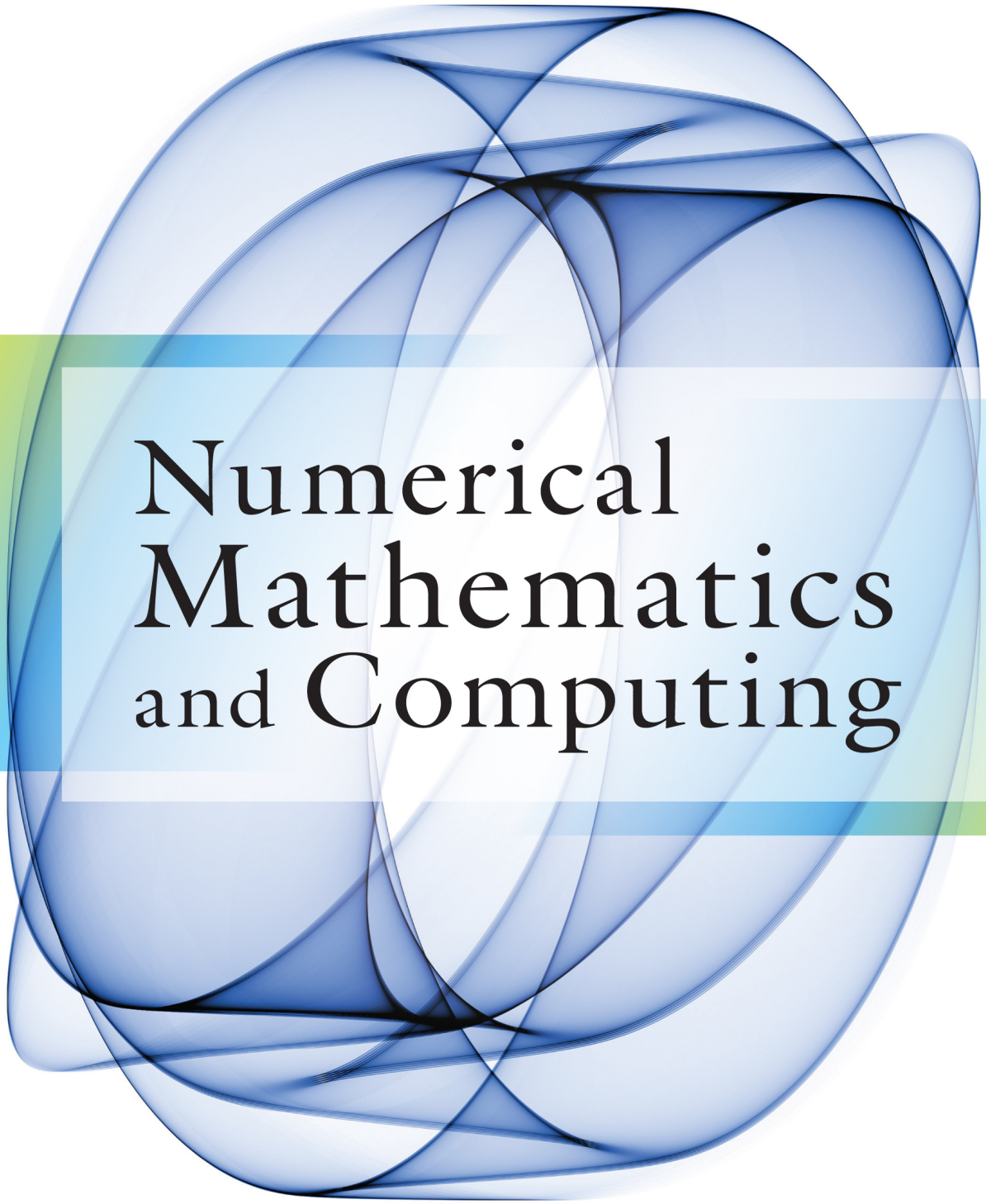


Numerical Mathematics and Computing

WARD CHENEY | DAVID KINCAID

SEVENTH EDITION

BROOKS/COLE
CENGAGE Learning

Numerical Mathematics and Computing, Seventh Edition
Ward Cheney & David Kincaid

Publisher: Richard Stratton

Acquisitions Editor: Molly Taylor

Assistant Editor: Shaylin Walsh Hogan

Editorial Assistant: Alex Gontar

Media Editor: Andrew Coppola

Marketing Manager: Jennifer P. Jones

Marketing Communications Manager: Mary Anne Payumo

Content Project Manager: Alison Eigel Zade

Senior Art Director: Linda May

Manufacturing Planner: Doug Bertke

Rights Acquisition Specialist: Shalice Shah-Caldwell

Production Service: MPS Limited

Interior Chapter Opener image: Roofoo/©istockphoto

Illustrator: MPS Limited

Cover Designer: KeDesign

Cover Image: Roofoo/©istockphoto

Compositor: MPS Limited

Library of Congress Control Number: 2012935124

ISBN-13: 978-1-133-10371-4

ISBN-10: 1-133-10371-5

Brooks/Cole
20 Channel Center Street
Boston, MA 02210
USA

Cengage Learning is a leading provider of customized learning solutions with office locations around the globe, including Singapore, the United Kingdom, Australia, Mexico, Brazil and Japan. Locate your local office at: **international.cengage.com/region.**

Cengage Learning products are represented in Canada by Nelson Education, Ltd.

For your course and learning solutions, visit **www.cengage.com.**

Purchase any of our products at your local college store or at our preferred online store **www.cengagebrain.com**

Instructors: Please visit **login.cengage.com** and log in to access instructor-specific resources.

Printed in the United States of America
1 2 3 4 5 6 7 16 15 14 13 12

1 Mathematical Preliminaries and Floating-Point Representation

The Taylor series for the natural logarithm $\ln(1 + x)$ is

$$\ln 2 = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5} - \frac{1}{6} + \frac{1}{7} - \frac{1}{8} + \cdots$$

Adding together the eight terms shown, we obtain $\ln 2 \approx 0.63452$,* which is a poor approximation to $\ln 2 = 0.69315\ldots$. On the other hand, the Taylor series for $\ln[(1 + x)/(1 - x)]$ gives us (with $x = \frac{1}{3}$)

$$\ln 2 = 2\left(3^{-1} + \frac{3^{-3}}{3} + \frac{3^{-5}}{5} + \frac{3^{-7}}{7} + \cdots\right)$$

By adding the four terms shown between the parentheses and multiplying by 2, we obtain $\ln 2 \approx 0.69313$. This illustrates the fact that rapid convergence of a Taylor series can be expected near the point of expansion but not at remote points. Evaluating the series $\ln[(1 + x)/(1 - x)]$ at $x = \frac{1}{3}$ is a mechanism for evaluating $\ln 2$ near the point of expansion. It also gives an example in which the properties of a function can be exploited to obtain a more rapidly convergent series. Taylor series and Taylor's Theorem are two of the principal topics we discuss in this chapter. They are ubiquitous features in much of numerical analysis.

Computers usually do *not* use base-10 arithmetic for storage or computation. Numbers that have a finite expression in one number system may have an infinite expression in another system. This phenomenon is illustrated when the familiar decimal number 1/10 is converted into the binary system:

$$(0.1)_{10} = (0.0\,0011\,0011\,0011\,0011\,0011\,0011\,0011\,0011\ldots)_2$$

We explain the floating-point number system and develop basic facts about roundoff errors. Another topic is loss of significance, which occurs when nearly equal numbers are subtracted. It is studied and shown to be avoidable by various programming techniques.

*The symbol $\approx$ means "approximately equal to."

1.1 Introduction

Objectives

The objective of this text is to help the reader to understand some of the many methods for solving scientific problems using computers. We intentionally limit ourselves to the typical problems that arise in science, engineering, and technology. Thus, we do *not* touch on problems of accounting, modeling in the social sciences, information retrieval, artificial intelligence, and so on.

Usually, our treatment of problems do *not* begin at the source, for that would take us far afield into such areas as physics, engineering, and chemistry. Instead, we consider problems after they have been cast into certain standard mathematical forms. The reader is therefore asked to accept on faith the assertion that the chosen topics are indeed important ones in **scientific computing**.

To survey many topics, we must treat some in a superficial way. But it is hoped that the reader acquires a good bird's-eye view of the subject and therefore is better prepared for a further, deeper study of **numerical analysis**.

For each principal topic, we list good current sources for more information. In any realistic computing situation, considerable thought should be given to the choice of method to be employed. Although most procedures presented here are useful and important, they may not be the optimum ones for a particular problem. In choosing among available methods for solving a problem, the analyst or programmer should consult recent references.

Limitations

Becoming familiar with basic numerical methods without realizing their limitations would be foolhardy. Numerical computations are almost invariably contaminated by errors, and it is important to understand the source, propagation, magnitude, and rate of growth of these errors. Numerical methods that provide approximations *and* error estimates are more valuable than those that provide only approximate answers. While we cannot help but be impressed by the speed and accuracy of the modern computer, we should temper our admiration with generous measures of skepticism. As the eminent numerical analyst Carl-Erik Fröberg once remarked:

> *Never in the history of mankind has it been possible to produce so many wrong answers so quickly!*

Thus, one of our goals is to help the reader arrive at this state of skepticism, armed with methods for detecting, estimating, and controlling errors.

Programming

The reader is expected to be familiar with the rudiments of programming. Algorithms are presented as **pseudocode**, and *no* particular programming language is adopted. The pseudocodes are an important intermediate step in translating the algorithms presented in the textbook before coding, running, and debugging them in a computer language and on a computer.

Some of the primary issues related to numerical methods are the nature of numerical errors, the propagation of errors, and the efficiency of the computations involved, as well as the number of operations and their possible reduction.

Many students have graphing calculators and access to mathematical software systems that can produce solutions to complicated numerical problems with minimal difficulty. The purpose of a numerical mathematics course is to examine the underlying algorithmic techniques so that students learn how the software or calculator found the answer. In this way, they would have a better understanding of the inherent limits on the accuracy that must be anticipated in working with such systems.

Strategies

One of the fundamental strategies behind many numerical methods is the replacement of a difficult problem with a string of simpler ones. By carrying out an iterative process, the solutions of the simpler problems can be put together to obtain the solution of the original, difficult problem. This strategy succeeds in solving linear systems (Chapters 2 and 8), finding zeros of functions (Chapter 3), interpolation (Chapter 4), numerical integration (Chapters 5), and more.

Students majoring in computer science and mathematics as well as those majoring in engineering and other sciences are usually well aware that numerical methods are needed to solve problems that they frequently encounter. It may *not* be as well recognized that scientific computing is quite important for solving problems that come from fields other than engineering and science, such as economics. For example, finding zeros of functions may arise in problems using the formulas for loans, interest, and payment schedules. Also, problems in areas such as those involving the stock market may require least-squares solutions (Chapter 9). In fact, the field of computational finance requires solving quite complex mathematical problems utilizing a great deal of computing power. Economic models routinely require the analysis of linear systems of equations with thousands of unknowns.

Significant Digits of Precision: Examples

Significant digits are digits beginning with the leftmost *nonzero* digit and ending with the rightmost *correct* digit, including final zeros that are exact.

EXAMPLE 1 Using an industrial laser cutting machine, a technician cuts a 2-meter by 3-meter rectangular sheet of steel into two equal triangular pieces.
What is the diagonal measurement of each triangle?
Can these pieces be slightly modified so the diagonals are exactly 3.6 meters?

Solution Since the piece is rectangular, the Pythagorean Theorem can be invoked. Thus, to compute the diagonal, d, in Figure 1.1 (p. 4), we write $2^2 + 3^2 = d^2$. It follows that

$$d = \sqrt{4 + 9} = \sqrt{13} = 3.60555\ 1275$$

This last number is obtained by using a handheld calculator. The accuracy of d as given can be verified by computing $(3.60555\ 1275) * (3.60555\ 1275) = 13$.

Is this numerical value for the diagonal to be taken seriously? Certainly not. To begin with, the given dimensions of the rectangle cannot be expected to be precisely 2 and 3. If the dimensions are accurate to 1 millimeter, the dimensions may be as large as 2.001 and 3.001.

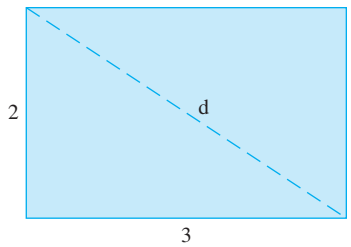


FIGURE 1.1
Rectangular sheet of steel

Using the Pythagorean Theorem again, one finds that the diagonal may be as large as

$$d = \sqrt{2.001^2 + 3.001^2} = \sqrt{4.00400\,1 + 9.00600\,1} = \sqrt{13.01002} \approx 3.6069$$

Similar reasoning indicates that d may be as small as 3.6042. These are both *worst cases*. We can conclude that

$$3.6042 \leqq d \leqq 3.6069$$

No greater accuracy can be claimed for the diagonal.

If we want the diagonal to be exactly 3.6, we require

$$(3 - c)^2 + (2 - c)^2 = 3.6^2$$

For simplicity, we reduce each side by the same amount. This leads to the equation

$$c^2 - 5c + 0.02 = 0$$

Using the quadratic formula, we find the smaller root to be

$$c = 2.5 - \sqrt{6.23} \approx 0.00400$$

By cutting off 4 millimeters from the two perpendicular sides, we have triangular pieces of sizes 1.996 by 2.996 meters. **Check:** $(1.996)^2 + (2.996)^2 \approx 3.6^2$. ■

To show the effect of the number of significant digits used in a calculation, we consider the problem of solving a linear system of equations.

EXAMPLE 2 Let's concentrate on solving for the variable y in this **linear system of equations** in two variables

$$\begin{cases} 0.1036\,x + 0.2122\,y = 0.7381 \\ 0.2081\,x + 0.4247\,y = 0.9327 \end{cases} \tag{1}$$

First, carry only three significant digits of precision in the calculations. Second, repeat with four significant digits throughout. Finally, use ten significant digits.

Solution **Step 1.** In the first task, we round all numbers in the original problem to three digits and round all the calculations, keeping only **three significant digits**. We take a multiple α of the first equation and subtract it from the second equation to eliminate the x-term in the second equation. The **multiplier** is $\alpha = 0.208/0.104 \approx 2.00$. Thus, in the second equation, the new coefficient of the x-term is $0.208 - (2.00)(0.104) \approx 0.208 - 0.208 = 0$ and the new y-term coefficient is $0.425 - (2.00)(0.212) \approx 0.425 - 0.424 = 0.001$. The right-hand side is $0.933 - (2.00)(0.738) = 0.933 - 1.48 = -0.547$. Hence, we find that

$$y = -0.547/(0.001) \approx -547$$

Step 2. We decide to keep **four significant digits** throughout and repeat the calculations. Now the multiplier is $\alpha = 0.2081/0.1036 \approx 2.009$. In the second equation, the new coefficient of the x-term is $0.2081 - (2.009)(0.1036) \approx 0.2081 - 0.2081 = 0$, the new coefficient of the y-term is $0.4247 - (2.009)(0.2122) \approx 0.4247 - 0.4263 = -0.00160\,0$, and the new right-hand side is $0.9327 - (2.009)(0.7381) \approx 0.9327 - 1.483 \approx -0.5503$. Hence, we find

$$y = -0.5503/(-.00160\,0) \approx 343.9$$

We are shocked to find that the *answer* has changed from -547 to 343.9, which is a huge difference!

Step 3. In fact, if we repeat this process and carry **ten significant digits**, we find that even 343.9 is *not* accurate, since we obtain $y \approx 356.29071\,99$.

