0) = S_0$, $I(0) = I_0$, and $R(0) = R_0$. In the above equations a and b are rate constants. Given the three groups, and the letters used to designate them, this is an example of what is known as a SIR model in mathematical epidemiology.

7.6.1.2 Newton's Second Law

According to Newton's second, $F = ma$. Letting $y(t)$ designate position, then this law takes the form

$$m\frac{d^2y}{dt^2} = F(t, y, y'), \quad \text{for } 0 < t. \tag{7.77}$$

Assuming that the initial position and velocity are specified, then the initial conditions for this problem are

$$y(0) = \alpha \;\text{ and }\; y'(0) = \beta. \tag{7.78}$$

It is possible to write the problem as a first-order system by introducing the velocity $v = y'$. Using the original differential equation (7.77), we obtain the following:

$$\begin{aligned} y' &= v, \\ v' &= \frac{1}{m}F(t, y, v). \end{aligned}$$

Introducing the vector $\mathbf{y}(t)$, defined as

$$\mathbf{y} = \begin{pmatrix} y \\ v \end{pmatrix},$$

then the IVP can be written as

$$\mathbf{y}'(t) = \mathbf{f}(t, \mathbf{y}), \quad \text{for } \; 0 < t, \tag{7.79}$$

where the initial conditions (7.78) take the form

$$\mathbf{y}(0) = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}.$$

The function $\mathbf{f}(t, \mathbf{y})$ appearing in (7.79) is

$$\mathbf{f}(t, \mathbf{y}) = \begin{pmatrix} v \\ \frac{1}{m}F(t, y, v) \end{pmatrix}.$$

What is significant is that the change of variables has transformed the second-order problem for $y(t)$ into a first-order problem for $\mathbf{y}(t)$. Like the original, (7.79) is nonlinear if F depends nonlinearly on either y or v.

7.6.2 Simple Approach

Writing down a numerical method to solve (7.75) is easy because all of the formulas we have derived for single equations apply to the vector case. For example, the approximation $y'(t_j) \approx (y(t_{j+1}) - y(t_j))/k$ becomes

$$\mathbf{y}'(t_j) \approx \frac{\mathbf{y}(t_{j+1}) - \mathbf{y}(t_j)}{k},$$

and the error is still $O(k)$. Note that the error is now a vector, and stating that it is $O(k)$ means that each element of the error vector is $O(k)$.

What this means is that every method listed in Table C.4 in Appendix C works with vector functions. So, for example, the trapezoidal method becomes

$$\mathbf{y}_{j+1} = \mathbf{y}_j + \frac{k}{2}(\mathbf{f}_j + \mathbf{f}_{j+1}), \tag{7.80}$$

where $\mathbf{f}_j = \mathbf{f}(t_j, \mathbf{y}_j)$ and $\mathbf{f}_{j+1} = \mathbf{f}(t_{j+1}, \mathbf{y}_{j+1})$. Similarly, the RK4 method takes the form

$$\mathbf{y}_{j+1} = \mathbf{y}_j + \frac{1}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4), \tag{7.81}$$

where $\mathbf{k}_1 = k\mathbf{f}(t_j, \mathbf{y}_j)$, $\mathbf{k}_2 = k\mathbf{f}(t_j + \frac{k}{2}, \mathbf{y}_j + \frac{1}{2}\mathbf{k}_1)$, $\mathbf{k}_3 = k\mathbf{f}(t_j + \frac{k}{2}, \mathbf{y}_j + \frac{1}{2}\mathbf{k}_2)$, and $\mathbf{k}_4 = k\mathbf{f}(t_{j+1}, \mathbf{y}_j + \mathbf{k}_3)$. Moreover, the stated properties, like being A-stable, are still the same. This is where a comment needs to be made about implicit methods. For single equations, an implicit method is a little more difficult to use than an explicit method, but for systems of IVPs an implicit method can be a lot more difficult. The reason is that at each time step it is necessary to solve a large system of nonlinear equations, and this usually means using Newton's method. As explained in Section 3.10, this requires calculation of a Jacobian and solving multiple matrix equations at each time step. Because of this, most IVP solvers use explicit methods whenever possible.

Example

The equation for the angular deflection of a pendulum is

$$\ell \frac{d^2\theta}{dt^2} = -g\sin(\theta), \tag{7.82}$$

where the initial angle $\theta(0)$ and the initial angular velocity $\theta'(0)$ are assumed to be given. Also, ℓ is the length of the pendulum and g is the gravitational acceleration constant. Introducing the angular velocity $v = \theta'$ then the equation can be written as the first-order system

$$\theta' = v, \tag{7.83}$$

$$v' = -\alpha\sin(\theta), \tag{7.84}$$

where $\alpha = g/\ell$. In regard to (7.75), $\mathbf{y} = (\theta, v)^T$ and $\mathbf{f} = (v, -\alpha\sin(\theta))^T$. The numerical solution using the RK4 method in (7.81) is shown in Figure 7.10, with $\alpha = 1$, $\theta(0) = \pi/4$, and $\theta'(0) = 0$. Two time intervals are shown, one at the beginning and another much later. The pendulum is doing what pendulums do, which is oscillate back and forth in a periodic manner. However, as seen in the figure on the right, the amplitude in the numerical solution has decreased substantially. It is possible to prove that the solution of the

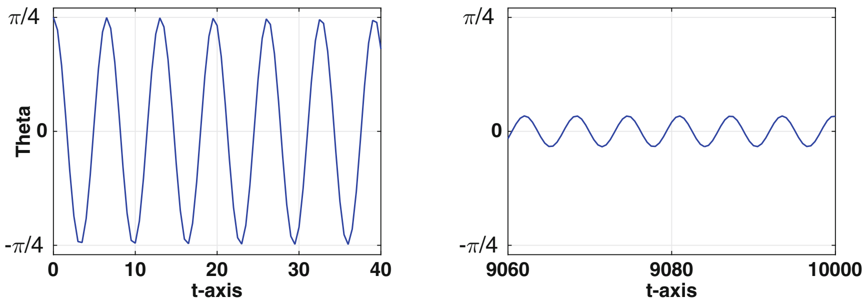


Figure 7.10 Solution of the pendulum equation (7.82) using the RK4 method (7.81). On the left is the computed solution at the beginning, and on the right is the computed solution for t close to 10,000.

pendulum problem does not decay, and the amplitude in the figure on the right should be $\pi/4$. The fact that the numerical solution decays is not particularly surprising. It is possible to reduce the decay by simply taking a smaller time step, but there is another way to do this, and this will be considered next. ■

7.6.3 Component Approach and Symplectic Methods

A consequence of a vector version of the RK4 method, as given in (7.81), is that every component of $\mathbf{y}$ is being approximated the same way. There are sometimes benefits of not doing this and to explain, consider the problem of solving $my'' = F(y)$. This can be written in component form as

$$y' = v, \tag{7.85}$$

$$v' = \frac{1}{m}F(y), \tag{7.86}$$

where we have introduced the velocity $v = y'$. If the vector version of the trapezoidal method is applied to this system, one obtains

$$\begin{aligned} y_{j+1} &= y_j + \frac{k}{2}(v_{j+1} + v_j), \\ v_{j+1} &= v_j + \frac{k}{2m}[F(y_{j+1}) + F(y_j)]. \end{aligned} \tag{7.87}$$

As is always the case with the trapezoidal method, the resulting equations are implicit. The question therefore arises as to whether it might be possible to tweak the above equations so they are explicit, similar to what we did to discover the RK2 methods. With this in mind, note that one of the culprits for the implicitness is the v_{j+1} term in (7.87). Can we find an approximation for this term that uses information at earlier time steps? One possibility is to use the Euler method (7.86), which gives us $v_{j+1} = v_j + \frac{k}{m}F(y_j)$. Introducing this into (7.87) yields

$$y_{j+1} = y_j + kv_j + \frac{1}{2m}k^2 F(y_j), \tag{7.88}$$

$$v_{j+1} = v_j + \frac{k}{2m}\left[F(y_{j+1}) + F(y_j)\right]. \tag{7.89}$$

Assuming we first use (7.88) to calculate y_{j+1} and then use (7.89) to find v_{j+1}, the procedure is explicit. It is known as the *velocity Verlet method* for solving (7.85) and (7.86), and we will try it using our earlier example.

Example

The solution of the pendulum problem in (7.82) using the velocity Verlet method is given in Figure 7.11. This shows the same two time intervals shown for RK4 in Figure 7.10 as well as uses the same time step k. Unlike the RK4 solution shown in Figure 7.10, the amplitude decay is not evident in the velocity Verlet solution. ■

The result in Figure 7.11 is surprising. It's possible to prove that the velocity Verlet method is $O(k^2)$, yet it has produced a better solution than the $O(k^4)$ RK4 method. Although it is always possible for a $O(k^2)$ method to produce a better answer than a $O(k^4)$ method on a particular problem, the reason here is more profound. What is happening is that velocity Verlet does a better job in approximating the energy in the system over a longer time interval than RK4. To investigate this, the equation for the energy can be obtained by multiplying (7.82) by the velocity θ' and integrating. From this, it is found that

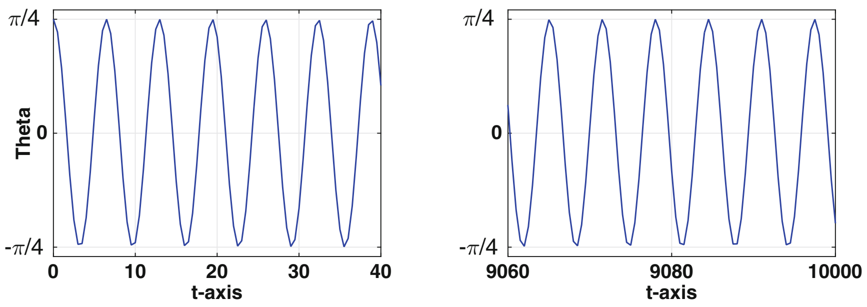


Figure 7.11 Solution of the pendulum equation (7.82) using the velocity Verlet method (7.88) and (7.89) over the same two time intervals shown in Figure 7.10.

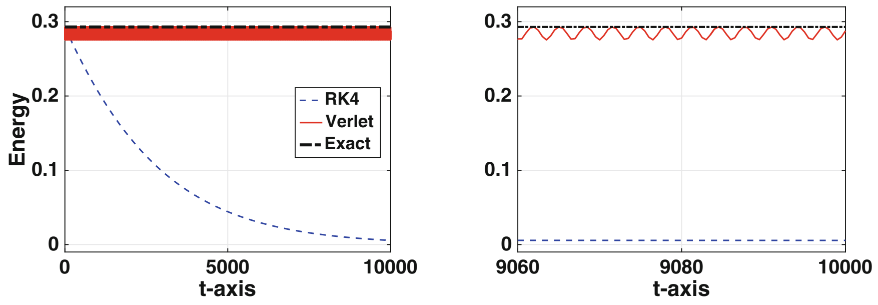


Figure 7.12 The energy (7.90) as computed using RK4 and the velocity Verlet method for the pendulum equation (7.82). On the left are the curves for $0 \leq t \leq$ 10,000, while on the right the curves are shown over a small time interval near $t =$ 10,000.

$$H(\theta, \theta') = \frac{1}{2}\ell(\theta')^2 + g(1 - \cos\theta) \tag{7.90}$$

is constant. The function H is a Hamiltonian (or, more precisely, a Hamiltonian per unit mass) for the pendulum. With the initial conditions $\theta(0) = \pi/4$ and $\theta'(0) = 0$, it then follows that $H(\theta, \theta') = g(1 - \sqrt{2}/2)$. The computed values for H, assuming $g = 1$, using the two numerical methods are plotted in Figure 7.12. For comparison, the exact value is also shown. The decay in the RK4 value is clearly seen, while the computed energy from velocity Verlet method oscillates, but the values stay very near the exact value. Note that these oscillations are the reason for the solid (red) bar in the left plot in the figure. The reason that velocity Verlet does so well is that it is an example of what is known as a *symplectic method*. However, even though velocity Verlet does well computing the energy, it does not do as well with the phase. For the example in Figure 7.12, using velocity Verlet the computed value of the period is about 6.47, while the exact value is about 6.53. Although the difference is small, given the large number of oscillations, the difference between the exact and computed value of the angular position $\theta(t)$ grows. This difference can be reduced, but not eliminated, by taking a smaller value for the step size k. More about this, and symplectic methods in general, can be found in Holmes [2007], Stuart and Humphries [1998], or Hairer et al. [2003].

7.7 Some Additional Questions and Ideas

1. Round-off error can be a problem for the finite difference formulas in Table 7.1 (see Figure 7.1). What do you do if you want an accurate numerical derivative for very small step sizes?

   Answer: Well, one option is to transform the formula, as discussed in

Section 7.2. Another way is to use a *complex Taylor series expansion* (CTSE), what is sometimes called the complex-step derivative approximation method. The standard example used for this is the centered difference formula

$$y'(t_j) = \frac{y(t_{j+1}) - y(t_{j-1})}{2k} + \frac{1}{6}k^2 y'''(t_j) + \cdots . \tag{7.91}$$

Although this has an error that is $O(k^2)$, as shown in Figure 7.1, it can have a problem when k is small. A way to avoid this is to use an imaginary step size. To explain, note, using Taylor's theorem, that

$$y(t+ik) = y(t) + iky'(t) - \frac{1}{2}k^2 y''(t_j) - \frac{1}{6}ik^2 y'''(t) + \cdots ,$$

where, as usual, $i = \sqrt{-1}$. Taking the imaginary part of this equation, and rearranging, we have that

$$y'(t) = \frac{Im[y(t+ik)]}{k} + \frac{1}{6}k^2 y'''(t) + \cdots . \tag{7.92}$$

This gives us a $O(k^2)$ approximation for the first derivative that does not have the round-off problems that arise with (7.91). To compare, the example for Figure 7.1 is redone in Figure 7.13 using (7.19) and the corresponding approximation coming from (7.92). The latter is

$$y'(t) \approx \frac{1}{k} Im\left[\sqrt{y(t+ik)}\right] . \tag{7.93}$$

The approximation in (7.93) is actually better than indicated in Figure 7.13, because for values of k smaller than 10^{-7}, the error is zero (using double precision). This very interesting idea does have limitations, such as how the function is defined when using a complex argument, and more discussion of this can be found in Martins et al. [2003] and Lai and Crassidis [2008].

2. The problem with numerical differentiation seen in Figure 7.1 did not seem to be an issue when solving IVPs, why not?

   Answer: The better IVP solvers do not require a small step size to produce an accurate solution, and because of this they avoid the numerical differentiation problem. As seen in Figure 7.7, to achieve an error of 10^{-9}, RK4 uses only 100 points and the trapezoidal method uses about 5000 points. Neither are close to the step sizes causing problems in Figure 7.1. As for the less accurate methods, like the two Euler methods, the number of time points needed to cause a problem is so large that most would not consider them viable options. As an example, for the Euler method to start to have trouble with numerical differentiation it is necessary to use 10^9 or more time points when solving the logistic equation (7.58). The solution

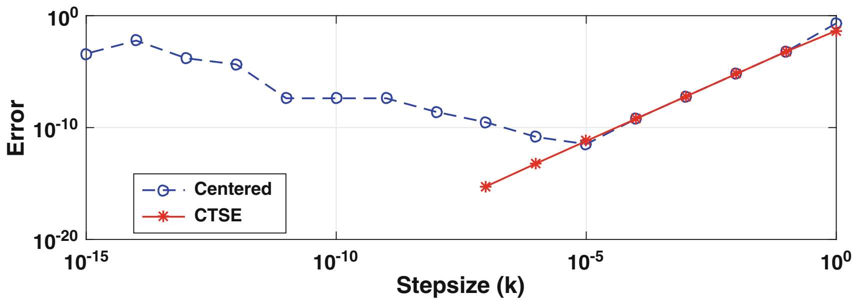


Figure 7.13 Error when using a numerical approximation of $y'(1)$, when $y(t) = \sqrt{t}$, using the centered difference (7.19), and the CTSE approximation is (7.93). Note that the CTSE error is zero for $k < 10^{-7}$.

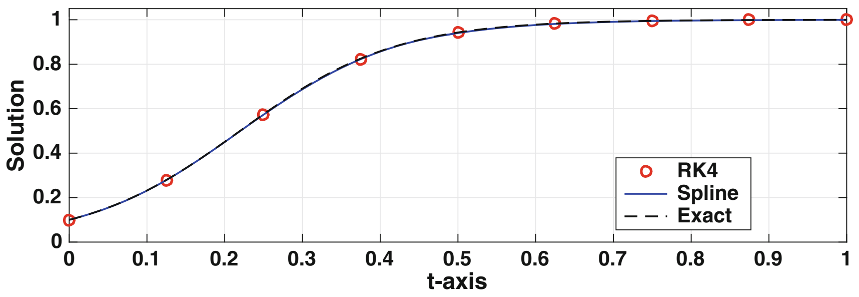


Figure 7.14 Solution of the logistic equation (7.33) using RK4. Also shown is the exact solution and the clamped cubic spline fitted to the RK4 values (these two curves are almost indistinguishable from each other).

accuracy in this case is no better than what would be obtained using 10^2 points with RK4 or about 10^3 points with the trapezoidal method.

3. It's nice that RK4 can find an accurate solution using just a few points scattered across the time interval, but I want a smooth curve. What do I do?

   Answer: The easy solution is to just use more points. However, a better solution is to use interpolation. It's possible in this case to use a clamped cubic spline because we know, or have a computed value for, y' at the endpoints. In particular, $y'(0) = f(0, \alpha)$ and $y'(T) = f(T, y_M)$. To illustrate how well this works, the solution of the logistic equation (7.33), with $y(0) = 0.1$, is shown in Figure 7.14. With this it is possible to obtain an accurate value for the solution anywhere in the time interval. An example of how this can be used in data fitting is explained in Section 9.5.

4. It is often stated that a physical system must satisfy causality, which, roughly speaking, means that the future does not affect the past. Doesn't a forward-difference approximation violate causality?

Answer: It does. In fact, all but two of the difference formulas in Table 7.1 violate causality. What is interesting is that the one method that is based on a causal approximation, which is backward Euler, is unconditionally stable. The non-causal approximations, in comparison, produce only conditionally stable methods. It is unclear what connections there might be between these two properties, but apparently there might be. For those who might be interested in computations and the connections with some of the basic laws of physics, the collection of articles in Zenil [2012] might be consulted.

7.7.1 RK4: Why Use Simpson's Rule?

The RK4 method was derived using Simpson's rule, and the question is why. To consider this, the general form of the RK4 assumption is

$$y_{j+1} = y_j + c_1k_1 + c_2k_2 + c_3k_3 + c_4k_4, \tag{7.94}$$

where

$$\begin{aligned} k_1 &= kf(t_j, y_j), \\ k_2 &= kf(t_j + \alpha_2 k, y_j + \beta_{21}k_1), \\ k_3 &= kf(t_j + \alpha_3 k, y_j + \beta_{31}k_1 + \beta_{32}k_2), \\ k_4 &= kf(t_j + k, y_j + \beta_{41}k_1 + \beta_{42}k_2 + \beta_{43}k_3). \end{aligned}$$

It is assumed that $0 \le \alpha_2 \le 1$ and $0 \le \alpha_3 \le 1$. The order conditions resulting from the above assumption are listed in Table 7.4. There are 11 equations and 12 unknowns, and the solutions are given in Ralston [1962]. Just so it's clear, every solution results in an IVP solver with an error that is $O(k^4)$.

To decide on which solution of the order equations to pick, we considered the problem of solving $y' = f(t)$. Applying (7.94) to this equation we get that

$$y_{j+1} = y_j + c_1kf(t_j) + c_2kf(t_j + \alpha_2 k) + c_3kf(t_j + \alpha_3 k) + c_4kf(t_j + k),$$

On the other hand, integrating the equation we get that

$$y(t_{j+1}) = y(t_j) + \int_{t_j}^{t_{j+1}} f(t)dt.$$

Combing these two equations, it must be that

$$\int_{t_j}^{t_{j+1}} f(t)dt \approx c_1kf(t_j) + c_2kf(t_j + \alpha_2 k) + c_3kf(t_j + \alpha_3 k) + c_4kf(t_j + k). \tag{7.95}$$

In other words, the solution of the order equations is associated with an integration rule. This is where Simpson's rule was used. The reason for picking Simpson is that its error is $O(k^5)$, and this is required to produce an IVP solver with a truncation error that is $O(k^4)$ (i.e., it is consistent with the error being sought for the RK4 method). However, is there a better choice? Asked a different way, are there choices for the c_i's and α_i's in the above expression that produce a better error?

The question asked about the c_i's and α_i's is the same question asked in the derivation of the Gaussian integration rules. The difference with (7.95) is that the endpoints are being used to determine the integration rule. This variation is known as Lobatto quadrature, and this is explained in Exercise 6.24. Using the result of Exercise 6.24(c), the Lobatto rule that produces the maximum precision has $c_1 = 1/12$. This gives a RK4 method of the form

$$y_{j+1} = y_j + \frac{1}{12}\left(k_1 + 5k_2 + 5k_3 + k_4\right), \tag{7.96}$$

$$\begin{aligned}
c_1 + c_2 + c_3 + c_4 &= 1\\
\beta_{21} &= \alpha_2\\
\beta_{31} + \beta_{32} &= \alpha_3\\
\beta_{41} + \beta_{42} + \beta_{43} &= 1\\
c_2\alpha_2 + c_3\alpha_3 + c_4 &= \frac{1}{2}\\
c_2\alpha_2^2 + c_3\alpha_3^2 + c_4 &= \frac{1}{3}\\
c_2\alpha_2^3 + c_3\alpha_3^3 + c_4 &= \frac{1}{4}\\
c_3\alpha_2\beta_{32} + c_4(\alpha_2\beta_{42} + \alpha_3\beta_{43}) &= \frac{1}{6}\\
c_3\alpha_2\alpha_3\beta_{32} + c_4(\alpha_2\beta_{42} + \alpha_3\beta_{43}) &= \frac{1}{8}\\
c_3\alpha_2^2\beta_{32} + c_4(\alpha_2^2\beta_{42} + \alpha_3^2\beta_{43}) &= \frac{1}{12}\\
c_4\alpha_2\beta_{32}\beta_{43} &= \frac{1}{24}
\end{aligned}$$

Table 7.4 Order conditions for a general RK4 method [Ralston, 1962].

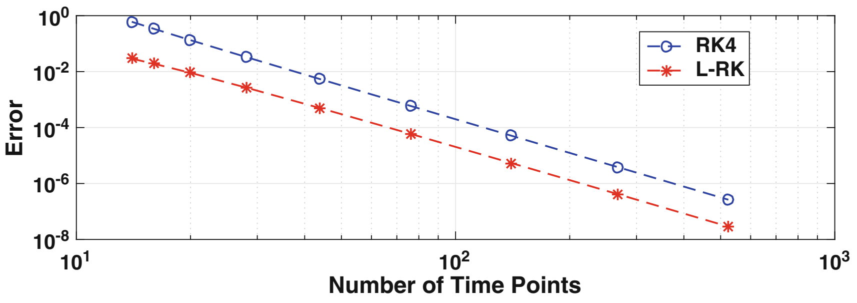


Figure 7.15 Comparison in the error for the classic RK4 and the Labatto RK4 method (7.96).

where

$$\alpha_2 = \frac{1}{2}\left(1-\frac{1}{\sqrt{5}}\right), \quad \beta_{21} = \frac{1}{2}\left(1-\frac{1}{\sqrt{5}}\right),$$
$$\alpha_3 = \frac{1}{2}\left(1+\frac{1}{\sqrt{5}}\right), \quad \beta_{31} = -\frac{1}{20}(5+3\sqrt{5}), \quad \beta_{32} = \frac{1}{4}(3+\sqrt{5}),$$
$$\beta_{41} = -\frac{1}{4}(1-5\sqrt{5}), \quad \beta_{42} = -\frac{1}{4}(5+3\sqrt{5}), \quad \beta_{43} = \frac{1}{2}(5-\sqrt{5}).$$

Note that as an integration rule the error for Labatto is $O(k^7)$, while for Simpson's rule it's $O(k^5)$.

Example

The solution of the IVP

$$y' = y - 2te^t \sin(t^2), \text{ for } 0 < t < 3,$$

where $y(0) = 1$, is $y = e^t \cos(t^2)$. The error in solving this problem using the classic RK4 method and the Labatto RK4 method is shown in Figure 7.15. Both curves show the expected $O(k^4)$ decrease, with the Labatto version giving a better error because of its more accurate quadrature approximation.

Exercises

7.1. This exercise concerns deriving finite difference approximations of derivatives.

(a) Derive a $O(k^2)$ approximation of $y'(t_j)$ that uses $y(t_j)$, $y(t_{j-1})$, and $y(t_{j+2})$.

x	0	1/3	2/3	1
u	1	0	-1	1

Table 7.5 Values for Exercise 7.4.

(b) Derive a $O(k^2)$ approximation of $y'(t_j)$ that uses $y(t_j)$, $y(t_{j+1})$, and $y(t_{j+2})$.
(c) Derive a $O(k^2)$ approximation of $y'(t_j)$ that uses $y(t_j)$, $y(t_{j-1})$, and $y(t_{j-2})$. Also, explain how your answer can also be obtained from one of the formulas in Table 7.1.

7.2. In the following, a claim is made about an approximation for the first derivative. Explain why the claim is wrong.
(a) A $O(k)$ approximation of $y'(t_j)$ is

$$y'(t_j) \approx \frac{2y(t_{j+1}) - y(t_j)}{k}.$$

(b) A $O(k^2)$ approximation of $y'(t_j)$ is

$$y'(t_j) \approx \frac{y(t_{j+2}) - y(t_{j-2})}{k}.$$

(c) A $O(k)$ approximation of $y'(t_j)$ is

$$y'(t_j) \approx \frac{y(t_{j+1}) - 3y(t_j) + 2y(t_{j-1})}{k}.$$

7.3. In this exercise, difference approximations are derived that make use of derivative information.
(a) Find a $O(k^2)$ approximation of $y''(t_{j+1})$ and uses $y'(t_{j+1})$ and $y'(t_{j-1})$.
(b) Find a $O(k^2)$ approximation of $y'(t_{j+1})$ and uses $y(t_j)$, $y(t_{j+1})$, and $y'(t_j)$.
(c) Find a $O(k^4)$ approximation of $y'(t_{j+1})$ and uses $y(t_{j-1})$, $y(t_{j+1})$, $y'(t_j)$, and $y'(t_{j-1})$.

7.4. For a linearly elastic material, the stress $T(x)$ is given as

$$T = E\frac{du}{dx},$$

where $u(x)$ is the displacement of the material and E is a positive constant known as the Young's modulus. You can assume the material is steel, so $E = 200$, and you should also assume that $u(0) = 0$ Using second-order approximations for $u'(x)$, and the data in Table 7.5, find T at $x = 0$, 1/3, 2/3, 1.

7.5. For a power law fluid, the shear stress $S(x)$ is given as

$$S = \alpha\left(\frac{dv}{dx}\right)^{\gamma},$$

where $v(x)$ is the shear velocity, and α and γ are positive constants. You can assume the fluid is ketchup at room temperature, so $\gamma = 1/4$ and $\alpha = 20$. Using second-order approximations for $v'(x)$, and the data in Table 7.6, find S at $x = 0$, $1/3$, $2/3$, 1.

7.6. The Bernoulli equation is

$$y' + y^3 = \frac{y}{1+t}.$$

(a) If the Euler method is used to solve this equation, what is the resulting finite difference equation?
(b) If the trapezoidal method is used to solve this equation, what is the resulting finite difference equation?
(c) If Heun's method is used to solve this equation, what is the resulting finite difference equation?
(d) If the RK4 method is used to solve this equation, what is the resulting finite difference equation?

7.7. The Michaelis-Menten equation is

$$\frac{dS}{dt} = -\frac{v_m S}{K_M + S},$$

where v_m and K_M are positive constants.
(a) If the Euler method is used to solve this equation, what is the resulting finite difference equation?
(b) If the trapezoidal method is used to solve this equation, what is the resulting finite difference equation?
(c) If Heun's method is used to solve this equation, what is the resulting finite difference equation?
(d) If the RK4 method is used to solve this equation, what is the resulting finite difference equation?

7.8. An IVP is solved using an explicit method using M time steps to reach $t = 1$. The value computed for y_M is given in Table 7.7 as a function of the time steps used in the calculation. Which explicit IVP solver was most likely used in this calculation? Make sure to explain why.

x	0	1/3	2/3	1
v	0	2	4	0

Table 7.6 Values for Exercise 7.5.

7.9. To solve $y' = f(t, y)$ one can use one of the following finite difference equations. Determine if the method is A-stable, conditionally A-stable, or unstable. If it is conditionally stable, make sure to state what the condition is.

(a) $y_{j+1} = y_j + \frac{k}{3}\left[f(t_j, y_j) + 2f(t_{j+1}, y_{j+1})\right]$

(b) $y_{j+1} = y_j + \frac{k}{4}\left[5f(t_j, y_j) - f(t_{j+1}, y_{j+1})\right]$

(c) $y_{j+1} = y_j + \frac{k}{5}\left[-2f(t_j, y_j) + 7f(t_{j+1}, y_{j+1})\right]$

7.10. The IVP for a mass-spring-dashpot system is

$$m\frac{d^2y}{dt^2} + c\frac{dy}{dt} + ky = F(t), \text{ for } t > 0$$

where $y(0) = a$ and $y'(0) = b$. In this problem m, c, k, a, b are given numbers and $F(t)$ is a given forcing function.

(a) Using finite difference approximations for y'' and y', derive a finite difference approximation for the differential equation that has truncation error that is $O(k^2)$.

(b) Convert the initial conditions so they apply to the finite difference approximation. Your approximation of $y'(0) = b$ must have a truncation error that is $O(k^2)$.

(c) Setting $v = y'$, find a first-order system for y and v.

(d) Write down a numerical method for solving the problem in part (c) that has a truncation error that is $O(k^2)$.

7.11.(a) The integration rule of the form

$$\int_{t_j}^{t_{j+1}} f(t)dt \approx a_1 f(t_j) + a_2 f(z),$$

which has the maximum precision is $a_1 = k/4$, $a_2 = 3k/4$, and $z = t_j + 2k/3$. Explain how this gives rise to the RK2 method given in (7.66).

M	y_M
16	1.062500000
32	1.031250000
64	1.015625000
128	1.007812500
256	1.003906250
512	1.001953125
1024	1.000976562

Table 7.7 Values for Exercise 7.8.

(b) What condition must be satisfied for the RK2 method given in (7.66) to be A-stable?

7.12. This problem explores a RK1 method. Assume the method has the form

$$y_{j+1} = y_j + akf(t_j, y_j).$$

Find the constant a using the same procedure used to derive the RK2 method. What is the truncation error?

7.13. This problem explores the stability condition for RK2.

(a) What does the RK2 method in (7.60) reduce to for the equation $y' = -ry$? Assume the constants satisfy the order conditions (7.63).

(b) Show that the amplification factor for RK2 can be written as $\kappa = 1 - z + \frac{1}{2}z^2$, where $z = rk$.

(c) Using the result from part (b), show that the RK2 method is conditionally A-stable and that the stability condition is $rk < 2$.

7.14. What condition must be satisfied for the (classic) RK4 method to be A-stable?

7.15. This problem concerns the differential equation $y' = f(t, y)$. In Figure 7.16, lines are shown which represent the absolute error as a function of the number of mesh points for several different methods.

(a) Which line, if any, should correspond to a RK2 method (e.g., Heun's method)?

(b) Which line, if any, should correspond to the trapezoidal method?

(c) Which line, if any, should correspond to Euler's method? Make sure to justify your answers. Also, a line may be used more than once.

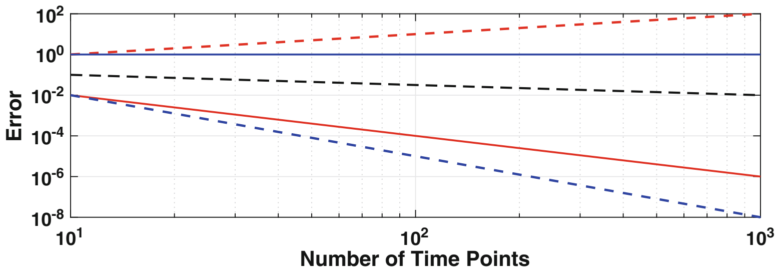


Figure 7.16 Graph used in Exercises 7.15 and 7.18.

7.16. To solve the differential equation $y' = f(t, y)$ one can use the finite difference equation

$$y_{j+1} = y_j + k[\theta f_j + (1 - \theta)f_{j+1}],$$

where θ is a number that satisfies $0 \leq \theta \leq 1$ (one first picks θ from this interval and then uses the above formula to calculate the solution).

(a) For what value(s) of θ is the method explicit, and for which value(s) of θ is it implicit?
(b) For what value(s) of θ is the method A-stable?
(c) For what value(s) of θ is this method second-order accurate? What is its order of accuracy for the other values of θ?

7.17. In deriving the trapezoidal method for solving the differential equation $y' = f(t, y)$ we integrated the equation over the interval $t_j \leq t \leq t_{j+1}$. In this problem you are to integrate the equation over the interval $t_{j-1} \leq t \leq t_{j+1}$.

(a) What numerical method is obtained if Simpson's rule is used on the resulting integral? What is the truncation error for this rule?
(b) What numerical method is obtained if the midpoint rule is used on the resulting integral? What is the truncation error for this rule?

7.18. This problem concerns the differential equation $y' = f(t, y)$ where $y(0) = a$.

(a) From the differentiation formula

$$y'(t) = \frac{y(t+2k) - y(t-2k)}{4k} + O(k^2)$$

derive a difference equation that can be used to numerically solve the differential equation.
(b) One of the curves in Figure 7.16 gives the error as a function of the number of grid points for the method you derived in part (a). Which one is it? Make sure to justify your answer.

7.19. Suppose you want to compute the solution of $y' = e^{-y} + t^5$ for $0 \leq t \leq 1$ using 100 time steps (so, $k = 0.01$). The methods to be tried are (i) the Euler method, (ii) the backward Euler method, (iii) the trapezoidal method, (iv) Heun, and (v) the RK4 method.

(a) Which one would you expect to complete the calculation the fastest? Why?
(b) Which one would you expect to be the most accurate? Why?
(c) If stability is a concern which method would be best? Why?

7.20. This problem concerns the Michaelis-Menten equation

$$\frac{dS}{dt} = -\frac{v_m S}{K_M + S},$$

where v_m and K_M are positive constants. The initial condition is $S(0) = S_0$. The biochemical applications of this equation, and how to solve it, were considered in Section 2.1.1 and in Exercise 2.17. In this exercise you are to solve this problem using the trapezoidal and RK4 methods.

(a) Solving this problem, one finds that the exact solution satisfies

$$K_M \ln(S/S_0) + S = S_0 - v_m t\,.$$

Verify that if S satisfies the above equation, then it is a solution of the IVP.

In the following questions, assume that $v_m = 0.76$ mM/min, $K_M = 16.7$ mM, and $S_0 = 100$ mM.

(b) Plot S for $0 \leq t \leq 500$. An easy way to do this is to rewrite the equation in part (a) as

$$t = \frac{1}{v_m}\left[S_0 - K_M \ln(S/S_0) - S\right]\,.$$

Picking values S_j, with $0 < S_j \leq S_0$, the above equation can be used to find the corresponding t_j. With this, the requested plot can be obtained by plotting the values for the (t_j, S_j)'s.

(c) On the same axes, plot the exact and the two numerical solutions for $0 \leq t \leq 500$ in the case of when $M = 20$.

(d) Redo (c) for $M = 5$, $M = 10$, and $M = 40$ (there should be one graph for each M).

(e) Plot the max error e_∞, defined in (7.44), as a function of M for each method, using $M = 10, 20, 40, 80$. The two curves should be in the same log-log plot.

(f) Compare the two methods based on your results from parts (c)–(e). This includes ease of use, speed of calculation, accuracy of results, and apparent stability.

7.21. This problem concerns the IVP involving the Bernoulli equation

$$y' + y^3 = \frac{y}{a+t}\,, \quad \text{for} \;\; t > 0,$$

where $y(0) = 1$. You are to solve this problem using the trapezoidal and RK4 methods.

(a) Verify that the exact solution is

$$y = \frac{a+t}{\sqrt{\beta + \frac{2}{3}(a+t)^3}}\,.$$

Also, determine the value of β from the initial condition.

(b) Assuming $a = 0.01$, on the same axes plot the exact and the two numerical solutions for $0 \leq t \leq 3$ in the case of when $M = 80$.

(c) Redo (b) for $M = 20$, $M = 40$, and $M = 160$ (there should be one graph for each M). If one of the methods is unstable you can exclude it from the plot (for that value of M) but make sure to state this in your write-up.

(d) Plot the max error e_∞, defined in (7.44), as a function of M for each method, using $M = 40, 80, 160, 320, 640$. The two curves should be in the same log-log plot.

(e) Compare the two methods based on your results from parts (b)–(d). This includes ease of use, speed of calculation, accuracy of results, and apparent stability.

7.22. The equation for the height $y(t)$ of a projectile is

$$\frac{d^2y}{dt^2} = -\frac{gR^2}{(R+y)^2}, \quad \text{for } 0 < t,$$

where g is the gravitational acceleration constant and R is the radius of the Earth. Also, $y(t)$ is the vertical distance from the surface of the Earth.

(a) Write this as a first-order system using the height y and the velocity $v = y'$.

(b) If $y(0) = 0$ and $v(0) = 100\,\text{m/s}$, find the time, to at least three significant digits, that it takes until the object hits the ground. Make sure to state what method you use to solve the IVP, why you picked the method, and how you used the solver to answer the question. In addition, you should plot y as a function of t, for $0 \le t \le T$, where T is your answer. It is advised that when you think you have found T that you double M to make sure that your solution does not change significantly.

(c) If $y(0) = 0$, then what does $v(0)$ have to be so the object stays airborne for 2 hours? In other words, if $T = 2\,\text{hr}$, then what does $v(0)$ have to be so that $y(T) = 0$. You should calculate $v(0)$ to six significant digits. In addition, you should plot y as a function of t, for $0 \le t \le T$. Make sure to state what method you use to solve the IVP, why you picked the method, and how you used the solver to answer the question.

7.23. Using polar coordinates in the orbital plane, the position of an object orbiting the sun is $(r(t), \theta(t))$. From Newton's laws, to find the position it is necessary to solve

$$r'' = \frac{\alpha^2}{r^3} - \frac{\mu}{r^2},$$

where $\alpha = \theta'(0)r(0)^2$ is constant, and μ is the gravitational parameter of the sun. Note that $\mu = 1.327 \times 10^{11}\,\text{km}^3/\text{sec}^2$.

(a) Write the above differential equation as a first-order system using the radial position r and radial velocity $v = r'$.

(b) The approximate values for Mars are: $r(0) = 2.244 \times 10^8\,\text{km}$, $r'(0) = 0$, and $\theta'(0) = 9.513 \times 10^{-8}\,\text{radians/sec}$. If one instead uses astronomical units (au), then $r(0) = 1.5\,\text{au}$ and $r'(0) = 0$. Also, measuring time in terms of a terrestrial year (where 1 ty = 365 days), then $\theta'(0) = 3\,\text{radians/ty}$. Explain why it is better to use the au, ty values rather than the km, sec values when computing the solution.

(c) Using the initial conditions from part (b), find $r(t)$, to at least three significant digits, after one terrestrial year. Make sure to state what method you used to solve the problem, why you picked that method, and how you know that the solution is correct to three significant digits.

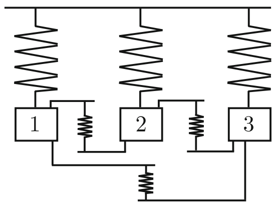


Figure 7.17 Three oscillators that are coupled by springs, as an example of the problem considered in Exercise 7.24.

(d) The angular coordinate of the object is determined using the following formula:

$$\theta(t) = \theta(0) + \alpha \int_0^t \frac{ds}{r^2(s)} .$$

Assuming that $\theta(0) = 0$, use your results from part (c) to find the angular coordinate for the object after one terrestrial year.

7.24. This exercise involves finding the solution of the problem for the coupled oscillators shown in Figure 7.17. The equation of motion in this case is $\mathbf{y}'' + \mathbf{K}\mathbf{y} = \mathbf{0}$, where

$$\mathbf{K} = \begin{pmatrix} 1 + k_{12} + k_{13} & -k_{12} & -k_{13} \\ -k_{12} & 1 + k_{12} + k_{23} & -k_{23} \\ -k_{13} & -k_{23} & 1 + k_{13} + k_{23} \end{pmatrix} .$$

In the above matrix, k_{ij} is the spring constant for the spring connecting the ith and jth oscillator, and it is positive (these are the three smaller springs shown in Figure 7.17). The initial conditions are $\mathbf{y}(0) = (-1, 0, 2)^T$ and $\mathbf{y}'(0) = (0, 0, 0)^T$.

(a) Assuming $\mathbf{y}(t) = (y_1(t), y_2(t), y_3(t))^T$, introduce the velocity $v_i(t) = y_i'(t)$. Letting $\mathbf{z}(t) = (y_1(t), v_1(t), y_2(t), v_2(t), y_3(t), v_3(t))^T$, show that the oscillator problem can be rewritten as $\mathbf{z}' = \mathbf{A}\mathbf{z}$. What is $\mathbf{z}(0)$?

(b) If Euler's method is used to solve the equation in part (a), what is the resulting finite difference equation? Your answer should be written in terms of $\mathbf{z}_i$ and $\mathbf{z}_{i+1}$.

(c) If the trapezoidal method is used to solve the equation in part (a), what is the resulting finite difference equation? Your answer should be written in terms of $\mathbf{z}_i$ and $\mathbf{z}_{i+1}$.

(d) If the RK4 method is used to solve the equation in part (a), what is the resulting finite difference equation? Your answer should be written in terms of $\mathbf{z}_i$ and $\mathbf{z}_{i+1}$.

(e) Using one of the methods from (b)–(d), on the same axis, plot y_1, y_2, y_3 for $0 \le t \le 10$. Assume that $k_{12} = k_{13} = k_{23} = 1/2$. Make sure to state what method you used to solve the problem and why you picked that method. Also, explain why you believe your solutions are accurate approximations of the exact solutions.

7.25. This exercise explores the differences between a symplectic method and a method which conserves energy exactly. Recall that the equation $my'' = F(y)$ can be written as the first-order system given in (7.85) and (7.86). A Hamiltonian for this system is

$$H(y,v) = \frac{1}{2}mv^2 - \int_0^y F(s)ds.$$

Also recall that H corresponds to the total energy of the system.

(a) It is assumed that the trapezoidal method given in (7.87) can be modified to produce a method that conserves energy. In particular, it is assumed that one can be found of the form

$$\begin{aligned} y_{j+1} &= y_j + \frac{k}{2}(v_{j+1} + v_j), \\ v_{j+1} &= v_j + \frac{k}{m}W(y_{j+1}, y_j). \end{aligned}$$

Using the above formula for the Hamiltonian, show that

$$\begin{aligned} &H(y_{j+1}, v_{j+1}) - H(y_j, v_j) \\ &\qquad = \frac{k}{2}(v_j + v_{j+1})\left(W(y_{j+1}, y_j) - \frac{1}{y_{j+1} - y_j}\int_{y_j}^{y_{j+1}} F(s)ds \right). \end{aligned}$$

From this, conclude that the method is conservative if

$$W(y_{j+1}, y_j) = \frac{1}{y_{j+1} - y_j}\int_{y_j}^{y_{j+1}} F(s)ds.$$

It is possible to show that the resulting method is second-order (you do not need to prove this).

(b) What finite difference equations do you obtain when the method in part (a) is applied to the pendulum system in (7.83), (7.84)? The resulting method is implicit. Show that finding θ_{j+1} and v_{j+1} reduces to solving $f(\theta_{j+1}) = 0$, where

$$f(\theta) = \theta - \theta_j - kv_j - \frac{1}{2}k^2\frac{\cos\theta - \cos\theta_j}{\theta - \theta_j}.$$

It is assumed here that $\ell = g = 1$.

(c) Show that in the case of when θ_{j+1} is close to θ_j, the solution of $f(\theta_{j+1}) = 0$ is close to $\theta_j + kv_j - \frac{1}{2}k^2\sin\theta_j$.

(d) Compute the solution using the method from part (b) for $0 \le t \le 10000$, using $k = 1/2$ and the initial conditions $\theta(0) = \pi/4$, $\theta'(0) = 0$. These are the same values used for Figures 7.10, 7.11, and 7.12. In your write-up, state what method you used to solve $f(\theta_{j+1}) = 0$, including the stopping condition and what you used for the initial guess(es) for θ_{j+1}.

(e) With the computed solution from part (d), plot θ for $0 \leq t \leq 40$ and for $9060 \leq t \leq 10000$. Also plot the energy $H(\theta, \theta')$ over the same time intervals. Comment on how these compare to the results obtained using RK4 and the velocity Verlet method. To help with this comparison, note that the energy values for velocity Verlet seen in Figure 7.12 oscillate between about 0.2755 and 0.2929.

(f) As stated in Section 7.6.3, using velocity Verlet the computed value of the period is about 6.47, while the exact value is about 6.53. Determine the period for your computed solution from part (d), and compare it to the velocity Verlet value.

Chapter 8
Optimization

8.1 Introduction

The problem central to this chapter is easy to state: given a function $F(\mathbf{v})$, find the point $\mathbf{v}_m \in \mathbb{R}^n$ where F achieves its minimum value. This can be written as

$$F(\mathbf{v}_m) = \min_{\mathbf{v}\in\mathbb{R}^n} F(\mathbf{v}). \tag{8.1}$$

The point $\mathbf{v}_m$ is called the minimizer or optimal solution. The function F is referred to the objective function, or the error function, or the cost function (depending on the application). Also, because the minimization is over all of $\mathbb{R}^n$, this is a problem in unconstrained optimization.

Some examples of such problems are below.

- $\mathbf{Av} = \mathbf{b}$: Given a linear equation $ax - b = 0$, it is possible to rewrite this as finding the minimum value of $F(x) = |ax - b|$ or of $F(x) = (ax - b)^2$. This is illustrated in Figure 8.1 in the case of when $a = 4$ and $b = 3$. In both cases, the minimum of $F(x)$ is unique and corresponds to the solution of the original equation. It is possible to use this same idea for $\mathbf{Av} = \mathbf{b}$. Recall that a vector norm $||\mathbf{w}||$ is never negative and it has the property that the one and only solution of $||\mathbf{w}|| = 0$ is $\mathbf{w} = \mathbf{0}$. Because of this, the linear system $\mathbf{Av} = \mathbf{b}$ can be rewritten in the form given in (8.1) by taking

$$F(\mathbf{v}) = ||\mathbf{Av} - \mathbf{b}||.$$

  It is also possible to use

$$F(\mathbf{v}) = ||\mathbf{Av} - \mathbf{b}||^2.$$

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_8

There is a considerable degree of flexibility here as one can consider the benefits, or drawbacks, of using the different norms that have been defined for vectors. Some of the ramifications of this will be considered later in this chapter.

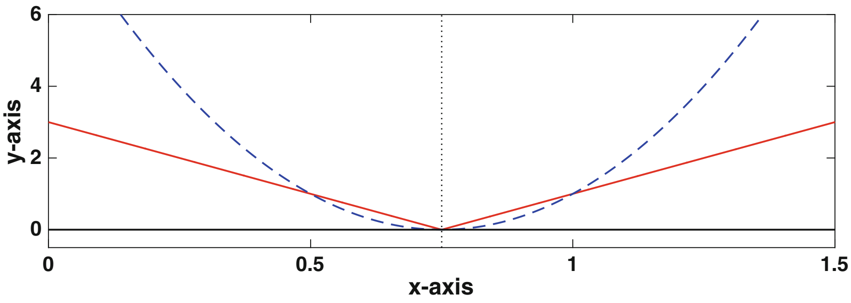


Figure 8.1 Plot of $F(x) = |4x - 3|$, the solid (red) line, and $F(x) = (4x - 3)^2$, the dashed (blue) line. Both functions have a minimum corresponding to the solution of $4x - 3 = 0$.

- In Section 3.10, Newton's method was used to solve

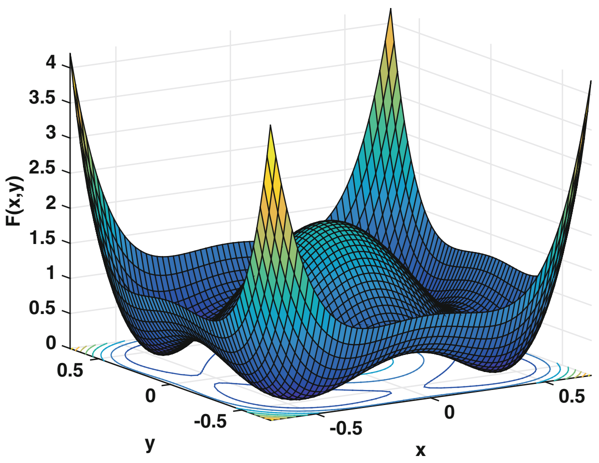
$$\begin{aligned} x^2 + 4y^2 &= 1, \\ 4x^2 + y^2 &= 1. \end{aligned}$$

Another way to solve it is to rewrite it as a minimization problem. This can be done, for example, by taking

$$F(x, y) = (x^2 + 4y^2 - 1)^2 + (4x^2 + y^2 - 1)^2. \tag{8.2}$$

With this, F is never negative, and it is only equal to zero when the original equations are satisfied. The surface plot of $F(x, y)$, and its associated contour plot, is given in Figure 8.2. The four local minimum points for $F(x, y)$ correspond to the four intersection points seen in Figure 3.4. Although the minima correspond exactly with the four solutions, the function $F(x, y)$ has multiple critical points. In particular $\nabla F = \mathbf{0}$ at the four minimum points, at the local maximum at $(x, y) = (0, 0)$, and at the four saddle points between the minimum points. What this means is that any method of finding the minima that involves trying to find where $\nabla F = \mathbf{0}$ will need to be able to have a way of deciding if the point is a minimum or not.

- Traveling Salesman Problem (TSP): A simple version of this is: given a set of points in the plane, find the path of minimum length that connects all of the points with a closed curve (see Figure 8.3). This problem has been studied intensely over the last few decades, and it has become the poster-child of hard problems. The reason is that it is an example of a NP-complete problem, which means that it is thought that the solution requires exponential time to solve. Specifically, to find the optimal solu-



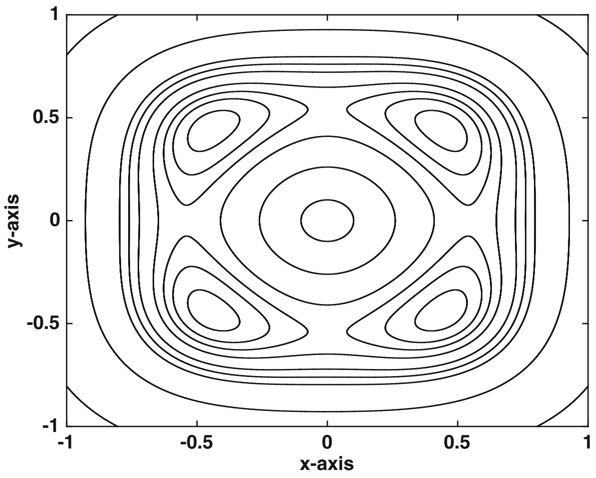


Figure 8.2 Top: Function given in (8.2). Bottom: Contour curves for the function.

tion you have to go through all possible paths, and the numbers of routes increases exponentially with the numbers of points. A more expanded discussion of the TSP can be found in Applegate et al. [2006].

- Regression: This refers to the problem of finding a function that best fits a set of data points. Two examples of this situation are shown in Figure 8.4. Unlike interpolation, the number of data points for regression is larger than the number of parameters in the function used to fit the data.

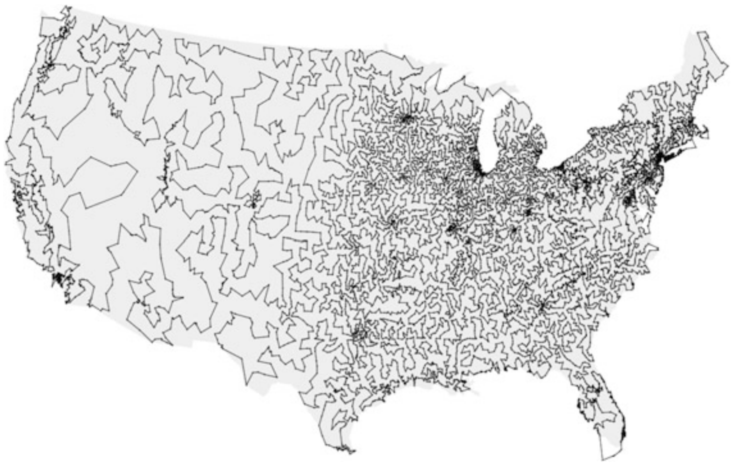


Figure 8.3 The shortest traveling salesman route going through all 13, 509 cities in the contiguous USA with a population of at least 500 [Applegate et al., 2006]

It is assumed in what follows that the function to be minimized is continuous, and it has one or more minima. For those who like their functions to have a unique minimum, additional assumptions are necessary. One often made is that the function is strictly convex. The function in Figure 8.2 does not have this property because it is possible to find a straight line connecting two points on the surface that intersects the surface other than at the two original points. This is not possible, for example, for the surface in Figure 8.14, and for this reason it is strictly convex. The assumption of convexity is not made here because in most applications you have no idea if this happens, and the methods considered do not require this to work.