Using a computer, we find

$$y \approx 3.562907442151324 \times 10^2$$

The lesson learned in this example is that data thought to be accurate should be carried with full precision and *not* be rounded prior to each of the calculations. ■

Figure 1.2 shows a geometric illustration of what can happen in solving two equations in two unknowns. The point of intersection of the two lines is the exact solution. As is shown by the dotted lines, there may be a **degree of uncertainty** from errors in the measurements or roundoff errors. So instead of a sharply defined point, there may be a small trapezoidal area containing many possible solutions. However, if the two lines are nearly parallel, then this area of possible solutions can increase dramatically! This is related to well-conditioned and ill-conditioned systems of linear equations, which are discussed more in Chapter 8.

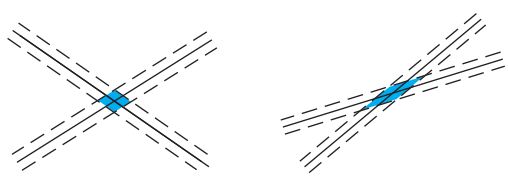


FIGURE 1.2
In 2D, well-conditioned and ill-conditioned linear systems

In most computers, the arithmetic operations are carried out in a double-length accumulator that has twice the precision of the stored quantities. However, even this may not avoid a loss of accuracy! Loss of accuracy can happen in various ways such as from roundoff errors and subtracting nearly equal numbers. We shall discuss loss of precision in Section 1.4, and the solving of linear systems in Chapter 2.

Errors: Absolute and Relative

Suppose that α and β are two numbers, of which one is regarded as an approximation to the other. The **error** of β as an approximation to α is $\alpha - \beta$; that is, the error equals the exact value minus the approximate value. The **absolute error** of β as an approximation to α is

$$|\alpha - \beta|$$

The **relative error** of β as an approximation to α is

$$\frac{|\alpha - \beta|}{|\alpha|}$$

Notice that in computing the absolute error, the roles of α and β are the same, whereas in computing the relative error, it is essential to distinguish one of the two numbers as *correct*. (Observe that the relative error is undefined in the case $\alpha = 0$.) For practical reasons, the relative error is usually more meaningful than the absolute error.

In summary, we have

Errors

$$\text{Absolute Error} = |\text{Exact Value} - \text{Approximate Value}|$$

$$\text{Relative Error} = \frac{|\text{Exact Value} - \text{Approximate Value}|}{|\text{Exact Value}|}$$

Here the exact value may be the *true* value or the *best* known value.

EXAMPLE 3 Let $\alpha_1 = 1.333$, $\beta_1 = 1.334$, and $\alpha_2 = 0.001$, $\beta_2 = 0.002$.
What are the absolute errors and relative errors of β_i as an approximation to α_i?

Solution The absolute error of β_i as an approximation to α_i is the same in both cases—namely, 10^{-3}. However, the relative errors are $\frac{3}{4} \times 10^{-3}$ and 1, respectively. The relative error clearly indicates that β_1 is a *good* approximation to α_1, but that β_2 is a *poor* approximation to α_2. ■

EXAMPLE 4 Consider $x = 0.00347$ rounded to $\widetilde{x} = 0.0035$ and $y = 30.158$ rounded to $\widehat{y} = 30.16$. In each case, what are the number of significant digits, absolute errors, and relative errors? Interpret the results.

Solution **Case 1.** $\widetilde{x} = 0.35 \times 10^{-2}$ has two significant digits, absolute error 0.3×10^{-4} and relative error 0.865×10^{-2}.

Case 2. $\widehat{y} = 0.3016 \times 10^{2}$ has four significant digits, absolute error 0.2×10^{-2} and relative error 0.66×10^{-4}.

Clearly, the relative error is a better indication of the number of significant digits than the absolute error. ■

Accuracy and Precision

Accurate to n decimal places means that you can trust n digits to the right of the decimal place. **Accurate to n significant digits** means that you can trust a total of n digits as being meaningful beginning with the leftmost nonzero digit.

Suppose you use a ruler graduated in millimeters to measure lengths. So the measurements are accurate to 1 millimeter, or 0.001m, which is three decimal places written in meters. A measurement such as 12.345m would be accurate to three decimal places. A measurement such as 12.34567 89m would be meaningless, since the ruler produces only three decimal places, and it should be 12.345m or 12.346m. If the measurement 12.345m has five dependable digits, then it is accurate to five significant figures. On the other hand, a measurement such as 0.076m has only two significant figures.

When using a calculator or computer in a laboratory experiment, one may get a *false sense* of having higher precision than is warranted by the data. For example, the result

$$(1.2) + (3.45) = 4.65$$

actually has only two significant digits of accuracy because the second digit in 1.2 may be the effect of rounding 1.24 down or rounding 1.16 up to two significant figures. Then the

left-hand side could be as large as

$$(1.249) + (3.454) = (4.703)$$

or as small as

$$(1.16) + (3.449) = (4.609)$$

There are really only two significant decimal places in the answer! In adding and subtracting numbers, the result is accurate only to the *smallest* number of significant digits used in any step of the calculation. In the above example, the term 1.2 has two significant digits; therefore, the final calculation has an *uncertainty* in the third digit.

In multiplication and division of numbers, the results may be even more *misleading*. For instance, perform these computations on a calculator:

$$(1.23)(4.5) = 5.535, \qquad (1.23)/(4.5) = 0.27333\,3333$$

You think that there are four and nine significant digits in the results, but there are really only two! As a rule of thumb, one should keep as many significant digits in a sequence of calculations as there are in the least accurate number involved in the computations.

Rounding and Chopping

Rounding reduces the number of significant digits in a number. The result of rounding is a number similar in magnitude that is a *shorter* number having fewer nonzero digits. There are several slightly different rules for rounding. The **round-to-even** method is also known as *statistician's rounding* or *bankers' rounding*. It is discussed next. Over a large set of data, the round-to-even rule tends to reduce the total rounding error with (on average) an equal portion of numbers rounding up as well as rounding down.

We say that a number x is **chopped to n digits** or figures when all digits that follow the nth digit are discarded and none of the remaining n digits are changed. Conversely, x is **rounded to n digits** or figures when x is replaced by an n-digit number that approximates x with *minimum* error.

The question of whether to round up or down an $(n+1)$-digit decimal number that ends with a 5 is best handled by always selecting the rounded n-digit number with an *even* nth digit. This may seem strange at first, but remarkably, this is essentially what computers do in rounding decimal calculations when using the standard floating-point arithmetic! (This is a topic discussed in Section 1.3.)

EXAMPLE 5 Give some examples of rounding three-decimal numbers to two digits.

Solution The results of rounding are

$$0.217 \approx 0.22, \quad 0.365 \approx 0.36, \quad 0.475 \approx 0.48, \quad 0.592 \approx 0.59$$

while chopping them gives

$$0.217 \approx 0.21, \quad 0.365 \approx 0.36, \quad 0.475 \approx 0.47, \quad 0.592 \approx 0.59$$

■

On computers, the user sometimes has the option to have all arithmetic operations done with either chopping or rounding. The latter is usually preferable, of course!

Computation of π

History of Computing π

Computing π has a rich and colorful history. In 1650 BCE, the Rhind Papyrus of ancient Egypt contained numerical algorithms such as this approximation

$$\pi \left(\frac{9}{2}\right)^2 \approx 8^2 \Rightarrow \pi \approx \frac{256}{81} \approx 3.1605$$

In 287–212 BC, Archimedes determined that

$$3.1408 \approx \frac{223}{71} < \pi < \frac{22}{7} \approx 3.142$$

by noting that the value of π is between the length of the permieter of a polygon inscribing and a polygon circumscribing a circle of radius one-half. Around 1700, John Machin dicovered the identity

$$\pi = 16 \tan \frac{1}{5} - 4 \tan \frac{1}{239}$$

and calculated the first hundred digits of π. In 1973, the first million digits of π were determined. Since then, many more sophisticated techniques have been developed for computing π. (See Moler [2011], for a recent article.)

Nested Multiplication and Horner's Algorithm

We begin with some remarks on evaluating a polynomial efficiently and on rounding and chopping real numbers. To evaluate the polynomial

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{n-1} x^{n-1} + a_n x^n \tag{2}$$

we group the terms in a **nested multiplication**:

$$p(x) = a_0 + x(a_1 + x(a_2 + \cdots + x(a_{n-1} + x(a_n)) \cdots))$$

The pseudocode[‡] that evaluates $p(x)$ starts with the innermost parentheses and works outward. It can be written as

Nested Multiplication Pseudocode

```
integer i, n; real p, x
real array (a_i)_{0:n}
p ← a_n
for i = n − 1 to 0
    p ← a_i + xp
end for
```

Here we assume that numerical values have been assigned to the integer variable n, the real variable x, as well as the coefficients $a_0, a_1, \ldots, a_n$, which are stored in a real linear array. (Throughout, we use semicolons between these declarative statements to save space.) The **left-pointing arrow** ($\leftarrow$) means that the value on the right is stored in the location

[‡]A **pseudocode** is a compact and informal description of an algorithm that uses the conventions of a programming language but omits the detailed syntax. When convenient, it may be augmented with natural language. Usually, writing the pseudocode is a good way of organizing the ideas in a mathematical algorithm before coding them in a particular programming language.

named on the left (i.e., **overwrites** from right to left). The for-loop index i runs backward, taking values $n-1, n-2, \ldots, 0$. The final value of p is the value of the polynomial at x. This nested multiplication procedure is also known as **Horner's algorithm** or **synthetic division**.

In the pseudocode (p. 8), there is exactly *one* addition and *one* multiplication each time the loop is traversed. Consequently, Horner's algorithm can evaluate a polynomial with only n additions and n multiplications. This is the minimum number of operations possible.

A *naive* method of evaluating a polynomial would require many more operations.

EXAMPLE 6 Show how $p(x) = 5 + 3x - 7x^2 + 2x^3$ should be computed.

Solution Let

$$p(x) = 5 + x(3 + x(-7 + x(2)))$$

for a given value of x. We have *avoided* all the exponentiation operations by using nested multiplication! ■

The polynomial in Equation (2) can be written in an alternative form by utilizing the mathematical symbols for **sum** $\sum$ and **product** $\prod$; namely,

Mathematical Notation

$$p(x) = \sum_{i=0}^{n} a_i x^i = \sum_{i=0}^{n} \left(a_i \prod_{j=1}^{i} x \right)$$

Recall that if $n \leqq m$, we write

$$\sum_{k=n}^{m} x_k = x_n + x_{n+1} + \cdots + x_m$$

and

$$\prod_{k=n}^{m} x_k = x_n x_{n+1} \cdots x_m$$

By convention, whenever $m < n$, we define

$$\sum_{k=n}^{m} x_k = 0 \quad \text{and} \quad \prod_{k=n}^{m} x_k = 1$$

Horner's algorithm can be used in the **deflation** of a polynomial. This is the process of removing a linear factor from a polynomial. If r is a **root** of the polynomial p, then $x - r$ is a factor of p. The remaining roots of p are the $n-1$ roots of a polynomial q of degree 1 less than the degree of p such that

Deflation

$$p(x) = (x - r)q(x) + p(r) \tag{3}$$

where

$$q(x) = b_0 + b_1 x + b_2 x^2 + \cdots + b_{n-1} x^{n-1} \tag{4}$$

The pseudocode for **Horner's algorithm** can be written as follows:

Synthetic Division Pseudocode

```
integer i, n; real r
real array (a_i)_{0:n}, (b_i)_{0:n-1}
b_{n-1} ← a_n
for i = n − 1 to 0
    b_{i-1} ← a_i + r b_i
end for
```

Notice that $b_{-1} = p(r)$ in this pseudocode. If f is an exact root, then $b_{-1} = p(r) = 0$.

If the calculation in Horner's algorithm is to be carried out with pencil and paper, the following arrangement is often used:

$$\begin{array}{cccccc} & a_n & a_{n-1} & a_{n-2} & \dots & a_1 & a_0 \\ r\,) & & rb_{n-1} & rb_{n-2} & \dots & rb_1 & rb_0 \\ \hline & b_{n-1} & b_{n-2} & b_{n-3} & \dots & b_0 & b_{-1} \end{array}$$

EXAMPLE 7 Use Horner's algorithm to evaluate $p(3)$, where p is the polynomial

$$p(x) = x^4 - 4x^3 + 7x^2 - 5x - 2$$

Solution We arrange the calculation as suggested:

$$\begin{array}{cccccc} & 1 & -4 & 7 & -5 & -2 \\ 3\,) & & 3 & -3 & 12 & 21 \\ \hline & 1 & -1 & 4 & 7 & \boxed{19} \end{array}$$

Thus, we obtain $p(3) = 19$, and we can write

$$p(x) = (x - 3)(x^3 - x^2 + 4x + 7) + 19$$ ■

In the **deflation process**, if r is a **zero** of the polynomial p, then $x - r$ is a factor of p, and conversely. The remaining zeros of p are the $n - 1$ zeros of $q(x)$.

EXAMPLE 8 Deflate the polynomial p of the preceding example, using the fact that 2 is one of its zeros.

Solution We use the same arrangement of computations as explained previously:

$$\begin{array}{cccccc} & 1 & -4 & 7 & -5 & -2 \\ 2\,) & & 2 & -4 & 6 & 2 \\ \hline & 1 & -2 & 3 & 1 & \boxed{0} \end{array}$$

Thus, we have $p(2) = 0$, and

$$x^4 - 4x^3 + 7x^2 - 5x - 2 = (x - 2)(x^3 - 2x^2 + 3x + 1)$$ ■

We can use Horner's algorithm for evaluating a derivative of a polynomial. By Equation (3), we can write polynomial p with root r as

$$p(x) = (x - r)q(x) + p(r)$$

where $q(x)$ is given by Equation (4). If we differentiate, we obtain

$$p'(x) = q(x) + (x - r)q'(x)$$

Clearly, we have

$$p'(r) = q(r)$$

The pseudocode for **Horner's algorithm** for determining $p(r)$ and $p'(r)$ can be written as follows:

Horner's Algorithm Pseudocode

```
integer i, n; real p, r
real array (a_i)_{0:n}, (b_i)_{0:n-1}
α ← a_n; β ← 0
for i = n − 1 to 0
    β ← α + rβ
    α ← a_i + rα
end for
```

Notice that the final values are $\alpha = p(r)$ and $\beta = p'(r)$.

If the calculation in Horner's algorithm (synthetic division) is to be carried out with pencil and paper, the following arrangement is often used:

$$\begin{array}{cccccccc}
 & a_n & a_{n-1} & a_{n-2} & \dots & a_2 & a_1 & a_0 \\
r\,) & & rb_{n-1} & rb_{n-2} & \dots & rb_2 & rb_1 & rb_0 \\
\hline
 & b_{n-1} & b_{n-2} & b_{n-3} & \dots & b_1 & b_0 & \boxed{b_{-1}} = p(r) \\
 & & rc_{n-2} & rc_{n-3} & \dots & c_2 & rc_1 & \\
\hline
 & c_{n-2} & c_{n-3} & c_{n-4} & \dots & c_0 & \boxed{c_{-1}} = p'(r) &
\end{array}$$

EXAMPLE 9 Given the polynomial

$$p(x) = x^4 - 4x^3 + 7x^2 - 5x - 3$$

Use synthetic division to find $p(3)$ and $p'(3)$.

Solution We use this arrangement of computation as explained previously:

$$\begin{array}{ccccccc}
 & 1 & -4 & 7 & -5 & -2 & \\
3\,) & & 3 & -3 & 12 & 21 & \\
\hline
 & 1 & -1 & 4 & 7 & \boxed{19} & = p(3) \\
 & & 3 & 6 & 30 & & \\
\hline
 & 1 & 2 & 10 & \boxed{37} & & = p'(3)
\end{array}$$

Thus, we have

$$x^4 - 4x^3 + 7x^2 - 5x - 2 = (x - 3)(x^3 - x^2 + 4x + 7) + 19$$

So $p(x) = (x-3)q(x)+19$ and $p'(x) = q(x)+(x-3)q'(x)$ where $q(x) = x^3-x^2+4x+7$. Hence, we have $p(3) = 19$ and $p'(3) = q(3) = 37$. ■

Pairs of Easy/Hard Problems

Sometimes in scientific computing, we encounter a pair of problems, one of which is *easy* and the other *hard*, and they are inverses of each other. This is the main idea in **cryptology**, in which multiplying two numbers together is trivial, but the reverse problem (factoring a huge number) verges on the impossible!

The same phenomenon arises with **polynomials**. Given the roots, we can easily find the power form of the polynomial as in (2). Given the power form, it may be a hard problem

to compute the roots (or it may be an *ill-conditioned problem*). Computer Exercise 1.1.24 calls for the writing of code to compute the coefficients in the power form of a polynomial from its roots. It is a do-loop with simple formulas. One adjoins one factor $(x - r)$ at a time. This theme arises again in linear algebra, in which computing $\boldsymbol{b} = \boldsymbol{Ax}$ is trivial, but finding $\boldsymbol{x}$ from $\boldsymbol{A}$ and $\boldsymbol{b}$ (the inverse problem) is hard. (See Section 2.1.)

Sample Problem Pairs

Easy/hard problems come up again in **two-point boundary value problems**. Finding Df, $f(0)$, and $f(1)$ when f is given and D is a differential operator is easy, but finding f from knowledge of Df, $f(0)$ and $f(1)$ is hard. (See Section 9.1.)

Likewise, computing the **eigenvalues/eigenvectors** of a matrix is a hard problem. Suppose we are given the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ of an $n \times n$ matrix and corresponding eigenvectors $\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_n$ of an $n \times n$ matrix. We can get $\boldsymbol{A}$ by putting the eigenvalues on the diagonal of a diagonal matrix $\boldsymbol{D}$ and the eigenvectors as columns in a matrix $\boldsymbol{V}$. Then $\boldsymbol{AV} = \boldsymbol{VD}$, and we can get $\boldsymbol{A}$ from this by solving the equation for $\boldsymbol{A}$. But finding λ_i and $\boldsymbol{v}_i$ from $\boldsymbol{A}$ itself is hard. (See Section 8.2.)

The reader may think of other examples.

It needs to be pointed out that there may be a huge gap between the *complexity* of factoring integers and that of the other problems mentioned, which are probably all solvable in *polynomial time*. Also, there may be a cultural difference at work here. To a numerical analyst, *hard* usually means: "I can solve it in polynomial time, but the degree of the polynomial is too high," whereas to a complexity theorist, *hard* means: "It is not solvable in polynomial time, as far as I know." On the other hand, there are NP-hard problems with a numerical flavor such as the *traveling salesman problem*.

First Programming Experiment

We conclude this section with a short programming experiment involving numerical computations. Here we consider, from the computational point of view, a familiar operation in calculus—namely, taking the derivative of a function. Recall that the **derivative** of a function f at a point x is defined by the equation

Derivative Formula

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

A computer has the capacity of imitating the limit operation by using a sequence of numbers h such as

$$h = 4^{-1}, 4^{-2}, 4^{-3}, \ldots, 4^{-n}, \ldots$$

This sequence certainly approaches zero rapidly! Of course, many other simple sequences are possible, such as $1/n$, $1/n^2$, and $1/10^n$. The sequence $1/4^n$ consists of machine numbers in a binary computer and, for this experiment on a 32-bit computer, it is sufficiently close to zero when n is 10.

If $f(x) = \sin x$, here is pseudocode for computing $f'(x)$ at the point $x = 0.5$:

***First* Pseudocode**

```
program First
integer i, imin, n ← 30
real error, y, x ← 0.5, h ← 1, emin ← 1
for i = 1 to n
    h ← 0.25h
    y ← [sin(x + h) − sin(x)]/h
```

(*Continued*)

```
        error ← |cos(x) − y|;
        output i, h, y, error
        if error < emin then
           emin ← error;   imin ← i
        end if
    end for
    output imin, emin
    end program First
```

We have neither explained the purpose of the experiment nor shown the output from this pseudocode. We invite the reader to discover this by coding and running it (or one like it) on a computer. (See Computer Exercises 1.1.1—1.1.3.)

Mathematical Software

The algorithms and programming problems in this book have been coded and tested in a variety of ways, and they are available on the website for this book as given in the Preface. Some are best done by using a scientific programming language such as C, C++, Fortran, or any other that allows for calculations with adequate precision. Sometimes it is instructive to utilize mathematical software systems such as MATLAB, Maple, Mathematica, or Octave, since they contain built-in problem-solving procedures. Alternatively, one could use a mathematical program library such as IMSL, NAG, or others when locally available. Some numerical libraries may have been specifically optimized for the processor, such as Intel and AMD. Software systems are particularly useful for obtaining graphical results as well as for experimenting with various numerical methods for solving a difficult problem. Mathematical software packages containing symbolic-manipulation capabilities, such as in Maple, Mathematica, and Macsyma, are particularly useful for obtaining exact as well as numerical solutions. In solving the computer problems, students should focus on gaining insights and better understanding of the numerical methods involved. Appendix A offers advice on computer programming for scientific computations. The suggestions are independent of the particular language being used.

Websites

With World Wide Web and the Internet, good mathematical software has become easy to locate and to use. Browsers, search engines, and websites may be used to find software that is applicable to a particular area of interest. Collections of mathematical software exist, ranging from large, comprehensive libraries to smaller versions of these libraries for personal computers; some of these may be interactive. Also, references to computer programs and collections of routines can be found in books and technical reports. The textbook website is given in the Preface. It contains an overview of mathematical software and other available supporting material.

Additional Reading

For additional study of topics found in this book, see the references in the Bibliography as well as an extensive list of items at the textbook website. Two interesting papers containing numerous examples of why numerical methods are critically important are Forsythe [1970]

and McCartin [1998]. See Briggs [2004] and Friedman and Littman [1994] for some industrial and real-world problems.

Summary 1.1

- Absolute Error = |Exact Value − Approximate Value|

  $$\text{Relative Error} = \frac{|\text{Exact Value} - \text{Approximate Value}|}{|\text{Exact Value}|}$$

- Use **nested multiplication** to evaluate a polynomial efficiently:

  $$\begin{aligned} p(x) &= a_0 + a_1x + a_2x^2 + \cdots + a_{n-1}x^{n-1} + a_nx^n \\ &= a_0 + x(a_1 + x(a_2 + \cdots + x(a_{n-1} + x(a_n))\cdots)) \end{aligned}$$

  A segment of pseudocode for doing this is

  ```
  p ← a_n
  for k = 1 to n
      p ← xp + a_{n−k}
  end for
  ```

- Deflation of the polynomial $p(x)$ is removing a linear factor:

  $$p(x) = (x - r)q(x) + p(r)$$

  where

  $$q(x) = b_0 + b_1x + b_2x^2 + \cdots + b_{n-1}x^{n-1}$$

  The pseudocode for **Horner's algorithm** for deflation of a polynomial is:

  ```
  b_{n−1} ← a_n
  for i = n − 1 to 0
      b_{i−1} ← a_i + r b_i
  end for
  ```

  Hence, we obtain $b_{-1} = p(r)$.

Exercises 1.1

1. In high school, some students have been misled to believe that $22/7$ is either the actual value of π or an acceptable approximation to π. Show that $355/113$ is a better approximation in terms of both absolute and relative errors. Find some other simple rational fractions n/m that approximate π. For example, ones for which

$$|\pi - n/m| < 10^{-9}$$

Hint: See Exercise 1.1.4.

*Exercises marked with [a] have answers in the back of the book.

[a]**2.** A real number x is represented approximately by 0.6032, and we are told that the relative error is at most 0.1%. What is x?
Note: There are *two* answers.

[a]**3.** What is the relative error involved in rounding 4.9997 to 5.000?

[a]**4.** The value of π can be generated by the computer to nearly full machine precision by the assignment statement

$pi \leftarrow 4.0 \arctan(1.0)$

Suggest at least four other ways to compute π using basic functions on your computer system.

5. A given doubly subscripted array $(a_{ij})_{n\times n}$ can be added in any order. Write the pseudocode segments for each of the following parts. Which is best?

[a]**a.** $\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}$ **b.** $\sum_{j=1}^{n}\sum_{i=1}^{n} a_{ij}$

c. $\sum_{i=1}^{n}\left(\sum_{j=1}^{i} a_{ij} + \sum_{j=1}^{i-1} a_{ji}\right)$

[a]**d.** $\sum_{k=0}^{n-1}\sum_{|i-j|=k} a_{ij}$ **e.** $\sum_{k=2}^{2n}\sum_{i+j=k}^{n} a_{ij}$

[a]**6.** Count the number of operations involved in evaluating a polynomial using nested multiplication. Do *not* count subscript calculations.

7. For small x, show that $(1+x)^2$ can sometimes be more accurately computed from $(x+2)x+1$. Explain. What other expressions can be used to compute it?

8. Show how these polynomials can be efficiently evaluated:

[a]**a.** $p(x) = x^{32}$ **b.** $p(x) = 3(x-1)^5 + 7(x-1)^9$

[a]**c.** $p(x) = 6(x+2)^3 + 9(x+2)^7 + 3(x+2)^{15} - (x+2)^{31}$

d. $p(x) = x^{127} - 5x^{37} + 10x^{17} - 3x^7$

9. Using the exponential function $\exp(x)$, write an efficient pseudocode segment for the statement $y = 5e^{3x} + 7e^{2x} + 9e^{x} + 11$.

[a]**10.** Write a pseudocode segment to evaluate the expression

$$z = \sum_{i=1}^{n} b_i^{-1} \prod_{j=1}^{i} a_j$$

where $(a_1, a_2, \ldots, a_n)$ and $(b_1, b_2, \ldots, b_n)$ are linear arrays containing given values.

11. Write segments of pseudocode to evaluate the following expressions efficiently:

a. $p(x) = \sum_{k=0}^{n-1} kx^k$

[a]**b.** $z = \sum_{i=1}^{n}\prod_{j=1}^{i} x^{n-j+1}$

c. $z = \prod_{i=1}^{n}\sum_{j=1}^{i} x_j$

d. $p(t) = \sum_{i=1}^{n} a_i \prod_{j=1}^{i-1}(t - x_j)$

12. Using summation and product notation, write mathematical expressions for the following pseudocode segments:

a.
```
integer i, n; real v, x
real array (a_i)_{0:n}
v ← a_0
for i = 1 to n
    v ← v + x a_i
end for
```

[a]**b.**
```
integer i, n; real v, x
real array (a_i)_{0:n}
v ← a_n
for i = 1 to n
    v ← v x + a_{n-i}
end for
```

c.
```
integer i, n; real v, x
real array (a_i)_{0:n}
v ← a_0
for i = 1 to n
    v ← v x + a_i
end for
```

d.
```
integer i, n; real v, x, z
real array (a_i)_{0:n}
v ← a_0
z ← x
for i = 1 to n
    v ← v + z a_i
    z ← x z
end for
```

[a]**e.**
```
integer i, n; real v
real array (a_i)_{0:n}
v ← a_n
for i = 1 to n
    v ← (v + a_{n-i}) x
end for
```

[a]**13.** Express in mathematical notation *without* parentheses the final value of z in the following pseudocode segment:

```
integer k, n; real z
real array (b_i)_{0:n}
z ← b_n + 1
for k = 1 to n − 2
    z ← z b_{n-k} + 1
end for
```

[a] **14.** How many multiplications occur in executing the following pseudocode segment?

```
integer i, j, n; real x
real array (a_ij)_{0:n×0:n}, (b_ij)_{0:n×0:n}
x ← 0.0
for j = 1 to n
    for i = 1 to j
        x ← x + a_ij b_ij
    end for
end for
```

15. Criticize the following pseudocode segments and write improved versions:

a.
```
integer i, n; real x, z; real array (a_i)_{0:n}
for i = 1 to n
    x ← z^2 + 5.7
    a_i ← x/i
end for
```

[a] **b.**
```
integer i, j, n
real array (a_ij)_{0:n×0:n}
for i = 1 to n
    for j = 1 to n
        a_ij ← 1/(i + j − 1)
    end for
end for
```

c.
```
integer i, j, n; real array (a_ij)_{0:n×0:n}
for j = 1 to n
    for i = 1 to n
        a_ij ← 1/(i + j − 1)
    end for
end for
```

16. **a.** Solve Example 1.1.2 to full precision.

b. Repeat for this augmented matrix

$$\left[\begin{array}{ll|l} 3.5713 & 2.1426 & 7.2158 \\ 10.714 & 6.4280 & 1.3379 \end{array}\right]$$

for a system of two equations and two unknowns x and y.

c. Can small changes in the data lead to massive change in the each of these solutions?

17. A base 60 approximation (circa 1750 BCE) is

$$\sqrt{2} \approx 1 + \frac{24}{60} + \frac{51}{60^2} + \frac{10}{60^3}$$

Determine how accurate it is. See Sauer [2006] for additional details.

18. Use Horner's algorithm to evaluate each of these polynomials at the point indicated.

a. $2x^4 + 9x^2 - 16x + 12$ at -6

b. $2x^4 - 3x^3 - 5x^2 + 3x + 8$ at 2

c. $3x^5 - 38x^3 + 5x^2 - 1$ at 4

Computer Exercises 1.1

1. Write and run a computer program that corresponds to the pseudocode program *First* described on pp. 12–13 and interpret the results.

2. (Continuation) Select a function f and a point x and carry out a computer experiment like the one given in the text. Interpret the results. Do not select too simple a function. For example, you might consider $1/x$, $\log x$, e^x, $\tan x$, $\cosh x$, or $x^3 - 23x$.

3. (Continuation) As we saw in the computer experiment *First*, the accuracy of a formula for numerical differentiation may deteriorate as the step-size h decreases. Study the following **central difference formula**:

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

as $h \to 0$. We learn in Section 4.3 that the **truncation error** for this formula is $-\frac{1}{6}h^2 f'''(\xi)$ for some ξ in the interval $(x-h, x+h)$.

Modify and run the code for the experiment *First* so that approximate values for the rounding error and truncation error are computed. On the same graph, plot the rounding error, the truncation error, and the total error (sum of these two errors) using a log-scale; that is, the axes in the plot should be $-\log_{10}|\text{error}|$ versus $\log_{10} h$. Analyze these results.

[a] **4.** The limit

$$e = \lim_{n\to\infty} \left(1 + \frac{1}{n}\right)^n$$

defines the number e in calculus. Estimate e by taking the value of this expression for $n = 8, 8^2, 8^3, \ldots, 8^{10}$.

Compare with e obtained from $e \leftarrow \exp(1.0)$. Interpret the results.

5. It is not difficult to see that the numbers

$$p_n = \int_0^1 x^n e^x \, dx$$

satisfy the inequalities $p_1 > p_2 > p_3 > \cdots > 0$. Establish this fact. Next, use integration by parts to show that

$$p_{n+1} = e - (n+1)p_n$$

and that $p_1 = 1$. In the computer, use the recurrence relation to generate the first 20 values of p_n and explain why the inequalities shown are violated. Do not use subscripted variables. (See Dorn and McCracken [1972], pp. 120–129.)

6. (Continuation) Let $p_{20} = \frac{1}{8}$ and use the formula in the preceding computer problem to compute $p_{19}, p_{18}, \ldots, p_2$, and p_1. Do the numbers generated obey the inequalities $1 = p_1 > p_2 > p_3 > \cdots > 0$? Explain the difference in the two procedures. Repeat with $p_{20} = 20$ or $p_{20} = 100$. Explain what happens.

7. Write an efficient routine that accepts as input a list of real numbers $a_1, a_2, \ldots, a_n$ and then computes the following:

Arithmetic mean $\quad m = \dfrac{1}{n}\displaystyle\sum_{k=1}^{n} a_k$

Variance $\quad v = \dfrac{1}{n-1}\displaystyle\sum_{k=1}^{n}(a_k - m)^2$

Standard deviation $\quad \sigma = \sqrt{v}$

Test the routine on a set of data of your choice.