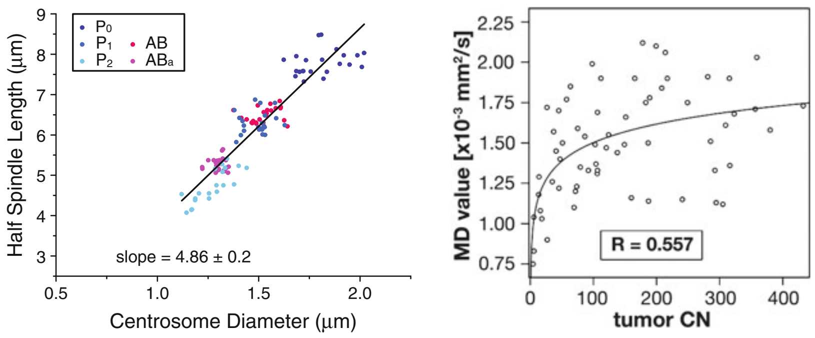


Figure 8.4 Examples of regression fits from Greenan et al. [2010] (L) and Stadlbauer et al. [2006] (R).

8.2 Regression: Introduction

In a typical regression problem one starts out with a set of data points, and then attempts to determine a function that has the same qualitative behavior seen in the data. Typical examples are shown in Figure 8.4. There are assumptions made when using regression and these will be considered in the development. The reason is that the assumptions have an arbitrariness to them and they can have a significant impact on the final result.

In what follows the data set will be: (x_1, y_1), (x_2, y_2), $\cdots$, (x_n, y_n). It is assumed that there are more data points than parameters in the function to be minimized. Also, unlike the situation for interpolation, we do not assume that the x_i's are distinct.

To carry out the regression analysis, we need to specify two functions: the model function and the error function. With this we will then need to determine what method should be used to calculate the minimizer.

8.2.1 Model Function

We begin with the model function, which means we specify the function that will be fitted to the data. Examples are:

- Linear Regression

$$
\begin{aligned}
g(x) &= v_1 + v_2 x \qquad (8.3)\\
g(x) &= v_1 + v_2 x + v_3 x^2\\
g(x) &= v_1 + v_2 x + \cdots + v_m x^{m-1}\\
g(x) &= v_1 e^x + v_2 e^{-x}
\end{aligned}
$$

- Nonlinear Regression

$$g(x) = v_1 + v_2 e^{v_3 x} \quad \text{- asymptotic regression function} \qquad (8.4)$$

$$g(x) = \frac{v_1 x}{v_2 + x} \quad \text{- Michaelis-Menten function} \qquad (8.5)$$

$$g(x) = \frac{v_1}{1 + v_2 \exp(-v_3 x)} \quad \text{- logistic function} \qquad (8.6)$$

In the above functions, the v_i's are the parameters determined by fitting the function to the given data, and x is the variable in the problem. What determines whether the function is linear or nonlinear is how it depends on the v_i's. Specifically, linear regression means that the fitting function $g(x)$ is a linear function of the v_i's. If you are unsure what this means, a simple test for linear regression is that for every parameter v_j, the derivative

$$\frac{\partial g}{\partial v_j}$$

does not depend on any of the v_i's. If even one of the derivatives depends on one or more of the v_i's then the function qualifies for nonlinear regression.

Exactly which $g(x)$ to use is determined by the application. For example, in Figure 8.4, the spindle length data looks to be approximately linear, and so one would be inclined to use (8.3). The function also might be known from theoretical considerations. Examples of linear relationships as in (8.3) include Hooke's law, Ohm's law, and Fick's law. The nonlinear functions can also come from theoretical considerations, and because of this they are often given names connected from the original problem they came from. For example, the one in (8.5) gets its name from a fundamental enzyme catalyzed reaction in biochemistry known as the Michaelis-Menten mechanism [Holmes, 2009]. The one in (8.6) gets its name from the fact that it is the solution of the logistic equation given in (7.6).

8.2.2 Error Function

The next decision concerns the error, or objective, function, which is used to fit the data. This is not a simple question and the choice can have a significant affect on the final solution. To illustrate the situation, three data points and a potential linear fit to these points are shown in Figure 8.5. To measure the error, one can use the vertical distance d_i, as in the left figure, or the true distances D_i, as in the right figure. It is easy to determine the vertical distance, and it is

$$d_i = |g(x_i) - y_i|.$$

The formula for the true distance depends on the model function. In what follows we will concentrate on the linear function (8.3), in which case

$$D_i = \frac{d_i}{\sqrt{1+v_1^2}} .$$

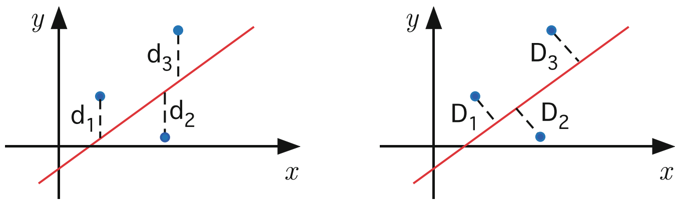


Figure 8.5 Different measures of error for the linear model function $g(x) = v_1 + v_2 x$.

Both of these are used. The d_i's are used in traditional least squares data fitting, while the D_i's are used in what is called *orthogonal regression* or *Deming regression*. It's necessary to be careful with the dimensional units using D_i. For example, the usual distance between two points requires the calculation of $D = \sqrt{(x_0 - x_1)^2 + (y_0 - y_1)^2}$. If the x's are measured, say, in terms of length, and the y's are measured in terms of, say, time then the formula for D has no meaning. The simplest way to avoid this problem is to scale, or nondimensionalize, the variables before using orthogonal regression. This scaling plays a central role in what is called a principal component analysis, which is one of the main tools for data analysis, and this is explained in Section 9.2.

8.2.2.1 Vertical Distance

The usual reason for data fitting is to obtain a function $g(x)$ that is capable of giving a representative, or approximate, value of the data as a function of x. This is a situation where using the vertical distances is appropriate. To examine how this can be done, we start with the linear model $g(x) = v_1 + v_2 x$. Suppose there are three data points, as illustrated in Figure 8.5. We want to find the values of v_1 and v_2 that will minimize d_1, d_2, and d_3. One way this can be done is to use the maximum error, which is

$$\begin{aligned} E_\infty &= \max\{d_1, d_2, d_3\} \\ &= \max_{i=1,2,3} |g(x_i) - y_i| \\ &= ||\mathbf{g} - \mathbf{y}||_\infty , \end{aligned} \tag{8.7}$$

where the ∞-norm is defined in Section 3.5, $\mathbf{y} = (y_1, y_2, y_3)^T$ is the data vector, and $\mathbf{g} = (g(x_1), g(x_2), g(x_3))^T$ is the corresponding vector of values of the model function. Other choices are the sum

$$\begin{aligned} E_1 &= d_1 + d_2 + d_3 \\ &= \sum_{i=1}^{3} |g(x_i) - y_i| \\ &= ||\mathbf{g} - \mathbf{y}||_1 , \end{aligned} \tag{8.8}$$

and the sum of squares

$$\begin{aligned} E_2 &= d_1^2 + d_2^2 + d_3^2 \\ &= \sum_{i=1}^{3} [g(x_i) - y_i]^2 \\ &= ||\mathbf{g} - \mathbf{y}||_2^2 . \end{aligned} \tag{8.9}$$

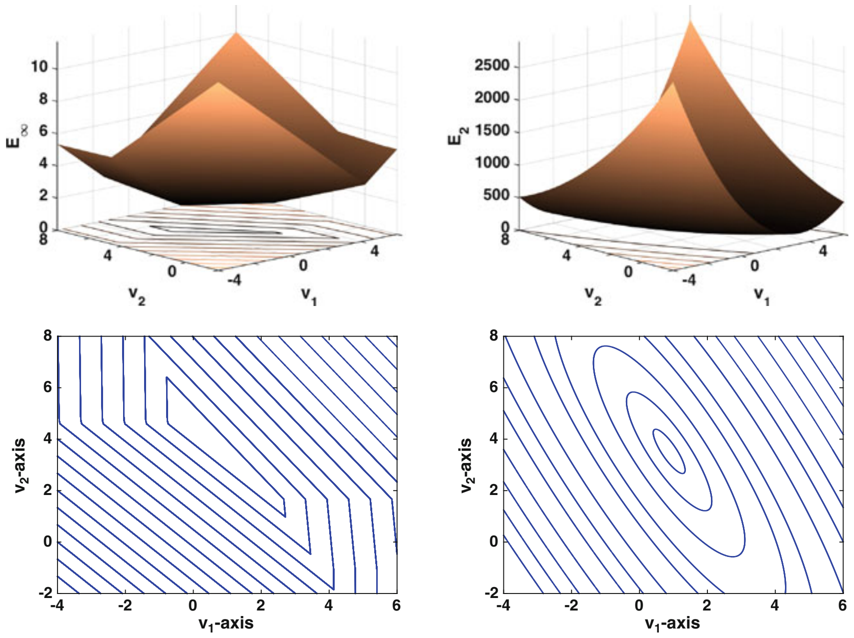


Figure 8.6 Plots of $E_\infty(v_1, v_2)$ and $E_2(v_1, v_2)$, both as a surface and as a contour plot, for the linear least squares function.

These functions have the properties associated with an error function, which are:

- they satisfy $E \geq 0$,
- in the (unlikely) case of when the curve passes through all of the data points, $E = 0$,
- they get larger the more $g(x_i)$ differs from y_i.

An example of a poor choice for an error function is $E = d_1 d_2 d_3$. One reason is that $E = 0$ if $d_1 = 0$, but d_2 or d_3 could be huge.

Of the three error functions given above, E_2 is the one most often used, producing what is called least squares. This function has the advantage of being differentiable if $g(x_i)$ is a differentiable function of the v_j's. This is illustrated in Figure 8.6 in the case of when $g(x) = v_1 + v_2 x$. It is evident that E_∞ has edges and corners, while E_2 is smooth. This smoothness allows the minimization problem to be solved using calculus methods, and this will be demonstrated below. Note, however, that the v_j's that minimize E_2 are not necessarily the same as those that minimize E_1 or E_∞. This will be discussed in more detail in Section 8.3.3.

8.3 Linear Least Squares

The assumption is that the model function $g(x)$ depends linearly on the parameters to be fitted to the data. This means it can be written as

$$g(x) = v_1\phi_1(x) + v_2\phi_2(x) + \cdots + v_m\phi_m(x),$$

where the ϕ_j's are known, or given, functions. For example, if $g(x) = v_1 + v_2x$, then $\phi_1 = 1$ and $\phi_2 = x$. As before, the data is assumed to be (x_1, y_1), (x_2, y_2), $\cdots$, (x_n, y_n). In what follows it is assumed there is more data than parameters, and this means that $m < n$.

8.3.1 Two Parameters

We begin with the case of when $g(x) = v_1 + v_2x$, which means that the associated error function is

$$E(v_1, v_2) = \sum_{i=1}^{n}(v_1 + v_2x_i - y_i)^2. \tag{8.10}$$

To find the minimum, we first find the first derivatives, which are

$$\frac{\partial E}{\partial v_1} = 2\sum_{i=1}^{n}(v_1 + v_2x_i - y_i) = 2\left(nv_1 + v_2\sum_{i=1}^{n}x_i - \sum_{i=1}^{n}y_i\right),$$

and

$$\frac{\partial E}{\partial v_2} = 2\sum_{i=1}^{n}(v_1 + v_2x_i - y_i)x_i = 2\left(v_1\sum_{i=1}^{n}x_i + v_2\sum_{i=1}^{n}x_i^2 - \sum_{i=1}^{n}x_iy_i\right).$$

Setting these derivatives to zero yields the system of equations

$$\begin{pmatrix} n & \sum x_i \\ \sum x_i & \sum x_i^2 \end{pmatrix}\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} \sum y_i \\ \sum x_iy_i \end{pmatrix}. \tag{8.11}$$

This is called the *normal equation* for the v_j's. Using the formula for the inverse matrix in (3.15), we have that

$$\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \frac{1}{n\sum x_i^2 - (\sum x_i)^2}\begin{pmatrix} \sum x_i^2 & -\sum x_i \\ -\sum x_i & n \end{pmatrix}\begin{pmatrix} \sum y_i \\ \sum x_iy_i \end{pmatrix}. \tag{8.12}$$

With this, we have solved the least squares problem when using a linear model to fit the data.

x_i	-1	2	0	1
y_i	1	-1	2	1

Table 8.1 Data used in linear least squares example.

Example

For the data in Table 8.1, $n = 4$, $\sum x_i = 2$, $\sum x_i^2 = 6$, $\sum y_i = 3$, and $\sum x_i y_i = -2$. In this case, (8.12) gives

$$\begin{aligned}\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} &= \frac{1}{20}\begin{pmatrix} 6 & -2 \\ -2 & 4 \end{pmatrix}\begin{pmatrix} 3 \\ -2 \end{pmatrix} \\ &= \frac{1}{10}\begin{pmatrix} 11 \\ -7 \end{pmatrix}.\end{aligned}$$

The linear fit $g(x) = \frac{1}{10}(11-7x)$, as well as the data, is shown in Figure 8.7. ■

A couple of mathematical comments are in order before continuing. First, the solution in (8.12) requires that the denominator be nonzero. It is not hard to prove that this holds except when all the x_i's are the same (also recall that we are assuming $n > 2$). Geometrically this makes sense because if all of the data points fall on a vertical line, then it is impossible to write the line as $g(x) = v_1 + v_2 x$. The second comment concerns whether the point is a minimum. It is possible to examine the second derivatives to determine this, but it's easier just to examine the function in (8.10). It is a quadratic function of v_1 or v_2 that opens upwards (i.e., $E \to \infty$ as either $v_1 \to \infty$ or $v_2 \to \infty$). Given that the solution is unique then the only possibility is that the solution is a minimum.

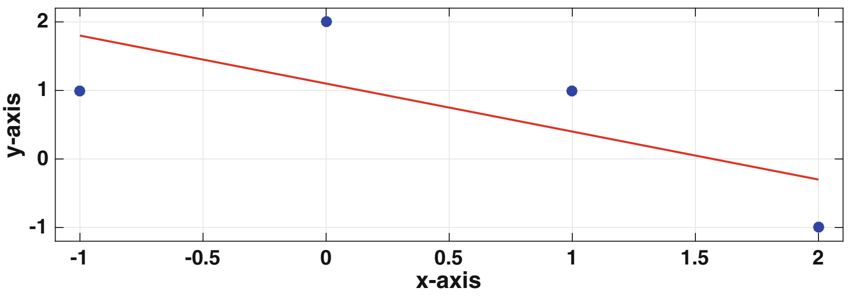


Figure 8.7 Linear fit obtained using the least squares formula in (8.12).

The more general two parameter least squares problem involving $g(x) = v_1\phi_1(x) + v_2\phi_2(x)$ can also be solved easily, and this is considered in Exercise 8.16.

8.3.2 General Case

The model function is now taken to have the general form

$$g(x) = v_1\phi_1(x) + v_2\phi_2(x) + \cdots + v_m\phi_m(x), \tag{8.13}$$

and the error function is

$$E(v_1, v_2, \cdots, v_m) = \sum_{i=1}^{n} [g(x_i) - y_i]^2. \tag{8.14}$$

It is not hard to show that this can be written in matrix form as

$$E(\mathbf{v}) = ||\mathbf{A}\mathbf{v} - \mathbf{y}||_2^2 \tag{8.15}$$

where $\mathbf{v} = (v_1, v_2, \cdots, v_m)^T$, $\mathbf{y} = (y_1, y_2, \cdots, y_n)^T$, and $\mathbf{A}$ is the $n \times m$ matrix

$$\mathbf{A} = \begin{pmatrix} \phi_1(x_1) & \phi_2(x_1) & \cdots & \phi_m(x_1) \\ \phi_1(x_2) & \phi_2(x_2) & \cdots & \phi_m(x_2) \\ \vdots & \vdots & & \vdots \\ \phi_1(x_n) & \phi_2(x_n) & \cdots & \phi_m(x_n) \end{pmatrix}. \tag{8.16}$$

There are various ways to find the $\mathbf{v}$ that minimizes E, and we will begin with the most direct method. It is assumed in what follows that the columns of $\mathbf{A}$ are independent, which means that the matrix has rank m.

8.3.2.1 Normal Equation

Using the calculus approach, one simply calculates the first derivatives of E, and then sets them to zero. After cleaning up the resulting expressions, one finds that the problem reduces to solving

$$(\mathbf{A}^T\mathbf{A})\mathbf{v} = \mathbf{A}^T\mathbf{y}. \tag{8.17}$$

This is the m-dimensional version of the matrix problem in (8.11). As before, it is called the *normal equation* for the v_j's. Because the columns of $\mathbf{A}$ are assumed to be independent, it is possible to prove that the symmetric $m \times m$ matrix $\mathbf{A}^T\mathbf{A}$ is positive definite. This means that the normal equation can be solved using a Cholesky factorization. If this is done, then for larger values of m and n, the computationally expensive steps are calculating $\mathbf{A}^T\mathbf{A}$ and then finding the Cholesky factorization. Altogether this adds up to about $nm^2 + \frac{1}{3}m^3$ flops.

Example

In the case of when

$$g(x) = v_1 + v_2 x + \cdots + v_m x^{m-1},$$

then

$$\mathbf{A} = \begin{pmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{m-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{m-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{m-1} \end{pmatrix}. \tag{8.18}$$

Although this matrix is not square, it strongly resembles the Vandermonde matrix we derived in Chapter 5 when taking the direct approach for polynomial interpolation. As you might recall, we found that the Vandermonde matrix is ill-conditioned when attempting to interpolate with larger degree polynomials. It should not be a surprise that the square matrix $\mathbf{A}^T\mathbf{A}$ in (8.17) suffers similar problems for this example. To check on this, the condition number for the matrix is given in Table 8.2, where the x_i's are picked randomly from the interval $0 < x < 1$. Based on the heuristic given in Section 3.6.3, it is likely that a numerical solution of the normal equation is correct to only about 3 digits when $m = 10$, and is meaningless when m is 12 or larger. ■

m	n	$\kappa_\infty(\mathbf{A}^T\mathbf{A})$	$\kappa_\infty(\mathbf{R}_m)$
4	100	2.84e+04	1.89e+02
6	100	2.51e+07	5.17e+03
8	100	3.12e+10	1.91e+05
10	100	3.79e+13	6.55e+06
12	100	4.92e+16	2.30e+08
14	100	2.56e+18	7.59e+09

Table 8.2 Condition number of matrices that arise when fitting a $m-1$ degree polynomial to data.

The question arises as to whether the ill-conditioned nature of the matrix in the last example is typical when solving the normal equation. One way to answer this is to run some numerical experiments and calculate $\kappa_\infty(\mathbf{A}^T\mathbf{A})$ for random $n\times m$ matrices. This is easy to do and you find that if the number of parameters m is not close to the number of data points n, then $\mathbf{A}^T\mathbf{A}$ is well-conditioned. However, as m gets close to n, the condition number starts to increase rapidly with the result that the matrix becomes ill-conditioned. It is also possible to find, as the last example illustrates, matrices which are ill-conditioned even though m is not close to n. In those cases when the matrix is ill-conditioned, or even if it is well-conditioned, the next method can be used.

8.3.2.2 QR Approach

Although it might not be immediately obvious, it is possible to use the QR factorization to solve the least square problem. To explain, as shown in Section 4.3.2, an invertible matrix can be factored as $\mathbf{B} = \mathbf{QR}$, where $\mathbf{Q}$ is an orthogonal matrix, and $\mathbf{R}$ is an upper triangular matrix. The procedure used to find this factorization can be used, without modification, on $n\times m$ matrices, assuming that the matrix has m linearly independent columns with $m \le n$. For the matrix $\mathbf{A}$ in the least squares problem, this means that it is possible to factor it as $\mathbf{A} = \mathbf{QR}$, where $\mathbf{Q}$ is an $n \times n$ orthogonal matrix and $\mathbf{R}$ is an $n \times m$ matrix of the form

$$\mathbf{R} = \begin{pmatrix} \mathbf{R}_m \\ \mathbf{O} \end{pmatrix}, \tag{8.19}$$

where $\mathbf{R}_m$ is an upper triangular $m \times m$ matrix and $\mathbf{O}$ is a $(n-m)\times m$ matrix containing only zeros. Using the useful fact that orthogonal matrices have the property that $||\mathbf{Qx}||_2 = ||\mathbf{x}||_2$, a straightforward calculation shows that (8.15) can be rewritten as

$$E(\mathbf{v}) = ||\mathbf{R}_m\mathbf{v} - \mathbf{b}_m||_2^2 + ||\mathbf{b}_{n-m}||_2^2,$$

where $\mathbf{b}_m$ is the vector coming from the first m rows of $\mathbf{Q}^T\mathbf{b}$ and $\mathbf{b}_{n-m}$ is the vector coming from the remaining $n-m$ rows of $\mathbf{Q}^T\mathbf{b}$. There is nothing we can do about the second term on the right-hand side, but we do have the ability to minimize the first term. In fact, the minimum occurs when

$$\mathbf{R}_m\mathbf{v} = \mathbf{b}_m. \tag{8.20}$$

This equation is easy to solve for $\mathbf{v}$ since $\mathbf{R}_m$ is upper triangular. The condition number of $\mathbf{R}_m$ is given in Table 8.2 for the previous example. This shows that in this case the QR approach produces a better conditioned matrix than when using the normal equation, but it also shows that $\mathbf{R}_m$ becomes ill-conditioned when a larger number of parameters are used. As for work, for larger values of m and n, the computationally expensive step using this method is finding the QR factorization, which requires about $2nm^2 - \frac{2}{3}m^3$ flops.

8.3.2.3 Parting Comments

There is no question that solving the least squares problem using the normal equation approach is simpler and easier to code. In terms of speed, based on the flops, when there is a lot of data compared to the number of fitting parameters, solving the normal equation should be about twice as fast as using the QR approach. When m and n are about the same then the two methods should, in theory, take about the same computing time. However, it is not really necessary to worry about this for data and parameter sets that are not particularly large (hundreds rather than millions), because the actual computing times are small in both cases. Consequently, because the QR approach has the advantage of producing a better conditioned matrix problem than when using normal equations, it is often the method of choice.

There are other methods, and one that is useful when the matrix is rank deficient (i.e., it does not have m linearly independent columns), uses the SVD factorization. Another approach, which has the additional benefit of not being limited to linear regression, is to return to the original minimization formulation and use descent methods (which are discussed later in the chapter).

8.3.3 Other Error Functions

The least squares error function in (8.10) was chosen, in part, simply because it's possible to differentiate it. Making the choice that is not based on physical reasoning but on mathematical convenience is not necessarily a good thing. So, it is worth examining what other error functions might produce. To explore this, we first state the least squares error function

$$E_2 = \sum d_i^2 \tag{8.21}$$

$$= \sum_{i=1}^{n} [g(x_i) - y_i]^2. \tag{8.22}$$

An alternative is the true distance, as illustrated in Figure 8.5. Using least squares, the error function in this case is

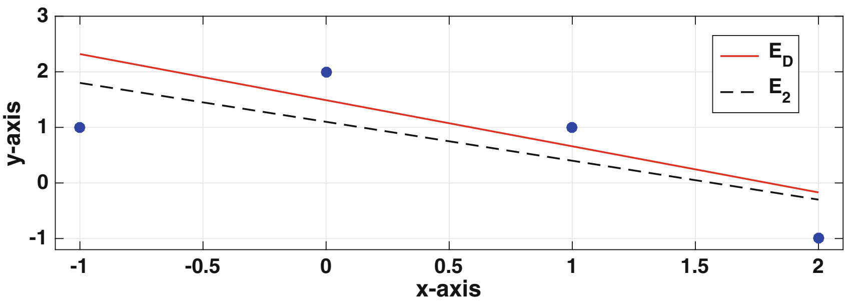


Figure 8.8 Linear fit obtained using E_D, solid red line, and E_2, the dashed black line.

$$E_D = \sum D_i^2 \tag{8.23}$$

$$= \frac{1}{1+v_1^2}\sum_{i=1}^{n}[g(x_i)-y_i]^2. \tag{8.24}$$

In the case of when $g(x) = v_1 + v_2x$, finding the values of v_1 and v_2 that minimizes E_D reduces to solving a cubic equation. Although it is possible to find a formula for the solution, it is easier to just find v_1 and v_2 numerically. In the example below, the values are calculated using the Nelder-Mead algorithm, which is described in Section 8.8.

Example

Earlier we used the data in Table 8.1 to find v_1 and v_2 when using the error function E_2. Using the same data, but using the error function E_D, one finds that $v_1 = 1.4895$ and $v_2 = -0.8298$. The resulting linear fit to the data is shown in Figure 8.8. For comparison, the earlier determined line using E_2 is also shown. The fact that the lines are different is no surprise. ■

Two other error functions mentioned earlier are

$$E_1 = \sum d_i$$

$$= \sum_{i=1}^{n} |g(x_i) - y_i|,$$

and

$$E_\infty = \max_{i=1,\cdots,n} |g(x_i) - y_i|.$$

As stated earlier, both of these have edges, so it is not possible to use the calculus solution for finding the minimum. However, there are derivative free

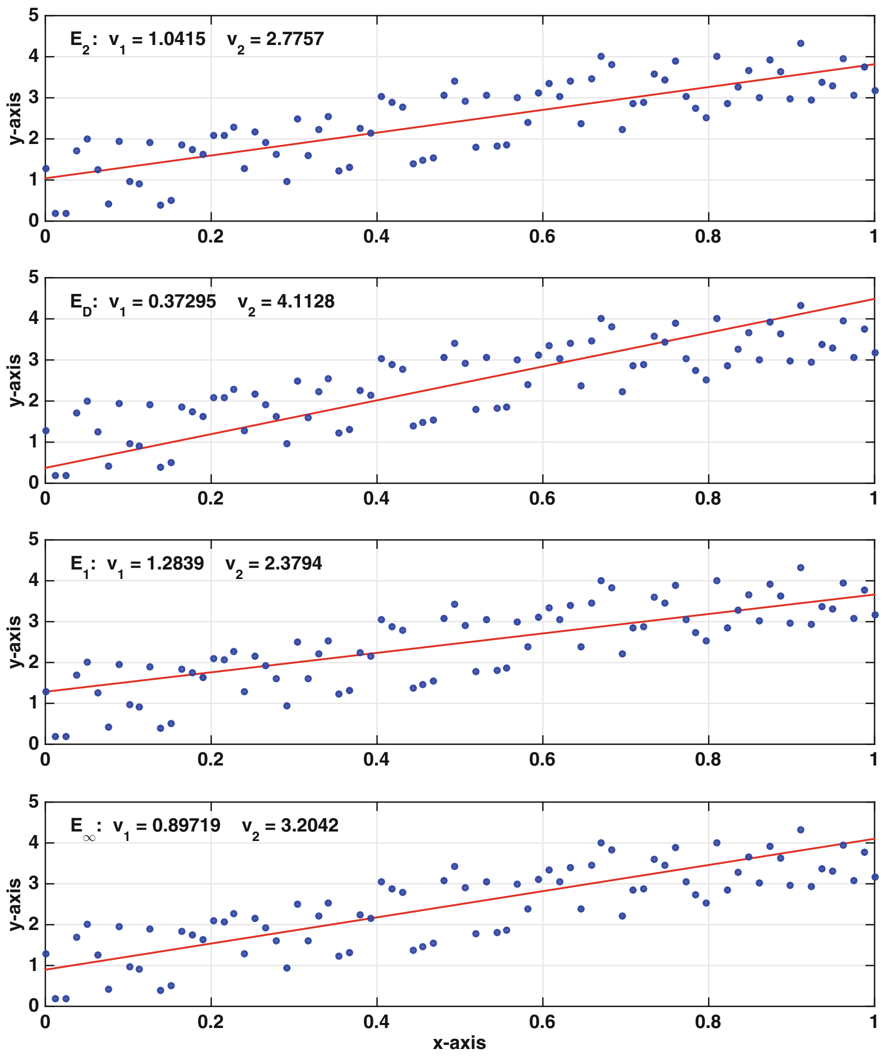


Figure 8.9 Linear fits obtained using different error functions on the same data set.

minimization methods, and one is described in Section 8.8. It is known as the Nelder-Mead algorithm, and this is used in the example below for all the error functions except E_2.

Example

An example of the linear fit obtained from each of these error functions is shown in Figure 8.9. It is difficult to make sweeping generalizations from this

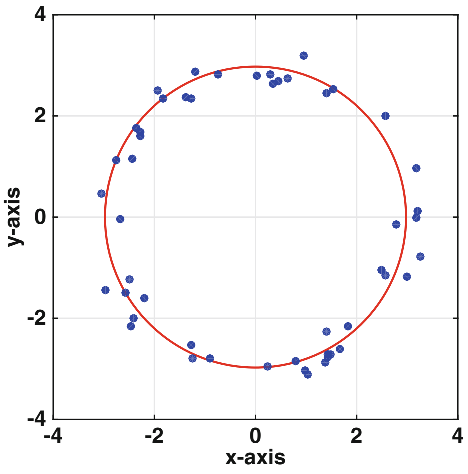


Figure 8.10 Data to be fitted by a circle, and the resulting circle obtained using least squares.

one comparison, but the values obtained using E_1 and E_2 are fairly close. They also differ from the values obtained using E_∞, even though all three of them are using vertical distances in some way. The one using true distances, E_D, produces values even more different than E_∞. One might be tempted to declare a winner, claiming one is better than the others. However, this requires a definition or criterion for winning, and in this sense they all win. Namely, each is based on a different definition of error, and they minimize their particular error function. ■

Example

As a third example, consider the data shown in Figure 8.10. The objective is to find the circle $x^2 + y^2 = r^2$ that best fits this data, with the parameter in this case being the radius r. The first step is to specify the error function. This is a situation where using the vertical distance runs into some serious complications. This is because for any data point to the right (so, $x_i > r$) or left (so, $x_i < -r$) of the circle, the vertical distance is not defined. Consequently, it is more appropriate to use the radial (true) distance between the curve and the data. So, given a data point (x_i, y_i), the corresponding radial coordinates are (r_i, θ_i), where $r_i^2 = x_i^2 + y_i^2$. The least squares error based on the radial distance is

$$E(r) = \sum_{i=1}^{n} (r - r_i)^2. \tag{8.25}$$

Taking the derivative of this with respect to r, setting the result to zero, and solving yields the solution

$$r = \frac{1}{n}\sum_{i=1}^{n} r_i. \tag{8.26}$$

This answer is one that you might have guested before undertaking the regression analysis because it states that the circle that best fits the data is the one whose radius is the average of the radii from the data. A comparison between the resulting circle and the data is shown in Figure 8.10. ■

8.4 Nonlinear Regression

To illustrate some of the complications that arise with nonlinear regression, we will consider the data shown in Figure 8.11, which is from Smith et al. [2010]. Based on what they refer to as a model for simple hyperbolic binding, they assumed the data can be fit with the function

$$g(x) = \frac{v_1 x}{v_2 + x}\,. \tag{8.27}$$

This is known as the Michaelis-Menten model function, and is used extensively in biochemistry. Using a least squares error function, then

$$E(v_1, v_2) = \sum_{i=1}^{n}\left(\frac{v_1 x_i}{v_2 + x_i} - y_i\right)^2. \tag{8.28}$$

Taking the first partials of E, and setting them to zero, we get the following system of equations to solve:

$$\begin{aligned}\sum_{i=1}^{n}\left(\frac{v_1 x_i}{v_2 + x_i} - y_i\right)\frac{x_i}{v_2 + x_i} &= 0,\\ \sum_{i=1}^{n}\left(\frac{v_1 x_i}{v_2 + x_i} - y_i\right)\frac{v_1 x_i}{(v_2 + x_i)^2} &= 0.\end{aligned}$$

In other words, we have to solve two nonlinear equations for v_1 and v_2. One possibility is to use Newton's method, see Section 3.10. However, a more direct approach is considered next.

8.4.1 Transforming to Linear Regression

Some nonlinear regression problems can be transformed into linear ones, although this usually only works when the model function is not too complicated. To illustrate, the Michaelis-Menten function, given in (8.27),

$$y = \frac{v_1 x}{v_2 + x},$$

can be written as

$$\begin{aligned}\frac{1}{y} &= \frac{v_2 + x}{v_1 x} \\ &= \frac{v_2}{v_1}\frac{1}{x} + \frac{1}{v_1}.\end{aligned}$$

Setting $Y = 1/y$ and $X = 1/x$, then the above model function takes the form

$$G(X) = V_1 + V_2 X. \tag{8.29}$$

where $V_1 = 1/v_1$ and $V_2 = v_2/v_1$. If (x_i, y_i) is one of the original data points, then the corresponding point for the transformed problem is $(X_i, Y_i) = (1/x_i, 1/y_i)$.

Using least squares, then the error function is

$$E(V_1, V_2) = \sum_{i=1}^{n} (V_1 + V_2 X_i - Y_i)^2. \tag{8.30}$$

According to (8.12), the resulting solution is

$$\begin{pmatrix} V_1 \\ V_2 \end{pmatrix} = \frac{1}{n\sum X_i^2 - (\sum X_i)^2} \begin{pmatrix} \sum X_i^2 & -\sum X_i \\ -\sum X_i & n \end{pmatrix} \begin{pmatrix} \sum Y_i \\ \sum X_i Y_i \end{pmatrix}. \tag{8.31}$$

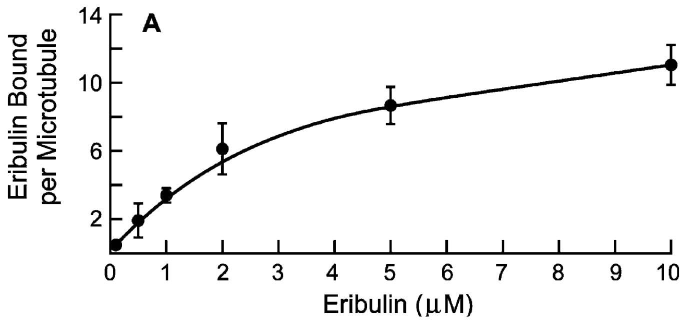


Figure 8.11 Data from Smith et al. [2010].

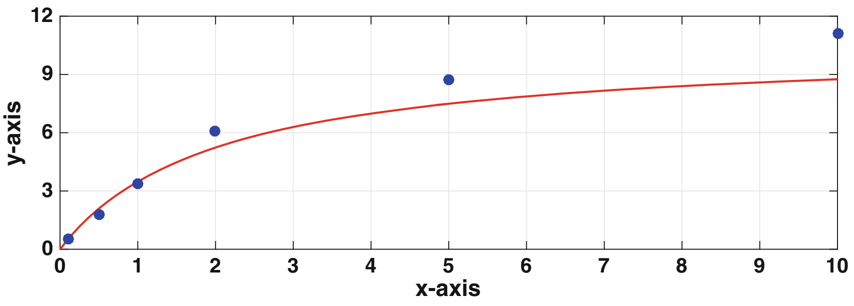


Figure 8.12 Fit of data from Smith et al. [2010] when using the error function in (8.30).

The last step is to transform back, using the formulas $v_1 = 1/V_1$ and $v_2 = V_2/V_1$. With this, we have been able to find a solution without the need for Newton's method.

Although the solution in (8.31) is often used, it has a rather serious flaw. To illustrate what this is, if you use the data in Figure 8.11, then the curve shown in Figure 8.12 is obtained. The poor fit occurs because, for this particular model function,

$$g(x_i) - y_i = \frac{G(X_i) - Y_i}{Y_i G(X_i)} .$$

So, having $G(X_i) - Y_i$ close to zero does not necessarily mean $g(x_i) - y_i$ is close to zero, and the reason is the denominator. As an example, the last data point in Figure 8.12 is $(x_6, y_6) = (10, 11.1)$, which results in $Y_6 G(X_6) \approx 0.008$ and $g(x_6) - y_6 \approx 123[G(X_6) - Y_6]$. This magnification of the error is the cause for the poor fit in Figure 8.12.

There is a simple fix, and it is to use the relative error E_R, which is given as

$$E_R(V_1, V_2) = \sum_{i=1}^{n} \left(\frac{V_1 + V_2 X_i - Y_i}{Y_i} \right)^2 . \tag{8.32}$$

Using the solution for the minimum given in Exercise 8.15, the resulting fit is shown in Figure 8.13. As is clear, the fit is definitely better than what was obtained earlier.

The transformation used above needs more discussion. First, it requires that the change of variables is defined for the data under consideration, which means that the x_i's and y_i's are nonzero. Second, the connection between the original and transformed minimization problems is tenuous. To illustrate, the linear least squares function (8.32), or the one in (8.32), is not the transformed version of the nonlinear least squares function in (8.28). The transformation method is based, in part, on the assumption that all three qualify as error functions for the data (transformed or not), and should produce v_1 and v_2 values that are close. The above example shows that there is merit to this

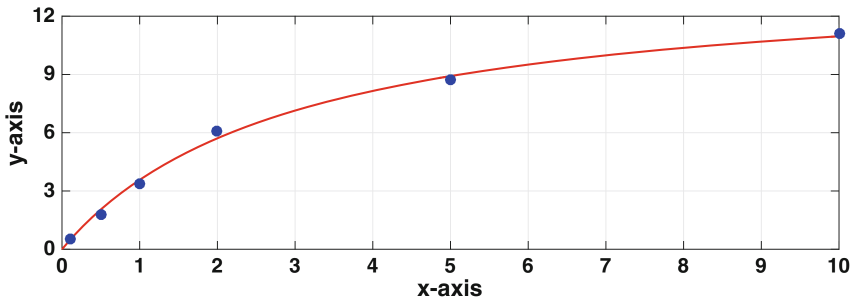


Figure 8.13 Fit of data from Smith et al. [2010] when using the error function in (8.32).

assumption, but it also shows that the nonlinear nature of the transformation can magnify the differences in these functions. What one could do is take the easily computable solutions using (8.32) and (8.32), and determine which produces the smallest value of (8.28). Or, for those set on minimizing (8.28), to use one of the solutions as the starting point when using Newton's method to minimize (8.28).

8.5 Descent Methods: Introduction

We are going to return to the original formulation of the problem, which is the following: given a function $F(\mathbf{v})$, find the point $\mathbf{v}_m \in \mathbb{R}^n$ where F achieves its minimum value. As was done for the regression problem, one possible approach is to use the calculus solution, and this means solving $\nabla F = \mathbf{0}$. We will, however, do something different, and to illustrate this situation, consider the linear least squares example in Figure 8.7. The values of the error function $E_2(v_1, v_2)$ are shown in Figure 8.14, both as a surface and as a contour plot. The basic idea of what we are going to do is easy to explain geometrically. If you were standing somewhere on this surface and wanted to go downhill, you would pick a direction of descent and then start walking in that direction. This simple idea can be used to solve the minimization problem, but two recurring decisions must be made: which particular downhill direction to use and just how far should you walk in this direction before picking another direction.

The basic algorithm consists of picking a starting position $\mathbf{v}_1$ and then constructing a sequence $\mathbf{v}_2$, $\mathbf{v}_3$, $\mathbf{v}_4, \ldots$ that (hopefully) converges to $\mathbf{v}_m$. In doing this it is required that the value of F decreases with each step. To get this to happen, suppose we are located at $\mathbf{v}_k$. We first need to identify a direction of descent $\mathbf{d}_k$ and we then need to "walk" a distance α_k in that direction. In mathematical terms we have the following formula:

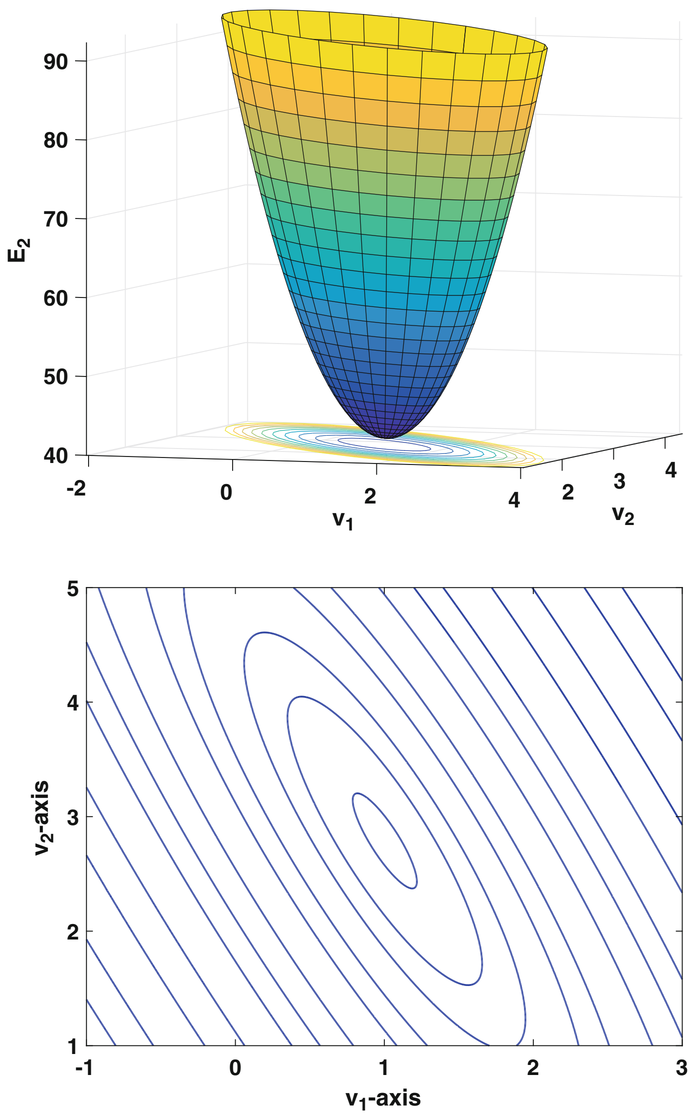


Figure 8.14 Plot of $E_2(v_1, v_2)$, both as a surface and as a contour plot, for the linear least squares function for data similar to what was used for Figure 8.7.

$$\mathbf{v}_{k+1} = \mathbf{v}_k + \alpha_k \mathbf{d}_k, \quad \text{for } \; k = 1, 2, 3, \ldots . \tag{8.33}$$

Assuming, for the moment, that we know $\mathbf{d}_k$ then the question arises as to how far we should walk in this direction. In other words, what is the value of α_k? The ideal solution is that α_k is the value of α that minimizes

$$q(\alpha) = F(\mathbf{v}_k + \alpha \mathbf{d}_k). \tag{8.34}$$

In optimization, determining the value of α_k is known as the *line search problem.* One possible way to solve this involves the calculus solution, which means that we solve $q'(\alpha) = 0$. We will certainly do this but we will also experiment with other possibilities. For example, the only requirement is that $F(\mathbf{v}_{k+1}) < F(\mathbf{v}_k)$. For at least one of the methods we will consider we will simply guess values for α_k. This might not be optimum but it avoids the potentially time-consuming problem of finding a minima for $q(\alpha)$.

8.5.1 Descent Directions

Almost everyone who first uses a descent method picks the direction of steepest descent for $\mathbf{d}_k$. In a way this is the most natural choice because water flowing down a hillside will try to follow a steepest descent path (however, it has a little problem with inertia that messes this up).

Recall from calculus that given a function $y = F(\mathbf{v})$, if $\nabla F(\mathbf{v}) \neq \mathbf{0}$, then the direction of steepest descent is $-\nabla F(\mathbf{v})$. Based on this, if one picks the direction of steepest descent at $\mathbf{v}_k$, then

$$\mathbf{d}_k = -\nabla F(\mathbf{v}_k). \tag{8.35}$$

This formula is used often in the pages that follow, and with it come some implicit assumptions. First, the function $F(\mathbf{v})$ is assumed to be $C^1(\mathbb{R}^n)$, which means that its first partial derivatives are defined and continuous everywhere. Second, whenever using this to determine a direction of descent it is assumed that $\nabla F(\mathbf{v}_k) \neq \mathbf{0}$.

We need a simple test to determine if any given $\mathbf{d}$ is a descent direction. Recall that the two vectors $\pm\nabla F(\mathbf{v})$ are normal, or perpendicular, to the level curves (or surfaces) of $F(\mathbf{v})$. Our descent directions must be on the same side as $-\nabla F(\mathbf{v})$, and this is illustrated in Figure 8.15. Said another way, the angle θ between $\mathbf{d}$ and $-\nabla F(\mathbf{v})$ must be less than 90^o. According to the law of cosines, $||\mathbf{d}|| \cdot ||\nabla F(\mathbf{v})|| \cos\theta = -\mathbf{d} \cdot \nabla F(\mathbf{v})$. From this it follows that the condition for $\mathbf{d}$ to be a descent direction at $\mathbf{v}$ is

$$\mathbf{d} \cdot \nabla F(\mathbf{v}) < 0. \tag{8.36}$$

It is difficult to say much more about the general form of the descent method because the choices one makes depend on the particular function being minimized. So, we consider an important special case, which is the problem of solving a matrix equation.

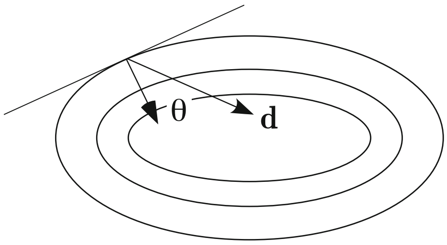


Figure 8.15 Angle θ between a direction of descent $\mathbf{d}$ and the direction of steepest descent.

8.6 Solving Linear Systems

It is easy to rewrite a linear system $\mathbf{Av} = \mathbf{b}$ as a minimization problem. For example, as mentioned earlier, one can take $F(\mathbf{v}) = ||\mathbf{Av} - \mathbf{b}||_2^2$. However, for certain matrices there is another possibility. To explain, we start with the $n = 1$ case, where the equation is simply $av = b$. We want to find a function $F(v)$ that has a minimum exactly when $av = b$. Recall that at a minimum of $F(v)$ one solves $F'(v) = 0$. If $av - b = 0$ corresponds to the equation $F'(v) = 0$ then we need $F'(v) = av - b$. Integrating gives us $F(v) = \frac{1}{2}av^2 - bv$ (the additive constant is not needed). For this to correspond to a minimum problem the quadratic must open upward, which means we have the positivity requirement $a > 0$.

To see what happens when there are more variables, let $n = 2$, so the equations have the form

$$\begin{aligned} a_{11}v_1 + a_{12}v_2 &= b_1, \\ a_{21}v_1 + a_{22}v_2 &= b_2. \end{aligned}$$

We want to find a smooth function $F(\mathbf{v})$ that has a minimum exactly when $\mathbf{Av} = \mathbf{b}$. To achieve this we require that F satisfies

$$\frac{\partial F}{\partial v_1} = a_{11}v_1 + a_{12}v_2 - b_1, \tag{8.37}$$

$$\frac{\partial F}{\partial v_2} = a_{21}v_1 + a_{22}v_2 - b_2. \tag{8.38}$$

For F to be smooth we need

$$\frac{\partial^2 F}{\partial v_1 \partial v_2} = \frac{\partial^2 F}{\partial v_2 \partial v_1},$$

and so from (8.37), (8.38) we conclude $a_{12} = a_{21}$. In other words, our construction requires that the matrix $\mathbf{A}$ be symmetric. Assuming this holds, then integrating (8.37) and (8.38) it follows that

$$F(\mathbf{v}) = \frac{1}{2}\mathbf{v}^T\mathbf{A}\mathbf{v} - \mathbf{b}\cdot\mathbf{v}. \tag{8.39}$$

To guarantee that this has a minimum there is a positivity requirement on $\mathbf{A}$, namely, it must be positive definite (as well as symmetric). Although this result has been derived in the particular case of $n = 2$, the quadratic form in (8.39) is what is obtained for general n and it is the function used in what follows. To summarize this discussion, we have the following result:

Theorem 8.1. *If $F(\mathbf{v}) = \frac{1}{2}\mathbf{v}^T\mathbf{A}\mathbf{v} - \mathbf{b}\cdot\mathbf{v}$, where $\mathbf{A}$ is symmetric and positive definite, then the minimum of F occurs at the same $\mathbf{v}$ that is the solution of $\mathbf{A}\mathbf{v} = \mathbf{b}$.*

8.6.1 Basic Descent Algorithm for $\mathbf{A}\mathbf{v} = \mathbf{b}$

To find the $\mathbf{v}$ that minimizes $F(\mathbf{v})$ in (8.39), the general form for the iteration given in (8.33) applies. Namely, given a descent direction $\mathbf{d}_k$ at $\mathbf{v}_k$, then $\mathbf{v}_{k+1} = \mathbf{v}_k + \alpha_k\mathbf{d}_k$. For α_k we will use the value of α that minimizes $q(\alpha) = F(\mathbf{v}_k + \alpha\mathbf{d}_k)$. Plugging in the formula for F from (8.39), taking the derivative with respect to α, and then setting the result to zero produces the solution

$$\alpha_k = \frac{\mathbf{d}_k\cdot\mathbf{r}_k}{\mathbf{d}_k\cdot\mathbf{q}_k}\ , \tag{8.40}$$

where

$$\mathbf{q}_k = \mathbf{A}\mathbf{d}_k, \tag{8.41}$$

and

$$\mathbf{r}_k = \mathbf{b} - \mathbf{A}\mathbf{v}_k. \tag{8.42}$$

The vector $\mathbf{r}_k$ is the *residual* and this was introduced in Section 3.6. It is also related to the direction of steepest descent. To explain, note that from (8.39)

$$-\nabla F(\mathbf{v}) = -\mathbf{A}\mathbf{v} + \mathbf{b}. \tag{8.43}$$

Therefore, the residual $\mathbf{r}_k$ at $\mathbf{v}_k$ is the direction of steepest descent at $\mathbf{v}_k$. This means, from (8.36), that for $\mathbf{d}_k$ to be a direction of descent it must satisfy

$$\mathbf{d}_k\cdot\mathbf{r}_k > 0. \tag{8.44}$$

Another observation worth mentioning is that, given $\mathbf{r}_1 = \mathbf{b} - \mathbf{A}\mathbf{v}_1$, then for the other values of k,

$$\begin{aligned}\mathbf{r}_k &= \mathbf{b} - \mathbf{A}\mathbf{v}_k\\ &= \mathbf{b} - \mathbf{A}(\mathbf{v}_{k-1} + \alpha_{k-1}\mathbf{d}_{k-1})\\ &= \mathbf{r}_{k-1} - \alpha_{k-1}\mathbf{q}_{k-1}.\end{aligned} \tag{8.45}$$

This makes computing the residual a bit easier as it avoids a matrix multiplication.

To summarize the above discussion, the descent method used to solve $\mathbf{Av} = \mathbf{b}$, when $\mathbf{A}$ is symmetric and positive definite, is: after picking $\mathbf{v}_1$, then

$$\mathbf{v}_{k+1} = \mathbf{v}_k + \alpha_k \mathbf{d}_k, \quad \text{for} \quad k = 1, 2, 3, \dots , \tag{8.46}$$

where $\mathbf{d}_k$ is a direction of descent at $\mathbf{v}_k$ and

$$\alpha_k = \frac{\mathbf{d}_k \cdot \mathbf{r}_k}{\mathbf{d}_k \cdot \mathbf{q}_k} . \tag{8.47}$$

In the above formula, $\mathbf{q}_k$ is given in (8.41) and the residual $\mathbf{r}_k$ is calculated using (8.45). The only thing left to do is specify the direction of descent.

8.6.2 Method of Steepest Descents for $\mathbf{Av} = \mathbf{b}$

We are going to consider what happens when we use the steepest descent direction to solve the minimization problem for $\mathbf{Av} = \mathbf{b}$. In other words, we will take $\mathbf{d}_k = \mathbf{r}_k$ in (8.46) and (8.47). Not surprisingly, this gives rise to what is known as the method of steepest descents (SDM). The resulting algorithm is: after picking $\mathbf{v}_1$, then

$$\mathbf{v}_{k+1} = \mathbf{v}_k + \alpha_k \mathbf{r}_k, \quad \text{for} \quad k = 1, 2, 3, \dots , \tag{8.48}$$

where $\mathbf{r}_k = \mathbf{b} - \mathbf{Av}_k$,

$$\alpha_k = \frac{\mathbf{r}_k \cdot \mathbf{r}_k}{\mathbf{r}_k \cdot \mathbf{q}_k} , \tag{8.49}$$

and $\mathbf{q}_k = \mathbf{Ar}_k$.

The vectors $\mathbf{v}_{k+1}$ computed using (8.46) continue getting closer to the solution $\mathbf{v}_m$, and we need a way to determine when we are close enough. One possibility is to use the iteration error (see Section 1.5), which means that we specify an error tolerance *tol* and then stop computing when $||\mathbf{v}_{k+1} - \mathbf{v}_k|| \leq tol$. Another possibility is to use the residual $\mathbf{r}_k$ given in (8.42). If we are fortunate enough to have $\mathbf{r}_k = \mathbf{0}$, then the problem is solved and $\mathbf{v}_m = \mathbf{v}_k$. Since $\|\mathbf{r}_k\| = 0$ only when $\mathbf{v}_k$ is the solution, having $\|\mathbf{r}_k\|$ small can be used as a stopping condition for the iteration. This works as long as $\mathbf{A}$ is not ill-conditioned, and we will return to this topic later.

Because the only significant operations involved with SDM are vector and matrix multiplication it is very easy to code. Moreover, it is capable of taking maximum advantage of the sparseness of $\mathbf{A}$. For example, to calculate $\mathbf{q}_k = \mathbf{Ad}_k$ we need to store, and use, only the nonzero entries in $\mathbf{A}$.

How effective is SDM in solving matrix equations? Let's find out by trying a couple of examples.

Examples

In the two examples below the stopping condition is $\|\mathbf{v}_{k+1} - \mathbf{v}_k\|_\infty < 10^{-4}$. Also, once the SDM has stopped, then the error $\|\mathbf{v}_m - \mathbf{v}_k\|_\infty$, which is the difference between the exact and SDM solutions, is computed to check to see how well the method has worked.

1. $\begin{pmatrix} 3 & -2 \\ -2 & 4 \end{pmatrix}\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$

   First note that the matrix is symmetric and, from Theorem 3.4, positive definite. Taking $\mathbf{v}_1 = (1,\ 2)^T$, to determine $\mathbf{v}_2$, note that

$$\mathbf{r}_1 = \begin{pmatrix} 1 \\ -1 \end{pmatrix} - \begin{pmatrix} 3 & -2 \\ -2 & 4 \end{pmatrix}\begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 2 \\ -7 \end{pmatrix},$$

$$\mathbf{q}_1 = \begin{pmatrix} 3 & -2 \\ -2 & 4 \end{pmatrix}\begin{pmatrix} 2 \\ -7 \end{pmatrix} = \begin{pmatrix} 20 \\ -32 \end{pmatrix}$$

   and

$$\alpha_1 = \frac{\mathbf{r}_1 \cdot \mathbf{r}_1}{\mathbf{r}_1 \cdot \mathbf{q}_1} = \frac{53}{264}.$$

   With this,

$$\mathbf{v}_2 = \mathbf{v}_1 + \alpha_1 \mathbf{r}_1 = \begin{pmatrix} 185/132 \\ 157/264 \end{pmatrix} \approx \begin{pmatrix} 1.4 \\ 0.6 \end{pmatrix}.$$

   The remaining $\mathbf{v}_k$'s are computed using MATLAB, and the resulting steps produced are shown in the contour plot on the left in Figure 8.16. There are two important observations to be made here. First, each path (line) is orthogonal to the contour it starts from. This is expected to given that

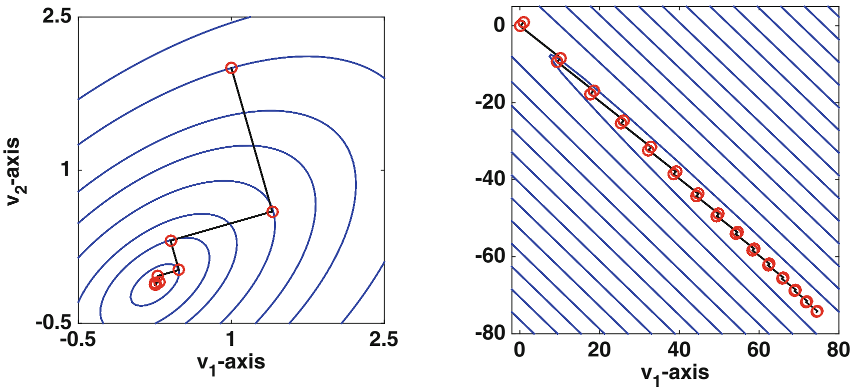


Figure 8.16 Contours of $F(v_1, v_2)$ and the first few steps taken by the method of steepest descent in the two example problems.

this is the method of steepest descents. The second observation is that the paths form right angles to each other. For example, the line from $\mathbf{v}_1$ to $\mathbf{v}_2$ is perpendicular to the one from $\mathbf{v}_2$ to $\mathbf{v}_3$. As will be explained below, this is not a good thing and we will modify the method to prevent this from happening. Finally, for this example, the method takes 14 iterations and the resulting error is 4×10^{-5}. For later reference, the condition number for this matrix is $\kappa_\infty \approx 4.5$. ■

2. $\begin{pmatrix} 5 & 4.99 \\ 4.99 & 5 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$

   Taking $\mathbf{v}_1 = (1,\ 1)^T$, the first few steps produced with SDM are shown in the contour plot on the right in Figure 8.16. The contours are so elongated in this example that the elliptical curves around the minimum are not evident (as they are in the contour plot to the left). It is found that the method takes a whopping 200 iteration steps and the resulting error is a mediocre 9×10^{-3}. For later reference, the condition number for this matrix is $\kappa_\infty \approx 1000$. ■

In reviewing the results from the above examples one might wonder why anyone would ever consider using the SDM because the method can be painfully slow to converge. This is not because the matrices are ill-conditioned. Although the condition number for the second matrix is not particularly small, it is not so large that we would anticipate a problem with round-off error. The difficulty the SDM has arises because of the alternating perpendicular paths it uses to locate the minimum.

8.6.3 Conjugate Gradient Method for $\mathbf{Av} = \mathbf{b}$

To improve the convergence of the SDM we need to modify the direction of descent. We will still include the direction of steepest decent, but we will add in a term to adjust the direction slightly to prevent what happens in Figure 8.16. Our choice will be to start off with $\mathbf{d}_1 = \mathbf{r}_1$ but from then on take

$$\mathbf{d}_k = \mathbf{r}_k + \beta_{k-1}\mathbf{d}_{k-1}, \quad \text{for } \ k = 2, 3, \ldots . \tag{8.50}$$

So, the question is, how to pick β_{k-1}? The choice is based on the observation that in the SDM the direction of steepest descent at $\mathbf{v}_k$ is orthogonal to the steepest descent direction at $\mathbf{v}_{k-1}$; that is, $\mathbf{r}_k \cdot \mathbf{r}_{k-1} = 0$. This is evident in Figure 8.16 because of the right angle the path makes at each $\mathbf{v}_k$. In this two dimensional setting, because there are only two nonzero perpendicular directions, then the SDM keeps repeating its descent directions. In particular,

```
Pick:  $\mathbf{v}_1$
Let:   $\mathbf{r}_1 = \mathbf{b} - \mathbf{A}\mathbf{v}_1$
       $\mathbf{d}_1 = \mathbf{r}_1$
Loop   For $k = 1, 2, 3, \cdots, n$
         $\mathbf{q}_k = \mathbf{A}\mathbf{d}_k$
         $\alpha_k = \dfrac{\mathbf{r}_k \cdot \mathbf{r}_k}{\mathbf{d}_k \cdot \mathbf{q}_k}$
         $\mathbf{v}_{k+1} = \mathbf{v}_k + \alpha_k \mathbf{d}_k$
         $\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{q}_k$
         $\beta_k = \dfrac{\mathbf{r}_{k+1} \cdot \mathbf{r}_{k+1}}{\mathbf{r}_k \cdot \mathbf{r}_k}$
         $\mathbf{d}_{k+1} = \mathbf{r}_{k+1} + \beta_k \mathbf{d}_k$
       End
```

Table 8.3 Outline of the conjugate gradient method (CGM) used to solve the linear system $\mathbf{A}\mathbf{v} = \mathbf{b}$, where $\mathbf{A}$ is an $n \times n$ symmetric positive definite matrix.

the direction it uses at $\mathbf{v}_1$ is the same one it uses at $\mathbf{v}_3$, $\mathbf{v}_5$, $\mathbf{v}_7$, etc. This is the primary culprit in slowing down the SDM and we will use β_{k-1} to stop this.

The way we will prevent $\mathbf{r}_k$ from being in the same direction as $\mathbf{r}_{k-2}$ is to select β_{k-1} so that $\mathbf{r}_k$ is actually orthogonal to $\mathbf{r}_{k-2}$. To determine how this can be accomplished, consider the first three points produced by the algorithm: $\mathbf{v}_1$, $\mathbf{v}_2$, and $\mathbf{v}_3$. The residuals at these points are: $\mathbf{r}_1$, $\mathbf{r}_2 = \mathbf{r}_1 - \alpha_1\mathbf{q}_1$, and $\mathbf{r}_3 = \mathbf{r}_2 - \alpha_2\mathbf{q}_2$, where $\mathbf{q}_k$ is given in (8.41) and α_k in (8.40). Given that $\mathbf{r}_1 \cdot \mathbf{r}_2 = 0$ then to have $\mathbf{r}_1 \cdot \mathbf{r}_3 = 0$, we require that $\mathbf{r}_1 \cdot \mathbf{q}_2 = 0$. Substituting in $\mathbf{d}_2 = \mathbf{r}_2 + \beta_1\mathbf{d}_1$ into the formula for $\mathbf{q}_2$, and then simplifying one finds that

$$\beta_1 = \frac{\mathbf{r}_2 \cdot \mathbf{r}_2}{\mathbf{r}_1 \cdot \mathbf{r}_1}\,.$$

Continuing in this way yields the following formula:

$$\beta_k = \frac{\mathbf{r}_{k+1} \cdot \mathbf{r}_{k+1}}{\mathbf{r}_k \cdot \mathbf{r}_k}\,, \quad \text{for} \ \ k = 1, 2, 3, \ldots\,. \tag{8.51}$$

Another outcome of the above analysis is that $\mathbf{d}_k \cdot \mathbf{r}_k = \mathbf{r}_r \cdot \mathbf{r}_k$. The resulting algorithm is given in Table 8.3.

It is assumed here that $\mathbf{r}_k \cdot \mathbf{r}_k \neq 0$, but if this were not the case we would have solved the equation exactly at the previous step and there would be no need to calculate β_k. This choice of β_k, along with the descent direction in (8.50), produces what is known as the *conjugate gradient method* (CGM), and the steps involved are summarized in Table 8.3. In terms of operations

k	1	2	3	4	$\cdots$
Position	$\mathbf{v}_1$	$\mathbf{v}_2$	$\mathbf{v}_3$	$\mathbf{v}_4$	$\cdots$
Residual	$\mathbf{r}_1$	$\mathbf{r}_2$	$\mathbf{r}_3$	$\mathbf{r}_4$	$\cdots$
Descent	$\mathbf{d}_1$	$\mathbf{d}_2$	$\mathbf{d}_3$	$\mathbf{d}_4$	$\cdots$

Table 8.4 The three important vectors used in the CGM algorithm.

per iteration step it is not much more than SDM. To calculate β_k and then construct $\mathbf{d}_k$ adds about $4n$ flops per step.

The CGM has a remarkable property that has profound consequences for how well it works. At each step, three vectors are computed that are of importance, and these are the position ($\mathbf{v}_{k+1}$), the residual ($\mathbf{r}_{k+1}$), and the direction of descent ($\mathbf{d}_{k+1}$). These are listed in Table 8.4. As explained earlier, the descent direction $\mathbf{d}_3$ is selected so that the residual $\mathbf{r}_3$ is orthogonal to the previous residuals, $\mathbf{r}_1$ and $\mathbf{r}_2$. Similarly, $\mathbf{d}_4$ is selected so $\mathbf{r}_4$ is orthogonal to $\mathbf{r}_1$, $\mathbf{r}_2$, and $\mathbf{r}_3$. The result is that the residuals $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \ldots$ are *mutually orthogonal*, which means that $\mathbf{r}_i \cdot \mathbf{r}_j = 0$, $\forall i \neq j$. In n dimensions, the only way for $n+1$ vectors to be mutually orthogonal is that one of them is the zero vector. If a residual is zero, then we have found the exact solution. Therefore, the conjugate gradient method produces the exact solution in no more than $n+1$ steps! This is known as the *finite termination property.* In contrast, iterative methods such as the SDM, Newton's method, or the bisection method, are not guaranteed to stop and only approach the solution in the limit (if they converge at all). This discussion is summarized in the following theorem:

Theorem 8.2. *The conjugate gradient method, when used to solve* $\mathbf{Ax} = \mathbf{b}$*, where* $\mathbf{A}$ *is an* $n \times n$ *symmetric positive definite matrix, will find the exact solution in* m *steps, where* $m \leq n+1$*.*

The proof of this can be found in Nocedal and Wright [2006].

Examples

1. $$\begin{pmatrix} 3 & -2 \\ -2 & 4 \end{pmatrix}\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

   Taking $\mathbf{v}_1 = (1,\ 2)^T$, all of the steps produced with CGM are shown in the contour plot on the left in Figure 8.17. It is found that the method calculates $\mathbf{v}_2$ and $\mathbf{v}_3$, at which point the error is 5.6×10^{-17} and the algorithm stops. Note that this is as accurate as can be expected using double precision. It should also be noted that $\mathbf{v}_1$ and $\mathbf{v}_2$ are the same as for SDM shown in Figure 8.16. This is expected because CGM uses the steepest descent direction at the first step, so if the SDM and CGM are started at the same point then they will calculate the same value for $\mathbf{v}_2$. However, unlike what is obtained using the SDM, the descent direction at $\mathbf{v}_2$ used by the CGM is not perpendicular to the level curve. ■

2. $\begin{pmatrix} 5 & 4.99 \\ 4.99 & 5 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$

Taking $\mathbf{v}_1 = (1,\ 1)^T$, all of the steps produced with CGM are shown in the contour plot on the right in Figure 8.17. It is found that the method stops after calculating $\mathbf{v}_3$, with the resulting error being 7×10^{-12}. ■

In comparing the search paths with Figures 8.16 and 8.17, the contrast between SDM and CGM is striking. As advertised, the CGM took three steps to solve the 2×2 matrix equations, and in doing so it showed itself to be far superior to the SDM. However, the reality is that round-off generally prevents CGM from producing the exact result. This is not unexpected, nor a criticism of the method, because as with all floating-point computations we can only get as close as round-off permits. The issue with CGM is that it is built on the assumption of exact orthogonality of the residuals at each step. With round-off, however, this will not happen. For well-conditioned matrices this is not an issue, and the above examples demonstrate this fact. However, if the condition number of the matrix gets too large then it is an issue and CGM will lose its finite termination property. The usual fix for this is to use a preconditioner, and what this entails is described in Section 3.11.

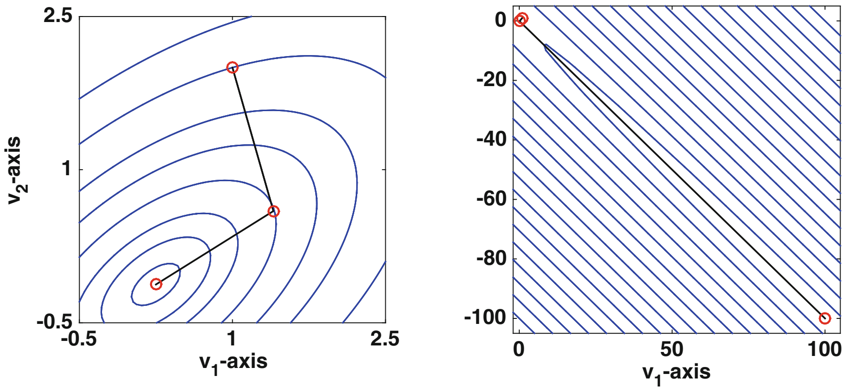


Figure 8.17 Contours of $F(v_1, v_2)$ and the paths taken by the conjugate gradient method in the two example problems.

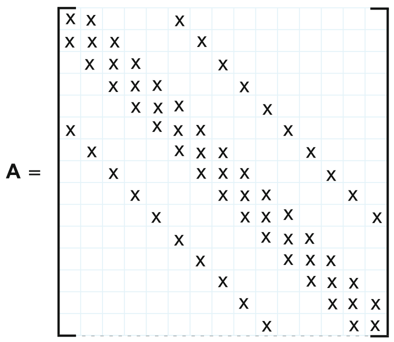


Figure 8.18 The x's designate the positions of the nonzero entries in the symmetric positive definite matrix used to test the convergence of the CGM.

8.6.3.1 Error and Rate of Convergence of CGM

One of the more important applications that produces large positive definite matrix equations involves numerically solving Laplace's equation. So, to investigate the effectiveness of the CGM for larger systems we will solve $\mathbf{Av} = \mathbf{b}$, where $\mathbf{A}$ is a matrix similar to those obtained for Laplace's equation on a rectangular domain. The matrix is illustrated schematically in Figure 8.18. It is symmetric and contains zeros everywhere except along 5 diagonals. In this example, the main diagonal entries are $a_{ii} = a$ and all the other nonzero entries are -1. Specifically, the super-diagonal entries are $a_{i,i+1} = -1$, the upper-most diagonal, which is m places to the right of the diagonal, are $a_{i,i+m} = -1$ and the nonzero entries below the diagonal can be determined from the symmetry of $\mathbf{A}$. Note that according to Theorem 3.4, in Section 3.7, this matrix is positive definite if $a > 4$.

To check on the accuracy of CGM we need to know the exact solution $\mathbf{v}_m$. This will be done by specifying $\mathbf{v}_m$ and then setting $\mathbf{b} = \mathbf{Av}_m$. This way we can observe the accuracy, and rate of convergence, of the CGM as it progresses. Also, there are various ways to measure error, and for this example we will consider the following:

$$\begin{aligned} \text{Error: } E_k &= \|\mathbf{v}_k - \mathbf{v}_m\|, \\ \text{Iteration Error: } I_k &= \|\mathbf{v}_k - \mathbf{v}_{k-1}\|, \\ \text{Residual Error: } R_k &= \|\mathbf{r}_k\|. \end{aligned}$$

The reason for considering I_k and R_k is that we would like to have some idea of whether I_k or R_k can be used as a reasonable substitute for E_k, and then using one of these quantities in the stopping condition in the code.

The results when $m = 10^2$ and $n = 10^4$ are shown in Figure 8.19. It shows that for this example the residual and iteration errors are effectively equivalent in their estimation of E_k. Except at the start, both underestimate the error and neither decreases monotonically. Another observation is that there are two stages in the convergence. For the first hundred steps, or so, the error reduction is modest. In fact, it is hard to tell if the error E_k decreases at all at the beginning. Shortly after that things change, and CGM converges relatively quickly. Where exactly this switch occurs depends on the eigenvalues of the matrix, and for most matrices this information is difficult to obtain. Those interested in how the rate of convergence depends on the eigenvalues should consult Nocedal and Wright [2006].

Another important point to make about Figure 8.19 concerns the number of iterations. According to Theorem 8.2, it could take up to $n + 1$ steps for the CGM to find the solution. Even so, the CGM has produced a reasonably accurate solution of the matrix equation in significantly fewer than n iterations.

To conclude, we consider the question of how the CGM compares with a direct solver for this example when n is large. An example of a direct solver is the Cholesky factorization (see Section 3.7.1). It is possible to take advantage of the structure of the matrix and use what is known as a band Cholesky factorization. For this example, the bandwidth for $\mathbf{A}$ is m. Assuming $m \ll n$, then solving the problem using a band Cholesky factorization requires about $nm(m + 7)$ flops [Golub and Van Loan, 2013]. Since $m = 10^2$ and $n = 10^4$, the flop count is about 10^8. For the CGM, in this example, calculating $\mathbf{v}_{k+1}$ takes approximately $19n$ flops. This means the CGM requires fewer flops if the solution can be found using fewer that about 520 iteration steps. According to Figure 8.19, if you are satisfied with an error of 10^{-4} or 10^{-8}, then the CGM takes fewer flops that band Cholesky, but if you want an error on the order of machine ε then band Cholesky looks to require fewer flops. Finally, in terms of memory, CGM requires storage of only the nonzero entries of $\mathbf{A}$ along with five n-vectors. Direct solvers cannot do better than this.

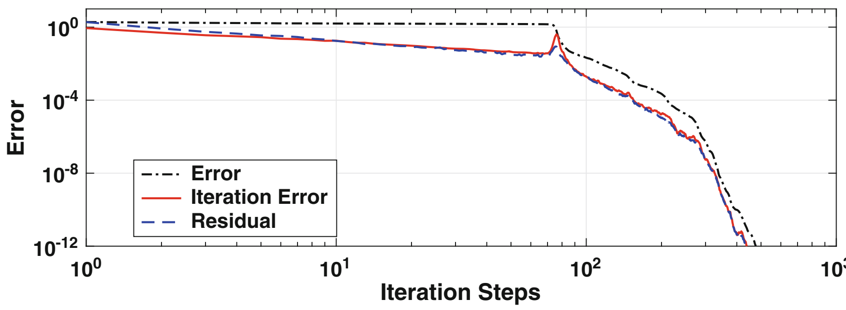


Figure 8.19 Error obtained at each iteration step of CGM to solve a equation involving the matrix illustrated in Figure 8.18. In this calculation, $a = 4.0001$, $m = 10^2$, and $n = 10^4$.

8.7 Descent Methods: General Nonlinear Problem

We now take up the problem of minimizing a general nonlinear function, and not necessarily a quadratic form as considered earlier. To be specific, we are going to consider how to numerically find a point $\mathbf{v}_m \in \mathbb{R}^n$ at which a given function $F(\mathbf{v})$ achieves a local minimum value. We are going to be using the gradient of F to determine descent directions, and so it is assumed that the function is smooth enough that this is possible.

The basic descent algorithm is: after picking a starting position $\mathbf{v}_1$ then

$$\mathbf{v}_{k+1} = \mathbf{v}_k + \alpha_k \mathbf{d}_k, \quad \text{for} \ \ k = 1, 2, 3, \ldots . \tag{8.52}$$

The complication now is how to pick directions of descent $\mathbf{d}_k$ and how to solve the line search problem to determine α_k.

8.7.1 Descent Direction

The vector $\mathbf{d}_k$ is a direction of descent at $\mathbf{v}_k$. This means that it satisfies (see Section 8.5.1)

$$\mathbf{d}_k \cdot \mathbf{g}_k < 0,$$

where $\mathbf{g}_k$ is the gradient vector at $\mathbf{v}_k$ and it is defined as

$$\mathbf{g}(\mathbf{v}) = \nabla F(\mathbf{v}).$$

As for possible descent directions, we have our two earlier favorites.

i) Steepest Descent Choice: $\mathbf{d}_k = -\mathbf{g}_k$

ii) Conjugate Gradient Choice: $\mathbf{d}_k = -\mathbf{g}_k + \beta_{k-1}\mathbf{d}_{k-1}$, where $\beta_0 = 0$ and otherwise

$$\beta_k = \frac{\mathbf{g}_{k+1} \cdot \mathbf{g}_{k+1}}{\mathbf{g}_k \cdot \mathbf{g}_k} . \tag{8.53}$$

Even though this choice for β_k corresponds exactly to what is given in the CGM algorithm, in nonlinear optimization it is known as the *Fletcher-Reeves method.*

Both of the above choices come directly from the case of when F is a quadratic form. There have been numerous others proposed, attempting to account for the non-quadratic nature of a general nonlinear function. One of particular note is

$$\beta_k = \frac{\mathbf{g}_{k+1} \cdot (\mathbf{g}_{k+1} - \mathbf{g}_k)}{\mathbf{g}_k \cdot \mathbf{g}_k} , \tag{8.54}$$

which gives rise to what is known as the *Polak-Ribière method.* In the case of when F is the quadratic form given in (8.39), $\mathbf{g}_{k+1} \cdot \mathbf{g}_k = 0$, so this formula reduces to the earlier choice. It is often stated that numerical testing shows (8.54) to be a better choice than using (8.53), and so it will be used in the examples to follow.

There are certainly other possibilities. For example, if $\mathbf{B}$ is a symmetric and positive definite matrix then one can take

$$\mathbf{d}_k = -\mathbf{B}\mathbf{g}_k.$$

Note that the steepest descent method is obtained if $\mathbf{B} = \mathbf{I}$. Another possibility is to take $\mathbf{B}^{-1}$ to be the Hessian of F at $\mathbf{v}_k$, which produces Newton's method for finding the minimum. To explain, the calculus solution for finding the minimum is to solve $\nabla F = \mathbf{0}$. As shown in Section 3.10, if Newton's method is used to solve this then you need to calculate the Jacobian of ∇F, and this is called the Hessian of F. For this to qualify for a descent method it is required that the Hessian be positive definite. Also, the matrix $\mathbf{B}$ in this case depends on $\mathbf{v}_k$, which means the LU factorization, or whatever method is used to solve the matrix equation, must be redone at each iteration step.

8.7.2 Line Search Problem

Determining the value of α_k is known as the line search problem. The basic objective is to find a value of α that reduces the value of

$$q(\alpha) = F(\mathbf{v}_k + \alpha\mathbf{d}_k).$$

It is usually not worthwhile to calculate an accurate value of α that minimizes $q(\alpha)$. This is because this is only one step in the iteration process, and will be mostly forgotten in the later iteration steps. What is needed is to find a value for α_k that reduces the function enough so the decent method continues to make good progress towards the solution.

With this in mind, most of the methods that are used start by sampling, which means that they have a method for picking a small number α's at which they evaluate $q(\alpha)$. They then use this information to determine α_k. How well this works depends on how clever they are in picking the α's.

To illustrate, we consider what is currently one of the "preferred" methods. The first step is to determine the line tangent to $q(\alpha)$ at $\alpha = 0$, and this is

$$\begin{aligned} q(\alpha) &\approx q(0) + \alpha\, q'(0) \\ &= F_k + m_k\alpha\,, \end{aligned}$$

where $m_k = \mathbf{g}_k \cdot \mathbf{d}_k$ is the slope of the line and $F_k = F(\mathbf{v}_k)$. Also note that the slope is negative because $\mathbf{d}_k$ is a descent direction. This tangent line, which is a linear function of α, is shown in Figure 8.20. A second line is also shown, and it is given as

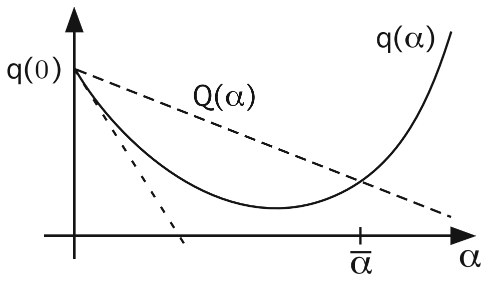


Figure 8.20 The lower dashed line is tangent to $q(\alpha)$ at $\alpha = 0$, and the upper dashed line is $Q(\alpha)$, which is given in (8.55). The goal of Armijo's method is to find a value of α that is close to $\overline{\alpha}$.

$$Q(\alpha) = F_k + \gamma\, m_k \alpha, \tag{8.55}$$

where $0 < \gamma < 1$ is a fixed number. So, Q is obtained from the tangent line, but it has a slope $\gamma\, m_k$ that is not as steep. A consequence of this is that near $\alpha = 0$ the curve $q(\alpha)$ is above the tangent line but below $Q(\alpha)$. As α increases, $q(\alpha)$ will eventually start to increase and intersect $Q(\alpha)$. Letting $\overline{\alpha}$ denote the value of α where this occurs, the algorithm to be described attempts to find a value of α that is closer to $\overline{\alpha}$ than to $\alpha = 0$. It does this by simply guessing a value a_1 for what $\overline{\alpha}$ might be. We want a_1 big enough that we keep making progress towards the minimum. However, once we get close to the minimum, a large value of a_1 can undo a lot of the progress we have made. The test used to decide if a_1 is too big is to check if $q(a_1) > Q(a_1)$. If so then a smaller value is tried, and the one used is $a_2 = \tau a_1$, where $0 < \tau < 1$. This is continued until one finds that $q(a_i) \leq Q(a_i)$, and this is the value used as the approximate solution of the line search problem.

The procedure described above is known as *Armijo's method*, and it consists of the following steps:

Step 0: Pick γ that satisfies $0 < \gamma < 1$ (e.g., $\gamma = \frac{1}{100}$),
and pick τ that satisfies $0 < \tau < 1$ (e.g., $\tau = \frac{1}{2}$).

Step 1: Pick a_1 that satisfies $0 < a_1$ (e.g., $a_1 = 1$)

Step 2: If $F(\mathbf{v}_k + a_i \mathbf{d}_k) < F_k + a_i\, \gamma\, \mathbf{g}_k \cdot \mathbf{d}_k$ then stop,
otherwise let $a_{i+1} = \tau a_i$ and repeat Step 2.

At termination, one takes $\alpha_k = a_i$.

To provide some insight into the parameters used here, note that if γ is close to zero then $Q(\alpha)$ is close to being horizontal. In contrast, if γ is chosen close to one then $Q(\alpha)$ gets closer to the tangent line shown in Figure 8.20. Because the goal is to try to make sufficient progress in the descent direction, the values usually suggested for γ are small, typically satisfying $10^{-3} \leq \gamma \leq 10^{-1}$. The value of τ determines how fast one reduces the value of a_i. Again, the goal is to not end up with a point near zero, and so the suggested values for τ usually satisfy $\frac{1}{2} \le \tau \le \frac{3}{4}$. Finally, a_1 is simply how far in the descent direction one starts the process.

One of its distinctive features of Armijo's method is that it uses a *back-tracking sampling method*, which means that it starts with the farthest point, determined by a_1, and then tries points that get progressively closer to $\alpha = 0$. This is done to try to maximize the step size in this particular direction. As a final comment, in the description of how an approximate solution of the line search problem is obtained, the issue of not wanting a_i either too big or too small came up (what you might call the Goldilocks problem). There is a more mathematical formulation of these requirements, and they are based on what are known as the Wolfe conditions. Those who are interested in learning more about this should consult Nocedal and Wright [2006].

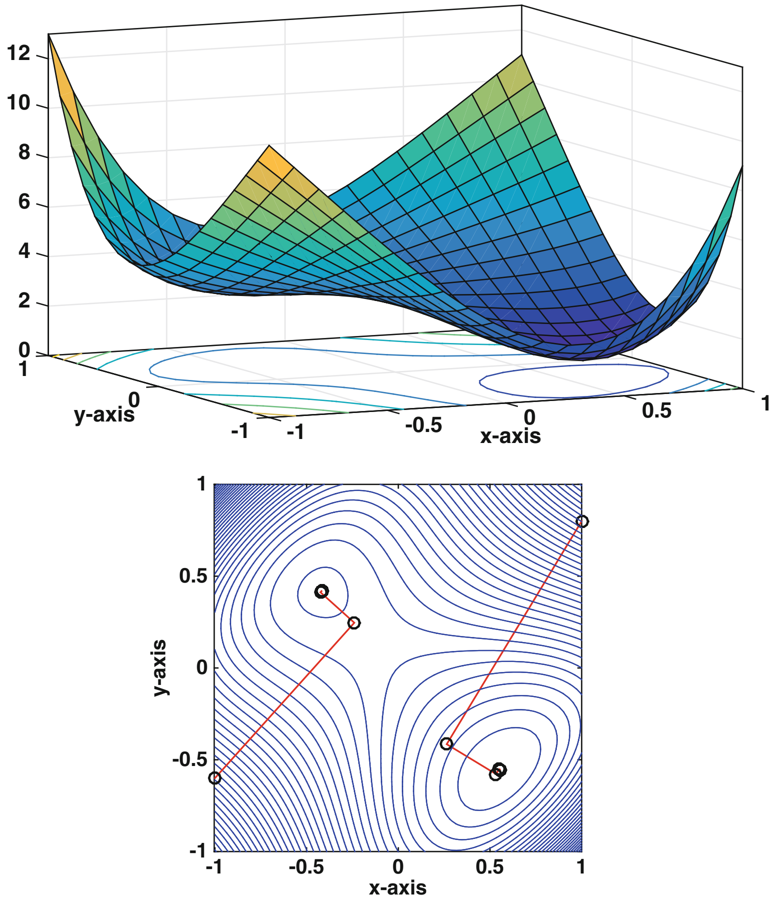
8.7.3 Examples

Example 1: Consider the function

$$F(x,y) = (x-y)^4 + 8xy - x + y + 3. \tag{8.56}$$

This is plotted in Figure 8.21, and it is evident that there are two local minimum points and a saddle point in-between them. We will try both the SDM and the CGM, using Polak-Ribière, on this problem. The SDM results are also shown in the figure, using two different starting points. The CGM results are shown as well, using the same two starting points. Note that the local minimum one finds depends on the location of the starting point, and the associated direction of steepest descent. Also, the difference between the SDM and CGM is most evident in the path on the right, which starts at $(x,y) = (1, 0.8)$. By construction the SDM selects directions that are perpendicular to the level contour it is located at, and this is seen clearly for the first and second point. In contrast, the CGM only does this at the first point but not at the second point. Finally, the Polak-Ribière method takes 6 iteration steps for the left path, while if you were to use the Fletcher-Reeves choice the number is 16 (a similar improvement also occurs for the right path). ■

It is worth commenting on SDM and CGM and the contours in Figure 8.21. For both methods, the first step is in the direction of the steepest descent. By following that direction it is possible to predict which minimum the method will find. The prediction is not certain because it is possible that the sampling in the line search will cause the method to jump over to the other minimum point. However, this did not happen in the four examples shown in Figure 8.21.



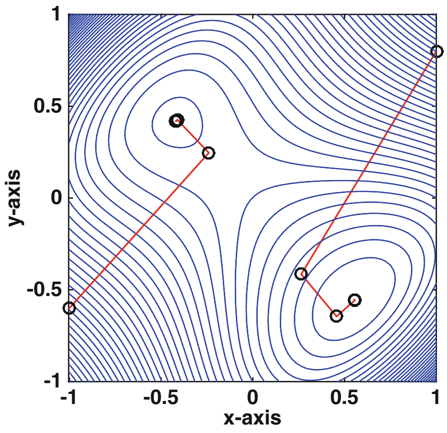


Figure 8.21 Top: Function given in (8.56). Middle: Decent paths, for two different starting points, obtained using SDM. Bottom: Decent paths, for two different starting points, obtained using Polak-Ribière version of the CGM.

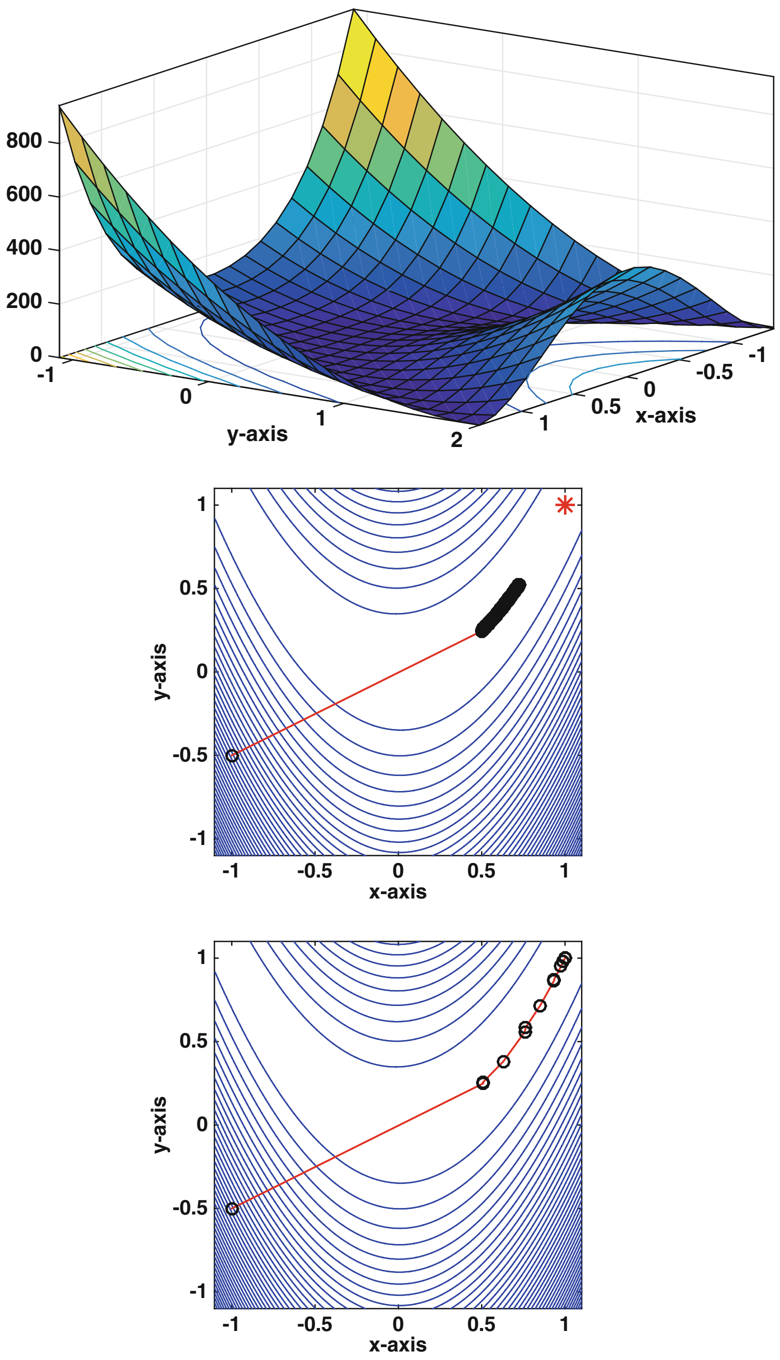


Figure 8.22 Top: Function given in (8.57). Middle: Decent path for the first 200 iteration steps using SDM. Bottom: Decent path obtained using the Polak-Ribière version of the CGM. The $*$ designates the location of the minimum.

Example 2: Consider the function

$$F(x, y) = 100(x^2 - y)^2 + (x - 1)^2. \tag{8.57}$$

This is known as the Rosenbrock function, and it is plotted in Figure 8.22. There is one minimum point and it is located at $(x, y) = (1, 1)$. The SDM and CGM (using Polak-Ribière) results are shown in Figure 8.22. The first 200 steps of the SDM are shown, and it is evident that it is very slow. If it is allowed to continue, it takes 11,529 iteration steps to achieve an error of less than 10^{-7}. In contrast, the CGM takes 15 steps to obtain a solution with an error of less than 10^{-7}. ■

As is evident in the above examples, the CGM looses its finite termination property for general nonlinear functions. However, it is still capable of finding the relative minimum points without much effort. The SDM, on the other hand, has many of the same difficulties seen for the quadratic case. On some problems it does just fine but it can be very slow on others, to the point that it is almost useless.

Example 3: As was explained earlier, it is possible to rewrite $\mathbf{Av} = \mathbf{b}$ as a minimization problem using

$$F(\mathbf{v}) = ||\mathbf{Av} - \mathbf{b}||_2^2.$$

In this example, this will be referred to as the least squares formulation. It has the advantage that it does not require the matrix to be symmetric or positive definite, unlike the equations considered in Section 8.6. So, the question arises as to how the CGM does solving the problem using the above function, versus the quadratic form in (8.39). To make this comparison it is required that the matrix be symmetric and positive definite, and we will use the matrix described in Section 8.6.3.1 with $a = 4.0001$, but now $n = 900$ and $m = 30$. Also, for the least squares problem, the Polak-Ribière version of the CGM is used. The error for the two methods is shown in Figure 8.23.

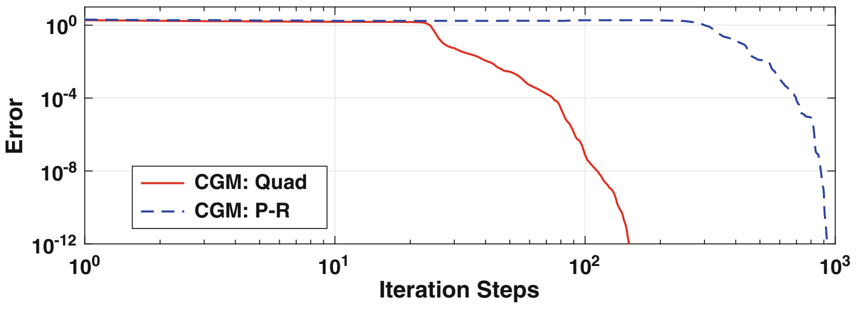


Figure 8.23 Comparison between using the CGM when solving $\mathbf{Av} = \mathbf{b}$ by minimizing the quadratic form or the least squares function.

It is not surprising that the CGM using the quadratic form finds the solution faster because it has the finite termination property, which the least squares version does not have. What the two versions do share is the ability to make advantage of the sparsity of the matrix, which can significantly reduce both the flop count per iteration step and the storage. ■

8.8 Minimization Without Differentiation

The last minimization method we will consider does not use the gradient, and is therefore applicable to functions that are just continuous. It is based on a simple geometrical argument and has probably been known since antiquity. However, the method described here is of more recent vintage, first proposed in 1965, and it is known as the *Nelder-Mead algorithm* [Nelder and Mead, 1965]. It is worth mentioning that according to Nelder, their method was not well received by some "professional optimizers" because of the lack of mathematical analysis [Nelder, 1979]. Another reason researchers did not take them seriously was their address, which was the National Vegetable Research Station. However, all is well now (almost) and it is one of the foundational methods in nonlinear optimization.

The basic idea underlying the method can be illustrated using either of the surfaces shown in Figures 8.21 and 8.22. Given three points on either surface, it's possible to construct an approximate direction of descent by positioning yourself at the highest point and then looking in a direction that passes between the other two points. The goal is to find a point in this direction that is lower than the highest value. A description of the procedure is given in Table 8.5 for the case of finding a local minimum of $F(\mathbf{v})$, where $\mathbf{v} \in \mathbb{R}^2$.

To explain some of the reasoning, in Step 1 the points are relabeled so $\mathbf{v}_3$ produces the largest value of the function. In Step 2 the midpoint between the two lower points, the centroid, is used to determine an approximate direction of descent by letting $\mathbf{d} = \mathbf{v}_c - \mathbf{v}_3$. Note that it is likely that this is a descent direction but it does not have to be, and contingencies are included later if this fails. With this, in Step 3, the procedure attempts to move in that direction by using reflection. If the value of the function at $\mathbf{v}_r$ is lower than all the other values then it will make a double step just to see if the function can be reduced even further (Step 4). However, if the value at $\mathbf{v}_r$ is not small enough then the procedure will try a contraction (Steps 5 or 6). If all this fails then there is a restart (Step 7), which is done by placing the new points near the point that currently produces the smallest value of the function. Note that, in each loop (from Step 1 back to Step 1), if a restart is not needed then the method evaluates the function F no more than twice.

Like all iteration methods, the question is when to stop. Given the geometric nature of the construction, a natural choice for a measure of the error is to use the area of the triangle. For example, given a tolerance *tol*, in Step 1

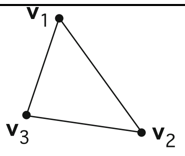
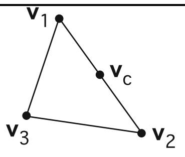
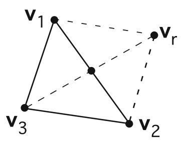
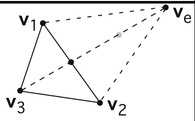
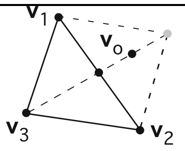
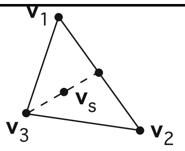
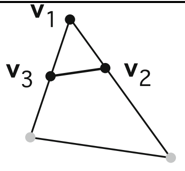
Step 0	Pick $\alpha > 0$ (e.g., $\alpha = 1$) and β with $0 < \beta < \alpha$ (e.g., $\beta = \frac{1}{2}$). Pick $0 < \delta < 1$ (e.g., $\delta = \frac{1}{2}$). Pick $\mathbf{v}_1$, $\mathbf{v}_2$, and $\mathbf{v}_3$ that are not collinear.	
Step 1	If necessary, relabel points so that $F_1 \le F_2 < F_3$, where $F_i = F(\mathbf{v}_i)$.	
Step 2	Calculate the centroid of the lowest side: $\mathbf{v}_c = \frac{1}{2}(\mathbf{v}_1 + \mathbf{v}_2)$, and calculate an approximate direction of descent: $\mathbf{d} = \mathbf{v}_c - \mathbf{v}_3$	 $\mathbf{v}_1$ $\mathbf{v}_c$ $\mathbf{v}_3$ $\mathbf{v}_2$
Step 3	Reflection: Calculate $\mathbf{v}_r = \mathbf{v}_c + \alpha\mathbf{d}$, and $F_r = F(\mathbf{v}_r)$. If $F_1 \le F_r < F_2$, let $\mathbf{v}_3 = \mathbf{v}_r$ and return to Step 1. If $F_r < F_1$ then go to Step 4 If $F_2 \le F_r < F_3$ then go to Step 5. If $F_3 \le F_r$ then go to Step 6.	 $\mathbf{v}_1$ $\mathbf{v}_r$ $\mathbf{v}_3$ $\mathbf{v}_2$
Step 4	Expansion: Calculate $\mathbf{v}_e = \mathbf{v}_c + 2\alpha\mathbf{d}$, and $F_e = F(\mathbf{v}_e)$. If $F_e < F_r$ then let $\mathbf{v}_3 = \mathbf{v}_e$ otherwise let $\mathbf{v}_3 = \mathbf{v}_r$. Return to Step 1.	 $\mathbf{v}_1$ $\mathbf{v}_e$ $\mathbf{v}_3$ $\mathbf{v}_2$
Step 5	Outside Contraction: Calculate $\mathbf{v}_o = \mathbf{v}_c + \beta\mathbf{d}$. If $F(\mathbf{v}_o) \le F_r$ then let $\mathbf{v}_3 = \mathbf{v}_o$ and return to Step 1, otherwise go to Step 7.	 $\mathbf{v}_1$ $\mathbf{v}_o$ $\mathbf{v}_3$ $\mathbf{v}_2$
Step 6	Inside Contraction: Calculate $\mathbf{v}_s = \mathbf{v}_3 + \beta\mathbf{d}$. If $F(\mathbf{v}_s) < F_3$ then let $\mathbf{v}_3 = \mathbf{v}_s$ and return to Step 1, otherwise go to Step 7.	 $\mathbf{v}_1$ $\mathbf{v}_s$ $\mathbf{v}_3$ $\mathbf{v}_2$
Step 7	Restart: Replace each $\mathbf{v}_i$ with $\mathbf{v}_1 + \delta(\mathbf{v}_i - \mathbf{v}_1)$ and return to Step 1.	

Table 8.5 Nelder-Mead algorithm for $\mathbf{v} \in \mathbb{R}^2$.

one can calculate the area A of the triangle and if $A < tol$ then the iteration is stopped and one uses $\mathbf{v}_1$ as the solution. The drawback with this is that the triangles may be elongated, so a somewhat better choice would be to use the distances between the three vertices.

Not every situation is covered by the above algorithm, and a complete version would need to explain, for example, what to do if $F_2 = F_3$, or if the three points calculated by this procedure become collinear. For those interested, the complete version is given in Lagarias et al. [1998] and Conn et al. [2009].

A few examples are given below using the Nelder-Mead algorithm outlined above. It will be seen that it finds the minimum fairly easily but the number of iteration steps is larger that what was needed when using the CGM. This is not surprising because some price is expected when using a method that works on a broader class of problems than the CGM (or any method that uses the gradient). However, the computational effort for Nelder-Mead is fairly low, usually requiring only 1 or 2 function evaluations per iteration.

8.8.1 Examples

Example 1: Consider the function

$$F(x, y) = 10x^2 + y^2. \tag{8.58}$$

The resulting surface is a paraboloid, which has a minimum at $(x, y) = (0, 0)$. The first five triangles produced by Nelder-Mead are shown in Figure 8.24. It is worth identifying how the steps in the algorithm produced these triangles. To get the 2nd, the first was reflected and then a double step was made. In other words, it comes from Step 4. The third comes from a single reflection, so Step 3 was used. In contrast, the 4th comes from an outside contraction, which is Step 5, and the same is true for the 5th triangle. Because the minimum point is now within the 5th triangle, it is expected that the majority of triangles produced by the method involves inside contractions (Step 6). If so then from this point on the method is reminiscent of the bisection method. This is because an inside contraction involves a single subdivision after which one picks which side the solution is on. Like the bisection method, this produces a dependable way to find the minimizer but it is not particularly fast in finding it. For this example, it takes the method 46 iteration steps to achieve an area error that is about 10^{-6}. ■

Example 2: Consider the function

$$F(x, y) = (x - y)^4 + 8xy - x + y + 3. \tag{8.59}$$

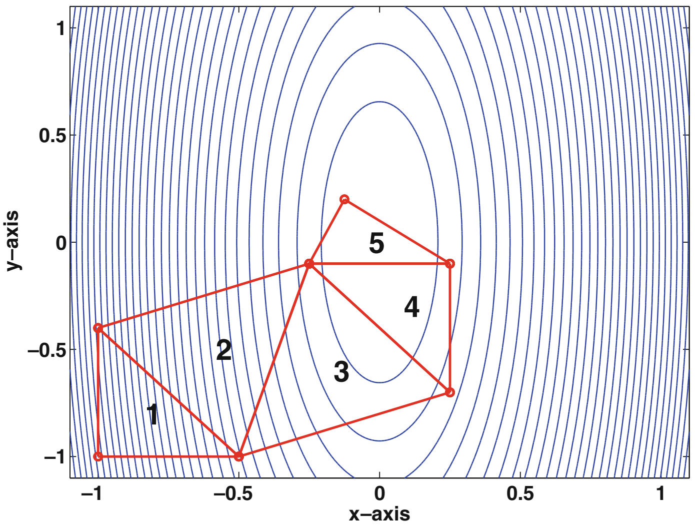


Figure 8.24 Triangles constructed using the Nelder-Mead algorithm to find the minimum of (8.58).

This surface is shown in Figure 8.21, and the first six triangles used by Nelder-Mead are shown in Figure 8.25. It takes the method 46 iteration steps to achieve an error that is less than 10^{-6}. ■

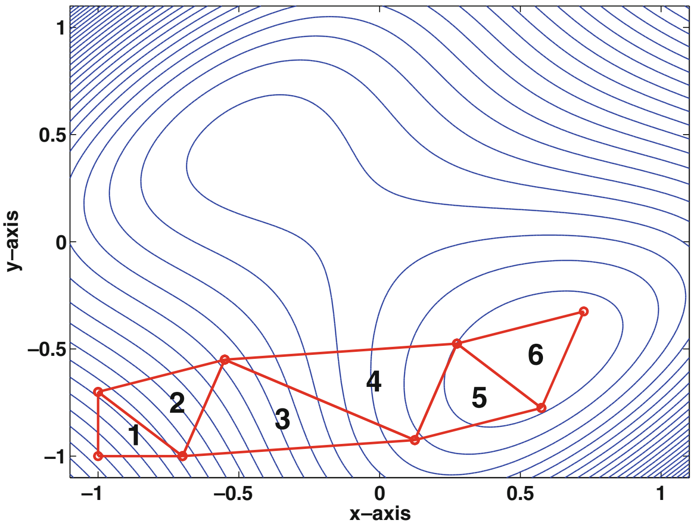


Figure 8.25 Triangles constructed using the Nelder-Mead algorithm to find the minimum of (8.59).

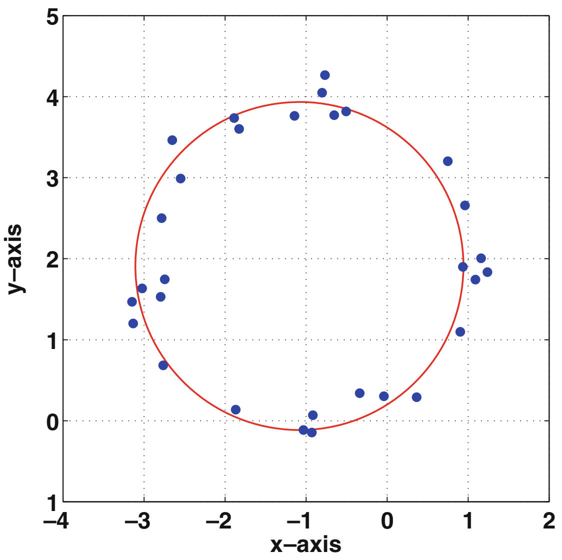


Figure 8.26 Data to be fitted by a circle, and the resulting circle obtained using least squares.