8. (Continuation) Show that another formula is

Variance $\quad v = \dfrac{1}{n-1}\left[\displaystyle\sum_{k=1}^{n} a_k^2 - nm^2\right]$

Of the two given formulas for v, which is more accurate in the computer? Verify on the computer with a data set. *Hint*: Use a large set of real numbers that vary in magnitude from very small to very large.

[a] **9.** Let a_1 be given. Write a program to compute for $1 \leqq n \leqq 1000$ the numbers

$$b_n = na_{n-1}, \qquad a_n = b_n/n$$

Print the numbers $a_{100}, a_{200}, \ldots, a_{1000}$. *Do not* use subscripted variables. What should a_n be? Account for the deviation of fact from theory. Determine four values for a_1 so that the computation does deviate from theory on your computer.

Hint: Consider extremely small and large numbers and print to full machine precision.

[a] **10.** In a computer, it can happen that $a + x = a$ when $x \neq 0$. Explain why. Describe the set of n for which $1 + 2^{-n} = 1$ in your computer. Write and run appropriate programs to illustrate the phenomenon.

11. Write a program to test the programming suggestion concerning the roundoff error in the computation of $t \leftarrow t+h$ versus $t \leftarrow t_0 + ih$. For example, use $h = \frac{1}{10}$ and compute $t \leftarrow t+h$ in double precision for the correct single-precision value of t; print the absolute values of the differences between this calculation and the values of the two procedures. What is the result of the test when h is a machine number, such as $h = \frac{1}{128}$, on a binary computer (with more than seven bits per word)?

[a] **12.** The Russian mathematician P. L. **Chebyshev** (1821–1894) spelled his name Qebywev. Many transliterations from the Cyrillic to the Latin alphabet are possible. *Cheb* can alternatively be rendered as Ceb, Tscheb, or Tcheb. The *y* can be rendered as *i*. *Shev* can also be rendered as schef, cev, cheff, or scheff. Taking all combinations of these variants, program a computer to print all possible spellings.

13. Compute $n!$ using logarithms, integer arithmetic, and double-precision floating-point arithmetic. For each part, print a table of values for $0 \leqq n \leqq 30$, and determine the largest correct value.

14. Given two arrays, a real array $v = (v_1, v_2, \ldots, v_n)$ and an integer permutation array $p = (p_1, p_2, \ldots, p_n)$ of integers $1, 2, \ldots, n$, can we form a new permuted array $v = (v_{p_1}, v_{p_2}, \ldots, v_{p_n})$ by overwriting v and *not* involving another array in memory? If so, write and test the code for doing it. If *not*, use an additional array and test. Consider these cases:

Case 1. $v = (6.3, 4.2, 9.3, 6.7, 7.8, 2.4, 3.8, 9.7)$,
$p = (2, 3, 8, 7, 1, 4, 6, 5)$

Case 2. $v = (0.7, 0.6, 0.1, 0.3, 0.2, 0.5, 0.4)$,
$p = (3, 5, 4, 7, 6, 2, 1)$

15. Using a computer algebra system (e.g., Maple, Mathematica, MATLAB, etc.), print 200 decimal digits of $\sqrt{10}$.

16. **a.** Repeat the Example 1.1.1 on loss of significant digits of accuracy, but perform the calculations with twice the precision before rounding them. Does this help?

b. Use Maple or some other mathematical software system in which you can set the number of digits of precision.
Hint: In Maple, use `Digits`.

17. In 1706, Machin used the formula

$$\pi = 16 \arctan\left(\frac{1}{5}\right) - 4 \arctan\left(\frac{1}{239}\right)$$

to compute 100 digits of π. Derive this formula. Reproduce Machin's calculations by using suitable software. *Hint*: Let $\tan\theta = \frac{1}{5}$, and use standard trigonometric identities.

18. Using a symbol-manipulating program such as Maple, Mathematica, MATLAB, or Macsyma, carry out the following tasks. Record your work in some manner, for example, by using a `diary` or `script` command.

a. Find the Taylor series, up to and including the term x^{10}, for the function $(\tan x)^2$, using 0 as the point x_0.

b. Find the indefinite integral of $(\cos x)^{-4}$.

c. Find the definite integral $\int_0^1 \log|\log x|\, dx$.

d. Find the first prime number greater than 27448.

e. Obtain the numerical value of $\int_0^1 \sqrt{1+\sin^3 x}\, dx$.

f. Find the solution of the differential equation $y' + y = (1+e^x)^{-1}$.

g. Define the function $f(x, y) = 9x^4 - y^4 + 2y^2 - 1$. You want to know the value of $f(40545, 70226)$. Compute this in the straightforward way by direct substitution of $x = 40545$ and $y = 70226$ in the definition of $f(x, y)$, using first 6-decimal accuracy, then 7-, 8-, and so on up to 24-decimal digits of accuracy. Next, prove by means of elementary algebra that

$$f(x, y) = (3x^2 - y^2 + 1)(3x^2 + y^2 - 1)$$

Use this formula to compute the same value of $f(x, y)$, again using different precisions, from 6-decimal to 24-decimal. Describe what you have learned. To force the program to do floating-point operations instead of integer arithmetic, write your numbers in the form 9.0, 40545.0, and so forth.

19. Consider the following pseudocode segments:

a.
```
integer i; real x, y, z
for i = 1 to 20
    x ← 2 + 1.0/8^i
    y ← arctan(x) − arctan(2)
    z ← 8^i y
    output x, y, z
end for
```

b.
```
real epsi ← 1
while 1 < 1 + epsi
    epsi ← epsi/2
    output epsi
end while
```

What is the purpose of each program? Is it achieved? Explain. Code and run each one to verify your conclusions.

20. Consider some oversights involving assignment statements.

[a] **a.** What is the difference between the following two assignment statements? Write a code that contains them and illustrate with specific examples to show that sometimes $x = y$ and sometimes $x \neq y$.

```
integer m, n; real x, y
x ← real(m/n)
y ← real(m)/real(n)
output x, y
```

b. What value does n receive?

```
integer n;   real x, y
x ← 7.4
y ← 3.8
n ← x + y
output n
```

What happens when the last statement is replaced with the following?

```
n ← integer(x) + integer(y)
```

21. Write a computer code that contains the following assignment statements exactly as shown. Analyze the results.

a. Print these values first using the default format and then with an extremely large format field:

```
real p, q, u, v, w, x, y, z
x ← 0.1
y ← 0.01
z ← x − y
p ← 1.0/3.0
q ← 3.0p
u ← 7.6
v ← 2.9
w ← u − v
output x, y, z, p, q, u, v, w
```

b. What values would be computed for x, y, and z if this code is used?

```
integer n; real x, y, z
for n = 1 to 10
    x ← (n − 1)/2
    y ← n^2/3.0
    z ← 1.0 + 1/n
    output x, y, z
end for
```

c. What values would the following assignment statements produce?

```
integer i, j; real c, f, x, half
x ← 10/3
i ← integer(x + 1/2)
half ← 1/2
j ← integer(half)
c ← (5/9)(f − 32)
f ← 9/5c + 32
output x, i, half, j, c, f
```

d. Discuss what is wrong with the following pseudocode segment:

```
real area, circum, radius
radius ← 1
area ← (22/7)(radius)^2
circum ← 2(3.1416)radius
output area, circum
```

22. Criticize the following pseudocode for evaluating $\lim_{x\to 0} \arctan(|x|)/x$. Code and run it to see what happens.

```
integer i; real x, y
x ← 1
for i = 1 to 24
    x ← x/2.0
    y ← arctan(|x|)/x
    output x, y
end for
```

23. Carry out some computer experiments to illustrate or test the programming suggestions in Appendix A. Specific topics to include are these:

a. When to avoid arrays.

b. When to limit iterations.

c. Checking for floating-point equality.

d. Ways for taking equal floating-point steps.

e. Various ways to evaluate functions.

Hint: Comparing single and double precision results may be helpful.

24. (**Easy/Difficult Problem Pairs**) Write a computer program to obtain the power form of a polynomial from its roots. Let the roots be $r_1, r_2, \ldots, r_n$. Then (except for a scalar factor) the polynomial is the product

$$p(x) = (x - r_1)(x - r_2)\cdots(x - r_n).$$

Find the coefficients in the expression

$$p(x) = \sum_{j=0}^{n} a_j x^j$$

Test your code on the **Wilkinson polynomials** in Computer Exercises 3.1.10 and 3.3.9. Explain why this task of getting the power form of the polynomial is *trivial*, whereas the inverse problem of finding the roots from the power form is quite difficult.

25. A **prime number** is a positive integer that has no integer factors other than itself and 1. How many prime numbers are there in each of these open intervals: (1, 40), (1, 80), (1, 160), and (1, 2000)? Make a guess as to the percentage of prime numbers among all numbers.

26. Mathematical software systems such as Maple, Mathematica, and MATLAB are able to do both numerical calculations and symbolic manipulations. Verify symbolically that a nested multiplication is correct for a general polynomial of degree ten.

27. In MATLAB, the `rat` function finds a rational fraction approximation (numerator and denominator) within a certain tolerance to a given floating-point number. For example, `[a,b]=rat(pi, 8000e-6)` return `a=22` and `b=7`. However, the relative error between 19/6 and π is 0.007981306248670 in format `long`, which is less than the tolerance 0.008. What's going on here? In terms of absolute and relative errors, is 19/6 or 22/7 the better approximation to π?

28. Use mathematical software to reproduce the three solutions to Example 1.1.2.
Hint: In MATLAB, use commands `str2nun(num2str(x,4))` for rounding to four significant decimal digits as well as `format long`.

29. Explain the results from coding and executing the following pseudocode using mathematical software such as in MATLAB with `format long`:

```
integer k; real dt, s, t
t ← 0;   s ← 1
dt ← 0.1
for k = 1 to 10
t ← t + dt
s ← s * dt
output k, t, s
end
```

Hint: Print results with a very large number of decimal places.

30. By plotting $\ln x$ and $\ln[(1+x)/(1-x)]$, show that they both contain the point $\ln 2$. Are there other values that match up?

1.2 Mathematical Preliminaries

Most students have encountered infinite series (particularly Taylor series) in their study of calculus without necessarily having acquired a good understanding of this topic. Consequently, this section is particularly important for numerical analysis and deserves careful study.

Once students are well grounded with a basic understanding of Taylor series, the Mean-Value Theorem, and alternating series (all topics in this section) as well as computer number representation (Section 1.3), they can proceed to study the fundamentals of numerical methods with better comprehension. Well-prepared students may wish to skip over some of this material.

Taylor Series

Familiar (and useful) examples of Taylor series are the following:

Common Taylor series

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots = \sum_{k=0}^{\infty} \frac{x^k}{k!} \qquad (|x| < \infty) \quad (1)$$

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots = \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k+1}}{(2k+1)!} \qquad (|x| < \infty) \quad (2)$$

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \cdots = \sum_{k=0}^{\infty} (-1)^k \frac{x^{2k}}{(2k)!} \qquad (|x| < \infty) \quad (3)$$

$$\frac{1}{1-x} = 1 + x + x^2 + x^3 + \cdots = \sum_{k=0}^{\infty} x^k \qquad (|x| < 1) \quad (4)$$

$$\ln(1+x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \cdots = \sum_{k=1}^{\infty} (-1)^{k-1} \frac{x^k}{k} \qquad (-1 < x \leqq 1) \quad (5)$$

For each case, the series represents the given function and converges in the interval specified. Series (1)–(5) are Taylor series expanded about $c = 0$.

A Taylor series expanded about $c = 1$ is

$$\ln(x) = (x-1) - \frac{(x-1)^2}{2} + \frac{(x-1)^3}{3} - \cdots = \sum_{k=1}^{\infty} (-1)^{k-1} \frac{(x-1)^k}{k}$$

where $0 < x \leqq 2$. The reader should recall the **factorial** notation

$$n! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot \cdots \cdot n$$

for $n \geqq 1$ and the special definition of $0! = 1$.

Series of this type are often used to compute good approximate values of complicated functions at specific points.

EXAMPLE 1 Use five terms in the $\ln(1+x)$ series (5) to approximate $\ln(1.1)$.

Solution Taking $x = 0.1$ in the first five terms of the series for $\ln(1+x)$ gives us

$$\ln(1.1) \approx 0.1 - \frac{0.01}{2} + \frac{0.001}{3} - \frac{0.0001}{4} + \frac{0.00001}{5} = 0.09531\,03333\ldots$$

where $\approx$ means **approximately equal**. This value is correct to six decimal places of accuracy. ■

On the other hand, such good results are not always obtained in using series.

EXAMPLE 2 Try to compute e^8 by using the e^x series (1).

Solution The result is

$$e^8 = 1 + 8 + \frac{64}{2} + \frac{512}{6} + \frac{4096}{24} + \frac{32768}{120} + \cdots$$

It is apparent that many terms are needed to compute e^8 with reasonable precision. By repeated squaring, we find $e^2 = 7.38905\,6$, $e^4 = 54.59815\,00$, and $e^8 = 2980.95798\,7$. The first six terms given yield $570.06666\,5$. ■

These examples illustrate a general rule:

■ **Rule**

Rule of Thumb

A Taylor series converges rapidly near the point of expansion and slowly (or not at all) at more remote points.

A graphical depiction of the phenomenon can be obtained by graphing a few partial sums of a Taylor series. In Figure 1.3 (p. 22), we show the function

$$y = \sin x$$

and the partial-sum functions

Partial Summations for sin x

$$S_1 = x$$
$$S_3 = x - \frac{x^3}{6}$$
$$S_5 = x - \frac{x^3}{6} + \frac{x^5}{120}$$

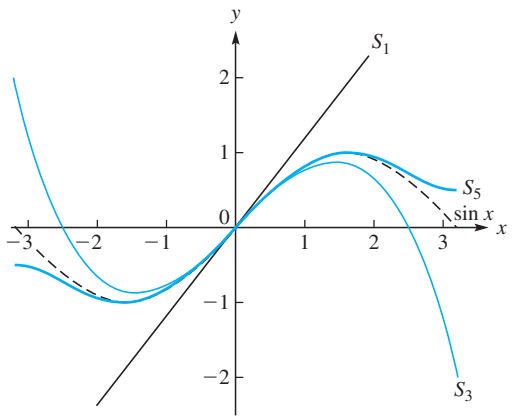


FIGURE 1.3
Approximations to sin x

which come from the sin x series (2). While S_1 may be an acceptable approximation to sin x when $x \approx 0$, the graphs for S_3 and S_5 match that of sin x on larger intervals about the origin.

All of the series illustrated here are examples of the following general series:

■ **Theorem 1**

Formal Taylor Series for f about c

$$f(x) \sim f(c) + f'(c)(x-c) + \frac{f''(c)}{2!}(x-c)^2 + \frac{f'''(c)}{3!}(x-c)^3 + \cdots$$

$$f(x) \sim \sum_{k=0}^{\infty} \frac{f^{(k)}(c)}{k!}(x-c)^k \tag{6}$$

Here, rather than using $=$, we have written $\sim$ to indicate that we are *not* allowed to assume that $f(x)$ *equals* the series on the right. All we have at the moment is a formal series that can be written down provided that the successive derivatives f', f'', f''', ... exist at the point c. Series (6) is called the **Taylor series of f at the point c**.

In the special case $c = 0$, the Formal Taylor series (6) is also called a **Maclaurin series**:

$$f(x) \sim f(0) + f'(0)x + \frac{f''(0)}{2!}x^2 + \frac{f'''(0)}{3!}x^3 + \cdots$$

$$f(x) \sim \sum_{k=0}^{\infty} \frac{f^{(k)}(0)}{k!}x^k \tag{7}$$

The first term is $f(0)$ when $k = 0$.

EXAMPLE 3 What is the Taylor series of the function

$$f(x) = 3x^5 - 2x^4 + 15x^3 + 13x^2 - 12x - 5$$

at the point $c = 2$?