Example 3: We return to an example considered earlier, which is how to fit a circle to data. Unlike before, the circle is not centered at the origin, and a sample data set is shown in Figure 8.26. The equation for the circle is

$$(x-a)^2+(y-b)^2=r^2,$$

and the goal is to determine a, b, and r from the minimization procedure. For given values of these parameters, the error will be calculated using the difference between the radius and the distance between the center (a, b) and each data point (x_i, y_i). In other words, the error function is

$$F(a,b,r)=\sum_{i=1}^{n}\left(\sqrt{(x_i-a)^2+(y_i-b)^2}-r\right)^2, \tag{8.60}$$

which is a generalization of the function given in (8.25). If you try to use the calculus solution, which means finding the first derivatives and setting them to zero, you end up with three nonlinear equations. It is not possible to solve these by hand, but one very useful piece of information is determined. Namely, from the equation

$$\frac{\partial F}{\partial r}=0,$$

it is found that

$$r = \frac{1}{n}\sum_{i=1}^{n} \sqrt{(x_i - a)^2 + (y_i - b)^2}\,.$$

This is the same solution given in (8.26), but before it was assumed that $a = b = 0$. What this result does do is make it so (8.60) is a function of two parameters: a and b. This means the Nelder-Mead procedure outline in Table 8.5 can be applied to this problem (without having to extend it to three dimensions). The method takes 55 steps and concludes that $a = -1.170734$, $b = 2.079167$, and $r = 1.9822$. The resulting circle is plotted in Figure 8.26. The problem of fitting geometric shapes to data arises in numerous applications, and those interested in circles and ellipses should consult Gander et al. [1994] and Chernov and Lesort [2005]. Those interested in polygons should consider looking at Persson et al. [2006] and Watson [2007]. ■

8.9 Variational Problems

There are several well-known minimization principles in physics that can be used to characterize how something will behave. For example, Fermat's principle states that the path taken by a ray of light minimizes the travel time. Another is the principle of minimum potential energy which states that out of all possible admissible displacements an object can undergo, the one that is realized is the one that minimizes the potential energy of the system. The question considered here is how these can be solved using one or more of the numerical optimization methods we have considered earlier in this chapter. Note that most of these principles are used to derive what are called Euler-Lagrange equations using the calculus of variations. This is not done here, and the problems will be solved by considering only the minimization principle.

8.9.1 Example: Minimum Potential Energy

An elastic string occupies the interval $0 \leq x \leq 1$, and is held at its two endpoints. When a transverse load $w(x)$ is applied, the string deflects. Letting $u(x)$ be the vertical displacement of the string, then the potential energy is

$$V = \int_0^1 \left(\frac{1}{2}Tu_x^2 - wu\right)dx, \tag{8.61}$$

where T is the tension (it is a positive constant). According to the principle of minimum potential energy, the displacement $u(x)$ is the function which minimizes V [Weinstock, 1974]. Also, note that the string is being held at its ends, and so any possible solution of this problem is required to satisfy $u(0) = u(1) = 0$.

We will find the minimizing function by introducing grid points along the x-axis, and then use them to numerically determine u_x as well as the value of the integral. To help make it clear how these approximations are used, the problem is written as

$$V = \int_0^1 f(x)dx, \tag{8.62}$$

where

$$f(x) = \frac{1}{2}Tu_x^2 - wu. \tag{8.63}$$

Suppose five points are used, and they are $x_0 = 0$, $x_1 = 1/4$, $x_2 = 1/2$, $x_3 = 3/4$, and $x_4 = 1$. We will use the composite trapezoidal rule to compute V (see Table C.2), and so

$$V \approx h\left(\frac{1}{2}f_0 + f_1 + f_2 + f_3 + \frac{1}{2}f_4\right), \tag{8.64}$$

where $f_i = f(x_i)$. Centered, or symmetric, approximations will be used when possible to compute f_i. Specifically, using the first-order approximations in Table 7.1 (with $h = 1/4$)

$$\begin{aligned}
u_x(0)^2 &\approx \left(\frac{u(x_1) - u(x_0)}{h}\right)^2, \\
u_x(x_i)^2 &\approx \frac{1}{2}\left[\left(\frac{u(x_{i+1}) - u(x_i)}{h}\right)^2 + \left(\frac{u(x_i) - u(x_{i-1})}{h}\right)^2\right], \quad \text{for } i = 1, 2, 3, \\
u_x(1)^2 &\approx \left(\frac{u(x_4) - u(x_3)}{h}\right)^2.
\end{aligned} \tag{8.65}$$

The reasoning behind using a symmetric difference formula will be explained after the examples are complete.

Combining (8.64) and (8.65), and using the boundary conditions $u_0 = u_4 = 0$, we have that

$$V \approx \frac{T}{2h}\left[u_1^2 + (u_2 - u_1)^2 + (u_3 - u_2)^2 + u_3^2\right] - h(w_1u_1 + w_2u_2 + w_3u_3), \tag{8.66}$$

where $w_i = w(x_i)$. It is worth observing that the above expression is a quadratic function of the u_i's.

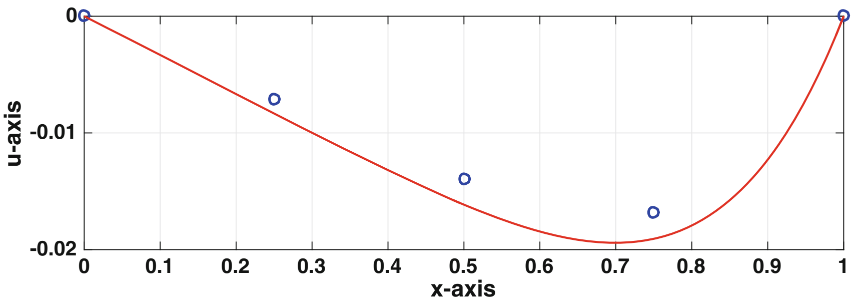


Figure 8.27 Numerical solution, the circles, for the function that minimizes (8.61) when using five grid points, and the exact solution, the solid (red) curve.

To find the values of u_1, u_2, and u_3 that minimize V we can use the CGM, Nelder-Mead, or the calculus solution. The latter is informative, and so we calculate the three derivatives

$$\frac{\partial V}{\partial u_i} \quad \text{for } i = 1, 2, 3$$

and then set them to zero. This leads to the matrix equation

$$\begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix} = \alpha \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix},$$

where $\alpha = h^2/T$. The above matrix is tri-diagonal, and this also happens when more grid points are used. Consequently, the equation can be solved very quickly using the Thomas algorithm (see Section 3.8).

Comparisons between the resulting solution and the exact solution are given in Figures 8.27 and 8.28 in the case of when $w = x^4$ and $T = 1$. The exact solution is

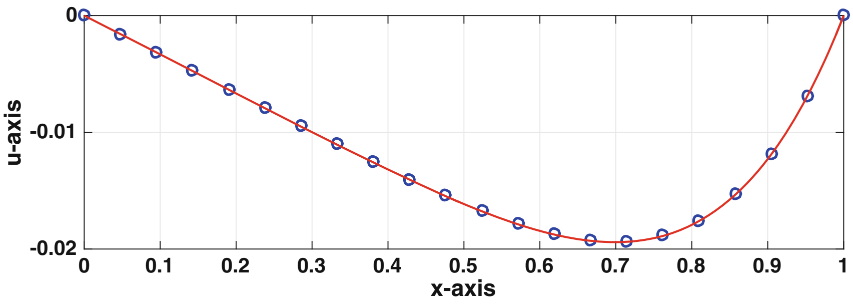


Figure 8.28 Numerical solution, the circles, for the function that minimizes (8.61) when using 22 grid points, and the exact solution, the solid (red) curve.

$$u = \frac{1}{30T}x(x^5 - 1).$$

Not unexpectedly, using 5 grid points does not produce a very accurate answer, but as more points are used the approximation closely matches the exact result. ■

8.9.2 Example: Brachistochrone Problem

Suppose two points A and B in a vertical plane are given, with A higher than B. A mass is going to move from A to B, while subjected to gravity, and the problem is to find the path it must take so the travel time is minimized. If A is directly above B the answer is easy, the mass should simply drop straight down. So, in what follows it is assumed that the two points are not on a vertical line. In particular, it is assumed that A is the origin, and B is the point $(1, b)$, where $b > 0$. It is assumed that y is positive in the downward direction. In this case, if the mass follows a path $y(x)$, then the travel time for that path is [Weinstock, 1974]

$$T = \int_0^1 f(x)dx, \tag{8.67}$$

where

$$f(x) = \sqrt{\frac{1}{2gy}\left[1 + \left(\frac{dy}{dx}\right)^2\right]}. \tag{8.68}$$

The objective is to find the path $y(x)$ that minimizes the value of this integral. In doing this it is required that $y(0) = 0$ and $y(1) = b$.

This problem is one of the first examples studied in the calculus of variations, but its numerical solution is not so simple. This is evident in (8.68) because the denominator is zero at the left endpoint, and this complicates using a numerical method. However, the situation is actually worse than this. To explain, the solution is found to be a cycloid, and it is given parametrically as

$$\begin{aligned} x &= A(\theta - \sin\theta), \\ y &= A(1 - \cos\theta), \end{aligned}$$

for $0 \le \theta \le \theta_M$. The values of A and θ_M are found from the requirements that $x = 1$ and $y = b$ at $\theta = \theta_M$. From this it is possible to show that near the left endpoint, $y \approx mx^{2/3}$, where m is a positive constant. This means that the $(y')^2$ term in the numerator in (8.68) is as bad as the y term in the denominator. As a final comment, the numerical solution will be compared to the above exact solution. To make the comparison, we will take $\theta_M = \pi/2$, which means that $A = b = 2/(\pi - 2)$.

One way to deal with the singularity at the left endpoint is to change variables in the integral. The particular choice will use the inverse function for y. If $y = f(x)$, then $x = f^{-1}(y)$. This means that $dx = (\frac{d}{dy}f^{-1})dy = \frac{dx}{dy}dy$ and $\frac{dy}{dx} = 1/\frac{dx}{dy}$. Substituting this into (8.67) we get that

$$T = \int_0^b F(y)dy, \tag{8.69}$$

where

$$F(y) = \sqrt{\frac{1}{2gy}\left[1 + \left(\frac{dx}{dy}\right)^2\right]}. \tag{8.70}$$

Note that the denominator is still zero at $y = 0$ but the $(\frac{dx}{dy})^2$ term is not singular at $y = 0$. This is because near the left end, $x \approx My^{3/2}$, where M is a positive constant. So, $(\frac{dx}{dy})^2 \approx (\frac{3}{2}M)^2 y$ and this is a continuous function at $y = 0$.

We will minimize (8.69) in a manner similar to what was done in the last example, which means a set of grid points will be used to find approximations for the derivative and integral. The difference now is that the grid points are placed along the y-axis, and we solve for the values of the x_i's. Also, to deal with the singularity at $y = 0$ we will separate it from the rest of the interval, and this will be done by writing

$$T = \int_0^\delta F(y)dy + \int_\delta^b F(y)dy, \tag{8.71}$$

where δ is small and positive. It should also be noted that the original requirements that $y(0) = 0$ and $y(1) = b$ are now written as $x(0) = 0$ and $x(b) = 1$.

The integral on the right in (8.71) has no singularity and the composite trapezoidal rule will be used to evaluate it. The grid points are $y_1 < y_2 < \cdots < y_{n+1}$, where $y_1 = \delta$, $y_{n+1} = b$, and $y_{i+1} - y_i = k$. The resulting approximation is

$$\int_\delta^b F(y)dy \approx k\left(\frac{1}{2}F_1 + F_2 + F_3 + \cdots + F_n + \frac{1}{2}F_{n+1}\right), \tag{8.72}$$

As before, centered, or symmetric, approximations will be used when possible to compute the derivative term in F. Specifically,

$$\left(\frac{dx}{dy}\right)^2 \approx \left(\frac{x(y_1)-x(y_2)}{k}\right)^2, \quad \text{for } i=1$$

$$\left(\frac{dx}{dy}\right)^2 \approx \frac{1}{2}\left[\left(\frac{x(y_{i+1})-x(y_i)}{k}\right)^2+\left(\frac{x(y_i)-x(y_{i-1})}{k}\right)^2\right], \quad \text{for } i=2,3,\cdots,n$$

$$\left(\frac{dx}{dy}\right)^2 \approx \left(\frac{x(y_{n+1})-x(y_n)}{k}\right)^2 \quad \text{for } i=n+1.$$

The left integral in (8.71) requires a bit more care. As explained in Section 6.2, one possibility is to pick a point c_0 in this interval and then use the approximation

$$\int_0^\delta F(y)dy \approx F(c_0)\delta.$$

We will do this for the "good" part of $F(y)$ but leave the term that is responsible for the singularity. Picking the midpoint, so $c_0=\delta/2$, leads to the approximation

$$\begin{aligned}\int_0^\delta F(y)dy &\approx \sqrt{\frac{1}{2g}\left[1+(x'(c_0))^2\right]}\int_0^\delta \frac{1}{\sqrt{y}}dy\\ &= \sqrt{\frac{2\delta}{g}\left[1+(x'(c_0))^2\right]},\end{aligned}$$

where

$$x'(c_0) \approx \frac{x(y_1)-x(0)}{\delta} = \frac{x(y_1)}{\delta}.$$

A comparison between the resulting numerical solution obtained from minimizing (8.69) and the exact solution is shown in Figure 8.29. In this calculation, $n=24$, $\delta=10^{-2}$, $b=2/(\pi-2)$, and the minimizer was found using

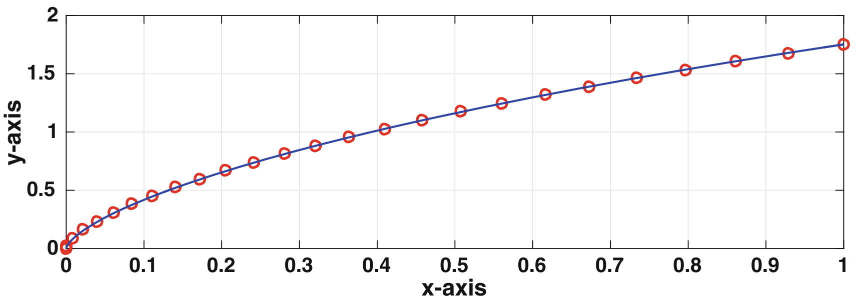


Figure 8.29 Numerical solution, the circles, of the brachistochrone problem, and the exact solution, the solid (blue) curve.

the Nelder-Mead algorithm. The initial guess used to start the solver is simply a straight line connecting the two points. As is evident in the figure, the solution is accurately computed using this procedure. ■

8.9.3 Parting Comments

The problem of finding a function which minimizes an integral looks to be straightforward, in the sense that it involves using numerical methods we have derived earlier. The fact is, however, that these problems are more finicky than what we have encountered elsewhere. This is one of the reasons for the symmetric differences used in the numerical procedure. To explain, both examples involve an integrand that includes a term of the form

$$\left(\frac{df}{dx}\right)^2 .$$

Without thinking about it too hard, one might try using a second-order difference of the form

$$\left(\frac{df}{dx}\right)^2 \approx \left(\frac{f_{i+1} - f_{i-1}}{2h}\right)^2 .$$

If you write this out for each grid point, you will notice that the values of f at the even nodes do not depend on the values of f at the odd nodes. This is a well-known problem in numerical fluid dynamics, but the solutions used for fluids (staggered grids, artificial dissipation, etc) are not directly applicable to the integrals considered here. This leads to dropping the idea of using a second-order approximation and reverting to just first-order differences

$$\left(\frac{df}{dx}\right)^2 \approx \left(\frac{f_{i+1} - f_i}{h}\right)^2 .$$

or

$$\left(\frac{df}{dx}\right)^2 \approx \left(\frac{f_i - f_{i-1}}{h}\right)^2 .$$

In minimization problems, the boundary conditions play a critical role in the formula for the quantity being minimized as well as determining the function which is the minimizer. The forward and backward differences given above have a biased direction implicit in their formulation, and this causes problems when interacting with the boundary conditions. This is the reason for looking for symmetric formulas, and examples are

$$\left(\frac{df}{dx}\right)^2 \approx \left|\frac{f_{i+1} - f_i}{h}\frac{f_i - f_{i-1}}{h}\right| ,$$

and

$$\left(\frac{df}{dx}\right)^2 \approx \frac{1}{2}\left[\left(\frac{f_{i+1}-f_i}{h}\right)^2 + \left(\frac{f_i-f_{i-1}}{h}\right)^2\right].$$

The second has the distinct advantage of being a smooth function of the f_i's, and for this reason was used in the formulation.

There have been numerous studies related to deriving numerical solutions that minimize integrals, and most discuss the difficulties encountered. For those interested, you might consult Dussault [2014], Levin et al. [2002], and Jameson and Vassberg [2001].

As a final comment, for the string example the calculus method was used to find the minimum but this was not done for the brachistochrone problem. The reason is that (8.66) is a quadratic function of the unknown u_i's, so setting the derivatives to zero results in a linear problem. For the brachistochrone problem, T is a non-quadratic function of the unknown x_i's, so the calculus method results in a system of nonlinear equations. Because of the difficulty of solving such a system, it is easier to just find the minimum directly, which was done using the Nelder-Mead algorithm.

8.10 Global Minimum

Although the objective is to find the global minimum of a function, assuming it exists, for many nonlinear functions this can be a very difficult thing to determine. All of the methods considered here are good at working with local minima, but they provide little if any information about whether the solution is the global minimum point. There are methods that can be used for this, such as simulated annealing or an evolutionary computational algorithm, but they are fairly slow.

Exercises

8.1. Rewrite the following as a minimization problem (you do not need to solve the problem).
(a) Solve:

$$\begin{aligned}(x-2)^4 + (x-2y)^2 &= 5\\ x^2 - y &= 0\end{aligned}$$

(b) Solve:

$$x + y^2 + e^{x+y} = 3$$
$$2y + 3x = \frac{x + y}{x^2 + y^2 + 1}$$

(c) Find the distance between $y = 3e^{-x} + 5x + 1$ and the point $(-1, -2)$.
(d) The positions of two objects moving in the x, y-plane are: $(x_1, y_1) = (t + \sin(3t), t + 3\cos(3t))$, and $(x_2, y_2) = (4\sin t, t^2 - 3)$. How close do they come to bumping into each other?
(e) Two points $\mathbf{p}$ and $\mathbf{q}$ are on a surface $z = f(x, y)$. Find a curve on this surface which connects these two points, and which has the smallest length.
(f) Consider a region $0 \le x \le a$, $0 \le y \le b$. Three heaters are going to placed in this region, at points $\mathbf{h}_1$, $\mathbf{h}_2$, and $\mathbf{h}_3$. The resulting temperature T at any point $\mathbf{x}$ in the region is

$$T(\mathbf{x}) = T_1F(||\mathbf{x} - \mathbf{h}_1||_2) + T_2F(||\mathbf{x} - \mathbf{h}_2||_2) + T_3F(||\mathbf{x} - \mathbf{h}_3||_2),$$

where $F(z) = 1/(1+z^2)$. Where should the heaters be placed in the region so the coldest point in the room is as hot as possible?

8.2. Given the data in Table 8.6, determine the least squares approximation of the form $f(x) = a + b\sin(x)$.

8.3. The function $y(t) = v_1t^2 + v_2(2-t)^3$ is to be fit to the data in Table 8.7.

(a) Find the values of v_1 and v_2 so the function best fits the data using least squares error.
(b) Explain why the following function would not be a good error function to use with this problem.

$$E(v_1, v_2) = \sum_{i=1}^{3}(y(t_i) - y_i)^3.$$

x_i	$-\pi/2$	0	$\pi/6$
f_i	2	0	-1

Table 8.6 Data for Exercise 8.2.

8.4. This exercise considers what happens when there is only one parameter when data fitting using linear least squares. Assume here that $f(x)$ is a given function.

(a) Suppose the model function is $y = v_1f(x)$. Using least squares error, what is the resulting formula for v_1? In doing this, assume that there is at least one data point with $f(x_i)$ nonzero.

t_i	0	1	2
y_i	2	0	−1

Table 8.7 Data for Exercise 8.3.

θ_i	0.00	1.57	3.14	4.71	6.28
r_i	0.62	1.23	16.14	1.35	0.62

Table 8.8 Data for the comet 27P/Crommelin, used in Exercise 8.5.

(b) Suppose the model function is $y = v_1 + f(x)$. Using least squares error, what is the resulting formula for v_1?

8.5. The elliptical path of a comet is described using the equation

$$r = \frac{p}{1 + \varepsilon \cos\theta},$$

where r is the radial distance to the Sun, θ is the angular position, ε is the eccentricity of the orbit, and p is a rectum parameter. In this exercise you are to use data for the comet 27P/Crommelin, which is given in Table 8.8.

(a) Writing the model function as $r = g(\theta)$, and using least squares, then the error function is

$$E(p, \varepsilon) = \sum_{i=1}^{n} [g(\theta_i) - r_i]^2.$$

What two equations need to be solved to find the value(s) of p and ε that minimize this function?

(b) By writing the model function as

$$\frac{1}{r} = \frac{1 + \varepsilon \cos\theta}{p},$$

explain how the nonlinear regression problem can be transformed into one with the model function $R = V_1 + V_2 \cos(\theta)$. Also, what happens to the data values?

(c) Writing the model function in part (b) as $R = G(\theta)$, and using the least squares error function

$$E(V_1, V_2) = \sum_{i=1}^{n} [G(\theta_i) - R_i]^2,$$

compute V_1 and V_2. Using these values, determine p and ε.

(d) Redo part (c) but use the relative least squares error function

$$E_R(V_1, V_2) = \sum_{i=1}^{n} \left(\frac{G(\theta_i) - R_i}{R_i} \right)^2 .$$

(e) Does part (c) or does part (d) produce the better answer? You need to provide an explanation for your conclusion, using a quantified comparison, graphs, and/or other information.

8.6. The exercise considers fitting data using the model function

$$g(x) = v_1 x^{v_2},$$

which is known as a power law function, and also as an allometric function. Two different methods are considered, one summarized in (a) and (b), and the second in (c) and (d). Assume that the data are (x_1, y_1), (x_2, y_2), $\cdots$, (x_n, y_n), where the x_i's and y_i's are positive.

(a) Writing $y = v_1 x^{v_2}$, and then taking the log of this equation, show that the transformed model function can be written as $G(X) = V_1 + V_2 X$, where $V_1 = \log v_1$ and $V_2 = v_2$. Also, show that the transformed data points (X_i, Y_i) are $X_i = \log x_i$ and $Y_i = \log y_i$.

(b) Continuing from part (a), using the least squares error $E(V_1, V_2) = \sum (V_1 + V_2 X_i - Y_i)^2$, and the common log, show that $v_1 = 10^{V_1}$ and $v_2 = V_2$, where V_1 and V_2 are given in (8.31).

(c) Show that to minimize the error function $E(v_1, v_2) = \sum (v_1 x_i^{v_2} - y_i)^2$, one gets that

$$v_1 = \frac{\sum y_i x_i^{v_2}}{\sum x_i^{2v_2}} .$$

(d) Continuing from part (c), show that finding the minimum of $E(v_1, v_2)$ reduces to solving an equation of the form $F(v_2) = 0$. Write down the function F, and explain why the secant method might be easier to use to solve the equation than Newton's method.

8.7. Allometric functions of the form

$$g(x) = v_1 x^{v_2},$$

are often used by experimentalists in biology and ecology. This exercise explores such an application, and the data are given in Table 8.9. What is given, for each animal, is its typical mass and its maximum relative speed. The latter is the animal's maximum speed divided by its body length. This exercise uses the results from Exercise 8.6(a),(b).

(a) Taking x to be the mass, fit the power law to the data in Table 8.9 and then plot the data and power law curve using a log-log plot.

(b) Based on your result from part (a), what was the running speed of a Tyrannosaurus rex?

Animal	Mass (kg)	Relative Speed (1/s)
canyon mouse	1.37e−02	39.1
chipmunk	5.10e−02	42.9
red squirrel	2.20e−01	20.5
snowshoe hare	1.50e+00	35.8
red fox	4.80e+00	28.7
human	7.00e+01	7.9
reindeer	1.00e+02	12.7
wildebeest	3.00e+02	11.0
giraffe	1.08e+03	3.8
rhinoceros	2.00e+03	1.8
elephant	6.00e+03	1.4

Table 8.9 Data for Exercise 8.7 adapted from Iriarte-Díaz [2002].

(c) Taking x to be the relative speed, fit the power law to the data in Table 8.9 and then plot the data and power law curve using a log-log plot.

(d) If speed $= \alpha(\text{mass})^\beta$, then mass $= a(\text{speed})^b$, where $b = 1/\beta$. Based on this, one might think that the exponents from parts (a) and (c) satisfy $b = 1/\beta$. Do they? Using a sketch similar to the ones in Figure 8.5, but in the X, Y-plane, explain why the error functions are different in the two cases. Because of this, it is not expected that $b = 1/\beta$.

8.8. The computing times for three matrix algorithms are given in Table 8.10, which is adapted from Table 4.13. The assumption is that the computing time T can be modeled using the function

$$T = \alpha N^\beta.$$

The goal of this exercise is to use regression to find α and β. Note that you do not need to know anything about how the matrix methods work to do this exercise.

(a) Using the results from Exercise 8.6(a),(b), fit the model function to the LU times, and then plot the model function and data on the same axis.

(b) Show that to minimize the least square error $E = \sum[T(N_i) - T_i]^2$ one gets that

$$\alpha = \frac{\sum T_i N_i^\beta}{\sum N_i^{2\beta}}.$$

(c) It would seem reasonable to expect that the computing time is a reflection of the flops used to calculate the factorization. If so, then β should be a

N	LU	QR	SVD
200	0.0003	0.0007	0.0062
400	0.0009	0.0025	0.0238
600	0.0025	0.0072	0.0591
800	0.0050	0.0143	0.1181
1000	0.0094	0.0264	0.2163
2000	0.1067	0.1881	2.5702
4000	0.4107	1.3351	14.2420

Table 8.10 Data for Exercise 8.8. Computing time, in seconds, for a LU, QR, and SVD factorization of a random $N \times N$ matrix using MATLAB.

positive integer. Using the resulting from part (b), and assuming β is a positive integer, find α and β. Also, using these values, plot the model function and data on the same axis. Explain how you find β, and also comment on how the assumption that β is an integer affects how well the model function fits the data.

(d) Redo (a) and (c) for the QR times.
(e) Redo (a) and (c) for the SVD times.

8.9. Typically, in reporting experimental data, at any given x_i, the mean y_i and standard deviation σ_i are given. Examples are shown in Figures 5.1 and 8.11. The objective of this exercise is to derive a regression procedure that produces a predicted value y that: i) of foremost importance, falls within the interval $y_i - \sigma_i < y < y_i + \sigma_i$, and ii) of secondary importance, has $y \approx y_i$.

(a) Setting

$$E_i = \left(\frac{y - y_i}{\sigma_i} \right)^p ,$$

on the same axes, sketch E_i as a function of y for $p = 2, 4, 8$. Use this to explain why taking a larger p has the effect of giving more weight to objective (i).

(b) Does E_i satisfy the three expected properties of an error function given in Section 8.2.2?

(c) Taking $p = 2$, $y = a + bx$, and the error function

$$E = \sum_{i=1}^{n} E_i ,$$

find a and b that minimize E.

(d) Suppose one gets better, and presumably more expensive, experimental equipment so the standard deviations are all reduced by a factor of, say, 10. Assuming the same y_i values are obtained, explain why the answer in part (c) is unchanged.

8.10. A matrix $\mathbf{A}$ is negative definite if $-\mathbf{A}$ is positive definite. Explain why the algorithm in Table 8.3 can be used to solve $\mathbf{Av} = \mathbf{b}$ if $\mathbf{A}$ is an $n \times n$ symmetric negative definite matrix.

8.11. Consider the equation

$$\begin{pmatrix} 2 & -1 \\ -1 & 4 \end{pmatrix}\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} 7 \\ 0 \end{pmatrix}.$$

(a) What is the quadratic form associated with this equation? Write it out as a polynomial.
(b) In this question you are to use the SDM. Taking $\mathbf{v}_1 = (1,\ 1)^T$, calculate $\mathbf{v}_2$.
(c) In this question you are to use the CGM. Taking $\mathbf{v}_1 = (1,\ 1)^T$, calculate $\mathbf{v}_2$ and $\mathbf{v}_3$.

8.12. Consider the equation

$$\begin{pmatrix} 3 & 1 \\ 1 & 2 \end{pmatrix}\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}.$$

(a) What is the quadratic form associated with this equation? Write it out as a polynomial.
(b) In this question you are to use the SDM. Taking $\mathbf{v}_1 = (-1,\ 2)^T$, calculate $\mathbf{v}_2$.
(c) In this question you are to use the CGM. Taking $\mathbf{v}_1 = (-1,\ 2)^T$, calculate $\mathbf{v}_2$ and $\mathbf{v}_3$.

8.13. The equation $\mathbf{Av} = \mathbf{0}$ is going to be solved using one of the following matrices:

$$\mathbf{A}_1 = \begin{pmatrix} -1 & 1 \\ 0 & 2 \end{pmatrix}\ \mathbf{A}_2 = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}\ \mathbf{A}_3 = \begin{pmatrix} 2 & 1 \\ -1 & 1 \end{pmatrix}\ \mathbf{A}_4 = \begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix}\ \mathbf{A}_5 = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}$$

(a) If the CGM is going to be used to solve the equation, only one matrix can be used. Which matrix is it? Make sure to explain why.
(b) Picking the matrix from part (a), and assuming that $\mathbf{v}_1 = (1\,,\,0)^T$, find $\mathbf{v}_2$ and $\mathbf{v}_3$.

8.14. Letting $F(\mathbf{v}) = \frac{1}{2}\mathbf{v}^T\mathbf{Bv} - \mathbf{b}\cdot\mathbf{v}$, suppose that $\nabla F = \mathbf{0}$ yields the equation $\mathbf{Av} = \mathbf{b}$. It was shown that for this to happen $\mathbf{A}$ must be symmetric. The purpose of this exercise is to show that the properties of **B** are more flexible. To do this, assume that

$$\mathbf{A} = \begin{pmatrix} 2 & 1 \\ 1 & 3 \end{pmatrix}.$$

(a) Show that **A** is positive definite.
(b) Find a nonsymmetric **B**.
(c) Find a symmetric but not positive definite **B**.
(d) Is it possible for **B** to be diagonal?
(e) Is it possible for **B** to be non-invertible?

8.15. The exercise considers a least squares problem using the relative error. As usual, the data are (x_1, y_1), (x_2, y_2), $\cdots$, (x_n, y_n). The error function in this case is

$$E_R = \sum_{i=1}^{n} \left(\frac{g(x_i) - y_i}{y_i} \right)^2 .$$

It is assumed here that the y_i's are nonzero.
(a) In the case of when $g(x) = v_1 + v_2 x$, show that the minimum occurs when

$$\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \frac{1}{ad - b^2} \begin{pmatrix} a & -b \\ -b & d \end{pmatrix} \begin{pmatrix} \sum 1/y_i \\ \sum x_i/y_i \end{pmatrix},$$

where $a = \sum x_i^2/y_i^2$, $b = \sum x_i/y_i^2$, and $d = \sum 1/y_i^2$.
(b) Calculate v_1 and v_2 for the data in Table 8.1. On the same axis, plot the resulting line, the data points, and the line found using (8.12).

8.16. The exercise considers the least squares problem using the general linear model function $g(x) = v_1 p(x) + v_2 q(x)$. As usual, the data are (x_1, y_1), (x_2, y_2), $\cdots$, (x_n, y_n), where $n > 2$.
(a) Using the error function

$$E(v_1, v_2) = \sum_{i=1}^{n} [g(x_i) - y_i]^2,$$

show that

$$\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \frac{1}{\sum p_i^2 \sum q_i^2 - (\sum p_i q_i)^2} \begin{pmatrix} \sum p_i^2 & \sum p_i q_i \\ \sum p_i q_i & \sum q_i^2 \end{pmatrix} \begin{pmatrix} \sum p_i y_i \\ \sum q_i y_i \end{pmatrix},$$

where $p_i = p(x_i)$ and $q_i = q(x_i)$.
(b) Show that the solution in part (a) reduces to the one given in (8.12) in the case of when $p(x) = 1$ and $q(x) = x$.
(c) Suppose the model function is $g(x) = v_1 x + v_2 x^2$ and the data is given in Table 8.11. Find v_1 and v_2.

x_i	$-\pi$	0	π
y_i	1	0	2

Table 8.11 Data for Exercise 8.16.

(d) Suppose the model function is $g(x) = v_1 \sin(x) + v_2 \sin(2x)$ and the data is given in Table 8.11. Explain why it is not possible to determine v_1 and v_2.

(e) The question is, what requirement is needed to prevent the problem in part (d), or more generally what is needed so the solution in part (a) is defined. Setting $\mathbf{p} = (p_1, p_2, \cdots, p_n)$ and $\mathbf{q} = (q_1, q_2, \cdots, q_n)$, explain why the needed condition is that the angle θ between $\mathbf{p}$ and $\mathbf{q}$ is not 0 or π. Note that if either $\mathbf{p}$ or $\mathbf{q}$ is the zero vector then $\theta = 0$.

(f) What is the matrix $\mathbf{A}$, as defined in (8.16), for this problem? Explain why the condition derived in part (e) is equivalent to the requirement that the columns of $\mathbf{A}$ are independent.

(g) What are $\mathbf{p}$ and $\mathbf{q}$ for part (d)?

(h) If $n = 3$ and $\mathbf{p} = (1, -1, 1)$, give two nonzero examples for $\mathbf{q}$ where the solution in part (a) is not defined.

8.17. The question considered in this exercise concerns whether the solution obtained using regression reduces to the solution obtained using interpolation in the case of when the number of data points equals the number of parameters.

(a) The least squares solution in (8.12) is for the model function $g(x) = v_1 + v_2 x$. Show that the solution reduces to the point-slope formula in the case of when $n = 2$ and $x_1 \neq x_2$. In other words, the regression solution produces the interpolation solution.

(b) In the case of when $n = 2$, $x_1 \neq x_2$, and $g(x) = v_1 + v_2 x$, show that the values of v_1 and v_2 obtained using the point-slope formula produce the minimum error for E_1, E_∞, and E_D (these are defined in Section 8.3.3).

8.18. This exercise explores fitting an ellipse to a data set, and an example of such data is given in Figure 8.30. The formula for the ellipse is

$$\left(\frac{x}{a}\right)^2 + \left(\frac{y}{b}\right)^2 = 1,$$

where a and b are positive and to be determined from the curve fitting. In what follows you are to find an error function and from this find a and b.

(a) For a given a and b, write down an expression that can be used as a measure of the error between a data point (x_i, y_i) and the curve. The requirements on this expression are: it is non-negative and only zero if (x_i, y_i) is on the ellipse.

(b) Using your expression from part (a), sum over the data points to produce an error function $E(a, b)$.
(c) Minimize $E(a, b)$ and find explicit formulas for a and b. Note that if your error function does not result in you being able to find explicit formulas, then you need to redo parts (a) and (b) so you can.

8.19. In this problem, assume that you are given data (x_i, y_i), with $i = 1, 2, \cdots, n$, and a nonlinear model function $g(x)$ that contains two parameters: v_1 and v_2. The objective is to find a change of variables from (x, y) to (X, Y) so the transformed model function has the form $G(X) = V_1 + V_2 X$. Find a change of variables that accomplishes this for the following model functions, give the resulting solution for V_1 and V_2, and the corresponding solution for v_1 and v_2.
(a) $g(x) = v_1 x e^{v_2 x}$
(b) $g(x) = 1/(v_1 + v_2 x)$
(c) $g(x) = v_1 \exp(-v_2 x^2)$

8.20. Consider the function

$$F(x, y) = -x^4 + \frac{1}{6}x^6 - 10xy + y + y^4.$$

(a) Plot the surface and contour curves for this function for $-4 \leq x \leq 4$ and $-4 \leq y \leq 4$.
(b) Use the SDM to find the (global) minimum point for this function. Make sure to state what stopping condition you used, the initial point, and the number of iterations needed.
(c) Redo (b) but use the Polak-Ribière method. Also, you must use the same starting point(s) as in part (b).

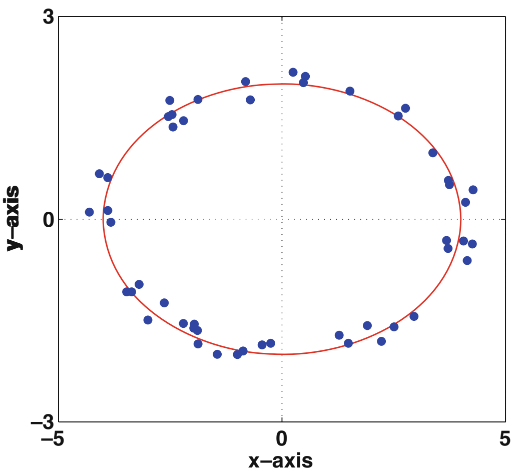


Figure 8.30 Example data set to be fitted by elliptic curve in Exercise 8.18.

8.21. Consider the function

$$F(x,y) = \frac{1}{10}(x+y)^4 + (x-1)^2 + 4y^2.$$

(a) Plot the surface and contour curves for this function for $-2 \leq x \leq 2$ and $-2 \leq y \leq 2$.
(b) Use the SDM to find the (global) minimum point for this function. Make sure to state what stopping condition you used, the initial point, and the number of iterations needed.
(c) Redo (b) but use the Polak-Ribière method. Also, you must use the same starting point(s) as in part (b).

8.22. Consider the problem of minimizing the following function:

$$f(x,y) = 2x^2 + 2xy + y^2 - x - 2y.$$

(a) If $\mathbf{v}_1 = (0,0)$ then find $\mathbf{v}_2$ and $\mathbf{v}_3$ using the steepest descent method.
(b) If $\mathbf{v}_1 = (0,0)$ then find $\mathbf{v}_2$ and $\mathbf{v}_3$ using the Polak-Ribière method.

8.23. The SDM is used to find the minimum of a function $F(\mathbf{v})$, and the contours for this function are shown in Figure 8.31.
(a) Shown in the plot is the second point $\mathbf{v}_2$ calculated using the SDM. If the first point $\mathbf{v}_1$ is located on the contour labeled with 28, where is $\mathbf{v}_1$ located? Make sure to explain how you arrive at your answer. Also, you only need to find one location for $\mathbf{v}_1$.

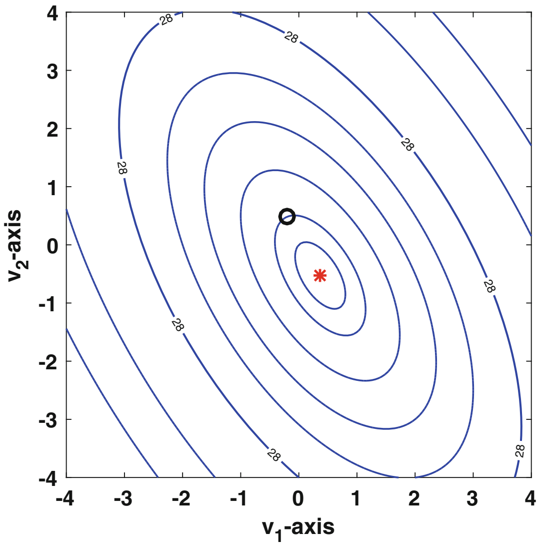


Figure 8.31 Contour plot for Exercise 8.23. The $*$ designates the location of the minimum.

(b) Suppose the starting point $\mathbf{v}_1$ is located on the contour labeled with 28. Where should it be so $\mathbf{v}_2$ is the exact solution?

8.24. Consider the function $F(\mathbf{v}) = \frac{1}{2}\mathbf{v}^T\mathbf{A}\mathbf{v} - \mathbf{b}\cdot\mathbf{v}$, where $\mathbf{A}$ is a symmetric and positive definite 2×2 matrix, and $\mathbf{v} = (v_1, v_2)^T$. The contour plot for F is shown below in Figure 8.32.

(a) Also shown in the figure is the starting point $\mathbf{v}_1$ and the point $\mathbf{v}_2$ calculated using the CGM. Determine where the next point $\mathbf{v}_3$ is located. Make sure to clearly explain how you arrive at your answer.

(b) If one uses the same starting point $\mathbf{v}_1$ for the SDM, where is $\mathbf{v}_2$? Make sure to clearly explain how you arrive at your answer.

(c) What starting point $\mathbf{v}_1$ should be used so the point $\mathbf{v}_2$ computed by the SDM is exactly at the minimum? The point $\mathbf{v}_1$ must be located on one of the four sides of the contour plot (in a similar manner as was done in part (a)). Make sure to clearly explain how you arrive at your answer.

8.25. It is possible to think a model function has three parameters, where in fact there are effectively only two (or just one). This exercise investigates such a situation.

(a) Suppose $g(x)$ is a linear function of the variable x. In trying to decide if two or three parameters are needed to fit a particular data set, the following possibilities are considered:

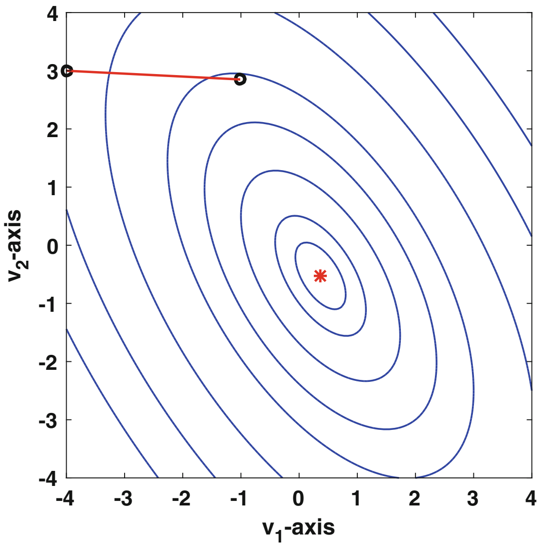


Figure 8.32 Contour plot for Exercise 8.24. The $*$ designates the location of the minimum.

i) $g_1(x) = v_1 + v_2 x$
ii) $g_2(x) = v_1 v_3 + v_2 x$
iii) $g_3(x) = (v_1 + v_2 x)/(1 + v_3)$
Suppose the minimum error using $g_2(x)$ occurs when $v_1 = 1$, $v_2 = 3$, and $v_3 = -2$. Explain why the minimum error using $g_1(x)$ occurs when $v_1 = -2$ and $v_2 = 3$. What values for v_1, v_2, and v_3 produce the minimum error using $g_3(x)$? Finally, explain why (i) corresponds to linear regression, while (ii) and (iii) correspond to nonlinear regression.

(b) For the model functions in part (a), suppose the minimum error using $g_1(x)$ occurs only when $v_1 = 6$ and $v_2 = -1$. Explain why if one uses either of the other two model functions that the minimum error does not occur at unique values for v_1, v_2, and v_3.

8.26. This problem considers the various steps used in the Nelder-Mead method.

(a) For the top contour plot in Figure 8.33, explain which step from Table 8.5 was used to determine each triangle.

(b) For the bottom contour plot in Figure 8.33, explain which step from Table 8.5 was used to determine each triangle.

8.27. This exercise considers the problem of finding the axial displacement $u(x)$ of an elastic bar, which is subject to a body force $w(x)$ as well as a force F_r on the right end $x = 1$. At the left end, where $x = 0$, the bar is fixed, which means that $u(0) = 0$. The potential energy is

$$V = \int_0^1 \left(\frac{1}{2}Du_x^2 + wu\right)dx - F_r u(1),$$

where D is a positive constant.

(a) Suppose the grid points are $x_0 = 0$, $x_1 = h$, $x_2 = 2h$, $\cdots$, $x_{n+1} = 1$, where $h = 1/(n+1)$. Writing $V = \int_0^1 F(x)dx$, write down the composite trapezoidal approximation for V.

(b) As with the string example, use a centered second-order approximation for u_x at x_1, x_2, $\cdots$, x_n, and a first-order approximation for u_x at x_0 and x_{n+1}. Using these with the result from part (a), what is the resulting approximation for V? Note that like the string example, $u_0 = 0$, but unlike the string example, u_{n+1} is an unknown in this problem.

(c) What is the resulting matrix equation that must be solved to find the minimum for the approximation for V found in part (b)?

(d) The minimum of V from part (b) can be found using the MATLAB command `fminsearch(@V,U)`, where V is the approximation from part (b) and $\mathbf{U}$ is an $(n+1)$-vector containing a starting guess for $(u_1, u_2, \cdots, u_{n+1})^T$. What would be a good, simple, and nonzero choice for $\mathbf{U}$, and why is it a good choice?

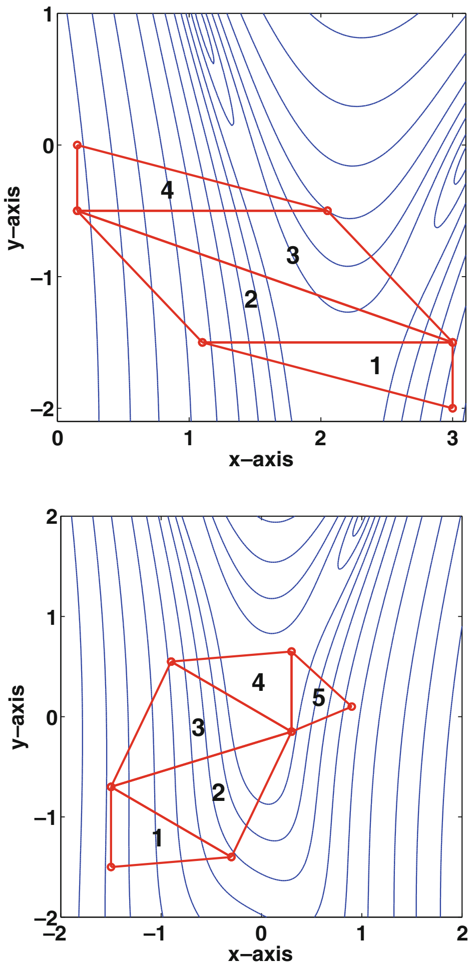


Figure 8.33 Plots used in Exercise 8.26.

(e) If $D = 1$, $F_r = 1$, $w = x^4$, then the exact solution is $u = x(x^5+24)/30$. On the same axis, plot the exact solution and the numerical solution when using 32 grid points. In doing this, explain how you found the numerical solution (using part (c) or part (d)), and why.

(f) The exact solution of the problem satisfies the boundary condition

$$D\frac{du}{dx}(1) = F_r.$$

So, the question is, does your numerical solution satisfy this condition. Using the first-order approximation you used in part (b), from the numerical solution calculate $u_x(1)$, using $n = 20, 40, 80, 160$. Are your answers consistent with the statement that the solution satisfies the above condition?

8.28. This exercise considers the problem of finding the traverse displacement $u(x)$ of an elastic beam, which is subject to a body force $w(x)$. The potential energy is

$$V = \int_0^1 \left(\frac{1}{2}EIu_{xx}^2 - wu\right)dx,$$

where EI is a positive constant. It is assumed that the beam has what are called simply supported ends. One consequence of this is that $u(0) = 0$ and $u(1) = 0$. The role of the boundary conditions will be discussed in more detail in part (f).

(a) Suppose the grid points are $x_0 = 0$, $x_1 = h$, $x_2 = 2h$, $\cdots$, $x_{n+1} = 1$, where $h = 1/(n+1)$. Writing $V = \int_0^1 F(x)dx$, write down the composite trapezoidal approximation for V.

(b) Use a centered second-order approximation for u_{xx} at x_1, x_2, $\cdots$, x_n. You can use a first-order approximations for u_{xx} at x_0 and x_{n+1}. Using these with the result from part (a), what is the resulting approximation for V? In doing this, remember that $u_0 = u_{n+1} = 0$.

(c) What is the resulting matrix equation that must be solved to find the minimum for the approximation for V found in part (b)?

(d) The minimum of V from part (b) can be found using the MATLAB command `fminsearch(@V,U)`, where V is the approximation from part (b) and $\mathbf{U}$ is an n-vector containing a starting guess for $(u_1, u_2, \cdots, u_n)^T$. What would be a good, simple, and nonzero choice for $\mathbf{U}$, and why is it a good choice? In answering this, it is worth knowing that a cable hanging between two poles is an elastic beam, subject to gravity.

(e) If $EI = 1$ and $w = x^4$, then the exact solution is

$$u = \frac{1}{1680}x^8 - \frac{1}{180}x^3 + \frac{5}{1008}x.$$

On the same axis, plot the exact solution and the numerical solution when using 22 grid points. In doing this, explain how you found the numerical solution (using part (c) or part (d)), and why.

(f) A simply supported beam is required to satisfy the four boundary conditions: $u(0) = 0$, $u(1) = 0$, $u_{xx}(0) = 0$, and $u_{xx}(1) = 0$. The numerical solution was derived without any mention of the last two conditions. The reason is that they are natural boundary conditions, which means that if u minimizes V, then it will automatically satisfy $u_{xx}(0) = 0$ and $u_{xx}(1) = 0$. In comparison, $u(0) = 0$ and $u(1) = 0$ are essential boundary conditions, which means that we must explicitly require this of our approximation (see part (b)). So, the question is, does your numerical solution satisfy (approximately) $u_{xx}(0) = 0$ and $u_{xx}(1) = 0$. Using the first-order approximations you used in part (b), from the numerical solution calculate $u_{xx}(0)$ and $u_{xx}(1)$, using $n = 20, 40, 80, 160$. Are your answers consistent with the statement that the minimizer satisfies $u_{xx}(0) = 0$ and $u_{xx}(1) = 0$?

8.29. This problem considers solving $\mathbf{Ax} = \mathbf{b}$. The $n \times n$ symmetric matrix $\mathbf{A}$ contains zeros except $a_{ii} = 3$, $a_{i,i-2} = a_{i,i+2} = -1$ (see below). Also the exact solution is $\mathbf{x} = (1, 1, \cdots, 1)^T$, and so calculate $\mathbf{b}$ using the formula $\mathbf{b} = \mathbf{Ax}$.

$$\mathbf{A} = \begin{pmatrix} 3 & 0 & -1 & & & & \\ 0 & 3 & 0 & -1 & & & \\ -1 & 0 & 3 & 0 & -1 & & \\ & -1 & 0 & 3 & 0 & -1 & \\ & & \ddots & & \ddots & & \ddots \\ & & & -1 & 0 & 3 & 0 \\ & & & & -1 & 0 & 3 \end{pmatrix}.$$

This is an example of what is called a penta-diagonal matrix.

(a) Show the matrix is positive definite.

(b) Taking $n = 1000$ and using the SDM, plot the error, iteration error, and the residual error as a function of the iteration number (as is done in Figure 8.19).

(c) Taking $n = 1000$ and using the CGM, plot the error, iteration error, and the residual error as a function of the iteration number (as is done in Figure 8.19).

8.30. This problem considers solving $\mathbf{Ax} = \mathbf{b}$. The $n \times n$ symmetric matrix $\mathbf{A}$ has elements $a_{ii} = \frac{1}{2}n(n+1)) + i(n-i) + 1$ and $a_{ij} = i + j$ for $i \neq j$. Also the exact solution is $\mathbf{x} = (1, 1, \cdots, 1)^T$, and so calculate $\mathbf{b}$ using the formula $\mathbf{b} = \mathbf{Ax}$.

(a) Write out the matrix in the case of when $n = 2$, $n = 3$, and $n = 4$, and explain why they are all positive definite. It is possible to prove that $\mathbf{A}$ is positive definite for all values of n (you do not need to show this).

(b) Taking $n = 1000$ and using the SDM, plot the error, iteration error, and the relative residual error as a function of the iteration number (as is done in Figure 8.19). Note that the relative residual is $\|\mathbf{r}\|/\|\mathbf{b}\|$.

(c) Taking $n = 1000$ and using the CGM, plot the error, iteration error, and the relative residual error as a function of the iteration number (as is done in Figure 8.19).

8.31. This exercise considers computing the curve that produces a surface of revolution with minimum area. Given two points (a, A) and (b, B) in the plane, with $a < b$ and A and B positive, assume that $y(x)$ is a smooth, positive, function connecting these points. If $y(x)$ is rotated about the x-axis, the resulting surface has area

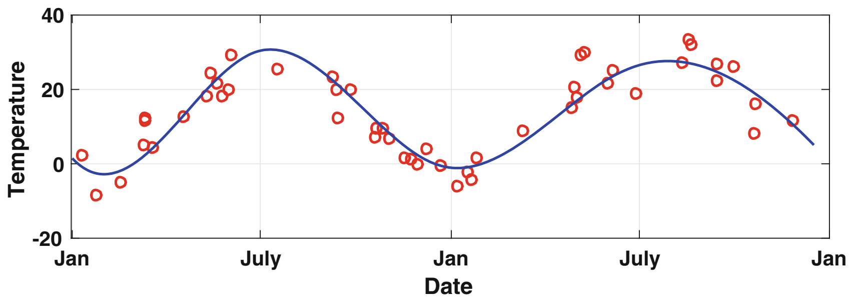


Figure 8.34 Temperature data over a two year period, and the cubic spline fit using least squares as considered in Exercise 8.32 [NCEI, 2015].

$$S = \int_a^b 2\pi y \sqrt{1 + (y')^2} dx.$$

It should be noted that the curve is required to satisfy $y(a) = A$ and $y(b) = B$. Also, unlike the brachistochrone problem, $y(x)$ does not have a singularity (at either end).

(a) Suppose the grid points are $x_0 = 0$, $x_1 = h$, $x_2 = 2h$, $\cdots$, $x_{n+1} = 1$, where $h = 1/(n+1)$. Writing $S = \int_0^1 F(x)dx$, write down the composite trapezoidal approximation for S.

(b) Use a centered second-order approximation for y' at $x_1, x_2, \cdots, x_n$, and a first-order approximation for y' at x_0 and x_{n+1}. Using these with the result from part (a), what is the resulting approximation for S?

(c) The minimum of S from part (b) can be found using the MATLAB command `fminsearch(@S,Y)`, where S is the approximation from part (b) and $\mathbf{Y}$ is an n-vector containing a starting guess for $(y_1, y_2, \cdots, y_n)^T$. What would be a good, simple, and nonzero choice for $\mathbf{Y}$, and why is it a good choice?

(d) Taking $a = 0$, $A = 1$, $b = 1$, and $B = 3$, plot the numerical solution for $y(x)$ for $n = 4$, $n = 9$, and $n = 19$.

(e) In calculus it is shown that the curve producing the minimum area is $y = (1/\alpha)\cosh(\alpha x + \beta)$, where α and β are determined from the requirements that $y(a) = A$ and $y(b) = B$. Determine the nonlinear equations that must be solved to find α and β. Discuss the numerical difficulties associated with solving these equations, compared to the direct solution in parts (a)–(c).
Note: The minimal surface problem has some interesting mathematical complications, and for more about this see Oprea [2007].

8.32. This exercise considers using a cubic spline as the model function when using least squares. An example of this is shown in Figure 8.34. Assume the data points are (x_1, y_1), (x_2, y_2), $\cdots$, (x_n, y_n). These are going to be fitted with the cubic spline function

x	1	48	94	144	193	242	286	334	382	430	474	532	592	651	715
y	-5.6	-5	10.6	17.2	25	27.2	17.2	6.1	-4.3	-3.3	23.3	22.8	31.7	25.6	12.8

Table 8.12 Temperature data for Exercise 8.32(c) [NCEI, 2015]. Note, x is measured in days, with $x = 1$ corresponding to January 1, and $x = 715$ corresponding to December 15 of the following year.

$$s(x) = \sum_{j=0}^{m+1} a_j B_j(x),$$

where $B_j(x)$ is given in (5.22). The nodes for the B-splines are assumed to be $\bar{x}_1$, $\bar{x}_2$, $\cdots$, $\bar{x}_m$, where the $\bar{x}_j$'s are equally spaced between $\bar{x}_1$ and $\bar{x}_m$. Note that in (5.22), h is the spacing between the $\bar{x}_j$ points. In this exercise, m is taken to be given, and then least squares is used to determine the a_j's. Also, the reason for including $j = 0$ and $j = m + 1$ in the sum is explained in Section 5.4.1.

(a) Show that to obtain the minimum of the error function

$$E(a_0, a_1, \cdots, a_{m+1}) = \sum_{i=1}^{n} [s(x_i) - y_i]^2$$

one needs to solve $\mathbf{Ca} = \mathbf{d}$, where $\mathbf{a} = (a_0, a_1, \cdots, a_{m+1})^T$, $\mathbf{C} = \mathbf{BB}^T$, $\mathbf{d} = \mathbf{By}$, $\mathbf{y} = (y_1, y_2, \cdots, y_n)^T$, and $\mathbf{B}$ is a $(m + 2) \times n$ matrix. In particular, the (ℓ, k) entry of $\mathbf{B}$ is $B_{\ell-1}(x_k)$.

(b) What value should you take for $\bar{x}_1$? What about $\bar{x}_m$?

(c) Taking $m = 3$, compute the a_j's using the data in Table 8.12. With this, plot $s(x)$ and the data on the same axis, for $1 \leq x \leq 715$. Also, compute $||\mathbf{C}||_\infty$ and comment on whether the matrix is ill-conditioned.

(d) Redo part (c), but take $m = 4$, $m = 5$, $m = 6$, and $m = 13$.

(e) Based on your results from parts (c) and (d), for a data set with n points, what do you recommend to use for m? In answering this, keep in mind that the model function should be capable of reproducing the more significant trends in the data, as well as provide a reasonable approximation over the entire interval. For example, the data in Table 8.12 comes from the same location, and time interval, as the data in Figure 8.34. Consequently, your recommendation using the data from Table 8.12 to determine $s(x)$ should result in a model function that does well with the data in Figure 8.34.

Chapter 9
Data Analysis

9.1 Introduction

In this chapter we consider a problem we have examined in earlier chapters, which is how to derive information from data. This was central to Chapter 5, when we derived interpolation formulas, and also in Chapter 8, where we investigated ways to use linear and nonlinear regression. In this chapter, four different situations are considered. The first three have a lot in common, and are examples illustrating the usefulness of the singular value decomposition (SVD) in data analysis. The SVD is explained in Section 4.5. These three methods also make use of the regression material covered in Section 8.2. The fourth method relates to what is sometimes called causal data, which means that there is an underlying mathematical model to explain the observed behavior, but it is necessary to fit the model to the data. This is similar to the regression problem, but in this case the model function comes from equations derived elsewhere, such as Newton's laws of mechanics, or Maxwell's equations of electrodynamics. This material will rely heavily on Section 5.4.1, which means cubic B-splines, and it uses the RK4 method, which is derived in Section 7.5.

9.2 Principal Component Analysis

The goal is to have a method that can be used to find connections between experimentally determined quantities, even though there is no obvious reason why they have to be connected. It is easiest to explain this using an example. In addition to introducing how the SVD can be used to find the connections, it will also show how the method overlaps with the regression material considered in the last chapter.

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_9

word	L	W	X	Y	x	y
bag	3	223	−3	117.70	−0.6	0.766
across	6	117	0	11.70	0.0	0.076
if	2	101	−4	−4.30	−0.8	−0.028
insane	6	40	0	−65.30	0.0	−0.425
by	2	217	−4	111.70	−0.8	0.727
detective	9	41	3	−64.30	0.6	−0.418
relief	6	153	0	47.70	0.0	0.310
slope	5	87	−1	−18.30	−0.2	−0.119
scoundrel	9	4	3	−101.30	0.6	−0.659
look	4	212	−2	106.70	−0.4	0.694
neither	7	51	1	−54.30	0.2	−0.353
pretentious	11	70	5	−35.30	1.0	−0.230
solid	5	259	−1	153.70	−0.2	1.000
gone	4	90	−2	−15.30	−0.4	−0.100
fun	3	78	−3	−27.30	−0.6	−0.178
therefore	9	13	3	−92.30	0.6	−0.601
generality	10	22	4	−83.30	0.8	−0.542
month	5	57	−1	−48.30	−0.2	−0.314
blot	4	185	−2	79.70	−0.4	0.519
infectious	10	86	4	−19.30	0.8	−0.126

Table 9.1 Number, L, of letters in a word, and the number, W, of words appearing in its definition in a dictionary. Also, X, Y are the centered values, and x, y are the scaled values.

9.2.1 Example: Word Length

The question considered is, is there a connection between the length of a word and how many words are used in its definition in the dictionary. Some sample data are given in Table 9.1, which were obtained from the Merriam-Webster Dictionary. We are going to see if there is a linear relationship between these values. In particular, if L is the number of letters in a word, and W is the number of words in its definition, we want to fit

$$W = \alpha L + \beta \tag{9.1}$$

to the data. This brings up the first observation about this problem, which is that writing the formula this way implies that L is the independent variable and W is the dependent variable. The idea of independent and dependent is almost arbitrary here, and we could just as well have written

$$L = aW + b. \tag{9.2}$$

Mathematically these two formulas are equivalent, with $\alpha = 1/a$ and $\beta = -b/a$. However, as explain in Section 8.3.3, traditional linear least squares applied to (9.1) will not produce equivalent coefficients to the values obtained when it is applied to (9.2). A way to avoid this is to use orthogonal regression, which uses the true distance between the point and line (see Figure 8.5). Not thinking too hard about this, and using the linear relationship in (9.1), one might claim that the error function is (see Section 8.3.3)

$$E(\alpha, \beta) = \frac{1}{1+\alpha^2}\sum_{i=1}^{n}(\alpha L_i + \beta - W_i)^2. \tag{9.3}$$

There are two complications with this, and how these are resolved are as follows:

Center the Data

The first issue concerns finding the minimizer of the above error function. Even for this rather simple two variable problem, finding the minimum of E requires solving a cubic. One way to avoid this is to center the data values using their mean, or average, values. In particular, letting

$$\overline{L} = \frac{1}{n}\sum_{i=1}^{n} L_i \qquad \text{and} \qquad \overline{W} = \frac{1}{n}\sum_{i=1}^{n} W_i, \tag{9.4}$$

then the centered data values are computed using the formulas

$$X_i = L_i - \overline{L} \qquad \text{and} \qquad Y_i = W_i - \overline{W}. \tag{9.5}$$

The resulting data values are given in Table 9.1. The significance of this step is that instead of the more general linear equation in (9.1), it is possible to now assume that

$$Y = \alpha X. \tag{9.6}$$

One reason this is possible is that, for it to hold, it is necessary that $\sum Y_i = \alpha \sum X_i$. This applies to our centered data because $\sum Y_i = 0$ and $\sum X_i = 0$. Moreover, as will be shown shortly, using this model function it is possible to find the minimizer rather easily.

Scale Data

The second complication with (9.3) concerns the differences in the dimensional units, and magnitudes, of the variables. The solution is to scale each variable using a characteristic value for that variable. Letting X_c and Y_c be characteristic values for X and Y, respectively, then the scaled data are obtained from the formulas

$$x_i = X_i/X_c \qquad \text{and} \qquad y_i = Y_i/Y_c. \tag{9.7}$$

There are various ways to pick X_c and Y_c, and these will be considered later. For this example we will pick the largest entry, in absolute value, for each variable. This means we will take $X_c = 5$ and $Y_c = 153.7$.

Based on the above adjustments, we will fit the line $y = \alpha x$ to the data values (x_i, y_i), using the error function

$$E(\alpha) = \frac{1}{1+\alpha^2}\sum_{i=1}^{n}(\alpha x_i - y_i)^2. \tag{9.8}$$

Taking the derivative of this expression, and setting it to zero, one finds that α satisfies $\alpha^2 + \lambda\alpha - 1 = 0$, where

$$\lambda = \frac{\sum(x_i^2 - y_i^2)}{\sum x_i y_i}. \tag{9.9}$$

The conclusion is that

$$\alpha = \frac{1}{2}\left(-\lambda \pm \sqrt{\lambda^2+4}\right), \tag{9.10}$$

where the $+$ is used if $\sum x_i y_i > 0$ and the $-$ is used if $\sum x_i y_i < 0$. It is also apparent that this conclusion requires that the normalized data vectors $\mathbf{x}$ and $\mathbf{y}$ not be orthogonal (since $\mathbf{x}\cdot\mathbf{y} = \sum x_i y_i$).

Converting the answer back into the original variables used to introduce the example, we have the formula

$$W = \overline{W} + m(L - \overline{L}), \tag{9.11}$$

where the slope is $m = \alpha Y_c/X_c$. The resulting linear fit to this data is shown in Figure 9.1.

What is rather surprising is that the linear fit obtained by minimizing the modified error function (9.8) can also be found using the singular value decomposition (SVD) of the transformed data matrix. To show this, let $\mathbf{P}$ be the 20×2 normalized data set obtained from the (x_i, y_i) values in Table 9.1. From the SVD, as given in Section 4.5.2,

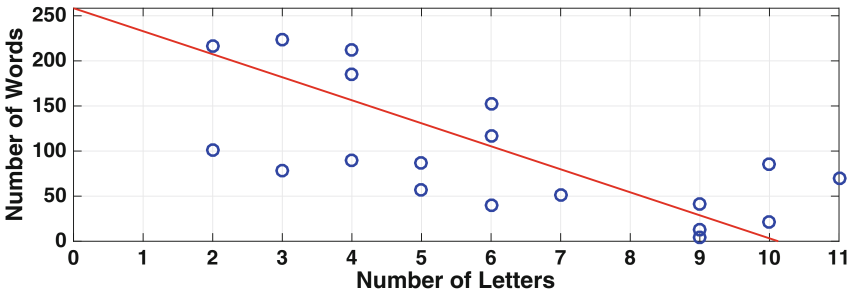


Figure 9.1 Linear fit connecting the length of a word and the number of words in its definition.

$$\mathbf{P} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T, \tag{9.12}$$

where $\mathbf{V}$ is a 2×2 orthogonal matrix whose column vectors $\mathbf{v}_i$ are orthonormal eigenvectors for $\mathbf{P}^T\mathbf{P}$, and $\boldsymbol{\Sigma}$ is a 20×2 diagonal-like matrix containing the two singular values σ_1 and σ_2. The latter are calculated using $\sigma_i = \sqrt{\lambda_i}$, where λ_i is an eigenvalue for $\mathbf{P}^T\mathbf{P}$ and labeled so that $\lambda_1 \geq \lambda_2$. Using the values given in Table 9.1, one finds that $\sigma_1 = 2.96$, $\sigma_2 = 1.42$, and

$$\mathbf{V} = \begin{pmatrix} 0.7694 & -0.6388 \\ -0.6388 & -0.7694 \end{pmatrix}.$$

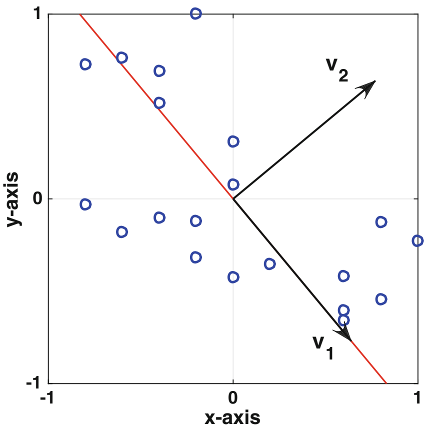


Figure 9.2 Eigenvectors $\mathbf{v}_1$ and $\mathbf{v}_2$ obtained using the SVD. Also shown are the normalized data from Table 9.1 and, by the solid (red) line, the linear fit determined by minimizing (9.8).

The two column vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ from this matrix, along with the normalized data from Table 9.1, are shown in Figure 9.2. As is immediately evident, the first vector $\mathbf{v}_1$ points in the direction determined by the linear fit, while the second column points in a direction orthogonal to the first. In other words, by using the first column from the matrix $\mathbf{V}$ we can solve the linear fit problem. As will be shown next, this is not a coincidence.

9.2.2 Principal Component Decomposition

It is assumed that there are m variables, or quantities, q_1, q_2, $\cdots$, q_m under consideration. Also, there are n observations, or data points, for these variables. These will be denoted as $\mathbf{q}_1$, $\mathbf{q}_2$, $\cdots$, $\mathbf{q}_n$, where each $\mathbf{q}_i$ is an m-vector. Letting $\mathbf{q}_i = (q_{i1}, q_{i2}, \cdots, q_{im})$, then q_{ij} is the value of q_j for the ith data point. This generates an $n \times m$ data matrix as illustrated in Table 9.2.

Before using a PCA, the data needs to be normalized, which involves two steps. The first is to center the values in each column using the mean value for that column. The resulting matrix entries are $Q_{ij} = q_{ij} - \overline{q}_j$, where

$$\overline{q}_j = \frac{1}{n}\sum_{i=1}^{n} q_{ij}$$

is the mean of the jth column. The second step is to scale the values in each column using a characteristic value for that column. The result is

$$p_{ij} = \frac{Q_{ij}}{S_j},$$

where S_j is a characteristic value for the jth column. In the word length example, we used the maximum entry in the jth column, in absolute value, for S_j. However, there are various choices one can take for S_j, and these will be discussed later.

	q_1	q_2	$\cdots$	q_m		p_1	p_2	$\cdots$	p_m
$\mathbf{q}_1$	q_{11}	q_{12}	$\cdots$	q_{1m}	$\mathbf{p}_1$	p_{11}	p_{12}	$\cdots$	p_{1m}
$\mathbf{q}_2$	q_{21}	q_{22}	$\cdots$	q_{2m}	$\mathbf{p}_2$	p_{21}	p_{22}	$\cdots$	p_{2m}
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$
$\mathbf{q}_n$	q_{n1}	q_{n2}	$\cdots$	q_{nm}	$\mathbf{p}_n$	p_{n1}	p_{n2}	$\cdots$	p_{nm}

Table 9.2 Original data matrix on the left, and its normalized version on the right.

Linear Approximation

We want to determine the best linear fit of the normalized data. What this means requires some explanation because of the multivariable nature of the data. In using a PCA, with m variables, setting $\mathbf{p} = (p_1, p_2, \cdots, p_m)^T$, one needs to select one of the following linear functions:

$$\mathbf{p} = \alpha_1 \mathbf{v}_1, \tag{9.13}$$

$$\mathbf{p} = \alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2, \tag{9.14}$$

$$\mathbf{p} = \alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 + \alpha_3 \mathbf{v}_3,$$

$$\vdots \qquad\qquad \vdots$$

$$\mathbf{p} = \alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 + \cdots + \alpha_{m-1} \mathbf{v}_{m-1}. \tag{9.15}$$

The data fitting that is part of a PCA is used to find the direction vectors $\mathbf{v}_j$ in the above expressions. It is assumed that the $\mathbf{v}_j$'s are m-vectors that have unit length, and they are mutually orthogonal. The first choice (9.13) can be interpreted as a line and (9.15) is an example of what is called a hyperplane, but geometrical interpretations for the others is not as obvious. For this reason, they are referred to by their dimensionality. So, (9.13) corresponds to a one-dimensional approximation, (9.14) corresponds to a two-dimensional approximation, etc.

The goal of a PCA is to find the principal directions for the data, and this is the role of the $\mathbf{v}_j$'s in the above functions. As an example, the data points shown in the upper plot in Figure 9.3 occupy, approximately, an ellipsoidal region. If we use (9.13), then the goal of PCA is to find a direction $\mathbf{v}_1$ that points along the main diagonal of this region, which is indicated with the solid (red) line. Finding this line is a three dimensional regression problem. Note that in doing this we will be using true distance between the line and data points (see Section 8.2.2).

Now, suppose the two dimensional approximation in (9.14) is used with the data in Figure 9.3. As in the one dimensional case, the vector $\mathbf{v}_1$ is required to identify the principal direction for the data, and so it points along the solid (red) line. To determine $\mathbf{v}_2$, in the lower plot in Figure 9.3 the data points are plotted in a plane perpendicular to the red line (the origin in this plot is a point on the red line). This is what you would see, for example, if you were to look at the data in the direction of the red line. The vector $\mathbf{v}_2$ is required to point in the principal direction of this data, which means that it points in the direction determined by the dashed (black) line.

Error Function

This brings us to the next step, which is to determine the formula for the error. So, suppose we pick the one-dimensional approximation (9.13). The error will be based on the distance between the data points and this line. To

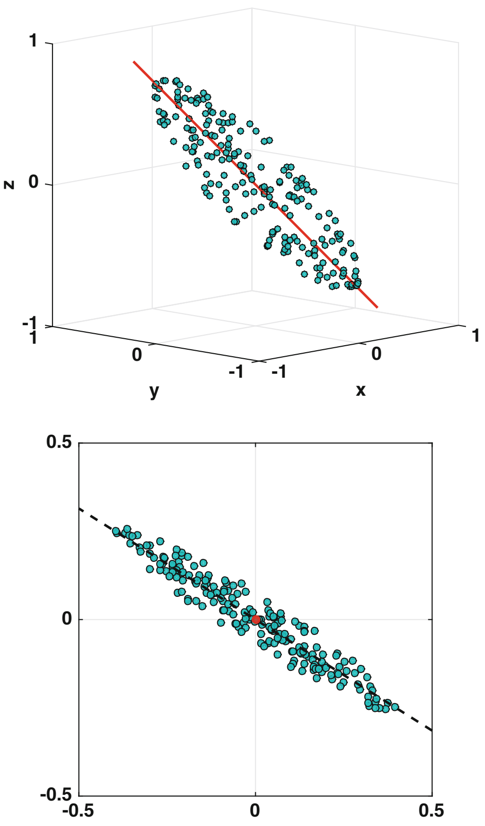


Figure 9.3 Example data set in the case of three variables. The upper plot is the data in three dimensions, and the lower plot is the data projected onto a plane that is perpendicular to the solid (red) line shown in the upper plot.

determine this, the distance between a point on the line and a data point $\mathbf{p}_i$ is $\overline{d}_i = ||\alpha_1\mathbf{v}_1 - \mathbf{p}_i||_2$. The distance d_i between $\mathbf{p}_i$ and the line is, by definition, the smallest value of $\bar{d}_i$. Taking the derivative of $\overline{d}_i$ with respect to α_1, and setting it to zero, one finds that the distance is

$$d_i = \sqrt{\mathbf{p}_i \cdot \mathbf{p}_i - (\mathbf{p}_i \cdot \mathbf{v}_1)^2}\,.$$

We are looking for the least squares error, and so the resulting error function obtain using the entire data set is

$$\begin{aligned} E_1 &= \sum_{i=1}^{n} d_i^2 \\ &= \sum_{i=1}^{n} \left[\mathbf{p}_i \cdot \mathbf{p}_i - (\mathbf{p}_i \cdot \mathbf{v}_1)^2\right]. \end{aligned} \tag{9.16}$$

We will look for a vector $\mathbf{v}_1$ that minimizes this function under the constraint that $||\mathbf{v}_1||_2 = 1$.

As a second example, suppose one picks the two-dimensional approximation (9.14). The distance between a point in this region and a data point $\mathbf{p}_i$ is

$$\overline{d}_i = ||\alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 - \mathbf{p}_i||_2.$$

The distance d_i between the two-dimensional region generated using (9.14) and $\mathbf{p}_i$ is the minimum of $\overline{d}_i$. Finding this, and then adding the squares of these distances for the data points, the resulting error function is

$$E_2 = \sum_{i=1}^{n} \left[\mathbf{p}_i \cdot \mathbf{p}_i - (\mathbf{p}_i \cdot \mathbf{v}_1)^2 - (\mathbf{p}_i \cdot \mathbf{v}_2)^2 \right]. \tag{9.17}$$

It should be remembered that when finding $\mathbf{v}_1$ and $\mathbf{v}_2$ which minimize E_2, it is required that $||\mathbf{v}_1||_2 = 1$, $||\mathbf{v}_2||_2 = 1$, and $\mathbf{v}_1 \cdot \mathbf{v}_2 = 0$.

The formula obtained when using more $\mathbf{v}_j$'s is a straightforward generalization of (9.16) and (9.17).

Minimization

We now turn to the problem of minimizing the error function, subject to the constraint that the $\mathbf{v}_j$'s have unit length. This can be done using the method of Lagrange multipliers. For the one-dimensional approximation, the function to minimize is

$$F(\mathbf{v}_1, \lambda_1) = E_1(\mathbf{v}_1) + \lambda_1(||\mathbf{v}_1||_2^2 - 1), \tag{9.18}$$

where λ_1 is the Lagrange multiplier. For the two dimensional approximation it is

$$F(\mathbf{v}_1, \mathbf{v}_2, \lambda_1, \lambda_2) = E_2(\mathbf{v}_1, \mathbf{v}_2) + \lambda_1(||\mathbf{v}_1||_2^2 - 1) + \lambda_2(||\mathbf{v}_2||_2^2 - 1), \tag{9.19}$$

where λ_1 and λ_2 are the Lagrange multipliers.

Finding the minimum is straightforward, but tedious. For (9.18), the first step is to differentiate F with respect to the components of $\mathbf{v}_1$, as well as λ_1, and then set the derivatives to zero. Something similar is done for (9.19). The details are left as an exercise, and one finds that the $\mathbf{v}_j$'s must satisfy

$$(\mathbf{P}^T\mathbf{P})\mathbf{v}_j = \lambda_j \mathbf{v}_j,$$

where $\mathbf{P}$ is the $n \times m$ matrix of normalized data values given in Table 9.2. The conclusion is that $\mathbf{v}_j$ must be an eigenvector for $\mathbf{P}^T\mathbf{P}$, and it has corresponding eigenvalue λ_j. With this, one finds that

$$\sum_{i=1}^{n} (\mathbf{p}_i \cdot \mathbf{v}_j)^2 = \lambda_j.$$

Substituting this into (9.16) we obtain

$$E_1 = -\lambda_1 + \sum_{i=1}^{n} (\mathbf{p}_i \cdot \mathbf{p}_i) \,, \tag{9.20}$$

and from (9.17) we have

$$E_2 = -\lambda_1 - \lambda_2 + \sum_{i=1}^{n} (\mathbf{p}_i \cdot \mathbf{p}_i) \,. \tag{9.21}$$

Therefore, to minimize E_1 we should use the eigenvector $\mathbf{v}_1$ corresponding to the largest eigenvalue of $\mathbf{P}^T\mathbf{P}$, and to minimize E_2 we should use the eigenvectors $\mathbf{v}_1$ and $\mathbf{v}_2$ corresponding to the largest, and second largest, eigenvalues of $\mathbf{P}^T\mathbf{P}$ (assuming, for the moment, that there is only one independent eigenvector for λ_1). Also, note that since $\mathbf{P}^T\mathbf{P}$ is symmetric, the eigenvectors are orthogonal, as required for the PCA.

This is the point where we need to recall some of the facts about the singular value decomposition (SVD), which were derived in Section 4.5. The SVD has the form $\mathbf{P} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$, where $\boldsymbol{\Sigma}$ is a diagonal-like matrix containing the singular values σ_i, and $\mathbf{V}$ is an orthogonal $m \times m$ matrix. Also, the eigenvalues λ_i of $\mathbf{P}^T\mathbf{P}$, and the singular values σ_i for $\mathbf{P}$, are connected through the formula $\lambda_i = \sigma_i^2$. Second, the columns of the matrix $\mathbf{V}$ are the eigenvectors for $\mathbf{P}^T\mathbf{P}$. Third, the SVD orders the singular values by size, so $\sigma_1 \geq \sigma_2 \geq \cdots \geq 0$. Based on this, the smallest value of E_1 in (9.16) is obtained when $\mathbf{v}_1$ is taken to be the first column from $\mathbf{V}$. This is because $\mathbf{v}_1$ is an eigenvector for the largest singular value $\sigma_1 = \sqrt{\lambda_1}$, and this produces the smallest value of the error E_1 as given in (9.20). Similarly, to obtain the smallest value of E_2, as given in (9.17), $\mathbf{v}_1$ and $\mathbf{v}_2$ should be the first two columns of $\mathbf{V}$.

The conclusions of the previous paragraph generalize to the case for higher order linear approximations, and this is summarized in the following theorem.

Theorem 9.1. *Let $\mathbf{P}$ be a normalized nonzero $n\times m$ data matrix, with $m < n$. Also, let the SVD of this matrix be $\mathbf{P} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$. Given k, with $1 \leq k < m$, consider a linear fit of the data of the form*

$$\mathbf{p} = \alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2 + \cdots + \alpha_k\mathbf{v}_k.$$

The smallest error is obtained when the vectors $\mathbf{v}_1$, $\mathbf{v}_2$, $\cdots$, $\mathbf{v}_k$ are chosen to be the first k columns of $\mathbf{V}$.

In the parlance of the subject, the columns of $\mathbf{V}$ are called the principal components.

Now the more practical question, which is, so what? The easiest way to answer this is thorough an example.

Word	L	W	1908	2008	p_1	p_2	p_3	p_4
bag	3	223	0.002	0.003	−0.6	0.766	−0.062	−0.064
across	6	117	0.002	0.003	0.0	0.076	−0.062	−0.064
if	2	101	0.176	0.173	−0.8	−0.028	1.000	1.000
insane	6	40	0.002	0.003	0.0	−0.425	−0.062	−0.064
by	2	217	0.002	0.003	−0.8	0.727	−0.062	−0.064
detective	9	41	0.000	0.001	0.6	−0.418	−0.070	−0.076
relief	6	153	0.002	0.003	0.0	0.310	−0.062	−0.064
slope	5	87	0.002	0.003	−0.2	−0.119	−0.062	−0.064
scoundrel	9	4	0.002	0.003	0.6	−0.659	−0.062	−0.064
look	4	212	0.019	0.028	−0.4	0.694	0.043	0.093
neither	7	51	0.002	0.003	0.2	−0.353	−0.062	−0.064
pretentious	11	70	0.002	0.003	1.0	−0.230	−0.062	−0.064
solid	5	259	0.002	0.003	−0.2	1.000	−0.062	−0.064
gone	4	90	0.011	0.010	−0.4	−0.100	−0.006	−0.020
fun	3	78	0.001	0.003	−0.6	−0.178	−0.067	−0.063
therefore	9	13	0.002	0.003	0.6	−0.601	−0.062	−0.064
generality	10	22	0.002	0.003	0.8	−0.542	−0.062	−0.064
month	5	57	0.008	0.007	−0.2	−0.314	−0.026	−0.038
blot	4	185	0.002	0.003	−0.4	0.519	−0.062	−0.064
infectious	10	86	0.002	0.003	0.8	−0.126	−0.062	−0.064

Table 9.3 Extension of the data given in Table 9.3, where 1908 and 2008 refer to the relative use of the word in the respective year. The variables p_1, p_2, p_3, and p_4 are the respective normalized values of the data.

Example

The word length data in Table 9.1 is expanded in Table 9.3 to include two new variables. One is the relative number of times the word appeared in print in 1908, and the other is the number of times for 2008 (both of these numbers are given in percentages). These were acquired using Google's ngram program [Michel et al., 2011]. The last four columns are the normalized values for the data, which were obtained by first centering each column and then dividing by the largest value for each variable (in absolute value) in that column.

Calculating the SVD of the normalized data matrix, one finds that

$$\mathbf{V} = \begin{pmatrix} 0.7712 & 0.3621 & 0.5236 & 0.0031 \\ -0.6169 & 0.6279 & 0.4744 & 0.0082 \\ -0.1086 & -0.4895 & 0.4944 & 0.7100 \\ -0.1134 & -0.4848 & 0.5064 & -0.7041 \end{pmatrix}.$$

Also, the means for the four original variables are $\overline{q}_1 = 6$, $\overline{q}_2 = 105.3$, $\overline{q}_3 = 0.0119$, and $\overline{q}_4 = 0.0131$, and the scaling factors used on the centered data are $S_1 = 5$, $S_2 = 153.7$, $S_3 = 0.1641$, and $S_4 = 0.1599$, respectively.

1. $k = 1$

   In this case, one selects $\mathbf{p} = \alpha_1\mathbf{v}_1$, where $\mathbf{v}_1$ is the vector coming from the first column of $\mathbf{V}$. Because of the single coefficient α_1, what is being assumed here is that knowing one of the variables, then $\mathbf{p} = \alpha_1\mathbf{v}_1$ can be used to determine approximate values for the other three. To illustrate, consider the word "anything," which has 8 letters. Given the normalization used in Table 9.3, for this word

$$\begin{aligned} p_1 &= \frac{8 - \overline{q}_1}{S_1} \\ &= 0.4. \end{aligned}$$

   From the equation $\mathbf{p} = \alpha_1\mathbf{v}_1$ we get that $p_1 = \alpha_1 v_{11}$. Since $v_{11} = 0.7712$, it then follows that $\alpha_1 = p_1/v_{11} = 0.5187$. From this we obtain the values, $p_2 = \alpha_1 v_{21} = -0.32$, $p_3 = \alpha_1 v_{31} = -0.0564$, and $p_4 = \alpha_1 v_{41} = -0.0588$. In original variables, the assumption that $\mathbf{p} = \alpha_1\mathbf{v}_1$ leads us to the conclusions that "anything" takes $\overline{q}_2 + S_2 p_2 \approx 56$ words in the dictionary, it appeared in $\overline{q}_3 + S_3 p_3 \approx 0.003\%$ of the printed literature in 1908, and appeared in $\overline{q}_4 + S_4 p_4 \approx 0.004\%$ in 2008.

2. $k = 2$

   The assumption now is that $\mathbf{p} = \alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2$, where $\mathbf{v}_1$ and $\mathbf{v}_2$ are the first two column vectors from $\mathbf{V}$. Again using the word "anything," it has 8 letters and, using Google's ngram program, it appeared in 0.0154% of the printed literature in 1908. As before, $p_1 = 0.4$. Also, $p_3 = (0.0154 - \overline{q}_3)/S_3 = 0.0212$. From the first and third entries in the equation $\mathbf{p} = \alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2$ we get that

$$\begin{aligned} p_1 &= \alpha_1 v_{11} + \alpha_2 v_{12} \\ p_3 &= \alpha_1 v_{31} + \alpha_2 v_{32}. \end{aligned}$$

   Substituting in the values for the p's and v's, and then solving one finds that $\alpha_1 = 0.6017$ and $\alpha_2 = -0.1769$. We can now use this information to find approximations for the other two variables, using the formula $p_i = \alpha_1 v_{i1} + \alpha_2 v_{i2}$. The conclusions are that the word "anything" takes about 31 words in the dictionary, and that it appeared in 0.016% of the printed literature in 2008. ■

As demonstrated in the previous example, using a PCA, one or more variables are used to predict the values of the others. Also, it does not make any difference which variables are taken to be "known" and which are "predicted." The exception to this statement occurs when there is a zero divisor. For example, when $k = 1$ in the previous example, if $v_{11} = 0$ then the value of α_1 would need to be determined from the second or third component of $\mathbf{v}_1$. Note that it would be concluded in this case that $p_1 = 0$.

Another point to make is that you might have noticed that nothing was said about how accurate the predictions are. The reason is that there is not enough data in the example to expect to be able to make accurate predictions. This is not the case of the next example, which involves crime data reported by the U.S. Census Bureau. The question of accuracy will be addressed, as it often is using a PCA, by splitting the data set into a training set, which is used to find the principal directions, and then a testing set which is used to test the accuracy of the approximations.

9.2.3 Scaling Factors

One question left unanswered is how to select a characteristic value S_j to scale the values in the jth column of the centered data matrix. A mathematical answer to this is to use a vector norm. If $\mathbf{c}_j$ is the jth column vector from the centered data matrix, then possibilities are

$$S_j = \frac{1}{n}||\mathbf{c}_j||_1, \quad S_j = \frac{1}{\sqrt{n}}||\mathbf{c}_j||_2, \text{ or } S_j = ||\mathbf{c}_j||_\infty.$$

Each of these is dimensionally consistent with the dimensions of $\mathbf{c}_j$. The multiplicative factors for the first two are there to scale for the size of the vector. For example, if $\mathbf{c}_j = (c, c, \cdots, c)^T$, with $c \geq 0$, then each of the above scaling factors yield $S_j = c$. Also, note that in the word length example, the ∞-norm choice was used for the scaling.

In applications, other choices are often used. One is *autoscaling*, and the formula is

$$S_j = \frac{1}{\sqrt{n-1}}||\mathbf{c}_j||_2 .$$

This is used in statistical applications because it corresponds to the standard deviation for the column. Another choice that is sometimes used comes from the difference between the largest and smallest entries in the uncentered data column, which results in the formula $S_j = \max_i q_{ij} - \min_i q_{ij}$. This is called *range scaling.*

Whatever choice is made, it is essential that S_j have the same dimensions as the values appearing in the jth column. Not unexpectedly, the choice can have an affect on the final answer. This is because different scalings produce different error functions, and, as illustrated in Figure 8.9, this will likely affect the value of the minimizer. A discussion of various ways to scale data, as well as other aspects of the normalization process, can be found in van den Berg et al. [2006] and Parente and Sutherland [2013].

9.2.4 Application: Crime Data

To illustrate how a PCA can be used, consider the major crime rates for the larger cities in the U.S. A portion of the dataset is shown in Table 9.4. In total there are 105 cities (note that three cities in the original dataset were removed because their data were incomplete). We will use this data and a PCA to see what connections there might be between the different crime rates. This will be done by splitting the data into two sets, one will be the *training set* and the other the *testing set*. The PCA will be carried out on the training set, and we will then see how well this does in predicting the data values for the cities in the testing set. For the training set we will use every other row in the dataset, which means using the data for New York, Houston, etc. The testing dataset will consist of those left out, which means Los Angeles, Phoenix, etc. Also, the normalization is determined using the training set, and for this example the 2-norm scaling given in the previous section will be used.

One Dimensional Approximation

The first question is, can one of the crime rates be used to estimate the other rates (as well as the population). For the training set, the singular values are $\sigma_1 = 18.8$, $\sigma_2 = 5.42$, $\sigma_3 = 4.28$, $\sigma_4 = 3.16$, $\sigma_5 = 1.60$, $\sigma_6 = 1.28$, $\sigma_7 = 0.651$, and $\sigma_8 = 0.184$. Also, the first column of $\mathbf{V}$ is

$$\mathbf{v}_1 = \begin{pmatrix} -0.2116 \\ -0.2961 \\ -0.3616 \\ -0.4760 \\ -0.3101 \\ -0.3915 \\ -0.3878 \\ -0.3323 \end{pmatrix}.$$

City	Population	Murder	Rape	Robbery	Assault	Burglary	Larceny	Vehicle Theft
New York, NY	8400907	46357	471	832	18597	142000	18780	112526
Los Angeles, CA	3848776	24070	312	903	12217	94240	18435	57414
Houston, TX	2273771	25593	287	823	11367	120933	29279	77058
Phoenix, AZ	1597397	8730	122	522	3757	65617	16281	39643
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Fremont, CA	202714	490	2	34	237	4978	1190	3237
Irving, TX	202447	604	4	34	352	8427	1913	5730
Yonkers, NY	202192	965	8	36	446	3145	620	2182

Table 9.4 Crime data rates for 105 of the largest cities in the U.S. in 2009 [U.S. Census Bureau, 2012]. The rates are per 100,000 population. Also given is the population of the city.

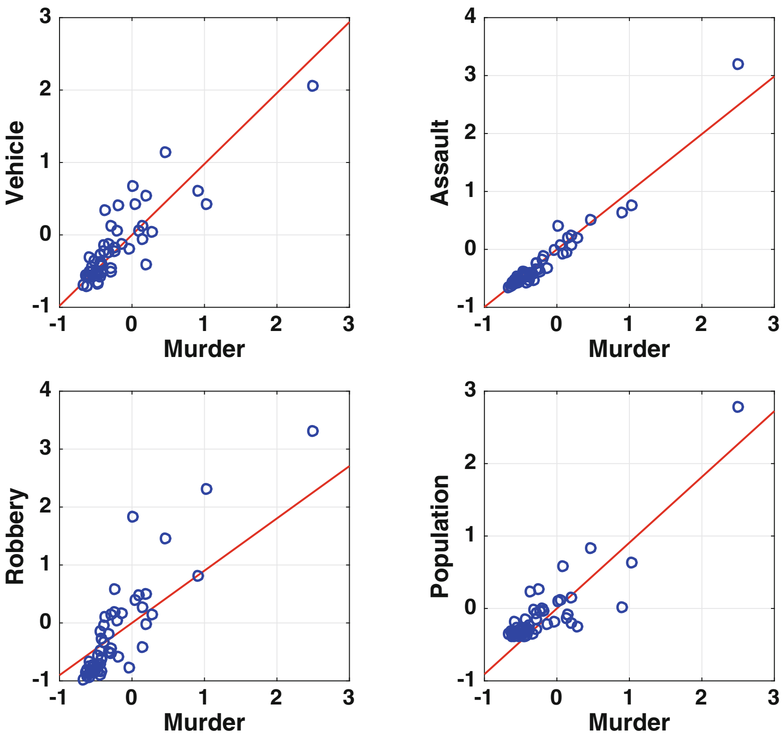


Figure 9.4 Using the murder rate to predict the rates of three other crimes and the population. The line comes from the training set, while the data comes from the testing set.

According to Theorem 9.1, the line of best fit is $\mathbf{p} = \alpha\mathbf{v}_1$. The question we are considering is, given one of the rates (i.e., one of the p_i's), what would we predict the other rates (and population) are. We will use the murder rate, which means we will select p_2. Since $p_2 = \alpha v_{21}$, then $\alpha = p_2/v_{21}$. From the equation $\mathbf{p} = \alpha\mathbf{v}_1$, the best line fits for the other rates (and population) are $p_i = a_i p_2$, where $a_i = v_{i1}/v_{21}$. The resulting lines, along with the data from the testing set, are shown in Figure 9.4. It is evident that the murder rate can be used to predict the assault rate fairly accurately, and it does reasonably well predicting the population of the city. The predictions for the other two rates are not quite as good, as there is more scatter in the data.

Seven Dimensional Approximation

We consider a second question, which is given the values for six of the crime rates and the population, how well can we predict the remaining crime rate? According to Theorem 9.1, the best fit seven dimensional plane is

$$\mathbf{p} = \alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2 + \cdots + \alpha_7\mathbf{v}_7.$$

This can be written out in matrix form as

$$\begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_8 \end{pmatrix} = \begin{pmatrix} v_{11} & v_{12} & \cdots & v_{17} \\ v_{21} & v_{22} & \cdots & v_{27} \\ \vdots & \vdots & \vdots & \vdots \\ v_{81} & v_{82} & \cdots & v_{87} \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_7 \end{pmatrix}. \tag{9.22}$$

We want to determine one of the p_i's, given the values for the others. It makes the mathematical formulas easier to write down if the variable of interest is p_1. In other words, it is assumed that the original data matrix is either written down, or column swapped, so the first column is the variable of interest here. With this, and (9.22), we need to find the α_i's in terms of p_2, p_3, $\cdots$, p_8. This is done by solving the reduced problem

$$\begin{pmatrix} p_2 \\ p_3 \\ \vdots \\ p_8 \end{pmatrix} = \begin{pmatrix} v_{21} & v_{22} & \cdots & v_{27} \\ v_{31} & v_{32} & \cdots & v_{37} \\ \vdots & \vdots & \vdots & \vdots \\ v_{81} & v_{82} & \cdots & v_{87} \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_7 \end{pmatrix}.$$

Writing the solution as $\alpha_i = \sum_{j=2}^{8} a_{ij} p_j$, then the first row from (9.22) can be written as

$$p_1 = a_2 p_2 + a_3 p_3 + \cdots + a_8 p_8, \tag{9.23}$$

where $a_i = \sum_{i=1}^{7} v_{1j} a_{ji}$. This is the formula we will use to predict the value of p_1 given the value of the other p_i's.

The resulting predictions for the vehicle theft rate, and the predictions for the murder rate, using the testing data are given in Figure 9.5. In these graphs, each symbol corresponds to the values for a city in the testing set.

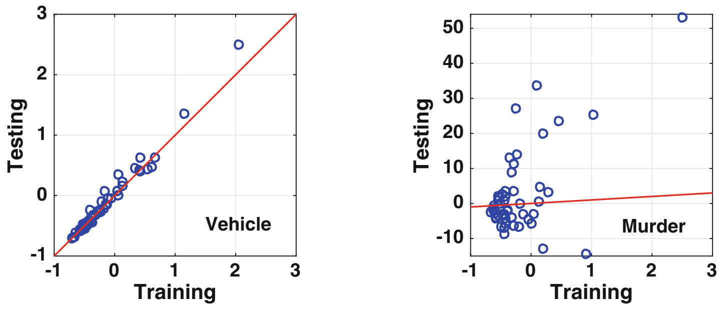


Figure 9.5 The testing value versus the value predicted using the training set. The (red) line is what would be obtained if the testing value and training value are equal.

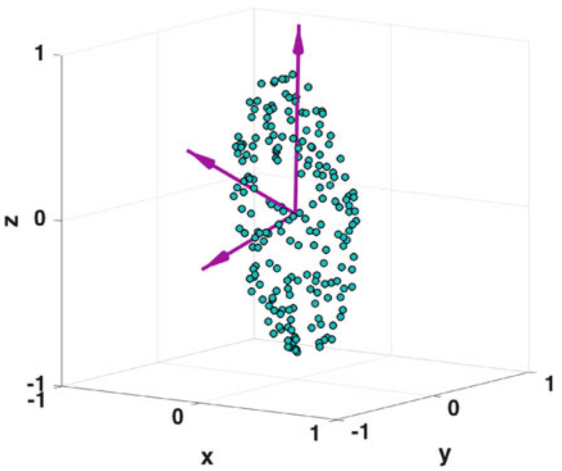


Figure 9.6 Example data set similar to the one shown in Figure 9.3, except now the three orthogonal vectors determined using a PCA are shown.

The training value for the city is the predicted value using (9.23). For example, to determine the predicted vehicle theft rate, the other rates (and population) for that city are used in (9.23) to calculate p_1. The second coordinate of the data points in Figure 9.5 are the known values from the testing dataset (for that particular crime). So, the better the prediction, the closer the data points will be to the line, which is what would be obtained if the two values are equal. What we conclude from this analysis is that it is possible to accurately predict the vehicle theft rate, but not the murder rate. This naturally leads to the question as to why this happens. The answer to this involves complicated social and economic reasons, and outside the prevue of mathematics. What the mathematics has done is made the connections, or non-connections, evident. It is up to the disciplinary experts to explain why.

9.2.5 Geometry and Data

The PCA was derived using a regression analysis, and the usefulness of this was demonstrated in the subsequent examples. However, it is worth considering what was done geometrically. For this we will use the data shown in Figure 9.6. The first step, which involved centering, is nothing more than finding the centroid of the data and then shifting the coordinate system so this is the origin. The regression step in a PCA is equivalent to rotating the coordinate system about the origin so it is aligned with the data, with the first coordinate vector pointing in the longest direction, the second coordinate pointing in the next longest direction, etc. This is illustrated in Figure 9.6.

Looking at the procedure geometrically helps identify some of the complications that might arise using a PCA. The one of most concern involves the scaling. This is done on each coordinate independently, which is equivalent to stretching (or contracting) the data points in the respective direction. Mathematically, it is possible to select scalings that dramatically distort the data. Consequently, it's important to select a scaling that reflects the data under consideration, and the vector norm based choices discussed in Section 9.2.3 are reasonable choices for this.

A second complication arises when the data form a circular, or spherical, pattern rather than an elliptical, or ellipsoidal, pattern shown in Figure 9.3. In such cases there is no longest direction. The PCA procedure works in such cases, but how it is used needs to be modified to be effective. This situation arises when the singular values repeat. As an example, suppose that there are three variables, and the singular values satisfy $\sigma_1 = \sigma_2$ and $\sigma_2 > \sigma_3$. The error, as given in (9.20), would be the same if we pick $\mathbf{v}_1$ or $\mathbf{v}_2$, and this means that the line of best fit is not unique. In such a case, the one dimensional approximation should be avoided and a two dimensional approximation used instead.

A third issue, with a more practical flavor, concerns the question of how many data points are needed for a PCA. The mathematical answer is simple, for a regression problem it is required that $m < n$. In considering this question, we begin with two variables, so $m = 2$. With this, the question is, how many points in the plane are needed so you can confidently determine the orientation of the elliptical pattern? The number is unclear, but it is certainly more than three. If it is assumed that 20 points are enough, then presumably for larger values of m the number is $10m$. This is known as the "rule of 10," and it is often used in experimental studies. Further discussion about such rules, including their limitations, can be found in Costello and Osborne [2005] and Mei et al. [2008].

9.2.6 Error

The error formulas used to derive Theorem 9.1 are useful enough that they should be considered in more detail. From (9.20) and (9.21), we found that the error using $\mathbf{v}_1$ is $E_1 = \sum \mathbf{p}_i \cdot \mathbf{p}_i - \sigma_1^2$, while using $\mathbf{v}_1$ and $\mathbf{v}_2$ the error is $E_2 = \sum \mathbf{p}_i \cdot \mathbf{p}_i - \sigma_1^2 - \sigma_2^2$. Generalizing this, if we use the first k columns from $\mathbf{V}$, then the error is

$$E_k = \sum_{i=1}^{n} \mathbf{p}_i \cdot \mathbf{p}_i - \left(\sigma_1^2 + \sigma_2^2 + \cdots + \sigma_k^2\right), \tag{9.24}$$

where the σ_j's are the singular values determined from the SVD for $\mathbf{P}$. It is reasonable to expect that if all of the columns of $\mathbf{V}$ are used, which provide a basis for the possible $\mathbf{p}$ values, then the error is zero. If so, then it must be that

$$\sum_{i=1}^{n} \mathbf{p}_i \cdot \mathbf{p}_i = \sum_{j=1}^{m} \sigma_j^2 .$$

It is possible to use the SVD of $\mathbf{P}$ to prove that this does indeed hold. The quantity $\sum \mathbf{p}_i \cdot \mathbf{p}_i$ has different names, depending on the discipline. In statistics it is known as the variance of the normalized data set. In mathematics it gives rise to what is called the Frobenius norm of a matrix, and this is defined as

$$||\mathbf{P}||_F \equiv \sqrt{\sum_{i,j} p_{ij}^2} = \sqrt{\sum_{i=1}^{n} \mathbf{p}_i \cdot \mathbf{p}_i} .$$

This is an example of a matrix norm that is not induced by a vector norm, what is referred to in Chapter 3 as an unnatural norm.

What the above formulas show is that the jth column of $\mathbf{V}$ contributes an amount σ_j^2 to the variance. It also shows that the relative error when using the first k columns from $\mathbf{V}$ is

$$R(k) = \frac{\sigma_{k+1}^2 + \sigma_{k+2}^2 + \cdots + \sigma_m^2}{\sigma_1^2 + \sigma_2^2 + \cdots + \sigma_m^2} , \tag{9.25}$$

assuming that $1 \leq k < m$. This is also called the relative residual variance.

The singular values and the relative error (9.25) for the normalized crime data are given in Figure 9.7. This information can be used to decide on how many columns (components) to use for a PCA. For example, one recommendation, known as the Guttman-Kaiser criterion, is that the number is determined by the average $\frac{1}{m}\sum \sigma_j$. Specifically, the columns with values of σ larger than this average are used. For the data shown in Figure 9.7, the average is 4.42, and so according to the Guttman-Kaiser criterion, the first two columns should be used. Another recommendation is based on a noticeable change in the plot of the singular values. This is known as Cattell's scree test, and it uses the point where there is a clear drop in the singular values, or they start to even out (at small values). Applying this to Figure 9.7, one would again decide on taking the first two columns. Finally, there are those who use the relative error (9.25), using a criterion of the form $R(k) < tol$, where *tol* is a number chosen somewhere between 0.25 and 0.05. In this case, from the plot in Figure 9.7, one would take 1 to 2 columns, depending on the choice for *tol*. Needless to say, there has been considerable discussion of this in the research literature, and some insights about this can be found in Jackson [1993], Peres-Neto et al. [2005], and Josse and Husson [2012].

9.2.7 Parting Comments

PCA is used in a wide variety of applications, and one reason is that it provides a relatively simple way to determine connections between variables in large datasets. As examples, it has been used in magnetic resonance imaging (MRI) of the brain [Jung et al., 2012; Andersen et al., 1999], for studying turbulent combustion [Parente and Sutherland, 2013], to determine connections between cancer and gene expression [Khan et al., 2001], for fraud detection for bodily injury claims in automobile insurance [Brockett et al., 2002], for assessing energy sustainability of rural communities [Doukas et al., 2012], and for cosmological tests of general relativity [Hojjati et al., 2012]. There are also variations on how to compute a PCA, or an approximate PCA. One of recent interest uses a randomized algorithm, that can be used on very large data sets, and this is discussed in Rokhlin et al. [2010].