Solution To compute the coefficients in the series, we need the numerical values of $f^{(k)}(2)$ for $k \geq 0$. Here are the details of the computation:

$$\begin{aligned}
f(x) &= 3x^5 - 2x^4 + 15x^3 + 13x^2 - 12x - 5 \quad &\Rightarrow \quad f(2) &= 207 \\
f'(x) &= 15x^4 - 8x^3 + 45x^2 + 26x - 12 \quad &\Rightarrow \quad f'(2) &= 396 \\
f''(x) &= 60x^3 - 24x^2 + 90x + 26 \quad &\Rightarrow \quad f''(2) &= 590 \\
f'''(x) &= 180x^2 - 48x + 90 \quad &\Rightarrow \quad f'''(2) &= 714 \\
f^{(4)}(x) &= 360x - 48 \quad &\Rightarrow \quad f^{(4)}(2) &= 672 \\
f^{(5)}(x) &= 360 \quad &\Rightarrow \quad f^{(5)}(2) &= 360 \\
f^{(k)}(x) &= 0 \quad &\Rightarrow \quad f^{(k)}(2) &= 0 \qquad (k \geq 6)
\end{aligned}$$

Therefore, we have

$$\begin{aligned}
f(x) \sim\ & 207 + 396(x-2) + 295(x-2)^2 \\
& + 119(x-2)^3 + 28(x-2)^4 + 3(x-2)^5
\end{aligned}$$

In this example, it is *not* difficult to see that $\sim$ may be replaced by $=$. Simply expand all the terms in the Taylor series and collect them to get the original form for f. Taylor's Theorem, discussed soon, allows us to draw this conclusion without doing any work! ■

It is interesting to show how well we can approximate a function $f(x)$ at a point $x = a$ by taking only a few terms of the Maclaurin series (7). We illustrated with three cases. With only the first term, the function is assumed to be a constant: $f(a) \approx f(0)$. With two terms, the slope at 0 is taken into account by way of the straight line from $f(0)$ to $f(a)$; namely, $f(a) \approx f(0) + f'(0)x$ when $x = a$. With three terms, the curvature due to $f''(0)$ comes into play and we obtain a parabolic curve: $f(a) \approx f(0) + f'(0)x + \frac{1}{2}f''(0)x^2$ when $x = a$. Each additional term improves the accuracy in the approximation for $f(a)$. In Figure 1.4, we show these partial sums in the Maclaurin series (7) when $f(x) = e^x$ and $x = a = 1$.

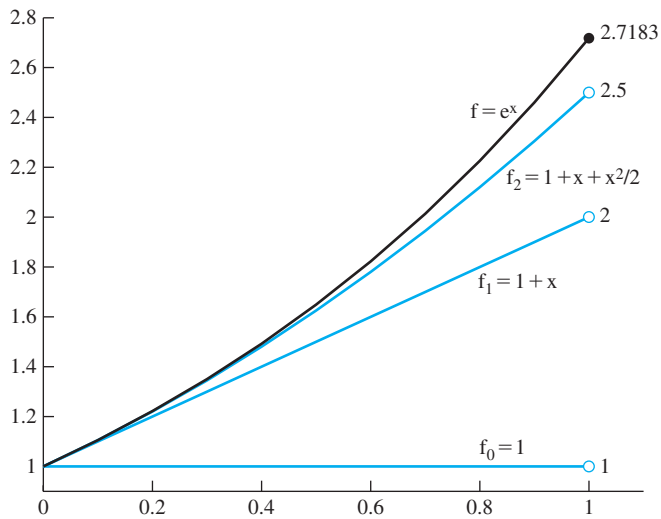


FIGURE 1.4
Approximations to e^x

Complete Horner's Algorithm

An application of Horner's algorithm is that of finding the Taylor expansion of a polynomial about any point. Let $p(x)$ be a given polynomial of degree n with coefficients a_k as in (2) in Section 1.1, and suppose that we desire the coefficients c_k in the equation

$$\begin{aligned} p(x) &= a_n x^n + a_{n-1}x^{n-1} + \cdots + a_0 \\ &= c_n(x-r)^n + c_{n-1}(x-r)^{n-1} + \cdots + c_1(x-r) + c_0 \end{aligned}$$

Of course, Taylor's Theorem asserts that $c_k = p^{(k)}(r)/k!$, but we seek a more efficient algorithm. Notice that $p(r) = c_0$, so this coefficient is obtained by applying Horner's algorithm to the polynomial p with the point r. The algorithm also yields the polynomial

$$q(x) = \frac{p(x) - p(r)}{x - r} = c_n(x-r)^{n-1} + c_{n-1}(x-r)^{n-2} + \cdots + c_1$$

This shows that the second coefficient, c_1, can be obtained by applying Horner's algorithm to the polynomial q with point r, because $c_1 = q(r)$. Notice that the first application of Horner's algorithm does not yield q in the form shown, but rather as a sum of powers of x. (See (3)–(4) in Section 1.1.) This process is repeated until all coefficients c_k are found.

We call the algorithm just described the **complete Horner's algorithm**. The pseudocode for executing it is arranged so that the coefficients c_k **overwrite** the input coefficients a_k.

Complete Horner's Algorithm Pseudocode

```
integer n, k, j;
real r; real array (a_i)_{0:n}
for k = 0 to n − 1
    for j = n − 1 to k
        a_j ← a_j + r a_{j+1}
    end for
end for
```

This procedure can be used in carrying out Newton's method for finding roots of a polynomial, which we discuss in Chapter 3. Moreover, it can be done in complex arithmetic to handle polynomials with complex roots or coefficients.

EXAMPLE 4 Using the complete Horner's algorithm, find the Taylor expansion of the polynomial

$$p(x) = x^4 - 4x^3 + 7x^2 - 5x + 2$$

about the point $r = 3$.

Solution The work can be arranged as follows:

$$\begin{array}{rrrrr} 1 & -4 & 7 & -5 & 2 \\ 3) & 3 & -3 & 12 & 21 \\ \hline 1 & -1 & 4 & 7 & \boxed{23} \\ & 3 & 6 & 30 & \\ \hline 1 & 2 & 10 & \boxed{37} & \\ & 3 & 15 & & \\ \hline 1 & 5 & \boxed{25} & & \\ & 3 & & & \\ \hline \boxed{1} & \boxed{8} & & & \end{array}$$

The calculation shows that

$$p(x) = (x-3)^4 + 8(x-3)^3 + 25(x-3)^2 + 37(x-3) + 23$$ ■

Taylor's Theorem in Terms of $(x - c)$

■ **Theorem 2**

Taylor's Theorem for $f(x)$

If the function f possesses continuous derivatives of orders 0, 1, 2, . . . , $(n + 1)$ in a closed interval $I = [a, b]$, then for any c and x in I,

$$f(x) = \sum_{k=0}^{n} \frac{f^{(k)}(c)}{k!}(x-c)^k + E_{n+1} \tag{8}$$

where the error term E_{n+1} can be given in the form

$$E_{n+1} = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x-c)^{n+1}$$

Here ξ is a point that lies between c and x and depends on both.

In practical computations with Taylor series, it is usually necessary to **truncate** the series because it is *not* possible to carry out an infinite number of additions. A series is said to be **truncated** if we ignore all terms after a certain point. Thus, if we truncate the exponential series (1) after seven terms, the result is

e^x Series

$$e^x \approx 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} + \frac{x^6}{6!}$$

This no longer represents e^x except when $x = 0$. But the truncated series should *approximate* e^x. Here is where we need Taylor's Theorem. With its help, we can assess the difference between a function f and its truncated Taylor series.

The explicit assumption in this theorem is that $f(x)$, $f'(x)$, $f''(x)$, . . . , $f^{(n+1)}(x)$ are all continuous functions in the interval $I = [a, b]$. The final term E_{n+1} in (8) is the **remainder** or **error term**. The given formula for E_{n+1} is valid when we assume only that $f^{(n+1)}$ exists at each point of the open interval (a, b). The error term is similar to the terms preceding it, but notice that $f^{(n+1)}$ must be evaluated at a point other than c. This point ξ depends on x and is in the open interval (c, x) or (x, c). Other forms of the remainder are possible; the one given here is **Lagrange's form**. (We do *not* prove Taylor's Theorem here.)

EXAMPLE 5 Derive the Taylor series for e^x at $c = 0$, and prove that it converges to e^x by using Taylor's Theorem.

Solution If $f(x) = e^x$, then $f^{(k)}(x) = e^x$ for $k \geq 0$. Therefore, $f^{(k)}(c) = f^{(k)}(0) = e^0 = 1$ for all k. From (8), we have

$$e^x = \sum_{k=0}^{n} \frac{x^k}{k!} + \frac{e^{\xi}}{(n+1)!}x^{n+1} \tag{9}$$

Now let us consider all the values of x in some symmetric interval around the origin, for example, $-s \leq x \leq s$. Then $|x| \leq s$, $|\xi| \leq s$, and $e^{\xi} \leq e^{s}$. Hence, the remainder term satisfies this inequality:

$$\lim_{n\to\infty} \left| \frac{e^{\xi}}{(n+1)!} x^{n+1} \right| \leq \lim_{n\to\infty} \frac{e^{s}}{(n+1)!} s^{n+1} = 0$$

Thus, if we take the limit as $n \to \infty$ on both sides of (9), we obtain

$$e^{x} = \lim_{n\to\infty} \sum_{k=0}^{n} \frac{x^{k}}{k!} = \sum_{k=0}^{\infty} \frac{x^{k}}{k!}$$

■

This example illustrates how we can establish, in specific cases, that a formal Taylor series (6) actually represents the function. Let's examine another example to see how the formal series can *fail* to represent the function.

EXAMPLE 6 Derive the formal Taylor series for $f(x) = \ln(1+x)$ at $c = 0$, and determine the range of positive x for which the series represents the function.

Solution We need $f^{(k)}(x)$ and $f^{(k)}(0)$ for $k \geq 1$. Here is the work:

$$\begin{aligned}
f(x) &= \ln(1+x) && \Rightarrow && f(0) &= 0 \\
f'(x) &= (1+x)^{-1} && \Rightarrow && f'(0) &= 1 \\
f''(x) &= -(1+x)^{-2} && \Rightarrow && f''(0) &= -1 \\
f'''(x) &= 2(1+x)^{-3} && \Rightarrow && f'''(0) &= 2 \\
f^{(4)}(x) &= -6(1+x)^{-4} && \Rightarrow && f^{(4)}(0) &= -6 \\
&\vdots && \Rightarrow && &\vdots \\
f^{(k)}(x) &= (-1)^{k-1}(k-1)!(1+x)^{-k} && \Rightarrow && f^{(k)}(0) &= (-1)^{k-1}(k-1)!
\end{aligned}$$

Hence by Taylor's Theorem, we obtain

$$\begin{aligned}
\ln(1+x) &= \sum_{k=1}^{n} (-1)^{k-1} \frac{(k-1)!}{k!} x^{k} + \frac{(-1)^{n} n! (1+\xi)^{-n-1}}{(n+1)!} x^{n+1} \\
&= \sum_{k=1}^{n} (-1)^{k-1} \frac{x^{k}}{k} + \frac{(-1)^{n}}{n+1} (1+\xi)^{-n-1} x^{n+1} \qquad (10)
\end{aligned}$$

For the *infinite* series to represent $\ln(1+x)$, it is necessary and sufficient that the error term converge to zero as $n \to \infty$. Assume that $0 \leq x \leq 1$. Then $0 \leq \xi \leq x$ (because *zero* is the point of expansion); thus, $0 \leq x/(1+\xi) \leq 1$. Hence, the error term converges to zero in this case. If $x > 1$, the terms in the series do not approach zero, and the series does not converge. Hence, the series represents $\ln(1+x)$ if $0 \leq x \leq 1$, but *not* if $x > 1$. (The series also represents $\ln(1+x)$ for $-1 < x < 0$, but not if $x \leq -1$.) ■

Mean-Value Theorem

The special case $n = 0$ in Taylor's Theorem is known as the **Mean-Value Theorem**. It is usually stated, however, in a somewhat more precise form.

■ **Theorem 3**

Mean-Value Theorem

If f is a continuous function on the closed interval $[a, b]$ and possesses a derivative at each point of the open interval (a, b), then

$$f(b) = f(a) + (b - a)f'(\xi)$$

for some ξ in (a, b).

Hence, the ratio $[f(b) - f(a)]/(b - a)$ is equal to the derivative of f at some point ξ between a and b; that is, for some $\xi \in (a, b)$,

$$f'(\xi) = \frac{f(b) - f(a)}{b - a}$$

The right-hand side could be used as an *approximation* for $f'(x)$ at any x within the interval (a, b). The approximation of derivatives is discussed more fully in Section 4.3.

Taylor's Theorem in Terms of *h*

Other forms of Taylor's Theorem are often useful. These can be obtained from (8) by changing the variables.

■ **Corollary 1**

Taylor's Theorem for $f(x + h)$

If the function f possesses continuous derivatives of order $0, 1, 2, \ldots, (n + 1)$ in a closed interval $I = [a, b]$, then for any x in I,

$$f(x + h) = \sum_{k=0}^{n} \frac{f^{(k)}(x)}{k!} h^k + E_{n+1} \tag{11}$$

where h is any value such that $x + h$ is in I and where

$$E_{n+1} = \frac{f^{(n+1)}(\xi)}{(n + 1)!} h^{n+1}$$

for some ξ between x and $x + h$.

The form (11) is obtained from (8) by replacing x by $x + h$ and replacing c by x. Notice that because h can be positive or negative, the requirement on ξ means $x < \xi < x + h$ if $h > 0$ or $x + h < \xi < x$ if $h < 0$.

The **error term** E_{n+1} depends on h in two ways: First, h^{n+1} is explicitly present; second, the point ξ generally depends on h. As h converges to zero, E_{n+1} converges to zero with essentially the same rapidity with which h^{n+1} converges to zero. For large n, this is quite rapid. To express this qualitative fact, we write

Big O Notation

$$E_{n+1} = \mathcal{O}(h^{n+1})$$

as $h \to 0$. This is called **big O notation**, and it is shorthand for the inequality

$$|E_{n+1}| \leqq C|h|^{n+1}$$

where C is a constant. In the present circumstances, this constant could be any number for which $|f^{(n+1)}(t)|/(n+1)! \leq C$, for all t in the initially given interval, I. Roughly speaking, $E_{n+1} = \mathcal{O}(h^{n+1})$ means that the behavior of E_{n+1} is similar to the much simpler expression h^{n+1}.

It is important to realize that (11) corresponds to an entire sequence of theorems, one for each value of n. For example, we can write out the cases $n = 0, 1, 2$ as follows:

$$\begin{aligned} f(x+h) &= f(x) + f'(\xi_1)h \\ &= f(x) + \mathcal{O}(h) \\ f(x+h) &= f(x) + f'(x)h + \frac{1}{2!}f''(\xi_2)h^2 \\ &= f(x) + f'(x)h + \mathcal{O}(h^2) \\ f(x+h) &= f(x) + f'(x)h + \frac{1}{2!}f''(x)h^2 + \frac{1}{3!}f'''(\xi_3)h^3 \\ &= f(x) + f'(x)h + \frac{1}{2!}f''(x)h^2 + \mathcal{O}(h^3) \end{aligned}$$

The importance of the error term in Taylor's Theorem cannot be stressed too much. In later chapters, many situations require an estimate of errors in a numerical process by use of Taylor's Theorem. Here are some elementary examples.

EXAMPLE 7 Expand $\sqrt{1+h}$ in powers of h. Then compute $\sqrt{1.00001}$ and $\sqrt{0.99999}$.