As successful as PCA has been, it is based on a major assumption, which is that the variables are related through a linear equation. In many problems the dependence is nonlinear, which limits the use of PCA. For example, suppose you are interested in the trajectory of a baseball, and your data variables are speed, position, and time. A PCA study would assume that these variables are linearly related, where a more appropriate assumption would be that the speed is determined by the ratio of the relative values for position and time, which is a nonlinear relationship. A significant effort has been invested in developing a nonlinear version of PCA, and this falls into the more general

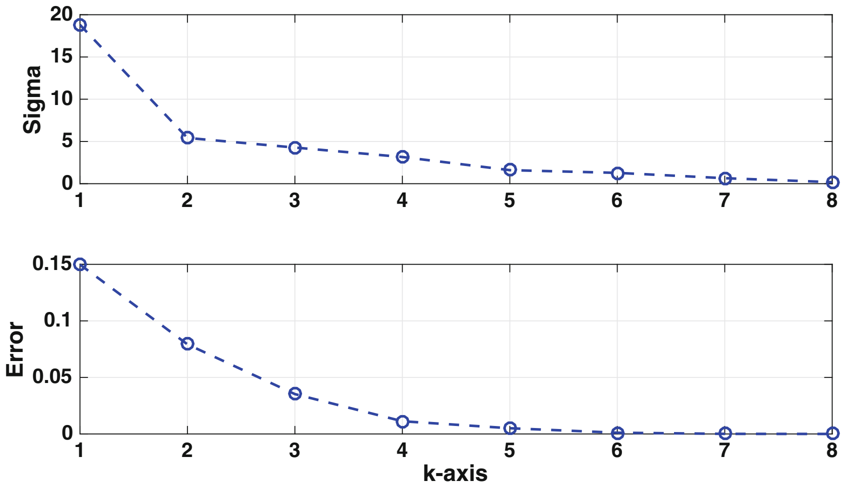


Figure 9.7 Top: Singular values for the normalized crime data. Bottom: Relative error (9.25) for the crime data.

problem of finding dimensionality reduction methods. There are numerous versions of this, including isomaps, diffusion maps, Laplacian eigenmaps, etc. A nice review of some of them, and how well they work, can be found in van der Maaten et al. [2009] and Burges [2010].

As a final comment, PCA is used extensively in statistics, which is a subject not considered in this text. For those who might be interested in the statistical implications of PCA, the word length example was based on a similar example considered in Abdi and Williams [2010]. They concentrate on the statistical applications, which includes correlations, statistical inference and confidence intervals, and it should be consulted to learn more about this.

9.3 Independent Component Analysis

To introduce the idea considered next, consider the situation of microphones recording sounds from multiple sources. Because the microphones are placed at different distances and angles from the sources, what they record will differ. The question is, from nothing more than what is recorded by each microphone, it is possible to separate out the independent sources (components) contributing to the sound record? This is an example of what is called a blind source separation problem.

As a simple example of what is being considered, suppose there are two sources, S_1 and S_2, and they are emitting the waves shown in Figure 9.8. There are also two microphones M_1 and M_2. It is assumed that the signals

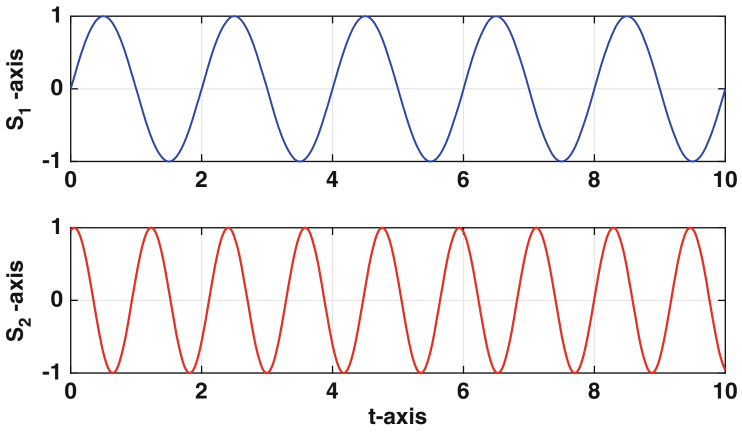


Figure 9.8 Waves being emitted by sources S_1 and S_2.

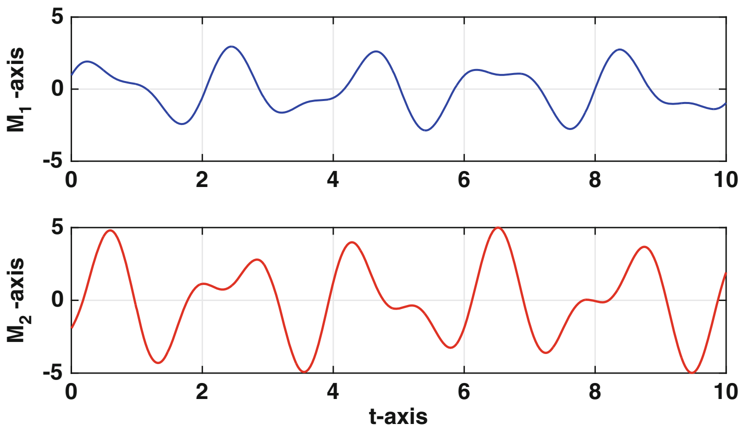


Figure 9.9 Waves recorded by microphones M_1 and M_2.

they record are $M_1 = 2S_1 + S_2$ and $M_2 = 3S_1 - 2S_2$, which produce the curves shown in Figure 9.9. So, the question is, given the data in Figure 9.9, how close can we come to reproducing the source data in Figure 9.8.

To answer this, we will make several assumptions. The first is that the recorded, or observed, data is determined linearly from the source values. In the case of two sources, s_1 and s_2, and two recorders, x_1 and x_2, it is assumed that

$$\begin{aligned} x_1 &= a_{11}s_1 + a_{12}s_2 \qquad (9.26) \\ x_2 &= a_{21}s_1 + a_{22}s_2. \end{aligned}$$

Written this way, the problem is to determine the a_{ij}'s and s_j's from the measured values of the x_i's. Given that there are six unknowns, and only two equations, it is apparent that the solution is not unique. Nevertheless, it is possible to derive useful information from this analysis, and this will be apparent later, after we complete the derivation of the method.

A few comments are needed before continuing. First, for the time series data in Figure 9.9, it is assumed that the same coefficients a_{ij} are used at each time point (i.e., they do not change with t). Also, note that we are using the same number of recorders as sources, and this will play an important role later. Finally, our solution will come close to determining the sources, but it will not be able to determine which is which (i.e., which one is s_1 and which is s_2).

9.3.1 Derivation of Method

The general form for the data set is given in Table 9.5. The assumption we are making is that at each $t = t_i$, the recorded and source signals are connected as in (9.26). This can be written in matrix form as

$$\mathbf{x}_i = \mathbf{A}\mathbf{s}_i \, , \tag{9.27}$$

where $\mathbf{A}$ is called the *mixing matrix* and it is assumed to be invertible. Note that in writing the equation this way, the problem is not limited to two recorders. In fact, in deriving the method, it's assumed there are m sources and m recorders. Consequently, $\mathbf{s}_i = (s_{i1}, s_{i2}, \cdots, s_{im})^T$ is the m-vector containing the values of the sources at time t_i, $\mathbf{x}_i = (x_{i1}, x_{i2}, \cdots, x_{im})^T$ is the m-vector containing the observed values at time t_i, and the mixing matrix is $m \times m$.

The objective is to find $\mathbf{A}$ using the $\mathbf{x}_i$'s. In the derivation to follow, at least at the beginning, this will be done using all of the data values. However, as the method unfolds, it will become evident that only a subset of the data values are needed to find $\mathbf{A}$. This is important because the data sets that arise in applications can be enormous, and finding the mixing matrix using the entire data set is essentially impossible from a computational point of view.

As was done for a PCA, the data will be centered by letting $\mathbf{x}_i^* = \mathbf{x}_i - \mathbf{x}_M$, where

$$\mathbf{x}_M = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i.$$

With this, (9.27) takes the form

$$\mathbf{x}_i^* = \mathbf{A}\mathbf{s}_i^*, \tag{9.28}$$

where $\mathbf{s}_i^* = \mathbf{s}_i - \mathbf{s}_M$ and $\mathbf{x}_M = \mathbf{A}\mathbf{s}_M$. Note that because $\mathbf{A}$ is invertible, it follows that

$$\frac{1}{n}\sum_{i=1}^{n} \mathbf{s}_i^* = \mathbf{0}. \tag{9.29}$$

In other words, the centered sources have zero mean.

The data values are also going to be scaled, but not in the straightforward method used for PCA. To explain what this is, the solution is going to be found using a singular value decomposition (SVD). As described in Section 4.5.2, a SVD involves the eigenvalues and eigenvectors of a matrix of the form $\mathbf{A}\mathbf{A}^T$. This will require us to consider the matrices $\mathbf{x}^*(\mathbf{x}^*)^T$ and $\mathbf{s}^*(\mathbf{s}^*)^T$, which are called outer products. For those who might not remember what these are, given a vector $\mathbf{x} = (x_1, x_2, \cdots, x_m)^T$, the outer product is

		x_1	x_2	$\cdots$	x_m		x_1^*	x_2^*	$\cdots$	x_m^*
t_1	$\mathbf{x}_1$	x_{11}	x_{12}	$\cdots$	x_{1m}	$\mathbf{x}_1^*$	x_{11}^*	x_{12}^*	$\cdots$	x_{1m}^*
t_2	$\mathbf{x}_2$	x_{21}	x_{22}	$\cdots$	x_{2m}	$\mathbf{x}_2^*$	x_{21}^*	x_{22}^*	$\cdots$	x_{2m}^*
$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$
t_n	$\mathbf{x}_n$	x_{n1}	x_{n2}	$\cdots$	x_{nm}	$\mathbf{x}_n^*$	x_{n1}^*	x_{n2}^*	$\cdots$	x_{nm}^*

Table 9.5 Original time series data for x_1, x_2, $\cdots$, x_m on the left, and their centered values x_1^*, x_2^*, $\cdots$, x_m^* on the right.

$$\mathbf{x}\mathbf{x}^T \equiv \begin{pmatrix} x_1^2 & x_1x_2 & x_1x_3 & \cdots & x_1x_m \\ x_1x_2 & x_2^2 & x_2x_3 & \cdots & x_2x_m \\ \vdots & \vdots & \vdots & & \vdots \\ x_1x_m & x_2x_m & x_3x_m & \cdots & x_m^2 \end{pmatrix}.$$

Also, as mentioned earlier, the solution of our problem is not unique, and one way to see this is that (9.28) can be rewritten as

$$\mathbf{x}^* = (\mathbf{A}\mathbf{C}^{-1})(\mathbf{C}\mathbf{s}^*),$$

where $\mathbf{C}$ is an arbitrary invertible matrix. What this means is that if $\mathbf{A}$ and $\mathbf{s}^*$ is claimed to be a solution, then so is $\mathbf{A}\mathbf{C}^{-1}$ and $\mathbf{C}\mathbf{s}^*$. Among the various solutions possible for this problem we will look for one that scales so that

$$\frac{1}{n}\sum_{i=1}^{n} \mathbf{s}_i^*(\mathbf{s}_i^*)^T = \mathbf{I}, \tag{9.30}$$

where $\mathbf{I}$ is the $m \times m$ identity matrix. The practical reason for making this assumption is that it will enable us to find a solution. In statistics this is referred to as *whitening* the source data, and it has the effect of removing the correlations in the input. Finally, given a data set, it is not difficult to whiten it, and the steps involved are outlined in Exercise 9.3.

Assuming there are m sources, and m recorders, then using a SVD (Section 4.5.2), it is possible to write

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T,$$

where $\mathbf{U}$ is an $m \times m$ orthogonal matrix, $\mathbf{V}$ is an $m \times m$ orthogonal matrix, and $\mathbf{\Sigma}$ is an $m \times m$ diagonal matrix. This would seem to make solving the problem more difficult because instead of finding one matrix, we are now looking for three. However, note that from (9.28),

$$\begin{aligned}
\frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i^*(\mathbf{x}_i^*)^T &= \frac{1}{n}\sum_{i=1}^{n}(\mathbf{A}\mathbf{s}_i^*)(\mathbf{A}\mathbf{s}_i^*)^T \\
&= \frac{1}{n}\sum_{i=1}^{n}\mathbf{A}(\mathbf{s}_i^*)(\mathbf{s}_i^*)^T\mathbf{A}^T \\
&= \mathbf{A}\left(\frac{1}{n}\sum_{i=1}^{n}\mathbf{s}_i^*(\mathbf{s}_i^*)^T\right)\mathbf{A}^T \\
&= (\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T)(\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T)^T \\
&= \mathbf{U}\boldsymbol{\Sigma}^2\mathbf{U}^T. \qquad (9.31)
\end{aligned}$$

Now, each outer product $\mathbf{x}_i^*(\mathbf{x}_i^*)^T$ is a symmetric matrix, which means the left hand side of the above equation is a symmetric matrix. It is also possible to show that its eigenvalues are never negative, which means it is a positive semidefinite matrix. Therefore, according to Theorem 4.7 (Section 4.5.2), it has a SVD of the form

$$\frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i^*(\mathbf{x}_i^*)^T = \mathbf{Q}\mathbf{D}\mathbf{Q}^T, \qquad (9.32)$$

where $\mathbf{D}$ is a diagonal matrix with non-negative entries, and $\mathbf{Q}$ is an orthogonal matrix. It is also possible to assume that $\mathbf{Q}$ is a proper orthogonal matrix, which means that it corresponds geometrically to a rotation.

It is important to note that both $\mathbf{D}$ and $\mathbf{Q}$ are known, or determinable, from the data. Comparing (9.31) and (9.32), it is seen that we are able to determine two of the matrices in the SVD for $\mathbf{A}$. Namely, we can take $\mathbf{U} = \mathbf{Q}$ and $\boldsymbol{\Sigma} = \mathbf{D}^{1/2}$. In this last expression, since $\mathbf{D}$ is a diagonal matrix with diagonals d_i, then $\mathbf{D}^{1/2}$ is a diagonal matrix with diagonals $\sqrt{d_i}$. We are going to need $\mathbf{D}$ to be invertible, and so the data must be such that the d_i's are positive. According to Theorem 4.8, this is true if the columns of the normalized data matrix are independent, which means that the matrix has rank m (see Exercise 9.7).

The outer product sum in (9.32) plays a central role in the analysis, and it can be expressed in different ways. To explain, let $\mathbf{X}^*$ denote the $n \times m$ centered data matrix (see Table 9.5). It is not hard to show that

$$\frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i^*(\mathbf{x}_i^*)^T = \frac{1}{n}\mathbf{X}^*(\mathbf{X}^*)^T. \qquad (9.33)$$

The right hand side of the above equation is known as the *covariance matrix* of $\mathbf{X}^*$. Another useful formula that comes directly from the above equation is

$$\frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i^*(\mathbf{x}_i^*)^T = \frac{1}{n}\begin{pmatrix} \mathbf{c}_1^*\cdot\mathbf{c}_1^* & \mathbf{c}_1^*\cdot\mathbf{c}_2^* & \mathbf{c}_1^*\cdot\mathbf{c}_3^* & \cdots & \mathbf{c}_1^*\cdot\mathbf{c}_m^* \\ \mathbf{c}_2^*\cdot\mathbf{c}_1^* & \mathbf{c}_2^*\cdot\mathbf{c}_2^* & \mathbf{c}_2^*\cdot\mathbf{c}_3^* & \cdots & \mathbf{c}_2^*\cdot\mathbf{c}_m^* \\ \vdots & \vdots & \vdots & & \vdots \\ \mathbf{c}_m^*\cdot\mathbf{c}_1^* & \mathbf{c}_m^*\cdot\mathbf{c}_2^* & \mathbf{c}_m^*\cdot\mathbf{c}_3^* & \cdots & \mathbf{c}_m^*\cdot\mathbf{c}_m^* \end{pmatrix}, \tag{9.34}$$

where $\mathbf{c}_1^*, \mathbf{c}_2^*, \cdots$, and $\mathbf{c}_m^*$ are the n-vectors coming from the m columns of $\mathbf{X}^*$. In linear algebra the above matrix is referred to as a Gram matrix. Although (9.34) looks more cumbersome than the formula in (9.33), it is actually more useful when computing the covariance matrix. Specifically, because of the symmetry of the matrix, it is only necessary to compute the entries on, and above, the diagonal. For larger data sets, this reduces the computing time by about half, and this is explored in Exercise 9.8.

9.3.2 Reduced Problem

What we have been able to establish, so far, is that the mixing matrix $\mathbf{A}$ in (9.27) has the form

$$\mathbf{A} = \mathbf{Q}\mathbf{D}^{1/2}\mathbf{V}^T. \tag{9.35}$$

In other words, the solution of the problem can be written as

$$\mathbf{s}_i^* = \mathbf{V}\mathbf{y}_i^*, \tag{9.36}$$

where $\mathbf{y}_i^* = \mathbf{D}^{-1/2}\mathbf{Q}^T\mathbf{x}_i^*$. In terms of the original variables, the solution is

$$\mathbf{s}_i = \mathbf{W}\mathbf{x}_i, \tag{9.37}$$

where $\mathbf{W} = \mathbf{V}\mathbf{D}^{-1/2}\mathbf{Q}^T$ is known as the *unmixing matrix*. What is left to do is determine the orthogonal matrix $\mathbf{V}$.

To help explain how $\mathbf{V}$ is determined, consider the original two source example shown in Figure 9.9. Because $m = 2$, and the matrix $\mathbf{V}$ is orthogonal, it is possible to write

$$\mathbf{V} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}. \tag{9.38}$$

Geometrically, $\mathbf{V}$ corresponds to a counter-clockwise rotation through an angle θ. This is useful to know, and to explain, in Figure 9.10, the experimental values for $\mathbf{s}_i^*$ are plotted in the (s_1^*, s_2^*)-plane, and the experimental values for $\mathbf{y}_i^*$ are plotted in the (y_1^*, y_2^*)-plane. Note that the $\mathbf{s}_i^*$ values have been whitened, which means they have been normalized so they satisfy (9.30). According to (9.36), the problem of finding $\mathbf{V}$ means we are looking for an angle that will rotate the $\mathbf{y}_i^*$ data in Figure 9.10 so they will approximate the $\mathbf{s}_i^*$ data. Looking at the $\mathbf{y}^*$ data region, it appears that a rotation of 20° to 30° will align it with the $\mathbf{s}^*$ region. It is also apparent there are four solutions,

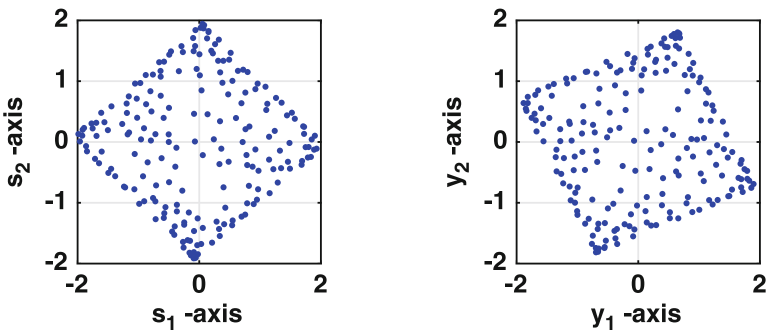


Figure 9.10 On the left the experimental values of $\mathbf{s}_i^*$ are plotted, and on the left the experimental values for $\mathbf{y}_i^*$ are plotted.

and they differ by approximately 90° from each other. A second useful observation coming from Figure 9.10 is that if the data regions happen to be circular, rather than square, then there is no rotation for the $\mathbf{y}_i^*$'s that will provide the best fit. In statistics, circular data patterns arise with Gaussian distributions, and avoiding these will play a central role in finding $\mathbf{V}$.

9.3.3 Contrast Function

The matrix $\mathbf{V}$ is found by introducing what is called a *contrast function*, which serves a similar purpose to the error function in regression. Before doing this, it is worth first determining what angle produces the best fit (so we will know if the contrast function is successful in finding it). This requires a measure of how different, or similar, the computed values for the sources are compared to the original values. To explore this idea, some of the possible solutions obtained from (9.36) and (9.38), for particular values of θ, are shown in Figure 9.11. The original source curves are also shown. One way to measure how close the dashed and solid curves are in this figure is to use the area between them (the closer they are, the smaller the area). This is easy to compute, and the result is shown in Figure 9.12 (see Exercise 6.25 for the specifics on how this is done). Locating the minimum for this curve, the conclusion is that the best fit occurs when $\theta \approx 0.63\pi$, and the corresponding solution is shown in the lower row in Figure 9.11. The third row in the figure corresponds to the angle producing the maximum area in Figure 9.12.

The reason for introducing a contrast function is to find the optimal angle. What makes the problem so challenging is that the minimum error must be determined without knowing the $\mathbf{s}_i$'s. Just so its clear, the area between the curves, which is used to produce Figure 9.12, can not be used as a contrast function, because it requires knowing the $\mathbf{s}_i$'s.

Kurtosis

The ICA method was originally developed as a statistical tool for data analysis, and the assumption central in the development is that the sources are independent. More precisely, they are described using independent probability distributions. This can be achieved if the distributions are non-Gaussian. As a measure of how Gaussian they are, the *kurtosis* is used. For a single distribution s^*, that satisfies (9.29) and (9.30), the kurtosis is computed using the formula

$$\kappa = \frac{1}{n}\sum_{i=1}^{n}(s_i^*)^4 - 3.$$

If s^* is Gaussian, the exact value for the kurtosis is $\kappa = 0$. For the ICA we want s^* to be as non-Gaussian as possible, and so the goal is to find a solution which produces a kurtosis value that is as far away from zero as possible. In other words, we need to find a solution which maximizes $|\kappa|$ or κ^2.

The complication for ICA is that there are multiple sources, and this means the measure of non-Gaussianity must be generalized. One possibility for a

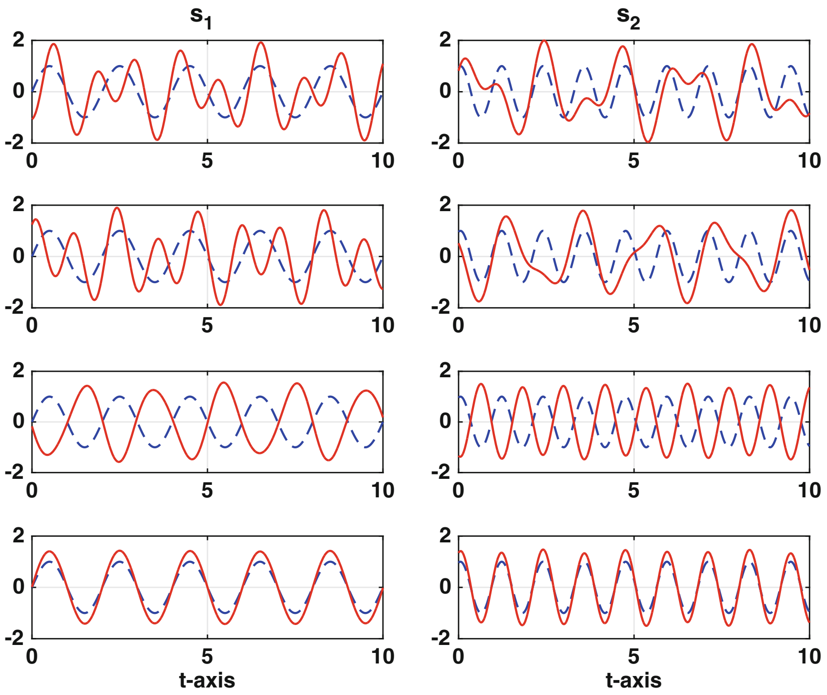


Figure 9.11 The solid (red) curves are the values for $\mathbf{s}_i^*$ obtained from (9.36) and (9.38) for four different values of θ. The dashed (blue) curves are the original sources used to generate the data. For each row, starting at the top, $\theta = \pi/3$, π, $5\pi/3$, and 0.63π.

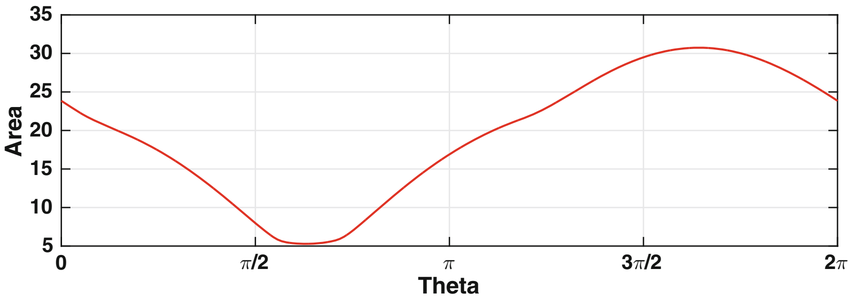


Figure 9.12 The area between the computed and actual source curves, as a function of the angle θ in (9.38).

contrast function is to simply add the functions together. Assuming there are m sources, then the result is

$$K = \sum_{j=1}^{m} |\kappa_j|, \tag{9.39}$$

where

$$\kappa_j = \frac{1}{n}\sum_{i=1}^{n}(s_{ij}^*)^4 - 3, \tag{9.40}$$

and $\mathbf{s}_i^* = (s_{i1}^*, s_{i2}^*, \cdots, s_{im}^*)$. The requirement is that, after substituting (9.36) into this expression, that $\mathbf{V}$ maximizes K. For the case of when $m = 2$, (9.39) is

$$K(\theta) = |\kappa_1| + |\kappa_2|, \tag{9.41}$$

where

$$\kappa_1 = \frac{1}{n}\sum_{i=1}^{n}(y_{i1}^* \cos\theta - y_{i2}^* \sin\theta)^4 - 3, \tag{9.42}$$

$$\kappa_2 = \frac{1}{n}\sum_{i=1}^{n}(y_{i1}^* \sin\theta + y_{i2}^* \cos\theta)^4 - 3, \tag{9.43}$$

and $\mathbf{y}_i^* = (y_{i1}^*, y_{i2}^*)^T$. The connection between $\mathbf{y}_i^*$ and the source data is given in (9.36).

As an example, K is plotted in Figure 9.13, using the data from Figure 9.9. There are four maximum points, corresponding to the values $\theta_1 = 0.434$, $\theta_2 = 2.01$, $\theta_3 = 3.58$, and $\theta_4 = 5.15$. The corresponding solutions for the sources are shown in Figure 9.14. Note that the respective curves for s_1, and for s_2, differ only by a multiple of -1, and for the ICA method they are equivalent. Also note that θ_2 is very close to the value obtained earlier using the area between the curves, which found that the optimal solution is

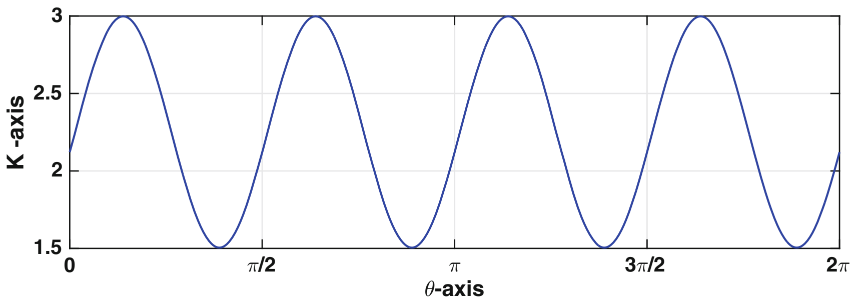


Figure 9.13 Function K, given in (9.41), plotted over the interval $0 \leq \theta \leq 2\pi$, using the data from Figure 9.9.

$\theta = 0.63\pi \approx 1.977$. As a final comment, finding four solutions is consistent with our earlier observation that there are four rotation angles in Figure 9.10.

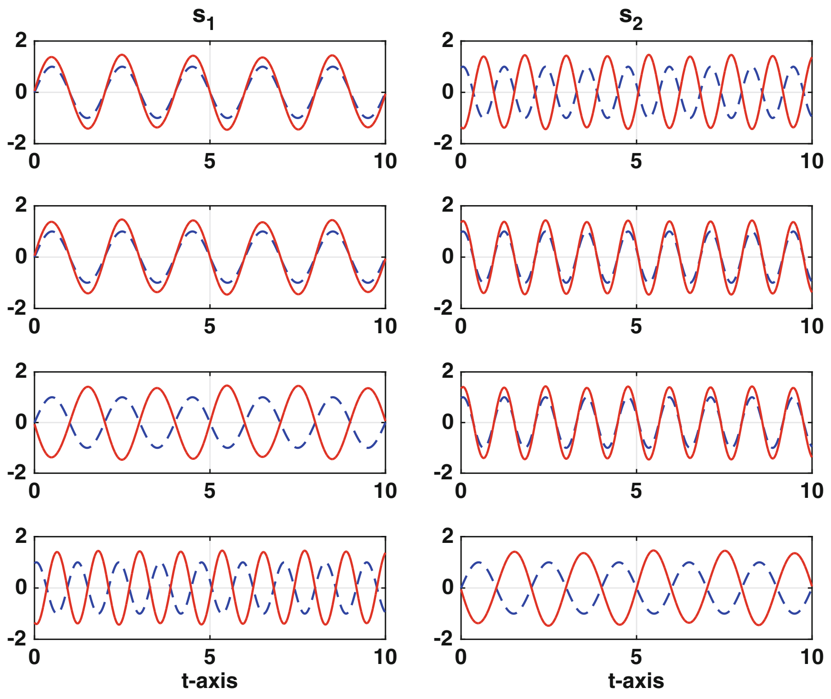


Figure 9.14 Solutions obtained for the four values for θ that maximize K, solid (red) curves, and the original signals from Figure 9.8, dashed (blue) curves.

Reduced Data Set

In the above example, $n = 6000$ and all of the $\mathbf{x}_i^*$ data points were used to find the mixing matrix $\mathbf{A}$. The computationally expensive step in the derivation involves computing $\sum \mathbf{x}_i^*(\mathbf{x}_i^*)^T$. It is possible to obtain a reasonably accurate approximation for this matrix using only a small subset of the data. To demonstrate this, first note that the order of the $\mathbf{y}_i^*$'s used to compute the kurtosis functions (9.41)–(9.43) is not important. In fact, if they are relabeled in a random order, the value of K is unchanged. So, suppose we randomly pick N data points. Using these points, the kurtosis function (9.41) can be evaluated, similar to the situation shown in Figure 9.13. What is of interest here is the location θ_2 of the second maximum, which is the angle producing the optimal solution. The value of θ_2, as a function of N, is shown in Figure 9.15. This shows that the values of θ_2 stabilize once N is 40 or larger.

There have been proposals for how to determine the minimum number of data points needed. For EEG signals, the assumption that the mixing matrix is time independent is applicable only over relatively short time periods, and the concern is that they are able to acquire enough data to carry out an ICA. The rule of thumb they often use is that one needs at least km^2 data points, where k is a constant between 5 and 32 [Särelä and Vigário, 2003; Onton and Makeig, 2006; Korats et al., 2013]. The result in Figure 9.15 is consistent with this, which shows that k should be 10 or larger for this example. However, for the image separation problem considered later, it appears that k should be at least 50.

An idea related to using a reduced sample size is known as downsampling. In this case, instead of using every data point, one uses every other point, or every third point, or every Kth point. This results in using, approximately, n/K data points. The difference here is that the recommendation is based on the number of data points, while the EEG rule is based on the number of unknowns in the mixing matrix.

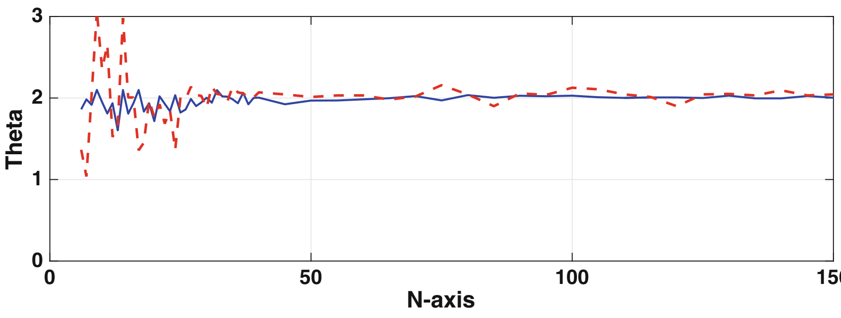


Figure 9.15 Value of the optimal angle as a function of the number N of random data points used. Shown are the values for a particular case, dashed (red) curve, and the average using 10 cases, solid (blue) curve.

Parting Comments

In the case of m sources, the contrast function in (9.41) can be written as $K = ||\boldsymbol{\kappa}||_1$, where $\boldsymbol{\kappa} = (\kappa_1, \kappa_2, \cdots, \kappa_m)^T$. It is also possible to use $K = ||\boldsymbol{\kappa}||_2^2$ or $K = ||\boldsymbol{\kappa}||_\infty$. When using m sources, the orthogonal matrix $\mathbf{V}$ contains $\frac{1}{2}m(m-1)$ independent entries, and these are determined by maximizing the contrast function. So, an ICA requires solving a multidimensional maximization problem, and the methods considered in Chapter 8 can be used in this case. The fly in the ointment is the contrast function. The kurtosis based functions considered here have the advantage of simplicity, but it is possible to find examples where they produce non-optimal answers (i.e., the solutions they produce are not the best possible). This has been the impetus for finding better contrast functions, as well as a more refined theoretical explanation of how they produce optimal solutions. Comprehensive introductions to the ICA method, which discuss different types of contrast functions, can be found in Comon and Jutten [2010] and Hyvärinen et al. [2001]. It is also possible to generalize the method to the problem of having more sources than recorders (overcompete ICA) and not as many (undercomplete ICA), and more about these can be found in Fabian et al. [2004] and Teh et al. [2003].

9.3.4 Summary of ICA

Given an $n \times m$ data set $\mathbf{X}$, corresponding to m sources and n observations, an ICA consists of the following steps:

1. Center the data to produce the data set $\mathbf{X}^*$.
2. Compute the matrix $\frac{1}{n}\mathbf{X}^*(\mathbf{X}^*)^T$ and then find its SVD, which can be written as $\frac{1}{n}\mathbf{X}^*(\mathbf{X}^*)^T = \mathbf{QDQ}^T$.
3. Compute $\mathbf{Y}^* = \mathbf{X}^*\mathbf{QD}^{-1/2}$.
4. Pick a contrast function $K(\mathbf{S}^*)$. Setting $\mathbf{S}^* = \mathbf{Y}^*\mathbf{V}^T$, determine the $m \times m$ orthogonal matrices $\mathbf{V}$ that minimize K.
5. Selecting one of the $\mathbf{V}$'s, the unmixing matrix is $\mathbf{W} = \mathbf{VD}^{-1/2}\mathbf{Q}^T$ and the corresponding source data are $\mathbf{S} = \mathbf{XW}^T$.

To use this procedure it is required that the columns of $\mathbf{X}$ are independent. Also, note that the above procedure is written in matrix form. To connect this with the vector version considered earlier, the ith row of the data matrix $\mathbf{X}$ is obtained from the data vector $\mathbf{x}_i$. The same applies to $\mathbf{X}^*$, $\mathbf{Y}^*$, $\mathbf{S}^*$, and $\mathbf{S}$. In doing this, because the original column vectors are put into the matrices as row vectors, the above formulas appear as transposes of the ones given earlier. As an example, the equations $\mathbf{s}_i = \mathbf{W}\mathbf{x}_i$ are written above in matrix form as $\mathbf{S} = \mathbf{XW}^T$.

9.3.5 Application: Image Separation

An interesting use of ICA is to separate images, and as an example the two images in the upper row in Figure 9.16 were mixed to produce the two images in the second row. To explain how this was done, each of the original grayscale images consists of an $M \times N$ array of integers, each with a value from 0 to 255. This number represents the intensity of gray for that pixel, with 0 corresponding to black and 255 corresponding to white. Letting the pixel matrices for the two source images be $\mathbf{S}_1$ and $\mathbf{S}_2$, then the pixel matrices for the two mixed images are $0.6\mathbf{S}_1 + 0.4\mathbf{S}_2$ and $0.4\mathbf{S}_1 + 0.6\mathbf{S}_2$.

Even though there is no time variable, it is possible to construct data vectors for each image. We will arrange the pixel information into vector form by simply putting the columns (or rows) together as one vector. Using columns, the first M entries of the data vector come from the first column of the pixel array, the next M entries come from the second column, etc. Applying this procedure to the two original images we obtain source data vectors $(s_{11}, s_{21}, \cdots, s_{n1})^T$ and $(s_{12}, s_{22}, \cdots, s_{n2})^T$, where $n = MN$. From this we then have $\mathbf{s}_i = (s_{i1}, s_{i2})^T$. The same procedure is applied to the mixed images to first obtain data vectors $(x_{11}, x_{21}, \cdots, x_{n1})^T$ and $(x_{12}, x_{22}, \cdots, x_{n2})^T$, and from this $\mathbf{x}_i = (x_{i1}, x_{i2})^T$. The images in Figure 9.16 are 1944×2592, so for this example $n = 5{,}038{,}848$. Also, a common question that arises is whether a different answer will be obtained if one uses rows instead of columns. The answer is no, and in fact, you can label the pixels randomly without changing the answer, as long as the same labeling order is used for both images. The reason is explained in the previous example when using a reduced data set.

The problem is now mathematically equivalent to the microphone example, which means the formulas we derived earlier are directly applicable to the data vectors for the images. In particular, the solution is given in (9.37), and the contrast function using the kurtosis is given in (9.39).

After finding the unmixed values using ICA, and then reassembling them into pixel arrays, the images shown in Figure 9.17 are obtained. The separation used the entire data set, although it is possible, and certainly faster, to use a reduced data set. To illustrate, the computed values of the kurtosis for a particular angle are shown in Figure 9.18, both as an average as well for a particular instance. It would appear that the value of k for the EEG rule of thumb, which was explained earlier, is about 50 in this case.

How you react to the images in Figure 9.17 depends on what you were planning to do with them. If you had hoped to have them framed and mounted on your wall, then you are likely disappointed. The reason being that the separation is incomplete (e.g., there is a residual image of the TV in the image on the right). Also, the image quality is not the same as the originals. If, on the other hand, you intended to use these images for feature detection, or motion control, then they are likely adequate. These are some of the primary applications of ICA, and particular examples include facial recognition [Bartlett et al., 2002], finding defects in fabrics [Zhang and Cheng, 2014], identifying

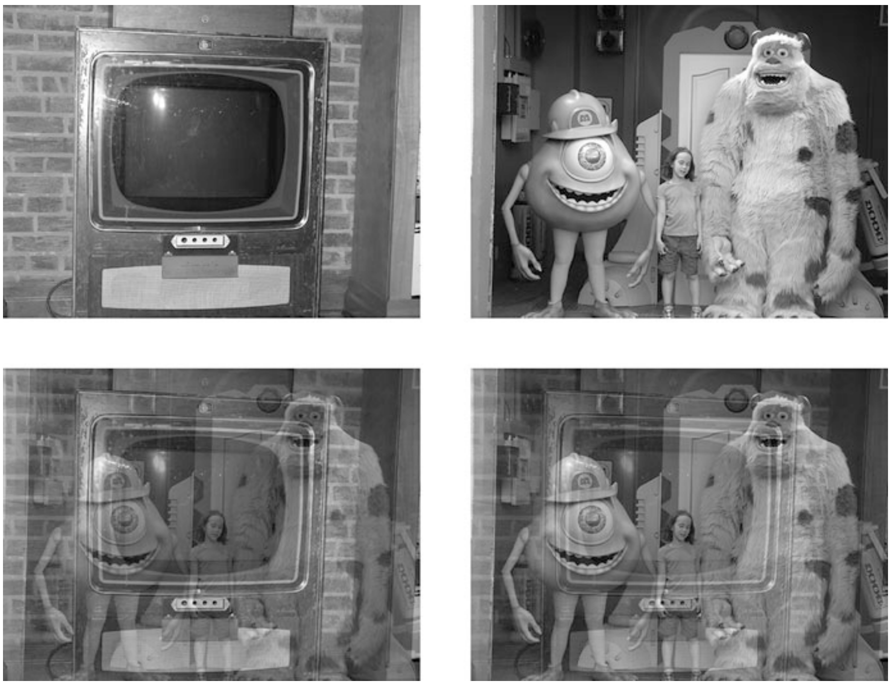


Figure 9.16 Original images taken in Disney World, top row, and the images obtained after they are mixed, bottom row.

3D shapes of the heart [Wu et al., 2014], detection of Down syndrome in newborns [Zhao et al., 2014], and for methods of watermarking images [Nguyen and Patra, 2008].

To address possible improvements for using ICA to separate images, note that it does not appear to be possible to get a better result by using a different contrast function. The images in Figure 9.17, which are found using the kurtosis function, are almost indistinguishable from the images one gets by

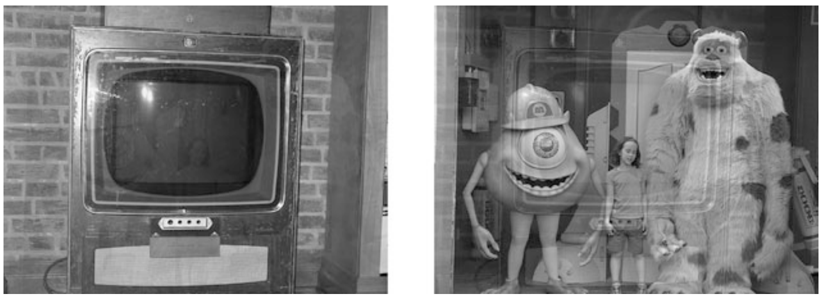


Figure 9.17 Result of using ICA to unmix the two images.

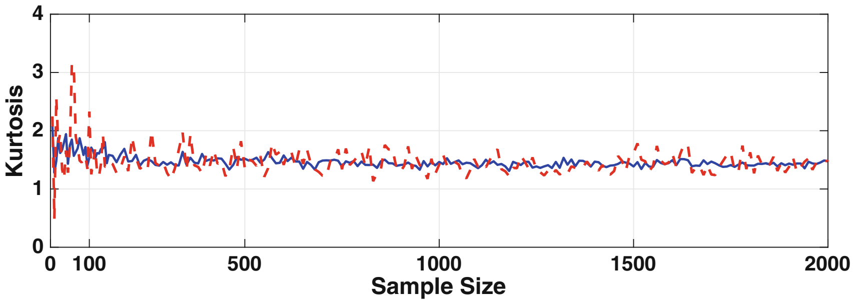


Figure 9.18 Value of the kurtosis function (9.41), for $\theta = \pi/4$, as a function of the number of random data points used. Shown are the values for a particular case, dashed (red) curve, and the average using 10 cases, solid (blue) curve.

minimizing the area. To be specific, using the kurtosis function, the angle is found to be $\theta = 1.1020$, which is very close to the optimal value $\theta = 0.9952$ determined using the 2-norm with the original sources, as explained in Exercise 9.9.

A second point that should be made is that the form of ICA used here is inherently lossy because floating point transformations are being applied to integer arrays. So, when converting the computed results back into integer arrays, round off error is almost inevitable, and this reduces the quality of the final images.

There have been several proposals on ways to measure how well ICA has separated the sources, what are called performance measures [Chen and Bickel, 2006; Nordhausen et al., 2011]. There has also been work on what is called robustification of ICA. It is not difficult to find examples where outliers in the data, produced by noise or other artifacts, can seriously degrade the quality of separation. Removing such data is called robustification, and this is discussed more in Anderson and Adali [2010] and Brys et al. [2005].

9.4 Modal Data Analysis

As usual, we will introduce the ideas underlying this method by considering an example. The one of interest has to do with the spread of flu. Over the course of a year, the portion of the population sick with the flu changes dramatically across the country, and we would like to have a way of understanding how it propagates through the population. There are fairly sophisticated models for the epidemiology of flu, but we are going to derive the information directly from experimental observations. In doing this, assume there are m locations at which the flu information is collected. It is assumed these are distributed across the country, but exactly where they are is not important for the moment. Suppose at time t_i the number of individuals with the flu at each location is counted, producing an m-vector $\mathbf{x}_i$. We then wait a period of time, and again count the flu cases at each location to produce an m-vector $\mathbf{y}_i$. It is going to be assumed that $\mathbf{y}_i$ can be predicted using a linear function of $\mathbf{x}_i$, and so

$$\mathbf{y}_i = \mathbf{A}\mathbf{x}_i + \mathbf{b}.$$

This can be simplified by normalizing the data, using centering and scaling in exactly the same way as was done for the PCA. Expressing this in matrix form, the normalized $\mathbf{x}$'s are

$$\mathbf{x}_i^* = \mathbf{S}_x^{-1}(\mathbf{x}_i - \mathbf{x}_M),$$

where $\mathbf{x}_M = \frac{1}{n}\sum \mathbf{x}_i$ is the mean, and $\mathbf{S}_x$ is a $m \times m$ diagonal matrix containing the scaling factors for the components of $\mathbf{x}$. Similarly, the normalized $\mathbf{y}$'s are

$$\mathbf{y}_i^* = \mathbf{S}_y^{-1}(\mathbf{y}_i - \mathbf{y}_M).$$

Using the scaled variables, the assumption of linearity can now be written as

$$\mathbf{y}_i^* = \mathbf{P}\mathbf{x}_i^*, \tag{9.44}$$

where $\mathbf{P}$ is called the *propagator matrix.* A critical observation here is that it is assumed that $\mathbf{P}$ does not depend on t_i. However, it is very likely that $\mathbf{P}$ does depend on the time interval between when $\mathbf{x}_i$ and $\mathbf{y}_i$ are measured, and it is therefore assumed that the same interval is used for all data points.

Suppose there are n time points at which we conduct our counting experiment. This will produce a sequence of observation vectors $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n$, along with the subsequent observations $\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_n$. This information can be used to determine $\mathbf{P}$. Treating this as a least squares problem, the error function is

$$E(\mathbf{P}) = \sum_{i=1}^{n} ||\mathbf{P}\mathbf{x}_i^* - \mathbf{y}_i^*||_2^2 .$$

Finding the m^2 entries in $\mathbf{P}$ that minimize this function can be found by simply differentiating E with respect to each of these entries, and then setting the derivatives equal to zero. The calculation is straightforward, and to write down the solution, we introduce the $n \times m$ normalized data matrices $\mathbf{X}^*$ and $\mathbf{Y}^*$. The ith row of $\mathbf{X}^*$ is $(\mathbf{x}_i^*)^T$, and the ith row of $\mathbf{Y}^*$ is $(\mathbf{y}_i^*)^T$. Denoting the ith row of $\mathbf{P}$ as $\mathbf{p}_i^T$, then minimizing E reduces to solving

$$\mathbf{G}\mathbf{p}_i = (\mathbf{X}^*)^T\mathbf{c}_i,$$

where $\mathbf{G} = (\mathbf{X}^*)^T\mathbf{X}^*$ and $\mathbf{c}_i$ is the ith column of $\mathbf{Y}^*$. Note that $\mathbf{G}$ is an example of what is called a Gram matrix. Multiplying by $\mathbf{G}^{-1}$, and expressing the answer in matrix form, we have that

$$\mathbf{P} = (\mathbf{Y}^*)^T\mathbf{X}^*\mathbf{G}^{-1}. \tag{9.45}$$

In writing down this expression, it is assumed that the columns of $\mathbf{X}^*$ are independent, which guarantees that $\mathbf{G}$ is invertible. The above solution can be written as

$$\mathbf{P} = (\mathbf{Y}^*)^T(\mathbf{X}^+)^T,$$

where $\mathbf{X}^+ = \left((\mathbf{X}^*)^T\mathbf{X}^*\right)^{-1}(\mathbf{X}^*)^T$ is known as the *Moore-Penrose pseudoinverse* of $\mathbf{X}^*$.

Once $\mathbf{P}$ is known, returning to the original (unnormalized) variables,

$$\mathbf{y}_i = \mathbf{A}\mathbf{x}_i + \mathbf{b}, \tag{9.46}$$

where

$$\mathbf{A} = \mathbf{S}_y\mathbf{P}\mathbf{S}_x^{-1},$$

and $\mathbf{b} = \mathbf{y}_M - \mathbf{A}\mathbf{x}_M$.

9.4.1 Application: Google's Flu Data

The data we will consider comes from Google Flu Trends. Using search data, Google developed a method for estimating the flu activity and how this is done is partly explained in Ginsberg et al. [2009]. They provide data for each state, including Washington DC, in the United States, and the data were collected every seven days from 2007 through 2015. A portion of the dataset is shown in Table 9.6. Using the notation introduced earlier, $m = 51$, $n = 398$, and $\mathbf{x}_i$ consists of the 51 flu values listed in the ith row of the data matrix. Also note that $\mathbf{y}_i = \mathbf{x}_{i+1}$, $\mathbf{X}^*$ comes from the first 398 rows in the normalized data matrix, and $\mathbf{Y}^*$ comes from rows 2 to 399.

This brings us to the question of how well the assumption (9.46) works on this data set when the projector matrix is computed using (9.45). It is easy to compute $\mathbf{P}$ using this formula, and the resulting comparison for three states is given in Figure 9.19. Apparently the assumption works very well. Nevertheless, it is worth comparing this method with other possibilities. An easy one uses what is called a persistence forecast, which means that one simply assumes $\mathbf{y}_i$ is the same as $\mathbf{x}_i$ (this is often used in weather forecasting). Mathematically, this is equivalent to assuming that $\mathbf{P} = \mathbf{I}$ in (9.44). Using the least squares error to determine the difference between the actual and predicted curves, the error using the persistence forecast is about a factor of two larger than what is obtained using the above method.

Date	Alabama	Alaska	Arizona	Arkansas	$\cdots$	West Virginia	Wisconsin	Wyoming
2007/12/2	1438	1517	1582	3325	$\cdots$	2268	1288	429
2007/12/9	1278	1635	1710	2993	$\cdots$	2148	1420	471
2007/12/16	1161	1715	2018	2809	$\cdots$	2158	1513	524
2007/12/23	1250	1868	2601	2778	$\cdots$	2151	1685	576
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$
2015/6/28	707	780	1113	1547	$\cdots$	1479	696	691
2015/7/5	589	755	1582	1449	$\cdots$	1341	693	593
2015/7/12	549	685	1065	1256	$\cdots$	1290	660	520
2015/7/19	575	696	1116	1235	$\cdots$	1271	662	514

Table 9.6 Google Flu Trends data for the estimated number of flu cases in each state and Washington DC over an approximately eight year period [Google, 2015].

9.4.2 Propagation Modes

It is possible to derive an understanding of how the flu spreads through the country by looking at the SVD expansion of the propagator matrix **P**. As explained in Section 4.5.3, from the SVD of **P**, the matrix can be written as

$$\mathbf{P} = \sum_{i=1}^{r} \sigma_i \mathbf{W}_i, \tag{9.47}$$

where $\mathbf{W}_i = \mathbf{u}_i \mathbf{v}_i^T$ is the outer product matrix obtained using the columns of the matrices **U** and **V** computed using the SVD. The entries in the $\mathbf{W}_i$'s satisfy $-1 \leq w \leq 1$. Also, the σ_i's are the nonzero singular values of **P**, and satisfy $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$.

To apply this to the flu problem, note that the singular values for the normalized flu data are

$$\sigma_1 = 8.59,\ \sigma_2 = 3.81,\ \sigma_3 = 3.10,\ \sigma_4 = 2.73,\ \cdots, \sigma_{51} = 0.02.$$

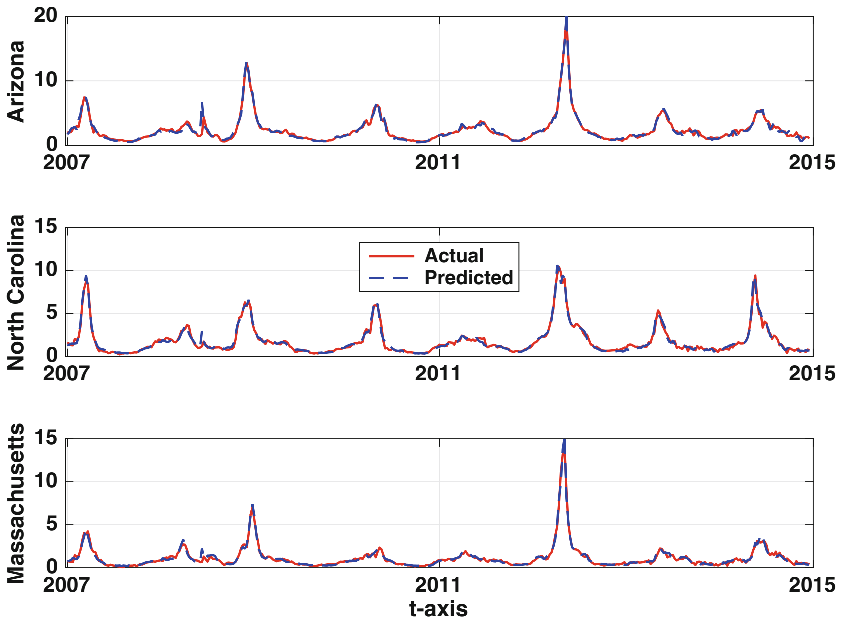


Figure 9.19 The Google flu values, solid (red) curve, and the values predicted using (9.46), dashed (blue) curve, for three different states. The values are per 1000.

Consequently, the terms in the sum (9.47) with singular values that are appreciably larger than one are the principal contributors to $\mathbf{P}$. To explore what can be learned from this, note that at time t_i, from (9.44), the values at location j satisfy

$$y^*_{ij} = p_{j1}x^*_{i2} + p_{j2}x^*_{i2} + \cdots + p_{jm}x^*_{im}\,.$$

Suppose we use only a first term approximation in (9.47), which gives us $\mathbf{P} \approx \sigma_1 \mathbf{W}_1$. Writing the entries of $\mathbf{W}_1$ as w_{ij}, then we have that

$$y^*_{ij} \approx \sigma_1(w_{j1}x^*_{i2} + w_{j2}x^*_{i2} + \cdots + w_{jm}x^*_{im}).$$

Assuming that the x^*_{ik} values in this equation are approximately equal and positive, then this approximation shows that states with a large w_{jk} value are the ones that are principally responsible for the larger increases of flu cases at location j.

As an example, for North Carolina, $j = 34$. When the $w_{34,k}$'s are sorted from largest to smallest one obtains the list

$$0.07,\ 0.055,\ 0.054,\ 0.052,\ 0.047,\ \cdots,\ -0.04,\ -0.12\,.$$

Writing this list in terms of the respective state, the first five are

$$\text{Texas, New Jersey, New Hampshire, Massachusetts, Illinois,}$$

while the last two are Wisconsin and Connecticut. Consequently, if you live in North Carolina and are concerned about the increased prevalence of flu, then according to this analysis you should pay attention to the current flu cases in Texas, and to a lesser extent to those in New Jersey, New Hampshire, etc. Why this happens is not clear. For example, one might expect that the best first-order approximation for how many flu cases to expect in any state would be determined by the number currently in that state, and not the number in an nonadjacent state. Looking for an explanation, the first question to address is whether $\mathbf{P} \approx \sigma_1 \mathbf{W}_1$ is an accurate approximation for NorthCarolina. A comparison is shown in Figure 9.20 for a one year inter-

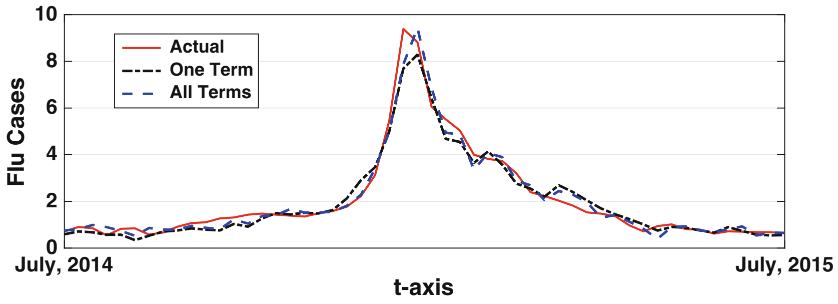


Figure 9.20 The Google flu values, solid (red) curve, for North Carolina over a one year period. Also shown are the values predicted using (9.46), dashed (blue) curve, and the one term approximation $\mathbf{P} \approx \sigma_1 \mathbf{W}_1$. The values are per 1000.

val, and it is apparent that it does provide a reasonable approximation. So, turning to the question of how much the current value of x^* for North Carolina affects the predicted value of y^* for North Carolina, the model predicts there is little dependence. This is because the computed value for $w_{34,34}$ is very close to zero. Pursuing this a bit further, the question is whether there are any states for which the increase in that state is primarily due to itself. Mathematically this corresponds to looking for states for which $w_{j,j}$ is close to the largest value of its $w_{j,k}$ values. This is easy to determine, and the list in decreasing order is

$$\text{Texas, New Jersey, New Hampshire, Massachusetts, Illinois, } \cdots .$$

It can not be a coincidence that the first five states in this list are also the first five states in the list for North Carolina. However, figuring out why this happens is outside the purview of this text, and is left for a topic for future study. What this does nicely demonstrate is that the mathematical analysis has generated several interesting interconnections in the data set that were not evident at the start.

Before leaving this example involving the Google flu values, there has been some criticism of using their method to determine, or estimate, the actual number of flu cases. This does not affect the conclusions or analysis used in the example, but those interested in this should consult Lazer et al. [2014] and Santillana et al. [2014].

9.4.3 Parting Comments

The central assumptions made here concern linearity and time independence, as expressed in (9.44) and (9.46). For the flu example there is a week separating the $\mathbf{x}$ and $\mathbf{y}$ values. During this period the actual number of flu cases is inevitability affected by weather conditions across the county, school schedules, and a host of other time dependent and potentially nonlinear affects. Nevertheless, we found that the assumptions work reasonably well. One could argue that the reason is that (9.46) is basically a two term Taylor series, about the mean, and it should therefore serve as a reasonable approximation so long as the nonlinear affects do not have the ability to strongly affect the number of flu cases over the given time interval. This argument is often given to justify, or explain, why (9.46) is used. For this reason, in the meteorological literature (9.46) is referred to as a tangent linear model. This also explains why if you use a longer time interval, like two weeks, then the error in the approximation increases, by about 30%.

Once the projector matrix is known there is a temptation to use it for longer-term forecasts. For example, knowing the normalized value $\mathbf{x}_i^*$ at t_i, we used it to forecast the value at t_{i+1} using the formula $\mathbf{y}_i^* = \mathbf{P}_i\mathbf{x}_i^*$. One could take this a step further and state that the predicted value at t_{i+2} is $\mathbf{P}(\mathbf{P}\mathbf{x}_i^*) = \mathbf{P}^2\mathbf{x}_i^*$. This can be continued, with the result that after k time steps the predicted value is $\mathbf{P}^k\mathbf{x}_i^*$. It is very unlikely that $\mathbf{P}^k\mathbf{x}_i^*$ is anywhere close to the actual value for $\mathbf{x}_{i+k}^*$ for k values much bigger than one. To illustrate, using the data at the beginning of July, 2014 then this method would predict there will be about 1600 cases of flu in North Carolina at the beginning of January, 2015, which differs significantly from the actual value of 8821 (see Figure 9.20).

The problem considered here falls into the category of predictive, or forecast, modeling. The most well known application of this arises with weather prediction, and considerable research has been invested into obtaining accurate data based weather models. How the pseudo-inverse and the SVD are used in such cases can be found by consulting Schneider and Griffies [1999] and Leutbecher and Palmer [2008]. A method very similar to the one considered here is used in fluid dynamics, and is known as dynamic mode decomposition. One of the objectives in this case is to identify coherent structures in the flow from the data, and more about this can be found in Tissot et al. [2014] and Tu et al. [2014].

9.5 Fitting IVPs to Data

A nonlinear regression problem that arises frequently in applications concerns how to find material coefficients from experiment. It is often the case that the equations governing the system are known. For example, the equations come from Newton's second law or from Maxwell's equations. What is not known are the material parameters for the particular system being studied. The question then arises is, if the response can be measured experimentally how can this information be used to find the parameters?

9.5.1 Logistic Equation

To give an example of this type of problem, it is often assumed that a population $y(t)$ of a species satisfies the logistic equation

$$\frac{dy}{dt} = \alpha y(\beta - y), \quad \text{for } \; 0 < t, \tag{9.48}$$

where $y(0) = a$. Note that although it is possible to solve this problem in closed form, in the discussion to follow this will not be used because in most problems such a solution is not available.

To illustrate, consider the data shown in Figure 9.21. Suppose it is known that the values, which are the average concentrations at something called 4-fold degenerate sites, are described using the logistic equation (9.48). The goal is to determine α and β from the data. We will use least squares to measure the error, and so the error function is

$$E(\alpha, \beta) = \sum_{i=1}^{n} [y(t_i) - y_i]^2. \tag{9.49}$$

To use our minimization methods we need to be able to evaluate this function for given values of α and β. There are various strategies on how this can be done, and we will consider two: a direct method, and an interpolation method.

Direct Method: For this one solves (9.48) numerically, and makes sure that the solver calculates the solution at all of the t_j's in the data set. If there are a large number of points, as is the case in Figure 9.21, this can add considerable computing time to the procedure. Nevertheless, because this type of data fitting is so common, many IVP solvers have the ability to control the time points. For example, in MATLAB this is accomplished using the *tspan* option in `ode45`.

Interpolation Method: The idea here is that the time steps are determined based on the objective of obtaining an accurate solution of the IVP. These

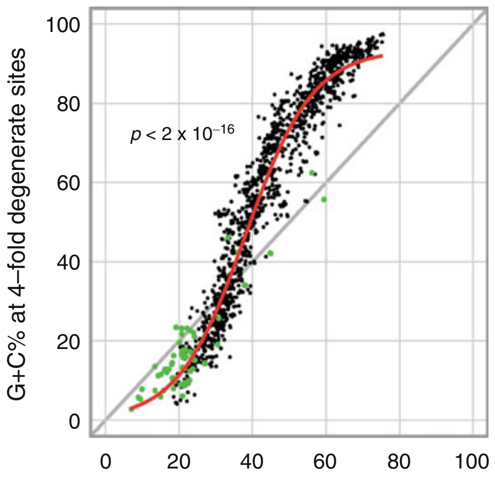


Figure 9.21 Data and curves from Raghavan et al. [2012].

points are then connected using an interpolation function, producing an approximation for $y(t)$ over the entire interval. The procedure consists of the following steps:

1. Solve the IVP using a coarse time step. It is assumed that the time step required for an accurate numerical solution is significantly larger than the distance between the time points in the data set. In the example below, RK4 is used with a time step of 0.2, while the average distance between the data time points is 0.02.

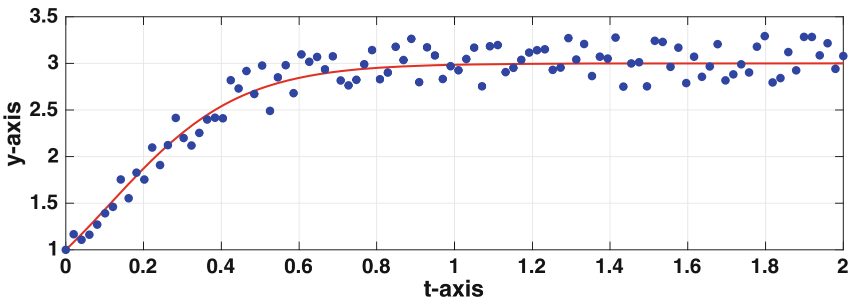


Figure 9.22 The numerical solution of the IVP using the values for the parameters obtained using the Nelder-Mead algorithm, using a least-squares error.

2. With the computed values of the solution of the IVP, use interpolation to produce a function that approximates $y(t)$ over the interval. In the example below, a clamped cubic spline is used. Note that this requires the value of $y'(t)$ at the two endpoints. However, given that we know $y(0)$, and we have just computed y at the right endpoint, we can compute $y'(t)$ at the

endpoints using the differential equation (the merits of this are discussed in Section 7.7).

3. Evaluate the interpolation function at the t_j's in the data set, and from this calculate the error in (9.49).

Whichever method is used to evaluate E, it is also necessary to decide on what method to use to find the minimum. The gradient descent methods described in Chapter 8 are certainly possible, but these require the calculation of the gradient of E. This, in turn, requires the calculation of the derivatives using a numerical method. For example,

$$\frac{\partial E}{\partial \alpha} \approx \frac{E(\alpha + \Delta\alpha, \beta) - E(\alpha, \beta)}{\Delta\alpha} \, .$$

Although this is often done in practice, there is another choice that avoids differentiation and this is the Nelder-Mead method. The latter is used in the examples to follow.

Example

To demonstrate the procedure we will use what is called synthetic data. What this means is that the value of α and β is selected, and then (9.48) is solved numerically. These computed values are then randomized to resemble what is obtained in an experiment. An example of such a data set, which consists of 100 points, is shown in Figure 9.22. The interpolation method is used to fit this data. To do this, RK4 with 11 time steps is used to produce the points needed to determine the clamped cubic spline. Using the Nelder-Mead algorithm one finds that after 38 iteration steps, $\alpha = 1.878$ and $\beta = 3.093$. Using these values the IVP was solved numerically, and the resulting curve is given in Figure 9.22. This is what is obtained using the interpolation method. If you use the direct method, then the computing time for this example increases by a factor of about 5, and the final solution is approximately the same (the parameter values agree to two or three digits). ■

Although there are several steps in the minimization process, the method is straightforward. It is essential to point out that even though it worked very well in the example, the procedure can easily fail. One complication is that it can be unpredictable what values the minimizer might use for α and β when searching for the minimum. For example, the minimizer might try to use unphysical values for the parameters, or values that cause the solution to behave in an unstable manner. In effect, this means that for many problems there are constraints on the minimization process, and one needs to build this into the algorithm. To illustrate, for the logistic equation, it is known that α must be positive. If the minimizer attempts to use a negative value this must be overridden, and an alternative put in its place.

9.5.2 FitzHugh-Nagumo Equations

In computational neuroscience, the FitzHugh-Nagumo equations are used as a model for the potential in the nerve. The equations in this case are

$$v' = c(v - \frac{1}{3}v^3 + w), \tag{9.50}$$

$$w' = -\frac{1}{c}(v - a - bw). \tag{9.51}$$

The numerical solution of this problem in the case of when $v(0) = -1$, $w(0) = 1$, $a = b = 0.2$ and $c = 3$ is shown in Figure 9.23. To mimic a typical data set from experiment, consider the synthetic data shown in Figure 9.24, which were obtained from the randomization of the solution in Figure 9.23. The error function to be used is

$$E(c, b) = \sum_{i=1}^{n} \{[v(t_i) - v_i]^2 + [w(t_i) - w_i]^2\}. \tag{9.52}$$

where (t_i, v_i) and (t_i, w_i) are the data, and $v(t)$ and $w(t)$ are the solution of the FitzHugh-Nagumo equations. Also note that we are going to assume $a = 0.2$ is known, and the objective is to use a fitting method to find the values of c and b. We will use the interpolation method, which means that to evaluate the error function for a particular (c, b), the FitzHugh-Nagumo equations are first solved numerically using a coarse time step (coarse in comparison to the step-size observed in the data). In this particular example, the RK4 method is used with $k \approx 0.25$. This solution is then used to construct a clamped cubic spline interpolation function that can be evaluated to obtain values for v and w at the various t_i's appearing in the error function. Finally, the Nelder-Mead algorithm is used to find the minimum. The fitting procedure takes 77 iteration steps and concludes that $c = 2.97$ and $b = 0.227$. The resulting solution cures, and the data, are shown in Figure 9.25.

9.5.3 Mass-Spring-Dashpot System

As another illustration, consider the equation for a damped oscillator

$$m\frac{d^2y}{dt^2} + c\frac{dy}{dt} + ky = 0, \quad \text{for} \ \ 0 < t, \tag{9.53}$$

where $y(0) = a$ and $y'(0) = b$. The material parameters are m, the mass, c, the damping coefficient, and k, the spring constant. There are a couple of reasons for considering this particular example. One is how often this equation arises in applications. A second is that there is a complication concerning which parameters can be determined from fitting the solution to a data set, and this is explained below.

A typical data set is shown in Figure 9.26. It is not possible from this data to determine the three material parameters, and the reason is uniqueness. Because the right hand side of the equation is zero, it is possible to multiply it by any nonzero constant and not affect the answer. This means the solution obtained using the values m_0, c_0, and k_0 is exactly the same as the solution using $2m_0$, $2c_0$, and $2k_0$. To avoid this problem we will rewrite the equation as

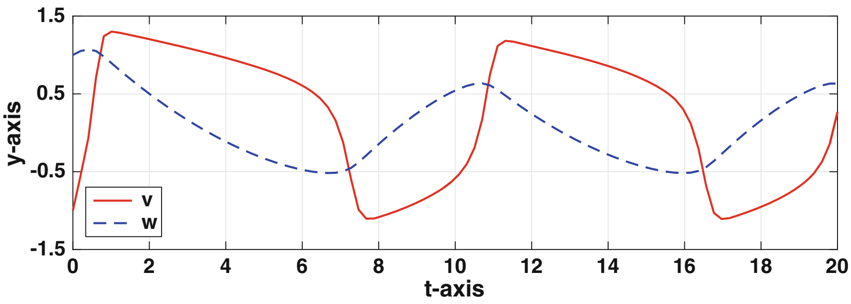


Figure 9.23 The numerical solution of the FitzHugh-Nagumo equations in (9.50), (9.51).

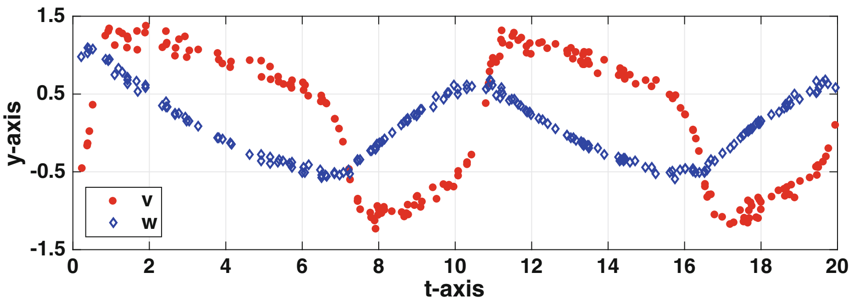


Figure 9.24 Synthetic data obtained from the solution of the FitzHugh-Nagumo equations in (9.50), (9.51).

$$\frac{d^2y}{dt^2} + \alpha\frac{dy}{dt} + \beta y = 0, \tag{9.54}$$

where $\alpha = c/m$ and $\beta = k/m$. As one check on whether this works, note that the same values of α and β are obtained if one uses m_0, c_0, and k_0 or uses $2m_0$, $2c_0$, and $2k_0$. The theory related to parameters, and the scaling of a problem, is an important component of analyzing a problem and a more complete discussion can be found in Holmes [2009]. The final step is to write the equation as a first-order system be letting $v = y'$. The result is

$$y' = v, \tag{9.55}$$

$$v' = -\alpha v - \beta y. \tag{9.56}$$

The data set for this example is shown in Figure 9.26, and as was the case for the two earlier examples, this is synthetic data derived from the exact solution. The error function is

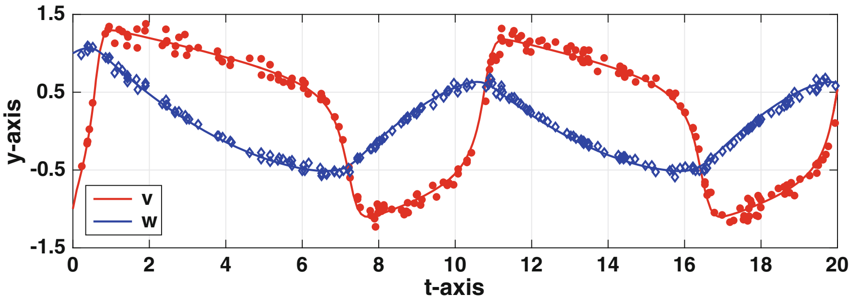


Figure 9.25 Synthetic data and corresponding numerical solution of the FitzHugh-Nagumo equations in (9.50), (9.51).

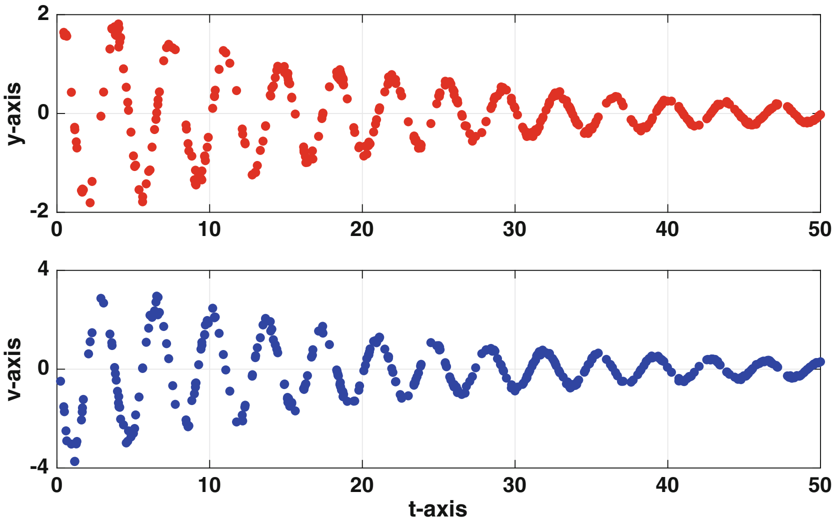


Figure 9.26 Synthetic data obtained from the solution of the mass-spring-dashpot equations in (9.55), (9.56).

$$E(\alpha, \beta) = \sum_{i=1}^{n} \{[y(t_i) - v_i]^2 + [v(t_i) - w_i]^2\}. \tag{9.57}$$

where (t_i, y_i) and (t_i, v_i) are the data, and $y(t)$ and $v(t) = y'(t)$ are the solution of (9.55), (9.56).

Using the Nelder-Mead algorithm one finds that after 58 iteration steps, $\alpha = 0.092$ and $\beta = 3.025$. Note that the stopping condition used in the calculation was that the area of the approximating triangle was less than 10^{-14}. Also, because of the wavelength of the oscillations in the solution, 100 time points were used when solving the IVP for constructing the cubic spline (versus 80 used for the FitzHugh-Nagumo example). The resulting comparison between the numerical solution of the IVP and the data is given in Figure 9.27.

9.5.4 Parting Comments

We used the blunt approach to find the coefficients of the IVPs, which means we just coded everything and then computed the answer. A better way would be to first analyze the problem, determine its basic mathematical properties, and then use a computational approach if necessary. As an example, for the logistic equation (9.48) a simple analysis shows that $y = \beta$ is a steady

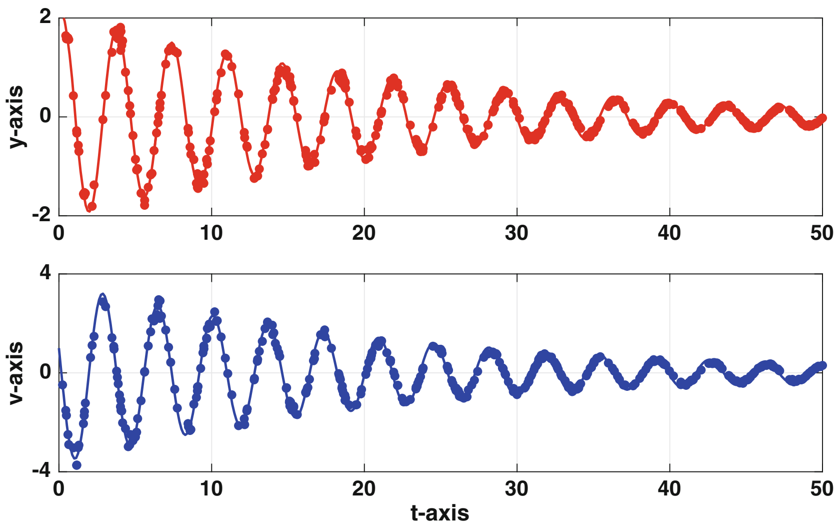


Figure 9.27 Data and numerical solution of the (9.55), (9.56) obtained using parameter values found using the fitting procedure.

state which is asymptotically stable if the parameters are positive. What this mean is that it is possible to just to look at the data in Figures 9.21 and 9.22 and produce a reasonably accurate value for β (it is the value the solution approaches as $t \to \infty$). One can also get an estimate of α by noting that at $t = 0$, $y'(0) = \alpha a(\beta - a)$. From the estimate of β, and using the approximation $y'(0) \approx (y(t_1) - y(0))/t_1 = (y_1 - a)/t_1$, one can obtain an estimate for α. This information can then be used to produce reasonably good guesses for the minimization algorithm.

Numerous methods have been proposed for finding material parameters from data. For example, Varah [1982] uses splines but fits them to the data, and then uses the spline to find the parameters. This is known as data smoothing, and more recent work on this can be found in Ramsay et al. [2007]. Other possibilities are discussed in Brewer et al. [2008] and Xue et al. [2010].

Exercises

9.1. This problem applies a PCA to the data given in Table 9.7.
(a) Find the normalized data set using the ∞-norm normalization (which is the same one used for the word length example).
(b) Find the SVD for the normalized data matrix from part (a).
(c) Suppose $x = 2.5$. What do you predict the values are for y and z?
(d) Suppose $x = 2.5$ and $y = 0.5$. What do you predict the value is for z?

x	y	z
2	1	2
1	−1	1
−1	1	−1
−2	−1	−2

Table 9.7 Data for Exercise 9.1

9.2. An often used model function in nonlinear regression is

$$g(x) = v_1 x^{v_2}.$$

As shown in Exercise 8.6, by using the log, the model function can be written as $G(X) = V_1 + V_2 X$, where $V_1 = \log v_1$ and $V_2 = v_2$. Also, a data point (x_i, y_i) transforms to (X_i, Y_i), where $X_i = \log x_i$ and $Y_i = \log y_i$. The data considered in this exercise is given in Table 9.8, which was also considered in Exercise 8.6.

(a) Apply the same normalization to the transformed (X_i, Y_i) data as was used in the word length example. This means you center the X and Y values, and then scale the values using the maximum entry, in absolute value. Label the normalized variables as p and q. Also, make sure to give the values of $\overline{X}$ and $\overline{Y}$, as well as the values used to scale the centered values.

(b) Assuming $p = \alpha q$, and using the true distance, then the error function is

$$E(\alpha) = \frac{1}{1+\alpha^2}\sum_{i=1}^{n}(\alpha p_i - q_i)^2.$$

The minimum of E occurs when α is given in (9.10). Calculate α using your normalized data from part (a).

(c) In this problem, instead of (9.11), we have $Y = \overline{Y} + m(X - \overline{X})$. Using the common log, show that $v_2 = m$ and

$$v_1 = 10^{\overline{Y} - m\overline{X}}.$$

(d) Using part (c), plot the (original) data and power law curve using a log-log plot.

(e) Based on your result from part (c), what was the running speed of a Tyrannosaurus rex?

9.3. This problem considers the process of whitening a data set with two sources. Let $\mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_n$ be the source data vectors, with $\mathbf{s}_i = (s_{i1}, s_{i2})^T$.

(a) Suppose it's possible to find an invertible 2×2 matrix $\mathbf{Z}$ with the property that

Animal	Mass (kg)	Relative Speed (1/s)
canyon mouse	1.37e−02	39.1
chipmunk	5.10e−02	42.9
red squirrel	2.20e−01	20.5
snowshoe hare	1.50e+00	35.8
red fox	4.80e+00	28.7
human	7.00e+01	7.9
reindeer	1.00e+02	12.7
wildebeest	3.00e+02	11.0
giraffe	1.08e+03	3.8
rhinoceros	2.00e+03	1.8
elephant	6.00e+03	1.4

Table 9.8 Data for Exercise 9.2 adapted from Iriarte-Díaz [2002].

i)

$\mathbf{s}_1$	1	3
$\mathbf{s}_2$	1	−1

ii)

$\mathbf{s}_1$	0	2
$\mathbf{s}_2$	−2	1

Table 9.9 Data for Exercise 9.3(b).

$$\mathbf{Z}\mathbf{Z}^T = \frac{1}{n}\sum_{i=1}^{n} \mathbf{s}_i(\mathbf{s}_i)^T .$$

Setting $\mathbf{s}_i^* = \mathbf{Z}^{-1}\mathbf{s}_i$, show that the $\mathbf{s}_i^*$'s satisfy (9.30). This shows that any matrix $\mathbf{Z}$ with the above property can be used to whiten the data set.

(b) Since $\frac{1}{n}\sum \mathbf{s}_i(\mathbf{s}_i)^T$ is symmetric, the SVD of this matrix can be written as

$$\frac{1}{n}\sum_{i=1}^{n} \mathbf{s}_i(\mathbf{s}_i)^T = \mathbf{U}\boldsymbol{\Sigma}\mathbf{U}^T .$$

Setting $\mathbf{s}_i^* = \boldsymbol{\Sigma}^{-1/2}\mathbf{U}^T\mathbf{s}_i$, show that the $\mathbf{s}_i^*$'s satisfy (9.30).

(c) Whiten one of the data sets in Table 9.9. Note that you should first center the data, so (9.29) is satisfied). Also, if you use the method from part (a), $\mathbf{Z}$ is not unique, and you should make a particular choice for this matrix.

9.4. Shown in Table 9.10 are the pixel values for two images (both images contain 4 pixels). Suppose that the images are mixed as follows: $0.4\mathbf{S}_1 + 0.6\mathbf{S}_2$ and $0.6\mathbf{S}_1 + 0.4\mathbf{S}_2$. Apply, by hand, the ICA to find the unmixed images, and compare your values with the originals.

9.5. In using the PCA, suppose that the data set contains two variables, and the normalized versions are labeled as p_1 and p_2. Also, after calculating the SVD, one obtains the matrix

$$\mathbf{V} = \begin{pmatrix} v_{11} & v_{12} \\ v_{21} & v_{22} \end{pmatrix} .$$

In this problem the PCA assumption is that $\mathbf{p} = \alpha_1\mathbf{v}_1$.

(a) Assuming $v_{11} \neq 0$, show that the assumption $\mathbf{p} = \alpha_1\mathbf{v}_1$ can be written as $p_2 = \alpha p_1$, where $\alpha = v_{21}/v_{11}$.

(b) Letting q_1 and q_2 be the original, unnormalized, data variables, show that the fitted line can be written as

$$q_2 = M_2 + a\,(q_1 - M_1),$$

where M_1 and M_2 are the means of q_1 and q_2, respectively. Also, $a = \alpha S_2/S_1$, where S_1 and S_2 are the scale factors used for the respective variables.

$\mathbf{S}_1$:

2	1
0	1

$\mathbf{S}_2$:

0	1
1	2

Table 9.10 Data for Exercise 9.4.

9.6. Suppose that the data set contains three variables, and the normalized versions are labeled as p_1, p_2, and p_3. Also, assume that the PCA assumption is $\mathbf{p} = \alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2$, where $\mathbf{v}_1 = (v_{11}, v_{21}, v_{31})^T$ and $\mathbf{v}_2 = (v_{12}, v_{22}, v_{32})^T$ are the first two columns in the matrix $\mathbf{V}$.

(a) Show that the PCA assumption can be written as $p_3 = \alpha p_1 + \beta p_2$, where $\alpha = (v_{31}v_{22} - v_{32}v_{12})/D$, $\beta = (v_{32}v_{11} - v_{31}v_{21})/D$, and $D = v_{11}v_{22} - v_{12}v_{21}$. This assumes, of course, that $D \neq 0$.

(b) Using the result from part (a), show that the PCA assumption, when expressed in terms of the original, unnormalized, data variables, can be written as

$$q_3 = M_3 + a\,(q_1 - M_1) + b\,(q_2 - M_2),$$

where M_1, M_2 and M_3 are the means of q_1, q_2 and q_3, respectively. Also, $a = \alpha S_3/S_1$ and $b = \beta S_3/S_2$, where S_1, S_2, and S_3 are the scale factors used for the respective variables.

9.7. The ICA method requires that the eigenvalues of $\frac{1}{n}\sum \mathbf{x}_i^*(\mathbf{x}_i^*)^T$ are positive, and this problem determines when this holds directly from the summation formula. It is assumed that $\mathbf{x}_i^* = (x_{i1}^*, x_{i2}^*)^T$ and at least one of the $\mathbf{x}_i^*$'s is nonzero.

(a) Setting

$$\mathbf{B} = \begin{pmatrix} a & b \\ b & d \end{pmatrix},$$

where a and d are positive, show that $\mathbf{B}$ has positive eigenvalues so long as $b^2 < ad$.

(b) Writing $\frac{1}{n}\sum \mathbf{x}_i^*(\mathbf{x}_i^*)^T = \mathbf{B}$, what are a, b, and d in terms of the x_{ij}^*'s?

(c) Using the connection between the dot product and the angle between two vectors, explain why the eigenvalues of $\frac{1}{n}\sum \mathbf{x}_i^*(\mathbf{x}_i^*)^T$ are positive so long as $\mathbf{c}_1^* = (x_{11}^*, x_{21}^*, \cdots, x_{n1}^*)^T$ and $\mathbf{c}_2^* = (x_{12}^*, x_{22}^*, \cdots, x_{n2}^*)^T$ are not multiples of each other.

9.8. Consider the problem of computing the covariance matrix

$$\mathbf{B} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i(\mathbf{x}_i)^T,$$

where $\mathbf{x}_i = (x_{i1}, x_{i2})^T$. The question addressed is, what is the fastest way to compute this matrix. In doing this, the $\mathbf{x}_i$'s should be randomly generated, and this should be done before answering the following questions. Also, you should answer the questions taking $n = 2{,}000$, $n = 20{,}000$, and $n = 200{,}000$. Finally, you should take advantage of any mathematical simplifications available (e.g., symmetry), to reduce the computational time.

(a) How long does it take to compute $\mathbf{B}$ using the summation formula? This means that each 2×2 matrix $\mathbf{x}_i(\mathbf{x}_i)^T$ is computed, and then added into the sum.

(b) How long does it take to compute $\mathbf{B}$ by first forming the matrix $\mathbf{X}$, and then using (9.33)? You should use MATLAB's matrix multiply command to compute the product.

(c) How long does it take to compute $\mathbf{B}$ by first forming the vectors $\mathbf{c}_1^*$ and $\mathbf{c}_2^*$, and then using (9.34)? You should use MATLAB's dot multiply command to compute the dot products.

(d) How long does it take to compute $\mathbf{B}$ by first forming the matrix $\mathbf{X}$, and then using MATLAB's `cov` command?

(e) Explain the differences, or non-differences, in the computing times. Also, comment if any of the methods failed, and why that happened.

9.9. This exercise explores modifications of the example used to introduce an ICA. The sources used to generate the curves in Figure 9.8 are $s_1(t) = \sin(\pi t)$ and $s_2(t) = \sin(1.7\pi t - 5)$, and the recorded signals for Figure 9.9 are $x_1(t) = 2s_1(t) + s_1(t)$ and $x_2(t) = 3s_1(t) - 2s_1(t)$. Also, 200 points were used over the interval $0 \leq t \leq 10$.

(a) The area in Figure 9.12 comes from using the composite trapezoidal rule to compute

$$\int_0^{10} |s_1 - s_1^*| dt + \int_0^{10} |s_2 - s_2^*| dt,$$

where s_1^* and s_2^* are the sources computed using (9.37). Note that the latter depend on the angle θ used to evaluate $\mathbf{V}$, but the original sources do not depend on θ. Plot a curve similar to the one in Figure 9.12, but use

$$\int_0^{10} |s_1 - s_1^*|^2 dt + \int_0^{10} |s_2 - s_2^*|^2 dt.$$

Does the value of the optimal angle change much using this error function? Note that because the ICA can not determine which source is s_1 or s_2, you will need to create two plots. One with s_1^* and s_2^* as above, and a second with s_1^* and s_2^* interchanged in the formula. Also, the above formula is associated with the 2-norm, and is also the error measure associated with least squares.

(b) The kurtosis plotted in Figure 9.13 comes from evaluating (9.41). Suppose instead one uses the 2-norm, and takes $K(\theta) = \kappa_1^2 + \kappa_2^2$. Plot this function and determine the locations of the local maxima. One of these, presumably, is close to the value obtained in part (a). Compare it with the value in part (a), as well as the solution obtained in the text.

The exercises to follow require synthetic data, which is either provided or you generate. For the latter, suppose that by using either the exact solution, or a numerical solution, you determine values y_0, y_1, $\cdots$, y_n of the solution at the time points t_0, t_1, $\cdots$, t_n. By appropriately modifying the MATLAB commands: `q=A*(2*rand-1); yd=(1+q)*y;` you can produce synthetic data. In doing this, the synthetic data values will satisfy $(1-A)y \le yd \le (1+A)y$, which means you should pick A so that $0 < A < 1$. Also, a different random number must be used at each time point.

9.10. This problem concerns the Bernoulli equation

$$y' + y^3 = \frac{y}{a+t}\,, \quad \text{for } \ t > 0,$$

where $y(0) = 1$. The exact solution is

$$y = \frac{a+t}{\sqrt{\beta + \frac{2}{3}(a+t)^3}}\,,$$

where $\beta = a^2(1 - 2a/3)$.

(a) Taking $a = 0.01$, generate synthetic data for this problem using 400 equally spaced time points over the interval $0 < t \le 2$.

(b) Using either the data from part (a), or a data set provided, find a. Also, plot the data and numerical solution using these parameter values.

9.11. In the description of the dynamics of excitons in semiconductor physics, one finds the following problem

$$\begin{aligned} n' &= \gamma - \alpha n^2 x, \\ x' &= \alpha n^2 x - \frac{x}{1+x}\,, \end{aligned}$$

where $n(0) = 1$ and $x(0) = 1$. Also, α and γ are constants that satisfy $0 < \gamma < 1$ and $\alpha > 0$. Note, $x(t)$ and $n(t)$ are densities and are therefore non-negative.

(a) Taking $\alpha = 0.1$ and $\gamma = 0.2$, generate synthetic data for n and x using 500 equally spaced time points over the interval $0 < t \le 200$.

(b) Using either the data from part (a), or a data set provided, find α and γ. Also, plot the data and numerical solution using these parameter values. If you modify the minimization code make sure to explain why you did so, and what exactly you did change.

Appendix A
Taylor's Theorem

The essential tool in the development of numerical methods is Taylor's theorem. The reason is simple, Taylor's theorem will enable us to approximate a function with a polynomial, and polynomials are easy to compute (most of the time). To start, we define what it means for a function to be C^n.

Definition A.1. Given a non-negative integer n, and an interval $a < x < b$, stating that $f \in C^n(a,b)$ means that $f(x)$, $f'(x)$, $f''(x)$, $\cdots$, $f^{(n)}(x)$ exist and are continuous functions on the interval $a < x < b$.

Note that this definition does not follow the usual convention for exponents. In particular, $f \in C(a,b)$ and $f \in C^0(a,b)$ are the same statement, which are both different than stating that $f \in C^1(a,b)$. If $f \in C(a,b)$, or equivalently if $f \in C^0(a,b)$, then the function is continuous on the interval. In contrast, $f \in C^1(a,b)$ means that $f(x)$ and $f'(x)$ are continuous on the interval. Also, to state that $f \in C^\infty(a,b)$ means $f(x)$ and all of its derivatives are defined and continuous for $a < x < b$.

We now state Taylor's theorem.

Theorem A.1. *Given a function $f(x)$, assume that $f \in C^{n+1}(x_L, x_R)$. In this case, if x and $x+h$ are points in the interval (x_L, x_R), then*

$$f(x+h) = f(x) + hf'(x) + \frac{1}{2}h^2 f''(x) + \cdots + \frac{1}{n!}h^n f^{(n)}(x) + R_{n+1}, \tag{A.1}$$

where the remainder is

$$R_{n+1} = \frac{1}{(n+1)!}h^{n+1} f^{(n+1)}(\eta), \tag{A.2}$$

and η is a point between x and $x+h$.

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

The result in (A.1) is known as Taylor's theorem with remainder. The mystery point η in (A.2) is not known other than it is somewhere in the given interval.

Writing out the first few cases we have that

$$\begin{aligned}
f(x+h) &= f(x) + hf'(\eta), \\
f(x+h) &= f(x) + hf'(x) + \frac{1}{2}h^2 f''(\eta), \\
f(x+h) &= f(x) + hf'(x) + \frac{1}{2}h^2 f''(x) + \frac{1}{6}h^3 f'''(\eta).
\end{aligned}$$

The η's in these formulas are not the same. Usually the exact value of η is not important because the remainder term is dropped when using Taylor's theorem to derive an approximation of a function. Doing this, the above expressions become

$$f(x+h) \approx f(x), \tag{A.3}$$

$$f(x+h) \approx f(x) + hf'(x), \tag{A.4}$$

$$f(x+h) \approx f(x) + hf'(x) + \frac{1}{2}h^2 f''(x). \tag{A.5}$$

As a function of h, (A.3) is a constant approximation, (A.4) is a linear approximation, and (A.5) is a quadratic approximation.

There are various ways to write a Taylor expansion. One is as stated in the above theorem, which is

$$f(x+h) = f(x) + hf'(x) + \frac{1}{2}h^2 f''(x) + \cdots + \frac{1}{n!}h^n f^{(n)}(x) + \cdots .$$

The assumption here is that h is close to zero. Another way to write the expansion is as

$$f(x) = f(a) + (x-a)f'(a) + \frac{1}{2}(x-a)^2 f''(a) + \cdots + \frac{1}{n!}(x-a)^n f^{(n)}(a) + \cdots .$$

In this case it is assumed that x is close to a. This gives rise to the linear approximation

$$f(x) \approx f(a) + (x-a)f'(a), \tag{A.6}$$

the quadratic approximation

$$f(x) \approx f(a) + (x-a)f'(a) + \frac{1}{2}(x-a)^2 f''(a), \tag{A.7}$$

and the cubic approximation

$$f(x) \approx f(a) + (x-a)f'(a) + \frac{1}{2}(x-a)^2 f''(a) + \frac{1}{6}(x-a)^3 f'''(a). \tag{A.8}$$

It's certainly possible to write down higher-order approximations, but they are not needed in this text.

A.1 Useful Taylor Series for x Near Zero

$$f(x) = f(0) + xf'(0) + \frac{1}{2}x^2 f''(0) + \frac{1}{6}x^3 f'''(0) + \cdots$$

Power Functions

$$\begin{aligned}
(a+x)^\gamma &= a^\gamma + \gamma x a^{\gamma-1} + \frac{1}{2}\gamma(\gamma-1)x^2 a^{\gamma-2} + \frac{1}{6}\gamma(\gamma-1)(\gamma-2)x^3 a^{\gamma-3} + \cdots \\
\frac{1}{1-x} &= 1 + x + x^2 + x^3 + \cdots \\
\frac{1}{(1-x)^2} &= 1 + 2x + 3x^2 + 4x^3 + \cdots \\
\sqrt{1+x} &= 1 + \frac{1}{2}x - \frac{1}{8}x^2 + \frac{1}{16}x^3 + \cdots \\
\frac{1}{\sqrt{1-x}} &= 1 + \frac{1}{2}x + \frac{3}{8}x^2 + \frac{5}{16}x^3 + \cdots
\end{aligned}$$

Trig Functions

$$\begin{aligned}
\sin(x) &= x - \frac{1}{3!}x^3 + \frac{1}{5!}x^5 + \cdots \\
\arcsin(x) &= x + \frac{1}{6}x^3 + \frac{3}{40}x^5 + \cdots \\
\cos(x) &= 1 - \frac{1}{2}x^2 + \frac{1}{4!}x^4 + \cdots \\
\arccos(x) &= \frac{\pi}{2} - x - \frac{1}{6}x^3 - \frac{3}{40}x^5 + \cdots \\
\tan(x) &= x + \frac{1}{3}x^3 + \frac{2}{15}x^5 + \cdots \\
\arctan(x) &= x - \frac{1}{3}x^3 + \frac{1}{40}x^5 + \cdots \\
\cot(x) &= \frac{1}{x} - \frac{1}{3}x - \frac{1}{45}x^3 + \cdots \\
\operatorname{arccot}(x) &= \frac{\pi}{2} - x + \frac{1}{3}x^3 - \frac{1}{5}x^5 + \cdots \\
\sin(a+x) &= \sin(a) + x\cos(a) - \frac{1}{2}x^2\sin(a) + \cdots \\
\cos(a+x) &= \cos(a) - x\sin(a) - \frac{1}{2}x^2\cos(a) + \cdots \\
\tan(a+x) &= \tan(a) + x\sec^2(a) + x^2\tan(a)\sec^2(a) + \cdots
\end{aligned}$$

Exponential and Log Functions

$$\begin{aligned}
e^x &= 1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \cdots \\
a^x &= e^{x\ln(a)} = 1 + x\ln(a) + \frac{1}{2}[x\ln(a)]^2 + \frac{1}{6}[x\ln(a)]^3 + \cdots \\
\ln(a+x) &= \ln(a) + \frac{x}{a} - \frac{1}{2}\left(\frac{x}{a}\right)^2 + \frac{1}{3}\left(\frac{x}{a}\right)^3 + \cdots
\end{aligned}$$

Hyperbolic Functions

$$\begin{aligned}
\sinh(x) &= x + \frac{1}{6}x^3 + \frac{1}{120}x^5 + \cdots \\
\text{arcsinh}(x) &= x - \frac{1}{6}x^3 + \frac{3}{40}x^5 + \cdots \\
\cosh(x) &= 1 + \frac{1}{2}x^2 + \frac{1}{24}x^4 + \cdots \\
\text{arccosh}(x) &= \sqrt{2x}\left(1 - \frac{1}{12}x + \frac{3}{160}x^2 + \cdots\right) \\
\tanh(x) &= x - \frac{1}{3}x^3 + \frac{2}{15}x^5 + \cdots \\
\text{arctanh}(x) &= x + \frac{1}{3}x^3 + \frac{1}{5}x^5 + \cdots
\end{aligned}$$

A.2 Order Symbol and Truncation Error

As a typical example of how we will use Taylor's theorem, for h close to zero

$$\sin(h) = h - \frac{1}{3!}h^3 + \frac{1}{5!}h^5 - \frac{1}{7!}h^7 + \cdots .$$

From this we have the approximations

$$\sin(h) \approx h,$$

and

$$\sin(h) \approx h - \frac{1}{3!}h^3.$$

It is useful to have a way to indicate how the next term in the series depends on h. The big-O notation is used for this, and we write

$$\sin(h) = h + O(h^3), \tag{A.9}$$

and

$$\sin(h) = h - \frac{1}{3!}h^3 + O(h^5). \tag{A.10}$$

In this text, the part of the series that is dropped when deriving an approximation is often designated as τ. Given where it comes from, τ is referred to as the *truncation error*. Using the above example, we will sometimes write (A.9) as

$$\sin(h) = h + \tau,$$

where $\tau = O(h^3)$. Similarly, (A.10) can be written as

$$\sin(h) = h - \frac{1}{3!}h^3 + \tau,$$

where $\tau = O(h^5)$.

The definition for big-O is given below. There are more general definitions, but they are not needed here.

Definition A.2. For h close to zero, $\tau = O(h^n)$ means that

$$\lim_{h\to 0} \frac{\tau}{h^n} = L,$$

where $-\infty < L < \infty$.

We will occasionally need to know how big-O terms combine. The rules that cover many of the situations we will come across are the following:

Lemma:

1) If $n < m$, then $O(h^n) + O(h^m) = O(h^n)$.
2) For any nonzero constant α, $O(\alpha h^n) = \alpha O(h^n) = O(h^n)$.

The proof of these statements comes directly from the definition. As an example of how they are used, if $f(h) = 1+2h+O(h^3)$ and $g(h) = -4+3h+O(h^4)$ then

$$\begin{aligned} f + 2g &= 1 + 2h + O(h^3) + 2[-4 + 3h + O(h^4)] \\ &= -7 + 8h + O(h^3) \end{aligned}$$

For the same reason,

$$-2f + 6g = -26 + 14h + O(h^3).$$

The last topic concerns two ways that the truncation error can be written. These come from writing the Taylor series using the remainder term, or else writing it out as a series. For example, one can write the series version

$$\sin(h) = h - \frac{1}{3!}h^3 + \frac{1}{5!}h^5 - \frac{1}{7!}h^7 + \cdots ,$$

as

$$\sin(h) = h + \tau,$$

where

$$\tau = -\frac{1}{3!}h^3 + \frac{1}{5!}h^5 - \frac{1}{7!}h^7 + \cdots . \tag{A.11}$$

In contrast, the remainder form, coming from (A.1) and (A.2), is

$$\sin(h) = h + \tau,$$

where

$$\tau = -\frac{1}{3!}h^3 \cos(\eta). \tag{A.12}$$

In the text, for both cases, the error term is written as $\tau = O(h^3)$. For the series version in (A.11) this should be interpreted as an asymptotic form of the error. What this means is that as h approaches zero, the first term approximation of τ has the stated dependence on h. More explanation about asymptotic forms of an approximation can be found in Holmes [2013].

Appendix B
B-Splines

B.1 Definition

Compact Version:

$$B_i(x) = Q\left(\frac{1}{h}(x - x_i)\right)$$

where

$$Q(x) = \begin{cases} 0 & \text{if} \quad |x| \geq 2, \\ \frac{1}{6}(2 - |x|)^3 & \text{if} \quad 1 \leq |x| \leq 2, \\ \frac{2}{3} - x^2\left(1 - \frac{1}{2}|x|\right) & \text{if} \quad 0 \leq |x| \leq 1. \end{cases}$$

Expanded Version:

$$B_i(x) = \begin{cases} 0 & \text{if} \quad x \leq x_{i-2}, \\ q_{i-2}(x) & \text{if} \quad x_{i-2} \leq x \leq x_{i-1}, \\ q_{i-1}(x) & \text{if} \quad x_{i-1} \leq x \leq x_i, \\ q_{i+1}(x) & \text{if} \quad x_i \leq x \leq x_{i+1}, \\ q_{i+2}(x) & \text{if} \quad x_{i+1} \leq x \leq x_{i+2}, \\ 0 & \text{if} \quad x_{i+2} \leq x, \end{cases}$$

where

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

$$\begin{aligned}
q_{i-2}(x) &= \frac{1}{6h^3}(x-x_{i-2})^3,\\
q_{i-1}(x) &= \frac{1}{6}+\frac{1}{2h}(x-x_{i-1})+\frac{1}{2h^2}(x-x_{i-1})^2-\frac{1}{2h^3}(x-x_{i-1})^3,\\
q_{+1}(x) &= \frac{1}{6}-\frac{1}{2h}(x-x_{i+1})+\frac{1}{2h^2}(x-x_{i+1})^2+\frac{1}{2h^3}(x-x_{i+1})^3,\\
q_{i+2}(x) &= -\frac{1}{6h^3}(x-x_{i+2})^3.
\end{aligned}$$

B.2 Plot

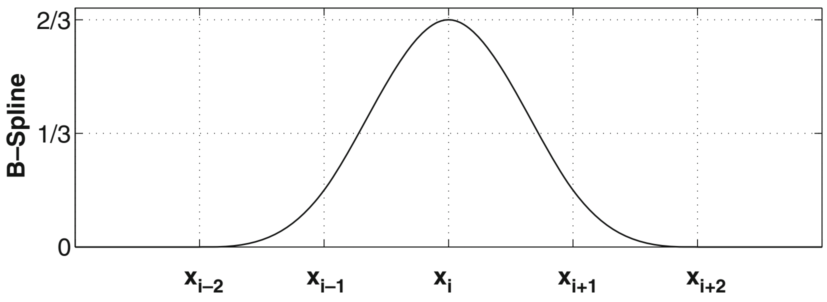


Figure B.1 Plot of the cubic B-spline $B_i(x)$.

B.3 Particular Values

B_i: $B_i(x_{i-1}) = \frac{1}{6}\,,\quad B_i(x_i) = \frac{2}{3}\,,\quad B_i(x_{i+1}) = \frac{1}{6}\,,$
$B_i(x_j) = 0$ for $j \neq i-1, i, i+1$

B_i': $B_i'(x_{i-1}) = \frac{1}{2h}\,,\quad B_i'(x_i) = 0\,,\quad B_i'(x_{i+1}) = -\frac{1}{2h}\,,$
$B_i'(x_j) = 0$ for $j \neq i-1, i, i+1$

B_i'': $B_i''(x_{i-1}) = \frac{1}{h^2}\,,\quad B_i''(x_i) = -\frac{2}{h^2}\,, B_i''(x_{i+1}) = \frac{1}{h^2}\,,$
$B_i''(x_j) = 0$ for $j \neq i-1, i, i+1$

B.4 Derivatives

$$B_i'(x) = \frac{1}{h} Q'\left(\frac{1}{h}(x - x_i)\right)$$

where

$$Q'(x) = \begin{cases} 0 & \text{if} \quad |x| \geq 2, \\ -\frac{s}{2}(2 - |x|) & \text{if} \quad 1 \leq |x| \leq 2, \\ -2x\left(1 - \frac{1}{2}|x|\right) + sx^2 & \text{if} \quad 0 \leq |x| \leq 1, \end{cases}$$

and $s = \text{sgn}(x)$.

B.5 Integrals

$$\begin{aligned} \int_{x_{i-2}}^{x_{i-1}} B_i(x)dx &= \int_{x_{i+1}}^{x_{i+2}} B_i(x)dx = \frac{1}{24}h \\ \int_{x_{i-1}}^{x_i} B_i(x)dx &= \int_{x_i}^{x_{i+1}} B_i(x)dx = \frac{11}{24}h \\ \int_{x_{i-2}}^{x_i} B_i(x)dx &= \int_{x_i}^{x_{i+2}} B_i(x)dx = \frac{1}{2}h \\ \int_{-\infty}^{\infty} B_i(x)dx &= \int_{x_{i-2}}^{x_{i+2}} B_i(x)dx = h \end{aligned}$$

Appendix C
Summary Tables

In several of the chapters, multiple methods were derived that do basically the same thing. The tables given here are a non-exhaustive listing for these methods and some of their basic properties (Tables C.1, C.2, C.3, and C.4). Not all of the notation is explained, and for that you need to consult the respective chapter.