Solution Let $f(x) = x^{1/2}$. Then $f'(x) = \frac{1}{2}x^{-1/2}$, $f''(x) = -\frac{1}{4}x^{-3/2}$, $f'''(x) = \frac{3}{8}x^{-5/2}$, and so on. Now use (11) with $x = 1$. Taking $n = 2$ for illustration, we have

$$\sqrt{1+h} = 1 + \frac{1}{2}h - \frac{1}{8}h^2 + \frac{1}{16}h^3\xi^{-5/2} \tag{12}$$

where ξ is an unknown number that satisfies $1 < \xi < 1+h$, if $h > 0$. It is important to notice that the function $f(x) = \sqrt{x}$ possesses derivatives of all orders at any point $x > 0$.

In (12), let $h = 10^{-5}$. Then

$$\sqrt{1.00001} \approx 1 + 0.5 \times 10^{-5} - 0.125 \times 10^{-10} = 1.00000\,49999\,87500$$

By substituting $-h$ for h in the series, we obtain

$$\sqrt{1-h} = 1 - \frac{1}{2}h - \frac{1}{8}h^2 - \frac{1}{16}h^3\xi^{-5/2}$$

Hence, we have

$$\sqrt{0.99999} \approx 0.99999\,49999\,87500$$

Since $1 < \xi < 1+h$, the absolute error does not exceed

$$\frac{1}{16}h^3\xi^{-5/2} < \frac{1}{16}10^{-15} = 0.00000\,00000\,00000\,0625$$

and both numerical values are correct to all 15 decimal places shown. ■

Alternating Series

Another theorem from calculus is often useful in establishing the convergence of a series and in estimating the error involved in truncation. From it, we have the following important principle for alternating series:

Alternative Series Principle

Principle. If the magnitudes of the terms in an alternating series converge monotonically to zero, then the error in truncating the series is no larger than the magnitude of the first omitted term.

This theorem applies only to **alternating series**—that is, series in which the successive terms are alternately positive and negative.

■ **Theorem 4**

Alternating Series Theorem

If $a_1 \geqq a_2 \geqq \cdots \geqq a_n \geqq \cdots 0$ for all n and $\lim_{n\to\infty} a_n = 0$, then the alternating series

$$a_1 - a_2 + a_3 - a_4 + \cdots$$

converges; that is,

$$\sum_{k=1}^{\infty}(-1)^{k-1}a_k = \lim_{n\to\infty}\sum_{k=1}^{n}(-1)^{k-1}a_k = \lim_{n\to\infty} S_n = S$$

where S is its sum and S_n is the nth partial sum. Moreover, for all n,

$$|S - S_n| \leqq a_{n+1}$$

EXAMPLE 8 If the sine series is to be used in computing $\sin 1$ with an error less than $\frac{1}{2} \times 10^{-6}$, how many terms are needed?

Solution From Series (2), we have

$$\sin 1 = 1 - \frac{1}{3!} + \frac{1}{5!} - \frac{1}{7!} + \cdots$$

If we stop at $1/(2n-1)!$, the error does not exceed the first neglected term, which is $1/(2n+1)!$. Thus, we should select n so that

$$\frac{1}{(2n+1)!} < \frac{1}{2} \times 10^{-6}$$

Using logarithms to base 10, we obtain $\log(2n+1)! > \log 2 + 6 = 6.3$. With a calculator, we compute a table of values for $\log n!$ and find that $\log 10! \approx 6.6$. Hence, if $n \geqq 5$, the error is acceptable. ■

EXAMPLE 9 If the logarithmic series (5) is to be used for computing $\ln 2$ with an error of less than $\frac{1}{2} \times 10^{-6}$, how many terms are required?

Solution To compute $\ln 2$, we take $x = 1$ in the series, and using $\approx$ to mean approximate equality, we have

$$S = \ln 2 \approx 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \cdots + \frac{(-1)^{n-1}}{n} = S_n$$

By the Alternating Series Theorem, the error involved when the series is truncated with n terms is

$$|S - S_n| \leqq \frac{1}{n+1}$$

We select n so that

$$\frac{1}{n+1} < \frac{1}{2} \times 10^{-6}$$

Hence, more than two million terms would be needed! We conclude that this method of computing $\ln 2$ is not practical. (See Exercises 1.2.10–1.2.12 for several good alternatives.) ■

Caution

A word of caution is needed about this technique of calculating the number of terms to be used in a series by just making the $(n+1)$st term less than some tolerance. This procedure is valid only for alternating series in which the terms decrease in magnitude to zero, although it is occasionally used to get rough estimates in other cases. For example, it can be used to identify a nonalternating series as one that converges slowly. When this technique cannot be used, a bound on the remaining terms of the series has to be established. Determining such a bound may be somewhat difficult.

EXAMPLE 10 It is known that

$$\frac{\pi^4}{90} = 1^{-4} + 2^{-4} + 3^{-4} + \cdots$$

How many terms should we take to compute $\pi^4/90$ with an error of at most $\frac{1}{2} \times 10^{-6}$?

Solution A naive approach is to take

$$1^{-4} + 2^{-4} + 3^{-4} + \cdots + n^{-4}$$

where n is chosen so that the next term, $(n+1)^{-4}$, is less that $\frac{1}{2} \times 10^{-6}$. This value of n is 37, but this is an erroneous answer because the partial sum

$$S_{37} = \sum_{k=1}^{37} k^{-4}$$

differs from $\pi^4/90$ by approximately 6×10^{-6}. What we should do, of course, is to select n so that *all* the omitted terms add up to less than $\frac{1}{2} \times 10^{-6}$; that is,

$$\sum_{k=n+1}^{\infty} k^{-4} < \frac{1}{2} \times 10^{-6}$$

By a technique familiar from calculus (see Figure 1.5), we have

$$\sum_{k=n+1}^{\infty} k^{-4} < \int_n^{\infty} x^{-4}\,dx = \frac{x^{-3}}{-3}\bigg|_n^{\infty} = \frac{1}{3n^3}$$

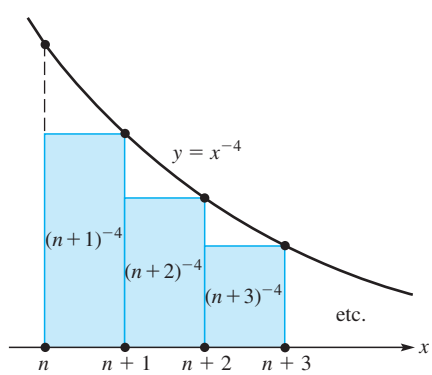


FIGURE 1.5 Illustrating Example 10

Thus, it suffices to select n so that $(3n^3)^{-1} < \frac{1}{2} \times 10^{-6}$, or $n \geqq 88$. (A more sophisticated analysis improves this considerably.) ■

Summary 1.2

- Complete Horner's Algorithm:

```
for k = 0 to n − 1
    for j = n − 1 to k
        a_j ← a_j + r a_{j+1}
    end for
end for
```

- The **Taylor series expansion** about c for $f(x)$ is

$$f(x) = \sum_{k=0}^{n} \frac{f^{(k)}(c)}{k!}(x-c)^k + E_{n+1}$$

with error term

$$E_{n+1} = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x-c)^{n+1}$$

A more useful form for us is the **Taylor series expansion** for $f(x+h)$, which is

$$f(x+h) = \sum_{k=0}^{n} \frac{f^{(k)}(x)}{k!}h^k + E_{n+1}$$

with error term

$$E_{n+1} = \frac{f^{(n+1)}(\xi)}{(n+1)!}h^{n+1} = \mathcal{O}(h^{n+1})$$

- An **alternating series**

$$S = \sum_{k=1}^{\infty}(-1)^{k-1}a_k$$

converges when the terms a_k converge downward to zero. Furthermore, the partial sums S_n differ from S by an amount that is bounded by

$$|S - S_n| \leqq a_{n+1}$$

Exercises 1.2

1. The Maclaurin series for $(1+x)^n$ is also known as the **binomial series**. It states that

$$(1+x)^n = 1 + nx + \frac{n(n-1)}{2!}x^2 + \cdots \quad (x^2 < 1)$$

Derive this series. Then give its particular forms in summation notation by letting $n = 2$, $n = 3$, and $n = \frac{1}{2}$. Next use the last form to compute $\sqrt{1.0001}$ correct to 15 decimal places (rounded).

2. (Continuation) Use the series in the preceding problem to obtain series (4). How could this series be used on a computing machine to produce x/y if only addition and multiplication are built-in operations?

3. (Continuation) Use the previous problem to obtain a series for $(1 + x^2)^{-1}$.

4. Why do the following functions not possess Taylor series expansions at $x = 0$?

[a]**a.** $f(x) = \sqrt{x}$ [a]**b.** $f(x) = |x|$

c. $f(x) = \arcsin(x - 1)$ **d.** $f(x) = \cot x$

[a]**e.** $f(x) = \log x$ **f.** $f(x) = x^{\pi}$

[a]**5.** Determine the Taylor series for $\cosh x$ about zero. Evaluate $\cosh(0.7)$ by summing four terms. Compare with the actual value.

6. Determine the first two nonzero terms of the series expansion about zero for the following:

[a]**a.** $e^{\cos x}$ [a]**b.** $\sin(\cos x)$

c. $(\cos x)^2(\sin x)$

[a]**7.** Find the smallest nonnegative integer m such that the Taylor series about m for $(x - 1)^{1/2}$ exists. Determine the coefficients in the series.

[a]**8.** Determine how many terms are needed to compute e correctly to 15 decimal places (rounded) using e^x series (1) for e^x.

[a]**9.** (Continuation) If $x < 0$ in the preceding problem, what are the signs of the terms in the series? Loss of significant digits can be a serious problem in using the series. Will the formula $e^{-x} = 1/e^x$ be helpful in reducing the error? Explain. (See Section 1.5 for further discussion.) Try high-precision computer arithmetic to see how bad the floating-point errors can be.

10. Show how the simple equation $\ln 2 = \ln[e(2/e)]$ can be used to speed up the calculation of $\ln 2$ in series (10).

[a]**11.** What is the series for $\ln(1 - x)$? What is the series for $\ln[(1 + x)/(1 - x)]$?

[a]**12.** (Continuation) In the series for $\ln[(1 + x)/(1 - x)]$, determine what value of x to use if we wish to compute $\ln 2$. Estimate the number of terms needed for ten digits (rounded) of accuracy. Is this method practical?

13. Use the Alternating Series Theorem to determine the number of terms in series (5) needed for computing $\ln 1.1$ with error less than $\frac{1}{2} \times 10^{-8}$.

14. Write the Taylor series for the function $f(x) = x^3 - 2x^2 + 4x - 1$, using $x = 2$ as the point of expansion; that is, write a formula for $f(2 + h)$.

15. Determine the first four nonzero terms in the series expansion about zero for

[a]**a.** $f(x) = (\sin x) + (\cos x)$ and find an approximate value for $f(0.001)$.

[a]**b.** $g(x) = (\sin x)(\cos x)$ and find an approximate value for $g(0.0006)$.

Compare the accuracy of these approximations to those obtained from tables or via a calculator.

[a]**16.** Verify this Taylor series and prove that it converges on the interval $-e < x \leqq e$.

$$\ln(e + x) = 1 + \frac{x}{e} - \frac{x^2}{2e^2} + \frac{x^3}{3e^3} - \frac{x^4}{4e^4} + \cdots$$
$$= 1 + \sum_{k=1}^{\infty} \frac{(-1)^{k-1}}{k} \left(\frac{x}{e}\right)^k$$

[a]**17.** How many terms are needed in series (3) to compute $\cos x$ for $|x| < \frac{1}{2}$ accurate to 12 decimal places (rounded)?

[a]**18.** A function f is defined by the series

$$f(x) = \sum_{k=1}^{\infty} (-1)^k \left(\frac{x^k}{k^4}\right)$$

Determine the minimum number of terms needed to compute $f(1)$ with error less than 10^{-8}.

19. Verify that the partial sums $s_k = \sum_{i=0}^{k} x^i / i!$ in the series (1) for e^x can be written recursively as $s_k = s_{k-1} + t_k$, where $s_0 = 1$, $t_1 = x$, and $t_k = (x/k)t_{k-1}$.

[a]**20.** What is the fifth term in the Taylor series of $(1 - 2h)^{1/2}$?

21. Show that if $E = \mathcal{O}(h^n)$, then $E = \mathcal{O}(h^m)$ for any nonnegative integer $m \leqq n$. Here $h \to 0$.

22. Show how $p(x) = 6(x + 3) + 9(x + 3)^5 - 5(x + 3)^8 - (x + 3)^{11}$ can be efficiently evaluated.

[a]**23.** What is the second term in the Taylor series of $\sqrt[4]{4x - 1}$ about 4.25?

[a]**24.** How would you compute a table of $\log n!$ for $1 \leqq n \leqq 1000$?

25. For small x, the approximation $\sin x \approx x$ is often used. For what range of x is this good to a *relative* accuracy of $\frac{1}{2} \times 10^{-14}$?

26. In the Taylor series for the function $3x^2 - 7 + \cos x$ (expanded in powers of x), what is the coefficient of x^2?

27. In the Taylor series (about $\pi/4$) for the function $\sin x + \cos x$, find the third nonzero term.

[a]**28.** By using Taylor's Theorem, one can be sure that for all x that satisfy $|x| < \frac{1}{2}$, $|\cos x - (1 - x^2/2)|$ is less than or equal to what numerical value?

29. Find the value of ξ that serves in Taylor's Theorem when $f(x) = \sin x$, with $x = \pi/4$, $c = 0$, and $n = 4$.

30. Use Taylor's Theorem to find a linear function that approximates $\cos x$ best in the vicinity of $x = 5\pi/6$.

31. For the alternating series $S_n = \sum_{k=0}^{n}(-1)^k a_k$, with $a_0 > a_1 > \cdots > 0$, show by induction that $S_0 > S_2 > S_4 > \cdots$, that $S_1 < S_3 < S_5 < \cdots$, and that $0 < S_{2n} - S_{2n+1} = a_{2n+1}$.

[a] **32.** What is the Maclaurin series for the function $f(x) = 3 + 7x - 1.33x^2 + 19.2x^4$? What is the Taylor series for this function about $c = 2$?

33. In the text, it was asserted that $\sum_{k=0}^{6} x^k/k!$ represents e^x only at the point $x = 0$. Prove this.

34. Determine the first three terms in the Taylor series in terms of h for e^{x-h}. Using three terms, one obtains $e^{0.999} \approx Ce$, where C is a constant. Determine C.

[a] **35.** What is the least number of terms required to compute π as 3.14 (rounded) using the series

$$\pi = 4 - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \cdots$$

36. Using the Taylor series expansion in terms of h, determine the first three terms in the series for $e^{\sin(x+h)}$. Evaluate $e^{\sin 90.01^\circ}$ accurately to ten decimal places as Ce for constant C.

37. Develop the first two terms and the error in the Taylor series in terms of h for $\ln(3 - 2h)$.

[a] **38.** Determine a Taylor series to represent $\cos(\pi/3+h)$. Evaluate $\cos(60.001^\circ)$ to eight decimal places (rounded). *Hint*: π radians equal 180 degrees.

[a] **39.** Determine a Taylor series to represent $\sin(\pi/4+h)$. Evaluate $\sin(45.0005^\circ)$ to nine decimal places (rounded).

40. Establish the first three terms in the Taylor series for $\csc(\pi/6 + h)$. Compute $\csc(30.00001^\circ)$ to the same accuracy as the given data.

41. Establish the Taylor series in terms of h for the following:

a. e^{x+2h} **b.** $\sin(x - 3h)$

c. $\ln[(x - h^2)/(x + h^2)]$

[a] **42.** Determine the first three terms in the Taylor series in terms of h for $(x - h)^m$, where m is an integer constant.

43. Given the series

$$-1 + 2^{-4} - 3^{-4} + 4^{-4} - \cdots$$

how many terms are needed to obtain four decimal places (chopped) of accuracy?