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

	General Form	**Subinterval** $x_i \le x \le x_{i+1}$	**Comments**	**Interpolation Error** $a \le x \le b$
Lagrange	$p_n(x) = \sum_{i=1}^{n+1} y_i \ell_i(x)$ $\ell_i(x) = \prod_{\substack{j=1 \\ j \ne i}}^{n+1} \frac{x - x_j}{x_i - x_j}$		$p_n \in C^\infty[a,b]$ Can have problems with large data sets with equally spaced points.	$E \le \frac{\lVert f^{(n+1)} \rVert_\infty}{4(n+1)} h^{n+1}$
Piecewise Linear	$g(x) = \sum_{i=1}^{n+1} y_i G_i(x)$	$g_i(x) = y_i + \frac{y_{i+1} - y_i}{x_{i+1} - x_i}(x - x_i)$	$g \in C[a,b]$ Quickly calculated but has corners.	$E \le \frac{1}{8} h^2 \lVert f'' \rVert_\infty$
Cubic Spline	$s(x) = \sum_{i=0}^{n+2} a_i B_i(x)$	$s_i(x) = a_i + b_i(x - x_i)$ $+ c_i(x - x_i)^2 + d_i(x - x_i)^3$	$s \in C^2[a,b]$ Requires solution of tridiagonal matrix equation.	Clamped $E \le \frac{5}{384} h^4 \lVert f'''' \rVert_\infty$
Chebyshev	$p_n(x) = \sum_{i=1}^{n+1} y_i \ell_i(x)$	$x_i = \frac{1}{2}[a + b + (b - a)z_i]$ $z_i = \cos\left(\frac{2i - 1}{2n}\pi\right)$	$p_n \in C^\infty[a,b]$	$E \le \frac{\alpha}{\sqrt{n}} R^{n+1} \lVert f^{(n+1)} \rVert_\infty$ $R = \frac{(b-a)e}{4(n+1)}$

Table C.1 Summary of interpolation methods. In the last column, an expression with a h means the x_i's are equally spaced.

	Rule	**Composite**
Midpoint	$\int_{x_i}^{x_{i+1}} f(x)dx = hf_{i+1/2} + \tau_r$ $\tau_r = \frac{1}{24}h^3 f''(\eta)$	$\int_a^b f(x)dx = h\left(f_{1+1/2} + f_{2+1/2} + \cdots + f_{n+1/2}\right) + \tau_c$ $\lvert\tau_c\rvert \leq \frac{b-a}{24}h^2 \lvert\lvert f''\rvert\rvert_\infty$
Trapezoid	$\int_{x_i}^{x_{i+1}} f(x)dx = \frac{h}{2}(f_{i+1} + f_i) + \tau_r$ $\tau_r = -\frac{1}{12}h^3 f''(\eta)$	$\int_a^b f(x)dx = h\left(\frac{1}{2}f_1 + f_2 + f_3 + \cdots + f_n + \frac{1}{2}f_{n+1}\right) + \tau_c$ $\lvert\tau_c\rvert \leq \frac{b-a}{12}h^2 \lvert\lvert f''\rvert\rvert_\infty$
Simpson	$\int_{x_{i-1}}^{x_{i+1}} f(x)dx = \frac{h}{3}(f_{i-1} + 4f_i + f_{i+1}) + \tau_r$ $\tau_r = -\frac{1}{90}h^5 f''''(\eta)$	$\int_a^b f(x)dx = \frac{h}{3}(f_1 + 4f_2 + 2f_3 + 4f_4 + 2f_5 + \cdots + 4f_n + f_{n+1}) + \tau_c$ $\lvert\tau_c\rvert \leq \frac{b-a}{90}h^4 \lvert\lvert f''''\rvert\rvert_\infty$
Hermite		$\int_a^b f(x)dx = h\left(\frac{1}{2}f_1 + f_2 + \cdots + f_n + \frac{1}{2}f_{n+1}\right) + \frac{1}{12}h^2(f_1' - f_{n+1}') + \tau_c$ $\lvert\tau_c\rvert \leq \frac{b-a}{720}h^4 \lvert\lvert f''''\rvert\rvert_\infty$
2-Point Gaussian	$\int_{x_{i-1}}^{x_{i+1}} f(x)dx = \frac{1}{2}h\left[f(z_i^+) + f(z_i^-)\right] + \tau_r$ $z_i^\pm = x_i + \frac{1}{2}h \pm \frac{1}{2\sqrt{3}}h$ $\tau_r = \frac{1}{4320}h^5 f''''(\eta)$	$\int_a^b f(x)dx = \frac{h}{2}\left(f_1^- + f_1^+ + f_2^- + f_2^+ \cdots + f_n^- + f_n^+\right) + \tau_c$ $\lvert\tau_c\rvert \leq \frac{b-a}{4320}h^4 \lvert\lvert f''''\rvert\rvert_\infty$

Table C.2 Summary of integration methods. Note that $h = x_{i+1} - x_i$, $f_i = f(x_i)$, $f_{i+1/2} = f(x_i + \frac{1}{2}h)$, and $f_i^\pm = f(z_i^\pm)$ where $z^\pm$ are defined in (6.33).

Methods for solving the differential equation

$$\frac{dy}{dt} = f(t, y)$$

Method	**Difference Equation**	τ_j	**Properties**
Euler	$y_{j+1} = y_j + kf_j$	$O(k)$	Explicit; Conditionally A-stable
Backward Euler	$y_{j+1} = y_j + kf_{j+1}$	$O(k)$	Implicit; A-stable
Trapezoidal	$y_{j+1} = y_j + \frac{k}{2}(f_j + f_{j+1})$	$O(k^2)$	Implicit; A-stable
Heun (RK2)	$y_{j+1} = y_j + \frac{1}{2}(k_1 + k_2)$ where $k_1 = kf_j, \quad k_2 = kf(t_{j+1}, y_j + k_1)$	$O(k^2)$	Explicit; Conditionally A-stable
Classic Runge–Kutta (RK4)	$y_{j+1} = y_j + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)$ where $k_1 = kf_j, \quad k_2 = kf(t_j + \frac{k}{2}, y_j + \frac{1}{2}k_1),$ $k_3 = kf(t_j + \frac{k}{2}, y_j + \frac{1}{2}k_2), \quad k_4 = kf(t_{j+1}, y_j + k_3)$	$O(k^4)$	Explicit; Conditionally A-stable

Table C.3 Finite difference methods for solving an IVP. The points $t_1, t_2, t_3, \ldots$ are equally spaced with step size $k = t_{j+1} - t_j$. Also, $f_j = f(t_j, y_j)$, $f_{j+1} = f(t_{j+1}, y_{j+1})$, and τ_j is the truncation error for the method.

Method	**Algorithm: For Solving $\mathbf{Av} = \mathbf{b}$**	**Algorithm: General Nonlinear $F(\mathbf{v})$**
Steepest Descent Method	Pick $\mathbf{v}_1$ For $k = 1, 2, 3, \ldots$ $\mathbf{v}_{k+1} = \mathbf{v}_k + \alpha_k \mathbf{r}_k$ $\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{q}_k$ where $\mathbf{q}_k = \mathbf{Ad}_k$ $\alpha_k = \dfrac{\mathbf{r}_k \cdot \mathbf{r}_k}{\mathbf{d}_k \cdot \mathbf{q}_k}$	Pick $\mathbf{v}_1$ For $k = 1, 2, 3, \ldots$ $\mathbf{v}_{k+1} = \mathbf{v}_k - \alpha_k \mathbf{g}_k$ where $\mathbf{g}_k = \nabla F(\mathbf{v}_k)$ $\alpha_k \ni F(\mathbf{v}_k - \alpha_k \mathbf{g}_k) < F(\mathbf{v}_k)$
Conjugate Gradient Method	Pick $\mathbf{v}_1$, set $\mathbf{r}_1 = \mathbf{b} - \mathbf{Av}_1$ and $\mathbf{d}_1 = \mathbf{r}_1$ For $k = 1, 2, 3, \ldots$ $\mathbf{v}_{k+1} = \mathbf{v}_k + \alpha_k \mathbf{d}_k$ $\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k \mathbf{q}_k$ $\mathbf{d}_{k+1} = \mathbf{r}_{k+1} + \beta_k \mathbf{d}_k$ where $\mathbf{q}_k = \mathbf{Ad}_k$ $\alpha_k = \dfrac{\mathbf{r}_k \cdot \mathbf{r}_k}{\mathbf{d}_k \cdot \mathbf{q}_k}$ $\beta_k = \dfrac{\mathbf{r}_{k+1} \cdot \mathbf{r}_{k+1}}{\mathbf{r}_k \cdot \mathbf{r}_k}$	Pick $\mathbf{v}_1$ and set $\mathbf{d}_1 = -\mathbf{g}_1$ For $k = 1, 2, 3, \ldots$ $\mathbf{v}_{k+1} = \mathbf{v}_k + \alpha_k \mathbf{d}_k$ $\mathbf{d}_{k+1} = -\mathbf{g}_{k+1} + \beta_k \mathbf{d}_k$ where $\alpha_k \ni F(\mathbf{v}_k + \alpha_k \mathbf{d}_k) < F(\mathbf{v}_k)$ $\beta_k = \dfrac{\mathbf{g}_{k+1} \cdot \mathbf{g}_{k+1}}{\mathbf{g}_k \cdot \mathbf{g}_k}$

Table C.4 Gradient decent methods for minimization. The matrix **A** is assumed to symmetric and positive definite.

References

Abdi, H., Williams, L.J.: Principal component analysis. Wiley Interdiscip. Rev. Comput. Stat. **2**(4),433–459 (2010). ISSN 1939-0068. doi:10.1002/wics.101

Anderson, M., Adali, T.: A general approach for robustification of ICA algorithms. In: Vigneron, V., Zarzoso, V., Moreau, E., Gribonval, R., Vincent, E. (eds.) Latent Variable Analysis and Signal Separation. Lecture Notes in Computer Science, vol. 6365, pp. 295–302. Springer, Berlin (2010). ISBN 978-3-642-15994-7. doi:10.1007/978-3-642-15995-4_37

Andersen, A.H., Gash, D.M., Avison, M.J.: Principal component analysis of the dynamic response measured by fMRI: a generalized linear systems framework. Magn. Reson. Imaging **17**(6), 795–815 (1999)

ANSI/IEEE: IEEE Standard for Binary Floating-Point Arithmetic, volume Std 754-1985. IEEE, New York (1985)

Applegate, D.L., Bixby, R.E., Chvátal, V., Cook, W.J.: The Traveling Salesman Problem: A Computational Study. Princeton Series in Applied Mathematics. Princeton University Press, Princeton (2006)

Ascher, U.M., Petzold, L.R.: Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations. SIAM, Philadelphia, PA (1998). ISBN 0898714125

Astier, P., Guy, J., Regnault, N., Pain, R., Aubourg, E., Balam, D., Basa, S., Carlberg, R.G., Fabbro, S., Fouchez, D., Hook, I.M., Howell, D.A., Lafoux, H., Neill, J.D., Palanque-Delabrouille, N., Perrett, K., Pritchet, C.J., Rich, J., Sullivan, M., Taillet, R., Aldering, G., Antilogus, P., Arsenijevic, V., Balland, C., Baumont, S., Bronder, J., Courtois, H., Ellis, R.S., Filiol, M., Gonçalves, A.C., Goobar, A., Guide, D., Hardin, D., Lusset, V., Lidman, C., McMahon, R., Mouchet, M., Mourao, A., Perlmutter, S., Ripoche, P., Tao, C., Walton, N.: The supernova legacy survey:measurement of

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

ω_m, ω_Λ and w from the first year data set. Astron. Astrophys. **447**(1), 31–48 (2006). doi:10.1051/0004-6361:20054185. http://dx.doi.org/10.1051/0004-6361:20054185

Bailey, D.H., Lee, K., Simon, H.D.: Using Strassen's algorithm to accelerate the solution of linear systems. J. Supercomput. **4**, 357–371 (1991)

Ballard, G., Demmel, J., Holtz, O., Schwartz, O.: Minimizing communication in numerical linear algebra. SIAM J. Matrix Anal. Appl. **32**(3), 866–901 (2011). doi:10.1137/090769156. http://dx.doi.org/10.1137/090769156

Bartlett, M.S., Movellan, J.R., Sejnowski, T.J.: Face recognition by independent component analysis. IEEE Trans. Neural Netw. **13**(6), 1450–1464 (2002). ISSN 1045-9227. doi:10.1109/tnn.2002.804287

Batterson, S., Smillie, J.: Rayleigh quotient iteration fails for nonsymmetric matrices. Appl. Math. Lett. **2**(1), 19–20 (1989). ISSN 0893-9659. doi:http://dx.doi.org/10.1016/0893-9659(89)90107-9

Beattie, C., Fox, D.: Localization criteria and containment for Rayleigh quotient iteration. SIAM J. Matrix Anal. Appl. **10**(1), 80–93 (1989). doi:10.1137/0610006

Benzi, M.: Preconditioning techniques for large linear systems: a survey. J. Comput. Phys. **182**(2), 418–477 (2002)

Berman, G.P., Izrailev, F.M.: The Fermi-Pasta-Ulam problem: fifty years of progress. Chaos **15**(1), 015104 (2005). ISSN 10541500

Bernoulli, J.: Meditationes de chordis vibrantibus. Comment. Acad. Sci. Imp. Petropol. **3**,13–28 (1728)

Bjöurck, A.: The calculation of linear least squares problems. Acta Numer. **13**, 1–53 (2004). ISSN 1474-0508. doi:10.1017/S0962492904000169. http://journals.cambridge.org/article_S0962492904000169

Bjöurck, A.: Numerical Methods in Matrix Computations. Springer, Cham (2015). ISBN 978-3-319-05088-1

Bogaert, I.: Iteration-free computation of Gauss–Legendre quadrature nodes and weights. SIAM J. Sci. Comput. **36**(3), A1008–A1026 (2014). doi:10.1137/140954969. http://dx.doi.org/10.1137/140954969

Boyd, J.P.: Solving Transcendental Equations: The Chebyshev Polynomial Proxy and Other Numerical Rootfinders, Perturbation Series, and Oracles. Other Titles in Applied Mathematics. SIAM, Philadelphia (2014). ISBN 9781611973525. https://books.google.com/books?id=29CgBAAAQBAJ

Brewer, D. Barenco, M., Callard, R., Hubank, M., Stark, J.: Fitting ordinary differential equations to short time course data. Philos. Trans. A **366** (1865), 519–544 (2008)

Brockett, P.L., Derrig, R.A., Golden, L.L., Levine, A., Alpert, M.: Fraud classification using principal component analysis of RIDITs. J. Risk Insur. **69**(3), 341–371 (2002). ISSN 1539-6975. doi:10.1111/1539-6975.00027. http://dx.doi.org/10.1111/1539-6975.00027

Brys, G., Hubert, M., Rousseeuw, P.J.: A robustification of independent component analysis. J. Chemometr. **19**(5–7), 364–375 (2005). ISSN 1099-128X. doi:10.1002/cem.940

Bunch, J.R., Hopcroft, J.E.: Triangular factorization and inversion by fast matrix multiplication. Math. Comput. **28**(125), 231–236 (1974)

Burges, C.J.C.: Geometric methods for feature extraction and dimensional reduction. In: Maimon, O., Rokach, L. (eds.) The Data Mining and Knowledge Discovery Handbook, 2nd edn., pp. 53–82. Springer, New York (2010)

Burkardt, J.: Hand-data. http://people.sc.fsu.edu/~jburkardt/m_src/hand_data/hand_data.html (2015)

Butcher, J.C.: The Numerical Analysis of Ordinary Differential Equations, 2nd edn. Wiley, Chichester (2008)

Chen, A., Bickel, P.J.: Efficient independent component analysis. Ann. Stat. **34**(6), 2825–2855 (2006). doi:10.1214/009053606000000939

Chernov, N., Lesort, C.: Least squares fitting of circles. J. Math. Imaging Vision **23**, 239–252 (2005)

Cline, A.K., Dhillon, I.S.: Computation of the singular value decomposition. In: Hogben, L. (ed.) Handbook of Linear Algebra, pp. 45.1–45.13. Chapman & Hall/CRC, Boca Raton (2007)

Collange, S., Defour, D., Graillat, S., Iakymchuk, R.: Numerical reproducibility for the parallel reduction on multi- and many-core architectures. Parallel Comput. **49**, 83–97 (2015). ISSN 0167-8191. doi:http://dx.doi.org/10.1016/j.parco.2015.09.001. http://www.sciencedirect.com/science/article/pii/S0167819115001155

Comon, P., Jutten, C. (eds.): Handbook of Blind Source Separation: Independent Component Analysis and Applications. Communications Engineering. Elsevier, Boston (2010). ISBN 978-0-12-374726-6

Conn, A.R., Scheinberg, K., Vicente, L.N.: Introduction to Derivative-Free Optimization. SIAM, Philadelphia (2009)

Cordero, A., Jordnn, C., Torregrosa, J.R.: One-point Newton-type iterative methods: a unified point of view. J. Comput. Appl. Math. **275**, 366–374 (2015). ISSN 0377-0427. doi:http://dx.doi.org/10.1016/j.cam.2014.07.009. http://www.sciencedirect.com/science/article/pii/S0377042714003288

Costello, A.B., Osborne, J.W.: Best practices in exploratory factor analysis: four recommendations for getting the most from your analysis. Pract. Assess. Res. Eval. **10**(7), 1–9 (2005). ISSN 1531-7714

Crandall, R, Pomerance, C.B.: Prime Numbers: A Computational Perspective, 2nd edn. Springer, New York (2010). ISBN 9780387289793

Davis, P.J., Rabinowitz, P.: Methods of Numerical Integration. Dover, New York (2007). ISBN 9780486453392. https://books.google.com/books?id=gGCKdqka0HAC

de Boor, C., Schoenberg, I.J.: Cardinal interpolation and spline functions VIII. The Budan-Fourier theorem for splines and applications. In: Bohmer, K., Meinardus, G., Schempp, W. (eds.) Spline Functions. Lecture Notes in Mathematics, vol. 501, pp. 1–79. Springer, Berlin (1976). ISBN 978-3-540-07543-1. doi:10.1007/BFb0079740 de Dinechin, F., Defour, D., Lauter, C.: Fast correct rounding of elementary functions in double precision using double-extended arithmetic. Technical Report 2004-10, Laboratoire de l'Informatique du Parallelisme, March 2004

Demmel, J.W.: Applied Numerical Linear Algebra. SIAM, Philadelphia, PA (1997). ISBN 0-89871-389-7

Demmel, J., Hida, Y.: Accurate and efficient floating point summation. SIAM J. Sci. Comput. **25**(4), 1214–1248 (2004)

Demmel, J, Nguyen, H.D.: Fast reproducible floating-point summation. In: 21st IEEE Symposium on Computer Arithmetic (ARITH), April 2013, pp. 163–172 (2013). doi:10.1109/ARITH.2013.9

Dennis, J.E., Jr., Schnabel, R.B.: Numerical Methods for Unconstrained Optimization and Nonlinear Equations. SIAM, Philadelphia, PA (1996). ISBN 0898713641

Dongarra, J., Sullivan, F.: The top 10 algorithms. Comput. Sci. Eng. **2**(1), 22–23 (2000)

Doukas, H., Papadopoulou, A., Savvakis, N., Tsoutsos, T., Psarras, J.: Assessing energy sustainability of rural communities using principal component analysis. Renew. Sust. Energ. Rev. **16**(4), 1949–1957 (2012). ISSN 1364-0321. doi:http://dx.doi.org/10.1016/j.rser.2012.01.018

Dussault, J.-P.: Solving trajectory optimization problems via nonlinear programming: the brachistochrone case study. Optim. Eng. **15**, 1–17 (2014). ISSN 1389-4420. doi:10.1007/s11081-013-9244-4

Erdmann, U., Ebeling, W., Mikhailov, A.S.: Noise-induced transition from translational to rotational motion of swarms. Phys. Rev. E **71**(5), 051904 (2005)

Evans, G.A., Webster, J.R.: A comparison of some methods for the evaluation of highly oscillatory integrals. J. Comput. Appl. Math. **112**, 55–69 (1999). ISSN 0377-0427. doi:http://dx.doi.org/10.1016/S0377-0427(99)00213-7

Fabian, J.T., Elmar, W.L., Carlos, G.P.: A geometric algorithm for overcomplete linear ICA. Neurocomputing **56**, 381–398 (2004). ISSN 0925-2312. doi:http://dx.doi.org/10.1016/j.neucom.2003.09.008

Ford, J.: The Fermi-Pasta-Ulam problem: paradox turns discovery. Phys. Rep. **213**(5), 271–310 (1992). ISSN 0370-1573. doi:http://dx.doi.org/10.1016/0370-1573(92)90116-H

Fourier, M.: Analyse des Équations Déterminées. Paris, Didot (1831). https://books.google.com/books?id=HIU_AAAAcAAJ

Gander, W., Gautschi, W.: Adaptive quadrature—revisited. BIT **40**(1), 84–101 (2000). ISSN 0006-3835. doi:10.1023/A:1022318402393

Gander, W., Golub, G.H., Strebel, R.: Least-squares fitting of circles and ellipses. BIT Numer. Math. **34**(4), 558–578 (1994). ISSN 0006-3835. doi:10.1007/BF01934268

GIMPS: Great Internet Mersenne Prime Search (2015). http://www.mersenne.org

Ginsberg, J., Mohebbi, M.H., Patel, R.S., Brammer, L., Smolinski, M.S., Brilliant, L.: Detecting influenza epidemics using search engine query data. Nature **457**, 1012–1014 (2009). doi:10.1038/nature07634

Giraud, L., Langou, J., Rozloznik, M.: The loss of orthogonality in the Gram-Schmidt orthogonalization process. Comput. Math. Appl. **50**(7), 1069–1075 (2005). ISSN 0898-1221

Gladwell, G.M.L.: Contact Problems in the Classical Theory of Elasticity. Sijthoff and Noordhoff, Germantown, MD (1980)

Goldberg, D.: What every computer scientist should know about floating-point arithmetic. ACM Comput. Surv. **23**(1), 5–48 (1991)

Golub, G., Uhlig, F.: The QR algorithm: 50 years later its genesis by John Francis and Vera Kublanovskaya and subsequent developments. IMA J. Numer. Anal. **29**(3), 467–485 (2009). doi:10.1093/imanum/drp012. http://imajna.oxfordjournals.org/content/29/3/467.abstract

Golub, G.H., Van Loan, C.F.: Matrix Computations, 4th edn. Johns Hopkins University Press, Baltimore, MD (2013). ISBN 1421408597

Golub, G.H., Welsch, J.H.: Calculation of Gauss quadrature rules. Math. Comput. **23**(106), 221–230 (1969). doi:10.3934/jcd.2014.1.391

Gonnet, P.: A review of error estimation in adaptive quadrature. ACM Comput. Surv. **44**(4), 22:1–22:36 (2012). ISSN 0360-0300. doi:10.1145/2333112.2333117

Google: Google flu trends. http://www.google.org/flutrends (2015)

Grasselli, M., Pelinovsky, D.: Numerical Mathematics. Jones and Bartlett Publishers, Sudbury, MA (2008)

Greenan, G., Brangwynne, C.P., Jaensch, S., Gharakhani, J., Julicher, F., Hyman, A.A.: Centrosome size sets mitotic spindle length in Caenorhabditis elegans embryos. Curr. Biol. **20**(4), 353–358 (2010)

Griffiths, D., Higham, D.J.: Numerical Methods for Ordinary Differential Equations. Springer, London (2010)

Gwinn, C.R., Johnson, M.D., Reynolds, J.E., Jauncey, D.L., Tzioumis, A.K., Dougherty, S., Carlson, B., Del Rizzo, D., Hirabayashi, H., Kobayashi, H., Murata, Y., Edwards, P.G., Quick, J.F.H., Flanagan, C.S., McCulloch, P.M.: Noise in the cross-power spectrum of the Vela pulsar. Astrophys. J. **758**(1), 6 (2012)

Hagan, P.S., West, G.: Interpolation methods for curve construction. Appl. Math. Finance **13**(2), 89–129 (2006). ISSN 1350486X. http://search.ebscohost.com.libproxy.rpi.edu/login.aspx?direct=true&db=buh&AN=21645450&site=ehost-live&scope=site

Hairer, E., Wanner, G.: Solving Ordinary Differential Equations II, 2nd edn. Springer, Berlin (2002)

Hairer, E., Lubich, C., Wanner, G.: Geometric numerical integration illustrated by the Stormer–Verlet method. Acta Numer. **12**, 399–450 (2003). ISSN 1474-0508. doi:10.1017/S0962492902000144

Halko, N., Martinsson, P.-G., Shkolnisky, Y., Tygert, M.: An algorithm for the principal component analysis of large data sets. SIAM J. Sci. Comput. **33**(5), 2580–2594 (2011). ISSN 1064-8275. doi:10.1137/100804139

Hall, C.A., Meyer, W.W.: Optimal error bounds for cubic spline interpolation. J. Approx. Theory **16**(2), 105–122 (1976). ISSN 0021-9045. doi:http://dx.doi.org/10.1016/0021-9045(76)90040-X

Hanrot, G., Lefevre, V., Stehle, D., Zimmermann, P.: Worst cases of a periodic function for large arguments. In: 18th IEEE Symposium on Computer Arithmetic, 2007 (ARITH '07), pp. 133–140 (2007). doi:10.1109/ARITH.2007.37

Harrison, J., Kubaska, T., Story, S., Tang, P.: The computation of transcendental functions on the IA-64 architecture. Intel Technol. J. **4**, 234–251 (1999)

Higham, N.J.: The accuracy of floating point summation. SIAM J. Sci. Comput. **14**, 783–799 (1993)

Higham, N.J.: Accuracy and Stability of Numerical Algorithms, 2nd edn. SIAM, Philadelphia, PA (2002). ISBN 0-89871-521-0

Hojjati, A., Zhao, G.-B., Pogosian, L., Silvestri, A., Crittenden, R., Koyama, K.: Cosmological tests of general relativity: a principal component analysis. Phys. Rev. D **85**, 043508 (2012). doi:10.1103/PhysRevD.85.043508

Holladay, J.C.: A smoothest curve approximation. Math. Tables Aids Comput. **11**(60), 233–243 (1957)

Holmes, M.H.: Introduction to Numerical Methods in Differential Equations. Springer, New York (2007)

Holmes, M.H.: Introduction to the Foundations of Applied Mathematics. Springer, New York (2009)

Holmes, M.H.: Introduction to Perturbation Methods, 2nd edn. Springer, New York (2013)

Holmes, M.H.: Connections between cubic splines and quadrature rules. Am. Math. Mon. **121**(7), 661–662 (2014)

Huss-Lederman, S., Jacobson, E.M., Johnson, J.R., Tsao, A., Turnbull, T.: Implementation of Strassen's algorithm for matrix multiplication. In: Proceedings of the 1996 ACM/IEEE Conference on Supercomputing, pp. 9–6 (1996)

Huynh, H.T.: Accurate monotone cubic interpolation. SIAM J. Numer. Anal. **30**(1), 57–100 (1993)

Hyvärinen, A., Karhunen, J., Oja, E.: Independent Component Analysis. Wiley-Interscience. Wiley, New York (2001). ISBN 047140540X

Iooss, G., James, G.: Localized waves in nonlinear oscillator chains. Chaos **15**(1), 015113 (2005)

Iriarte-Díaz, J.: Differential scaling of locomotor performance in small and large terrestrial mammals. J. Exp. Biol. **205**(18), 2897–2908 (2002). http://jeb.biologists.org/content/205/18/2897.abstract

Isaacson, E., Keller, H.B.: Analysis of Numerical Methods. Dover Publications, New York (1994). ISBN 9780486680293. https://books.google.com/books?id=y77n2ySMJHUC

Iserles, A., Nørsett, S.P., Olver, S.: Highly oscillatory quadrature: the story so far. In: de Castro, A.B., Gomez, D., Quintela, P., Salgado, P. (eds.) Numerical Mathematics and Advanced Applications, pp. 97–118. Springer, Berlin (2006). ISBN 978-3-540-34287-8. doi:10.1007/978-3-540-34288-5_6

Jackson, D.A.: Stopping rules in principal components analysis: a comparison of heuristical and statistical approaches. Ecology **74**(8), 2204–2214 (1993)

James, G., Kevrekidis, P.G., Cuevas, J.: Breathers in oscillator chains with Hertzian interactions. Phys. D **251**, 39–59 (2013). ISSN 0167-2789. doi:http://dx.doi.org/10.1016/j.physd.2013.01.017

Jameson, A., Vassberg, J.: Studies of alternate numerical optimization methods applied to the brachistochrone problem. CFD J. **9**(3), 281–296 (2001)

Johnson, K.A., Goody, R.S.: The original Michaelis constant: translation of the 1913 Michaelis-Menten paper. Biochemistry **50**(39), 8264–8269 (2011)

Josse, J., Husson, F.: Selecting the number of components in principal component analysis using cross-validation approximations. Comput. Stat. Data Anal. **56**(6), 1869–1879 (2012). ISSN 0167-9473. doi:http://dx.doi.org/10.1016/j.csda.2011.11.012. http://www.sciencedirect.com/science/article/pii/S0167947311004099

Jung, B.C., Choi, S.I., Du, A.X., Cuzzocreo, J.L., Geng, Z.Z., Ying, H.S., Perlman, S.L., Toga, A.W., Prince, J.L., Ying, S.H.: Principal component analysis of cerebellar shape on MRI separates SCA types 2 and 6 into two archetypal modes of degeneration. Cerebellum **11**(4), 887–895 (2012). ISSN 1473-4222. doi:10.1007/s12311-011-0334-6

Kahan, W.M.: Numerical linear algebra. Can. Math. Bull. **9**, 757–801 (1966)

Kershaw, D.: A note on the convergence of interpolatory cubic splines. SIAM J. Numer. Anal. **8**(1), 67–74 (1971)

Khan, J., Wei, J.S., Ringner, M., Saal, L.H., Ladanyi, M., Westermann, F., Berthold, F., Schwab, M., Antonescu, C.R., Peterson, C., Meltzer, P.S.: Classification and diagnostic prediction of cancers using gene expression profiling and artificial neural networks. Nat. Med. **7**(6), 673–679 (2001) ISSN 10788956

Knoll, D.A., Keyes, D.E.: Jacobian-free Newton-Krylov methods: a survey of approaches and applications. J. Comput. Phys. **193**(2), 357–397 (2004). ISSN 0021-9991. doi:10.1016/j.jcp.2003.08.010

Knuth, D.E.: The Art of Computer Programming: Seminumerical Algorithms, vol. 2, 3rd edn. Addison-Wesley Longman Publishing Co., Boston, MA (1997). ISBN 0-201-89684-2

Kollerstrom, N.: Thomas Simpson and 'Newton's method of approximation': an enduring myth. Br. J. Hist. Sci. **25**, 347–354 (1992). ISSN 1474-001X. doi:10.1017/S0007087400029150. http://journals.cambridge.org/article_S0007087400029150

Kondrat, S., Perez, C.R., Presser, V., Gogotsi, Y., Kornyshev, A.A.: Effect of pore size and its dispersity on the energy storage in nanoporous supercapacitors. Energy Environ. Sci. **5**, 6474–6479 (2012). doi:10.1039/C2EE03092F. http://dx.doi.org/10.1039/C2EE03092F

Korats, G., Le Cam, S., Ranta, R., Hamid, M.: Applying ICA in EEG: choice of the window length and of the decorrelation method. In: Gabriel, J., Schier, J., Van Huffel, S., Conchon, E., Correia, C., Fred, A., Gamboa, H. (eds.) Biomedical Engineering Systems and Technologies. Communications in Computer and Information Science, vol. 357, pp. 269–286. Springer, Berlin (2013). ISBN 978-3-642-38255-0. doi:10.1007/978-3-642-38256-7_18. http://dx.doi.org/10.1007/978-3-642-38256-7_18

Kuczyński, J., Woźniakowski, H.: Estimating the largest eigenvalue by the power and Lanczos algorithms with a random start. SIAM J. Matrix Anal. Appl. **13**(4), 1094–1122 (1992). doi:10.1137/0613066

Lagarias, J.C., Reeds, J.A., Wright, M.H., Wright, P.E.: Convergence properties of the Nelder-Mead simplex method in low dimensions. SIAM J. Optim. **9**, 112–147 (1998)

Lai, K.-L., Crassidis, J.L.: Extensions of the first and second complex-step derivative approximations. J. Comput. Appl. Math. **219**(1), 276–293 (2008). ISSN 0377-0427. doi:http://dx.doi.org/10.1016/j.cam.2007.07.026

Lambert, J.D.: Numerical Methods for Ordinary Differential Systems: The Initial Value Problem. Wiley, Chichester (1991)

Lazer, D., Kennedy, R., King, G., Vespignani, A.: The parable of Google flu: traps in big data analysis. Science **343**(6176), 1203–1205 (2014). doi:10.1126/science.1248506. http://www.sciencemag.org/content/343/6176/1203.short

Leutbecher, M., Palmer, T.N.: Ensemble forecasting. J. Comput. Phys. **227**(7), 3515–3539 (2008). ISSN 0021-9991. doi:http://dx.doi.org/10.1016/j.jcp.2007.02.014. http://www.sciencedirect.com/science/article/pii/S0021999107000812. Predicting weather, climate and extreme events

Levin, Y., Nediak, M., Ben-Israel, A.: A direct Newton method for calculus of variations. J. Comput. Appl. Math. **139**(2), 197–213 (2002). ISSN 0377-0427. doi:http://dx.doi.org/10.1016/S0377-0427(01)00427-7

Liu, J.: The multifrontal method for sparse matrix solution: theory and practice. SIAM Rev. **34**(1), 82–109 (1992). doi:10.1137/1034004

Loffeld, J., Tokman, M.: Comparative performance of exponential, implicit, and explicit integrators for stiff systems of ODEs. J. Comput. Appl. Math. **241**, 45–67 (2013). ISSN 0377-0427. doi:http://dx.doi.org/10.1016/j.cam.2012.09.038

Love, C.H.: Abscissas and Weights for Gaussian Quadrature for n=2 to 100, and n=125, 150, 175, and 200. National Bureau of Standards, U.S. Government Printing Office, Washington, DC (1966)

Marsden, M.J.: Quadratic spline interpolation. Bull. Am. Math. Soc. **80**(5), 903–906, (1974) Martins, J.R.R.A., Sturdza, P., Alonso, J.J.: The complex-step derivative approximation. ACM Trans. Math. Softw. **29**(3), 245–262 (2003)

Mei, L., Figl, M., Darzi, A., Rueckert, D., Edwards, P.: Sample sufficiency and PCA dimension for statistical shape models. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) Computer Vision—ECCV 2008. Lecture Notes in Computer Science, vol. 5305, pp. 492–503. Springer, Berlin (2008). ISBN 978-3-540-88692-1. doi:10.1007/978-3-540-88693-8_36. http://dx.doi.org/10.1007/978-3-540-88693-8_36

Michel, J.-B., Shen, Y.K., Aiden, A.P., Veres, A., Gray, M.K., The Google Books Team, Pickett, J.P., Hoiberg, D., Clancy, D., Norvig, P., Orwant, J., Pinker, S., Nowak, M.A., Aiden, E.L.: Quantitative analysis of culture using millions of digitized books. Science **331**(6014), 176–182 (2011). doi:10.1126/science.1199644

Morin, A., Urban J., Adams, P.D., Foster, I., Sali, A., Baker, D., Sliz, P.: Shining light into black boxes. Science **336**(6078), 159–160 (2012). doi:10.1126/science.1218263. http://www.sciencemag.org/content/336/6078/159.short

Mortenson, M.E.: Geometric Modeling, 2nd edn. Wiley, New York (1997)

Muller, J.-M.: Elementary Functions: Algorithms and Implementation, 2nd edn. Birkhauser, Boston (2005)

Muller, J.-M., Brisebarre, N., de Dinechin, F., Jeannerod, C.-P., Lefèvre, V., Melquiond, G., Revol, N., Stehlé, D., Torres, S.: Handbook of Floating-Point Arithmetic. Birkhäuser, Boston (2010)

Nakatsukasa, Y., Higham, N.: Stable and efficient spectral divide and conquer algorithms for the symmetric eigenvalue decomposition and the SVD. SIAM J. Sci. Comput. **35**(3), A1325–A1349 (2013). doi:10.1137/120876605

NCEI: Daily summary for Albany, NY. In: National Centers for Environmental Information, Climate Data Online. Washington, DC. http://www.ncdc.noaa.gov/cdo-web/search?datasetid=GHCND (2015)

Nelder, J.A.: This week's citation classic. Citation Classics Commentaries April **9**(15), 22 (1979)

Nelder, J.A., Mead, R.: A simplex method for function minimization. Comput. J. **7**(4), 308–313 (1965)

Nenad, U., Roberts, A.J.: A corrected quadrature formula and applications. ANZIAM J. **45**, E41–E56 (2008)

Nguyen, T.V., Patra, J.C.: A simple ICA-based digital image watermarking scheme. Digital Signal Process. **18**(5), 762–776 (2008). ISSN 1051-2004. doi:http://dx.doi.org/10.1016/j.dsp.2007.10.004

Nocedal, J., Wright, S.: Numerical Optimization. Springer Series in Operations Research and Financial Engineering. Springer, New York (2006). ISBN 9780387303031

Nordhausen, K., Ollila, E., Oja, H.: On the performance indices of ICA and blind source separation. In: IEEE 12th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), June 2011, pp. 486–490 (2011). doi:10.1109/SPAWC.2011.5990458

Olver, F.W.J., Lozier, D.W., Boisvert, R.F., Clark, C.W.: NIST Handbook of Mathematical Functions. Cambridge University Press, Cambridge (2010)

Onton, J., Makeig, S.: Information-based modeling of event-related brain dynamics. In: Neuper, C., Klimesch, W. (eds.) Event-Related Dynamics of Brain Oscillations. Progress in Brain Research, vol. 159, pp. 99–120. Elsevier, Amsterdam (2006). doi:http://dx.doi.org/10.1016/S0079-6123(06)59007-7. http://www.sciencedirect.com/science/article/pii/S0079612306590077

Oprea, J.: Differential Geometry and Its Applications. Mathematical Association of America, Washington, DC (2007). ISBN 9780883857489

Overton, M.L.: Numerical Computing with IEEE Floating Point Arithmetic. SIAM, Philadelphia, PA (2001)

Pantazis, R.D., Szyld, D.B.: Regions of convergence of the Rayleigh quotient iteration method. Numer. Linear Algebra Appl. **2**(3), 251–269 (1995). ISSN 1099-1506. doi:10.1002/nla.1680020307

Parente, A., Sutherland, J.C.: Principal component analysis of turbulent combustion data: data pre-processing and manifold sensitivity. Combust. Flame **160**(2), 340–350 (2013). ISSN 0010-2180. doi:http://dx.doi.org/10.1016/j.combustflame.2012.09.016

Parlett, B.N.: The Rayleigh quotient iteration and some generalizations for nonnormal matrices. Math. Comput. **28**, 679–693 (1974)

Parlett, B.N.: The Symmetric Eigenvalue Problem. SIAM, Philadelphia, PA (1998). doi:10.1137/1.9781611971163

Peckham, M.: RIKEN plans exascale supercomputer '30 times faster' than today's fastest in six years. Time, 30 Dec 2013

Peng, R.D.: Reproducible research in computational science. Science **334** (6060), 1226–1227 (2011). doi:10.1126/science.1213847

Peres-Neto, P.R., Jackson, D.A., Somers, K.M.: How many principal components? Stopping rules for determining the number of non-trivial axes revisited. Comput. Stat. Data Anal. **49**(4), 974–997 (2005). ISSN 0167-9473. doi:http://dx.doi.org/10.1016/j.csda.2004.06.015. http://www.sciencedirect.com/science/article/pii/S0167947304002014

Persson, A.H., Bondesson, L., Borlin, N.: Estimation of polygons and areas. Scand. J. Stat. **33**(3), 541–559 (2006). ISSN 1467-9469. doi:10.1111/j.1467-9469.2006.00507.x

Peters, G., Wilkinson, J.H.: Inverse iteration, ill-conditioned equations and Newton's method. SIAM Rev. **21**(3), 339–360 (1979). ISSN 00361445. http://www.jstor.org/stable/2029572

Raghavan, R., Kelkar, Y.D., Ochman, H.: A selective force favoring increased g+c content in bacterial genes. Proc. Natl. Acad. Sci. U. S. A. **109**(36), 14504–14507 (2012). doi:10.1073/pnas.1205683109. http://www.pnas.org/content/109/36/14504.abstract

Ralston, A.: Runge-Kutta methods with minimum error bounds. Math. Comput. **16**,431–437 (1962)

Ramsay, J.O., Hooker, G., Campbell, D., Cao, J.: Parameter estimation for differential equations: a generalized smoothing approach. J. R. Stat. Soc. B **69**(5), 741–796 (2007). ISSN 1467-9868

Raphson, J.: Analysis Aequationum Universalis. London (1690). http://books.google.com/books?id=JM4ZuQAACAAJ

Rokhlin, V., Szlam, A., Tygert, M.: A randomized algorithm for principal component analysis. SIAM J. Matrix Anal. Appl. **31**(3), 1100–1124 (2010). doi:10.1137/080736417

Rosser, J.B., Lanczos, C., Hestenes, M.R., Karush, W.: Separation of close eigenvalues of a real symmetric matrix. J. Res. Natl. Bur. Stand. **47**(4), 291–297 (1951)

Rump, S., Ogita, T., Oishi, S.: Accurate floating-point summation part I: faithful rounding. SIAM J. Sci. Comput. **31**(1), 189–224 (2008). doi:10.1137/050645671

Salomon, D.: Curves and Surfaces for Computer Graphics. Springer, New York (2006)

Sandor, J., Debnath, L.: On certain inequalities involving the constant e and their applications. J. Math. Anal. Appl. **249**(2), 569–582 (2000). ISSN 0022-247X. doi:http://dx.doi.org/10.1006/jmaa.2000.6911. http://www.sciencedirect.com/science/article/pii/S0022247X00969117

Santillana, M., Zhang, D., Althouse, B.M.: What can digital disease detection learn from (an external revision to) Google flu trends? Am. J. Prev. Med. **47**(3), 341–347 (2014). doi:10.1016/j.amepre.2014.05.020

Särelä, J., Vigário, R.: Overlearning in marginal distribution-based ICA: analysis and solutions. J. Mach. Learn. Res. **4**, 1447–1469 (2003). ISSN 1532-4435. doi:10.1162/jmlr.2003.4.7-8.1447. http://dx.doi.org/10.1162/jmlr.2003.4.7-8.1447

Schneider, T., Griffies, S.M.: A conceptual framework for predictability studies. J. Climate **12**(10), 3133–3155 (1999). doi:10.1175/1520-0442(1999)012⟨3133:ACFFPS⟩2.0.CO;2

Shure, L.: Comparing single-threaded vs. multithreaded floating point calculations. In: MATLAB Central, Loren on the Art of MATLAB (2009). http://blogs.mathworks.com/loren/2009/12/04/comparing-single-threaded-vs-multithreaded-floating-point-calculations/

Singh, S.: The Simpsons and Their Mathematical Secrets. Bloomsbury, New York (2013)

Smith, J.A., Wilson, L., Azarenko, O., Zhu, X., Lewis, B.M., Littlefield, B.A., Jordan, M.A.: Eribulin binds at microtubule ends to a single site on tubulin to suppress dynamic instability. Biochemistry **49**(6), 1331–1337 (2010)

Stadlbauer, A., Ganslandt, O., Buslei, R., Hammen, T., Gruber, S., Moser, E., Buchfelder, M., Salomonowitz, E., Nimsky, C.: Gliomas: histopathologic evaluation of changes in directionality and magnitude of water diffusion at diffusion-tensor MR imaging. Radiology **240**(3), 803–810 (2006)

Strassen, V.: Gaussian elimination is not optimal. Numer. Math. **13**(4), 354–356 (1969). ISSN 0029-599X. doi:10.1007/BF02165411

Stuart, A., Humphries, A.R.: Dynamical Systems and Numerical Analysis. Cambridge University Press, Cambridge (1998)

Süli, E., Mayers, D.F.: An Introduction to Numerical Analysis. Cambridge University Press, Cambridge (2003)

Teh, Y.W., Welling, M., Osindero, S., Hinton, G.E.: Energy-based models for sparse overcomplete representations. J. Mach. Learn. Res. **4**, 1235–1260 (2003). ISSN 1532-4435. http://dl.acm.org/citation.cfm?id=945365.964304

Tissot, G., Cordier, L., Benard, N., Noack, B.R.: Model reduction using dynamic mode decomposition. C.R. Mec. **342**(6–7), 410–416 (2014) ISSN 1631-0721. doi:http://dx.doi.org/10.1016/j.crme.2013.12.011. http://www.sciencedirect.com/science/article/pii/S163107211400103X

Traub, J.F.: Iterative Methods for the Solution of Equations. AMS Chelsea Publishing Series. Chelsea, New Jersey (1982). ISBN 9780828403122. http://books.google.com/books?id=se3YdgFgz4YC

Trefethen, L.N.: Is gauss quadrature better than Clenshaw–Curtis? SIAM Rev. **50**(1), 67–87 (2008). doi:10.1137/060659831. http://dx.doi.org/10.1137/060659831

Trefethen, L.N.: Approximation Theory and Approximation Practice. SIAM, Philadelphia, PA (2012). ISBN 1611972396, 9781611972399

Trefethen, L.N., Bau, D.: Numerical Linear Algebra. SIAM, Philadelphia, PA (1997). ISBN 0898713617

Trefethen, L.N., Weideman, J.A.C.: The exponentially convergent trapezoidal rule. SIAM Rev. **56**(3), 385–458 (2014). doi:10.1137/130932132

Tu, J.H., Rowley, C.W., Luchtenburg, D.M., Brunton, S.L., Kutz, J.N.: On dynamic mode decomposition: theory and applications. J. Comput. Dyn. **1** (2), 391–421 (2014). ISSN 2158-2491. doi:10.3934/jcd.2014.1.391. http://aimsciences.org/journals/displayArticlesnew.jsp?paperID=10631

U.S. Census Bureau: Crime rates by type: selected large cities. In: Statistical Abstract of the United States: 2012, 131st edn. Washington, DC (2012)

van den Berg, R.A., Hoefsloot, H.C.J., Westerhuis, J.A., Smilde, A.K., van der Werf, M.: Centering, scaling, and transformations: improving the biological information content of metabolomics data. BMC Genomics **7**(1), 142 (2006). doi:10.1186/1471-2164-7-142

van der Maaten, L.J.P., Postma, E.O., van den Herik, H.J.: Dimensionality Reduction: A Comparative Review. Technical report, Tilburg University (2009). http://homepage.tudelft.nl/19j49/Matlab_Toolbox_for_Dimensionality_Reduction_files/TR_Dimensiereductie.pdf

Varah, J.: A spline least squares method for numerical parameter estimation in differential equations. SIAM J. Sci. Stat. Comput. **3**(1), 28–46 (1982). doi:10.1137/0903003

Varga, R.S.: Gershgorin and His Circles. Springer Series in Computational Mathematics. Springer, Berlin (2004)

Voglis, N.: Waves derived from galactic orbits. In: Galaxies and Chaos. Lecture Notes in Physics. vol. 626, pp. 56–74. Springer, Berlin (2003)

Waldvogel, J.: Towards a general error theory of the trapezoidal rule. In: Gautschi, W., Mastroianni, G., Rassias, T.M. (eds.) Approximation and Computation. Springer Optimization and Its Applications, vol. 42, pp. 267–282. Springer, New York (2011)

Wallis, J.: Operum Mathematicorum. Number v. 2. Typis Leon, Oxford (1699). http://books.google.com/books?id=dhetMBKF4gkC

Watkins, D.S.: The QR algorithm revisited. SIAM Rev. **50**(1), 133–145 (2008). ISSN 0036-1445. doi:10.1137/060659454

Watson, G.A.: A Levenberg-Marquardt method for estimating polygonal regions. J. Comput. Appl. Math. **208**(2), 331–340 (2007). ISSN 0377-0427. doi:http://dx.doi.org/10.1016/j.cam.2006.10.001

Weideman, J.A.C.: Numerical integration of periodic functions: a few examples. Am. Math. Mon. **109**(1), 21–36 (2002). ISSN 00029890. http://www.jstor.org/stable/2695765

Weinstock, R.: Calculus of Variations: With Applications to Physics and Engineering. Dover, New York (1974). ISBN 9780486630694

White, F.M.: Viscous Fluid Flow. McGraw Hill Series in Mechanical Engineering, 3rd edn. McGraw-Hill, New York (2005)

Wilkins, G., Gu, M.: A modified Brent's method for finding zeros of functions. Numer. Math. **123**(1), 177–188 (2013). ISSN 0029-599X. doi:10.1007/s00211-012-0480-x. http://dx.doi.org/10.1007/s00211-012-0480-x

Williams, V.V.: Multiplying matrices faster than Coppersmith-Winograd. In: Proceedings of the 44th Annual ACM Symposium on Theory of Computing, STOC'12, pp. 887–898. ACM, New York, NY (2012). ISBN 978-1-4503-1245-5. doi:10.1145/2213977.2214056. http://doi.acm.org/10.1145/2213977.2214056

Wu, J., Brigham, K.G., Simon, M.A., Brigham, J.C.: An implementation of independent component analysis for 3D statistical shape analysis. Biomed. Signal Process. Control **13**, 345–356 (2014). ISSN 1746-8094. doi:http://dx.doi.org/10.1016/j.bspc.2014.06.003

Xue, H., Miao, H., Wu, H.: Sieve estimation of constant and time-varying coefficients in nonlinear ordinary differential equation models by considering both numerical error and measurement error. Ann. Stat. **38**(4), 2351–2387 (2010). doi:10.1214/09-AOS784

Yamazaki, I., Li, X.S.: New scheduling strategies and hybrid programming for a parallel right-looking sparse LU factorization algorithm on multicore cluster systems. In: IEEE 26th International Parallel Distributed Processing Symposium (IPDPS), May 2012, pp. 619–630 (2012). doi:10.1109/IPDPS.2012.63

Ypma, T.J.: Historical development of the Newton-Raphson method. SIAM Rev. **37**(4), 531–551 (1995). ISSN 0036-1445. doi:10.1137/1037125. http://dx.doi.org/10.1137/1037125

Zenil, H.: A Computable Universe: Understanding and Exploring Nature as Computation. World Scientific, Singapore (2012)

Zhang, H., Cheng, Z.: The performance evaluation of classic ICA algorithms for blind separation of fabric defects. J. Fiber Bioeng. Inform. **7**, 377–386 (2014)

Zhao, Q., Okada, K., Rosenbaum, K., Kehoe, L., Zand, D.J., Sze, R., Summar, M., Linguraru, M.G.: Digital facial dysmorphology for genetic screening: hierarchical constrained local model using ICA. Med. Image Anal. **18** (5), 699–710 (2014). ISSN 1361-8415. doi:http://dx.doi.org/10.1016/j.media.2014.04.002

Index

© Springer International Publishing Switzerland 2016
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

Series Editors

Timothy J. Barth
NASA Ames Research Center
NAS Division
Moffett Field, CA 94035, USA
barth@nas.nasa.gov

Michael Griebel
Institut für Numerische Simulation
der Universität Bonn
Wegelerstr. 6
53115 Bonn, Germany
griebel@ins.uni-bonn.de

David E. Keyes
Mathematical and Computer Sciences
and Engineering
King Abdullah University of Science
and Technology
P.O. Box 55455
Jeddah 21534, Saudi Arabia
david.keyes@kaust.edu.sa

and

Department of Applied Physics
and Applied Mathematics
Columbia University
500 W. 120th Street
New York, NY 10027, USA
kd2112@columbia.edu

Risto M. Nieminen
Department of Applied Physics
Aalto University School of Science
and Technology
00076 Aalto, Finland
risto.nieminen@tkk.fi

Dirk Roose
Department of Computer Science
Katholieke Universiteit Leuven
Celestijnenlaan 200A
3001 Leuven-Heverlee, Belgium
dirk.roose@cs.kuleuven.be

Tamar Schlick
Department of Chemistry
and Courant Institute
of Mathematical Sciences
New York University
251 Mercer Street
New York, NY 10012, USA
schlick@nyu.edu

Editor for Computational Science
and Engineering at Springer:
Martin Peters
Springer-Verlag
Mathematics Editorial IV
Tiergartenstrasse 17
69121 Heidelberg, Germany
martin.peters@springer.com