44. How many terms are needed in the series

$$\operatorname{arccot} x = \frac{\pi}{2} - x + \frac{x^3}{3} - \frac{x^5}{5} + \frac{x^7}{7} - \cdots$$

to compute $\operatorname{arccot} x$ for $x^2 < 1$ accurate to 12 decimal places (rounded)?

45. Determine the first three terms in the Taylor series to represent $\sinh(x + h)$. Evaluate $\sinh(0.0001)$ to 20 decimal places (rounded) using this series.

46. Determine a Taylor series to represent C^{x+h} for constant C. Use the series to find an approximate value of $10^{1.0001}$ to five decimal places (rounded).

[a] **47.** **Stirling's formula** states that $n!$ is greater than, and very close to, $\sqrt{2\pi n}\, n^n e^{-n}$. Use this to find an n for which $1/n! < \frac{1}{2} \times 10^{-14}$.

48. Develop the first two nonzero terms and the error term in the Taylor series in powers of h for $\ln[1 - (h/2)]$. Approximate $\ln(0.9998)$ using these two terms.

49. **L'Hôpital's rule** states that under suitable conditions,

$$\lim_{x \to a} \frac{f(x)}{g(x)} = \frac{f'(a)}{g'(a)}$$

It is true, for instance, when f and g have continuous derivatives in an open interval containing a, and $f(a) = g(a) = 0 \neq g'(a)$. Establish L'Hôpital's rule using the Mean-Value Theorem.

50. (Continuation) Evaluate the following numerically and use the previous problem to show that

a. $\lim_{x\to 0} \dfrac{\sin x}{x} = 1$ [a] **b.** $\lim_{x\to 0} \dfrac{\arctan x}{x} = 1$

[a] **c.** $\lim_{x\to \pi} \dfrac{\cos x + 1}{\sin x} = 0$

[a] **51.** Verify that if we take only the terms up to and including $x^{2n-1}/(2n-1)!$ in the series (2) for $\sin x$ and if $|x| < \sqrt{6}$, then the error involved does not exceed $|x|^{2n+1}/(2n+1)!$. How many terms are needed to compute $\sin(23)$ with an error of at most 10^{-8}? What problems do you foresee in using the series to compute $\sin(23)$? Show how to use periodicity to compute $\sin(23)$. Show that each term in the series can be obtained from the preceding one by a simple arithmetic operation.

[a] **52.** Expand the **error function**

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\, dt$$

in a series by using the exponential series and integrating. Obtain the Taylor series of $\operatorname{erf}(x)$ about zero directly. Are the two series the same? Evaluate $\operatorname{erf}(1)$ by adding four terms of the series and compare with the value

$\operatorname{erf}(1) \approx 0.8427$, which is correct to four decimal places.
Hint: Recall from the **Fundamental Theorem of Calculus** that

$$\frac{d}{dx}\int_0^x f(t)\,dt = f(x)$$

[a]**53.** Establish the validity of the Taylor series

$$\arctan x = \sum_{k=1}^{\infty}(-1)^{k+1}\frac{x^{2k-1}}{2k-1} \qquad (-1 \leqq x \leqq 1)$$

Is it practical to use this series directly to compute $\arctan(1)$ if ten decimal places (rounded) of accuracy are required? How many terms of the series would be needed? Will loss of significance occur?
Hint: Start with the series for $1/(1+x^2)$ and integrate term by term. Note that this procedure is only formal; the convergence of the resulting series can be proved by appealing to certain theorems of advanced calculus.

[a]**54.** It is known that

$$\pi = 4 - 8\sum_{k=1}^{\infty}(16k^2-1)^{-1}$$

Discuss the numerical aspects of computing π by means of this formula. How many terms would be needed to yield ten decimal places (rounded) of accuracy?

55. Taylor's Theorem for $f(x)$ expanded about c involves this equation:

$$f(x) = f(c) + (x-c)f'(c) + \frac{1}{2}(x-c)^2 f''(c) + \cdots + \frac{1}{n!}(x-c)^n f^{(n)}(\xi)$$

Use this to determine how many terms in the series for e^x are needed to compute e with error at most 10^{-10}.
Hint: Use these approximate values of $n!$: $9! = 3.6\times 10^5$, $11! = 4.0\times 10^7$, $12! = 4.8\times 10^8$, $13! = 6.2\times 10^9$, $14! = 8.7\times 10^{10}$, and $15! = 1.3\times 10^{12}$.

56. To illustrate your understanding of the material, do the following:

a. Repeat Example 1.2.3 using the complete Horner's algorithm.

b. Repeat Example 1.2.6 using the Taylor series of the polynomial $p(x)$.

Computer Exercises 1.2

[a]**1.** Everyone knows the quadratic formula $(-b \pm \sqrt{b^2-4ac})/(2a)$ for the roots of the quadratic equation $ax^2+bx+c=0$. Using this formula, by hand and by computer, solve the equation $x^2 + 10^8 x + c = 0$ when $c = 1$ and 10^8. Interpret the results.

2. Use a computer algebra system to obtain graphs of the first five partial sums of the series

$$\arctan x = \sum_{k=1}^{\infty}(-1)^{k+1}\frac{x^{2k-1}}{2k-1}$$

3. Use a graphical computer package to reproduce the graphs in Figure 1.3 as well as the next two partial sums—that is, S_4 and S_5. Analyze the results.

4. Use a computer algebra system to obtain the Taylor series given in (1)–(5), obtaining the final form at once without displaying all the derivatives.

5. Use two or more computer algebra systems to carry out Example 1.2.6 to 50 decimal places. Are their answers the same and correct to all digits obtained? Repeat using $\sqrt{x}$ expanded about $x_0 = 1$.

6. Use a computer algebra system to verify the results in Examples 1.2.7 and 1.2.9.

7. Design and carry out an experiment to check the computation of x^y on your computer.
Hint: Compare the computations of some examples, such as $32^{2.5}$ and $81^{1.25}$, to their correct values. A more elaborate test can be made by comparing single-precision results to double-precision results in various cases.

8. Verify that $x^y = e^{y\ln x}$. Try to find values of x and y for which these two expressions differ in your computer. Interpret the results.

9. (Continuation) For

$$\cos(x-y) = (\cos x)(\cos y) + (\sin x)(\sin y)$$

repeat the preceding computer problem.

10. The number of combinations of n distinct items taken m at a time is given by the **binomial coefficient**

$$\binom{n}{m} = \frac{n!}{m!\,(n-m)!}$$

for integers m and n, with $0 \leqq m \leqq n$.
Recall that $\binom{n}{0} = \binom{n}{n} = 1$.

a. Write

> **integer function** *ibin*(n, m)

which uses the definition above to compute $\binom{n}{m}$.

b. Verify the formula

$$\binom{n}{m} = \prod_{k=1}^{\min(m,n-m)} \left[\frac{n-k+1}{k}\right]$$

for computing the binomial coefficients. Write

> **integer function** *jbin*(n, m)

that is based on this formula.

c. Verify the formulas (**Pascal's triangle**)

$$\begin{cases} a_{i0} = a_{ii} = 1 \quad (0 \leqq i \leqq n) \\ a_{ij} = a_{i-1,\,j-1} + a_{i-1,\,j} \quad (2 \leqq i \leqq n,\ 1 \leqq j \leqq i-1) \end{cases}$$

Using Pascal's triangle, compute the binomial coefficients

$$\binom{i}{j} = a_{i,\,j} \qquad (0 \leqq i,\, j \leqq n)$$

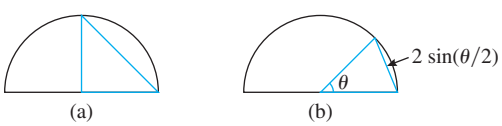
and store them in the lower triangular part of the array $(a_{ij})_{n\times n}$. Write

> **integer function** *kbin*(n, m)

that does an array look-up after first allocating and computing entries in the array.

11. The length of the curved part of a **unit semicircle** is π. We can approximate π by using triangles and elementary mathematics. Consider the semicircle with the arc bisected as in Figure (a). The hypotenuse of the right triangle is $\sqrt{2}$. Hence, a rough approximation to π is given by $2\sqrt{2} \approx 2.8284$. In Figure (b), we consider an angle θ that is a fraction $1/k$ of the semicircle. The secant shown has length $2\sin(\theta/2)$, and so an approximation to π is $2k\sin(\theta/2)$. From trigonometry, we have

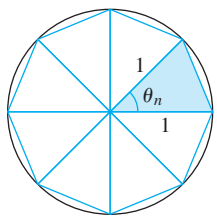
$$\sin^2\frac{1}{2}\theta = \frac{1}{2}(1-\cos\theta) = \frac{1}{2}\left(1-\sqrt{1-\sin^2\theta}\right)$$
$$= \frac{\sin^2\theta}{2+2\sqrt{1-\sin^2\theta}}$$



(a) (b)

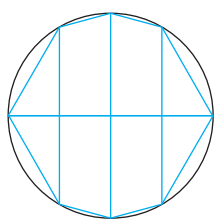
Now let θ_n be the angle that results from division of the semicircular arc into 2^{n-1} pieces. Next let $S_n = \sin^2\theta_n$ and $P_n = 2^n\sqrt{S_{n+1}}$. Show that $S_{n+1} = S_n/(2+2\sqrt{1-S_n})$ and P_n is an approximation to π. Starting with $S_2 = 1$ and $P_1 = 2$, compute S_{n+1} and P_n recursively for $2 \leqq n \leqq 20$.

12. The **irrational number** π can be computed by approximating the area of a unit circle as the limit of a sequence $p_1, p_2, \ldots$ described as follows. Divide the unit circle into 2^n sectors. (The figure shows the case $n = 3$.)



Approximate the area of the sector by the area of the isosceles triangle. The angle θ_n is $2\pi/2^n$. The area of the triangle is $\frac{1}{2}\sin\theta_n$. (Verify.) The nth approximation to π is then $p_n = 2^{n-1}\sin\theta_n$. Prove that $\sin\theta_n = \sin\theta_{n-1}/\{2[1+(1-\sin^2\theta_{n-1})^{1/2}]\}^{1/2}$ by means of well-known trigonometric identities. Use this recurrence relation to generate the sequences $\sin\theta_n$ and p_n $(3 \leqq n \leqq 20)$ starting with $\sin\theta_2 = 1$. Compare with the computation $4.0\arctan(1.0)$.

13. (Continuation) Calculate π by a method similar to that of the preceding computer problem, where the area of the unit circle is approximated by a sequence of trapezoids as illustrated by the figure.



[a] **14.** Write a routine in double or long double precision to implement the following algorithm for computing π.

> **integer** k; **real** a, b, c, d, e, f, g
> $a \leftarrow 0$
> $b \leftarrow 1$
> $c \leftarrow 1/\sqrt{2}$
> $d \leftarrow 0.25$
> $e \leftarrow 1$

```
for k = 1 to 5
    a ← b
    b ← (b + c)/2
    c ← √(ca)
    d ← d − e(b − a)^2
    e ← 2e
    f ← b^2/d
    g ← (b + c)^2/(4d)
    output k, f, |f − π|, g, |g − π|
end for
```

Which converges faster, f or g? How accurate are the final values? Also compare with the double- or long-double-precision computation of $4.0\arctan(1.0)$.
Hint: The value of π correct to 36 digits is

3.14159 26535 89793 23846 26433 83279 50288

Note: A new formula for computing π was discovered in the early 1970s. This algorithm is based on that formula, which is a direct consequence of a method developed by Gauss for calculating elliptic integrals and of Legendre's elliptic integral relation, both known for over 150 years! The error analysis shows that rapid convergence occurs in the computation of π, and the number of significant digits doubles after each step. (The interested reader should consult Brent [1976], Borwein and Borwein [1987], and Salamin [1976].)

15. Another **quadratically convergent** scheme for computing π was discovered by Borwein and Borwein [1984] and can be written as

```
integer k; real a, b, t, x
a ← √2
b ← 0
x ← 2 + √2
for k = 1 to 5
    t ← √a
    b ← t(1 + b)/(a + b)
    a ← (1/2)(t + 1/t)
    x ← xb(1 + a)/(1 + b)
    output k, x, |x − π|
end for
```

Numerically verify that $|x - \pi| \leqq 10^{-2k}$.
Note: Ludolf van Ceulen (1540–1610) was able to calculate π to 36 digits. With modern mathematical software packages such as MATLAB, Maple, and Mathematica, anyone can easily compute π to tens of thousands of digits in seconds!

[a] **16.** The **Fibonacci sequence** 1, 1, 2, 3, 5, 8, 13, 21, ... is defined by the linear recurrence relation

$$\begin{cases} \lambda_1 = 1, & \lambda_2 = 1 \\ \lambda_n = \lambda_{n-1} + \lambda_{n-2} & (n \geqq 3) \end{cases}$$

A formula for the nth **Fibonacci number** is

$$\lambda_n = \frac{1}{\sqrt{5}}\left\{\left[\frac{1}{2}\left(1+\sqrt{5}\right)\right]^n - \left[\frac{1}{2}\left(1-\sqrt{5}\right)\right]^n\right\}$$

Compute λ_n $(1 \leqq n \leqq 50)$, using both the recurrence relation and the formula. Write three programs that use integer, single-precision, and double-precision arithmetic, respectively. For each n, print the results using integer, single-precision, and double-precision formats, respectively.

[a] **17.** (Continuation) Repeat the experiment, using the sequence given by the recurrence relation

$$\begin{cases} \alpha_1 = 1, & \alpha_2 = \frac{1}{2}\left(1+\sqrt{5}\right) \\ \alpha_n = \alpha_{n-1} + \alpha_{n-2} & (n \geqq 3) \end{cases}$$

A closed-form formula is

$$\alpha_n = \left[\frac{1}{2}\left(1+\sqrt{5}\right)\right]^n$$

18. (Continuation) Change $+\sqrt{5}$ to $-\sqrt{5}$, and repeat the computation of α_n. Explain the results.

19. The **Bessel functions** J_n are defined by

$$J_n(x) = \frac{1}{\pi}\int_0^{\pi} \cos(x\sin\theta - n\theta)\, d\theta$$

Establish that $|J_n(x)| \leqq 1$.

a. It is known that

$$J_{n+1}(x) = 2nx^{-1}J_n(x) - J_{n-1}(x)$$

Use this recurrence relation to compute $J_0(1)$, $J_1(1), \ldots, J_{20}(1)$, starting from known values $J_0(1) \approx 0.76519\,76865$ and $J_1(1) \approx 0.44005\,05857$. Account for the fact that the inequality $|J_n(x)| \leqq 1$ is violated.

b. Another recursive relation is

$$J_{n-1}(x) = 2nx^{-1}J_n(x) - J_{n+1}(x)$$

Starting with the known values $J_{20}(1) \approx 3.87350\,3009 \times 10^{-25}$ and $J_{19}(1) \approx 1.54847\,8441 \times 10^{-23}$, use this equation to compute $J_{18}(1)$, $J_{17}(1), \ldots, J_1(1), J_0(1)$. Analyze the results.

20. A calculus student is asked to determine $\lim_{n\to\infty}(100^n/n!)$ and writes a program to evaluate $x_0, x_1, x_2, \ldots, x_n$ as follows:

```
integer parameter n ← 100
integer i; real x; x ← 1
for i = 1 to n
    x ← 100x/i
    output i, x
end for
```

The numbers printed become ever larger, and the student concludes that $\lim_{n\to\infty} x_n = \infty$. What is the moral here?

21. (**Maclaurin Series Function Approximations**) By using the truncated Maclaurin series, a function $f(x)$ with n continuous derivatives can be approximated by an nth-degree polynomial

$$f(x) \approx p_n(x) = \sum_{i=0}^{n} c_i x^i$$

where $c_i = f^{(i)}(0)/i!$.

a. Produce and compare computer plots for $f(x) = e^x$ and the polynomials $p_2(x)$, $p_3(x)$, $p_4(x)$, $p_5(x)$. Do the higher-order polynomials approximate the exponential function e^x satisfactorily on increasing intervals about zero?

b. Repeat for $g(x) = \ln(1 + x)$.

22. (Continuation) **Padé rational approximation** is the *best* approximation of a function by a rational function of a given order. Often it gives a better approximation of the function than truncating its Taylor series, and it may work even when the Taylor series does not converge! Consequently, the Padé rational approximations are frequently used in computer calculations such as for the basic function $\sin x$ as discussed in Computer Exercise 2.2.17. Rather than using high-order polynomials, we use ratios of low-order polynomials. These are called **rational approximations**. Let

$$f(x) \approx \frac{p_m(x)}{q_k(x)} = \frac{\sum_{i=0}^{m} a_i x^i}{\sum_{j=0}^{k} b_j x^j} = R_{m,k}(x)$$

where $b_0 = 1$. Here we have normalized with respect to $b_0 \neq 0$ and the values of m and k are modest. We choose the k coefficients b_j and the $m+1$ coefficients a_i in $R_{m,k}$ to match f and a specified number of its derivatives at the fixed point $x = 0$.

First, we construct the truncated Maclaurin series $\sum_{i=0}^{n} c_i x^i$ in which $c_i = f^{(i)}(0)/i!$ and $c_i = 0$ for $i < 0$. Next, we match the first $m + k + 1$ derivatives of $R_{m,k}$ with respect to x at $x = 0$ to the first $m+k+1$ coefficients c_i. This leads to the following displayed equations. Since $b_0 = 1$, we solve this $k \times k$ system of equations for $b_1, b_2, \ldots, b_k$

$$\begin{bmatrix} c_m & c_{m-1} & \cdots & c_{m-(k-2)} & c_{m-(k-1)} \\ c_{m+1} & c_m & \cdots & c_{m-(k-3)} & c_{m-(k-2)} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ c_{m+(k-2)} & c_{m+(k-3)} & \cdots & c_m & c_{m-1} \\ c_{m+(k-1)} & c_{m+(k-2)} & \cdots & c_{m+1} & c_m \end{bmatrix} \times \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_{k-1} \\ b_k \end{bmatrix} = \begin{bmatrix} -c_{m+1} \\ -c_{m+2} \\ \vdots \\ -c_{m+(k-1)} \\ -c_{m+k} \end{bmatrix}$$

(Solving systems of linear equations numerically is discussed in Chapter 2.) Finally, we evaluate these $m + 1$ equations for $a_0, a_1, \ldots, a_m$.

$$a_j = \sum_{\ell=0}^{j} c_{j-\ell} b_\ell \qquad (j = 0, 1, \ldots, m)$$

Note that $a_j = 0$ for $j > m$ and $b_j = 0$ for $j > k$. Also, if $k = 0$, then $R_{m,0}$ is a truncated Maclaurin series for f. Moreover, the Padé approximations may contain singularities.

a. Determine the **rational functions** $R_{1,1}(x)$ and $R_{2,2}(x)$. Produce and compare computer plots for $f(x) = e^x$, $R_{1,1}$, and $R_{2,2}$. Do these low-order rational functions approximate the exponential function e^x satisfactorily within $[-1, 1]$? How do they compare to the truncated Maclaurin polynomials of the preceding problem?

b. Repeat using $R_{2,2}(x)$ and $R_{3,1}(x)$ for the function $g(x) = \ln(1 + x)$.

Information on the life and work of the French mathematician **Herni Eugène Padé** (1863–1953) can be found in Wood [1999]. This reference also has examples and exercises similar to these. Further examples of Padé approximation can be seen.

23. (Continuation) Repeat for the **Bessel function** $J_0(2x)$, whose Maclaurin series is

$$1 - x^2 + \frac{x^4}{4} - \frac{x^6}{36} + \cdots = \sum_{i=0}^{\infty} (-1)^i \left(\frac{x^i}{i!}\right)^2$$

Then determine $R_{2,2}(x)$, $R_{4,3}(x)$, and $R_{2,4}(x)$ as well as comparing plots.

24. Carry out the details in the introductory example to this chapter by first deriving the Taylor series for $\ln(1 + x)$

and computing $\ln 2 \approx 0.63452$ using the first eight terms. Then establish the series $\ln[(1+x)/(1-x)]$ and calculate $\ln 2 \approx 0.69313$ using the terms shown. Determine the absolute error and relative errors for each of these answers.

25. Reproduce Figure 1.3 using your computer as well as adding the curve for S_4.

26. Use a mathematical software system that does symbolic manipulations such as Maple or Mathematica to carry out

a. Example 1.2.3 **b.** Example 1.2.6

27. Can you obtain the following numerical results?

$$\sqrt{1.00001} = 1.00000\,49999\,87500\,06249\,96093\,77734\,37500\,0000$$

$$\sqrt{0.99999} = 0.99999\,49999\,87499\,93749\,96093\,72265\,62500\,00000$$

Are these answers accurate to all digits shown?

28. Sometimes the values of a Taylor series cannot be easily reformulated. For 15 values of $x = 1, 0.1, 0.01, \ldots$, compare the following.

a. $(-1+e^x)/x$ versus eight terms in the truncated Taylor series.

b. $(1-\cos x)/x^2$ versus $\sin^2 x/[x^2+\cos x]$.

1.3 Floating-Point Representation

The standard way to represent a nonnegative real number in decimal form is with an **integer part** and a **fractional part** with a **decimal point** between them such as

$$37.21829, \qquad 0.00227\,1828, \qquad 30\,00527.11059$$

(We group five digits together as shown.)

Another standard form, often called **normalized scientific notation**, is obtained by shifting the decimal point and supplying appropriate powers of 10. Thus, the preceding numbers have alternative representations as

$$37.21829 = 0.37218\,29 \times 10^2$$
$$0.00227\,1828 = 0.22718\,28 \times 10^{-2}$$
$$30\,00527.11059 = 0.30005\,27110\,59 \times 10^7$$

In normalized scientific notation, the number is represented by a fraction multiplied by a power of 10, and the leading digit in the fraction is *not* zero (except when the number involved *is* zero). Thus, we write 79325 as 0.79325×10^5, *not* $0.07932\,5 \times 10^6$ or 7.9325×10^4 or some other way.

Normalized Floating-Point Representation

In the context of computer science, normalized scientific notation is also called **normalized floating-point representation**. In the decimal system, any real number x (other than zero) can be represented in normalized floating-point form as

$$x = \pm 0.d_1 d_2 d_3 \ldots \times 10^n$$

where $d_1 \neq 0$ and n is an integer (positive, negative, or zero). Each of the numbers d_1, d_2, $d_3, \ldots$ are the decimal digits 0, 1, 2, 3, 4, 5, 6, 7, 8, or 9.

Stated another way, the real number x, if different from zero, can be represented in normalized floating-point decimal form as

$$x = \pm r \times 10^n \qquad \left(\tfrac{1}{10} \leqq r < 1\right)$$

This representation consists of three parts: a sign that is either $+$ or $-$, a number r in the interval $\left[\frac{1}{10}, 1\right)$, and an integer power of 10. The number r is called the **normalized mantissa** and n the **exponent**.

The floating-point representation in the binary system is similar to that in the decimal system in several ways. If $x \neq 0$, it can be written as

$$x = \pm q \times 2^m \qquad \left(\tfrac{1}{2} \leqq q < 1\right)$$

The mantissa q would be expressed as a sequence of zeros or ones in the form $q = (0.b_1b_2b_3\ldots)_2$, where $b_1 \neq 0$. Hence, $b_1 = 1$ and then necessarily $q \geqq \frac{1}{2}$ and $q < 1$.

A floating-point number system within a computer is similar to what we have just described, with one important difference: Every computer has only a finite word length and a finite total capacity, so only numbers with a finite number of digits can be represented. A number is allotted only one word of storage in the single-precision mode (two or more words in double or long-double precision). In either case, the degree of precision is strictly limited. Clearly, irrational numbers cannot be represented, nor can those rational numbers that do *not* fit the finite format imposed by the computer. Furthermore, numbers may be either too large or too small to be representable. The real numbers that are representable in a computer are called its **machine numbers**.

Since any number used in calculations with a computer must conform to the format of numbers in that computer system, it must have a **finite expansion**. Numbers that have a nonterminating expansion cannot be accommodated precisely. Moreover, a number that has a terminating expansion in one base may have a nonterminating expansion in another. A good example of this is the following simple fraction as given in the introductory example to this chapter:

$$\begin{aligned}\frac{1}{10} = (0.1)_{10} &= (0.06314\,6314\,6314\,6314\ldots)_8 \\ &= (0.0\,0011\,0011\,0011\,0011\,0011\,0011\,0011\,0011\ldots)_2\end{aligned}$$

The important point here is that most real numbers cannot be represented exactly in a computer. (See Appendix B for a discussion of representation of numbers in different bases.)

We illustrate that the effective number system for a computer is *not* a continuum, but a rather peculiar discrete set.

EXAMPLE 1 List all the floating-point numbers that can be expressed in the form

$$x = \pm(0.b_1b_2b_3)_2 \times 2^{\pm k}$$

where b_1, b_2, b_3, and m are allowed to have only the value 0 or 1. Then allow only *normalized* floating-point numbers; that is, all numbers (with the exception of zero) having the form

$$x = \pm(0.b_1b_2b_3)_2 \times 2^{\pm k}$$

Solution There are two choices for the $\pm$, two choices for b_1, two choices for b_2, two choices for b_3, and three choices for the exponent. Thus, at first, one would expect

$$2 \times 2 \times 2 \times 2 \times 3 = 48$$

different numbers. For example, all of the possible nonnegative numbers in this system are as follows:

$$
\begin{aligned}
(0.000)_2 \times 2^{-1} &= 0, & (0.000)_2 \times 2^{0} &= 0, & (0.000)_2 \times 2^{1} &= 0 \\
(0.001)_2 \times 2^{-1} &= \frac{1}{16}, & (0.001)_2 \times 2^{0} &= \frac{1}{8}, & (0.001)_2 \times 2^{1} &= \frac{1}{4} \\
(0.010)_2 \times 2^{-1} &= \frac{2}{16}, & (0.010)_2 \times 2^{0} &= \frac{2}{8}, & (0.010)_2 \times 2^{1} &= \frac{2}{4} \\
(0.011)_2 \times 2^{-1} &= \frac{3}{16}, & (0.011)_2 \times 2^{0} &= \frac{3}{8}, & (0.011)_2 \times 2^{1} &= \frac{3}{4} \\
(0.100)_2 \times 2^{-1} &= \frac{4}{16}, & (0.100)_2 \times 2^{0} &= \frac{4}{8}, & (0.100)_2 \times 2^{1} &= \frac{4}{4} \\
(0.101)_2 \times 2^{-1} &= \frac{5}{16}, & (0.101)_2 \times 2^{0} &= \frac{5}{8}, & (0.101)_2 \times 2^{1} &= \frac{5}{4} \\
(0.110)_2 \times 2^{-1} &= \frac{6}{16}, & (0.110)_2 \times 2^{0} &= \frac{6}{8}, & (0.110)_2 \times 2^{1} &= \frac{6}{4} \\
(0.111)_2 \times 2^{-1} &= \frac{7}{16}, & (0.111)_2 \times 2^{1} &= \frac{7}{4}, & (0.111)_2 \times 2^{0} &= \frac{7}{8}
\end{aligned}
$$

Here there are many duplications! So we obtain only these nonnegative numbers $\frac{1}{16}, \frac{3}{16}, \frac{5}{16}, \frac{7}{16}; \frac{1}{8}, \frac{3}{8}, \frac{5}{8}, \frac{7}{8}; \frac{1}{4}, \frac{3}{4}, \frac{5}{4}, \frac{7}{4}; \frac{1}{2}, \frac{3}{2}; 0, 1$. Altogether there are 31 distinct numbers in the system. The nonnegative numbers obtained are shown as dots on a line in Figure 1.6.

FIGURE 1.6 Example 1: Nonnegative machine numbers

Observe that the numbers are symmetrically, but unevenly distributed, about zero.

Now allowing only normalized floating-point numbers ($b_2 = 1$), we cannot represent $\frac{1}{16}$, $\frac{1}{8}$, and $\frac{3}{16}$. Hence, there are only 25 distinct numbers, and the nonnegative machine numbers are now distributed as in Figure 1.7.

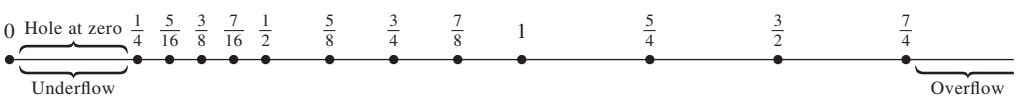


FIGURE 1.7 Example 1: Nonnegative normalized machine numbers

If, in the course of a computation, a number x is produced which is outside the computer's permissible range, then we say that an **overflow** has occurred or that x is **outside the range of the computer**. Generally, an overflow results in a fatal error (or **exception**), and the normal execution of the program stops! An **underflow** is usually treated automatically by setting x to zero without any interruption of the program and without a warning message in most computers.

In a computer whose floating-point numbers are restricted to the form in Example 1, any number closer to zero than $\frac{1}{4}$ would *underflow* to zero, and any number outside the range to the left of -1.75 to the right of $+1.75$ would *overflow* to machine $\pm$ infinity, respectively. Notice that there is a relatively wide gap between zero and the smallest positive machine number, which is $(0.100)_2 \times 2^{-1} = \frac{1}{4}$. This creates a phenomenon known as the

hole at zero. Figure 1.7 illustrates concepts such as the hole-at-zero, underflow, and overflow for Example 1.

We can store the normalized floating-point numbers from Example 1 in a five-bit computer with one bit for the sign of the number, two bits for the exponent, and two bits for the mantissa:

$$\boxed{\pm}\boxed{e_1}\boxed{e_2}\boxed{b_2}\boxed{b_3}$$

All possible combinations of positive normalized floating-point numbers are

$$(0.1b_2b_3)_2 \times 2^m = \left\{ \begin{array}{l} (0.100)_2 = \frac{1}{2} \\ (0.101)_2 = \frac{5}{8} \\ (0.110)_2 = \frac{3}{4} \\ (0.111)_2 = \frac{7}{8} \end{array} \right\} \times 2^{-1,0,1} = \left\{ \begin{array}{l} \frac{1}{4}, \frac{1}{2}, 1 \\ \frac{5}{16}, \frac{5}{8}, \frac{5}{4} \\ \frac{3}{8}, \frac{3}{4}, \frac{3}{2} \\ \frac{7}{16}, \frac{7}{8}, \frac{7}{4} \end{array} \right.$$

A machine number in floating-point single-precision is of the form

$$(-1)^s q \times 2^m = (-1)^s \times 2^{c-1} \times (1.b_2b_3)_2$$

Here we set the exponent to $m = c - 1$. (In a 32-bit computer with an 8-bit exponent, $m = c - 127$. Why?) Some special cases are for ± 0, $\pm\infty$, and so on.

Floating-Point Representation

Before the Standard for Floating-Point Arithmetic (IEEE-754) was established in the early 1980s, computers used many different forms of floating-point representation, differing in the word length, the format of the representation, and the rounding used between operations! Now IEEE-754 has been accepted by almost all hardware and software manufactures worldwide. It defines the floating-point number system used by computers and it offers several rounding schemes, which affects the accuracy, among other things. In most computers, there are three common levels of precision for floating-point numbers, with the number of bits allocated for each organized as shown in the following table.

Precision	Bits	Sign	Exponent	Mantissa
Single	32	1	8	23
Double	64	1	11	52
Long Double	80	1	15	64

A computer that operates in floating-point mode represents numbers as described earlier except for the limitations imposed by the finite word length. Many binary computers have a word length of 32 or 64 bits (binary digits). We shall describe a machine of this type whose features mimic many workstations and personal computers in widespread use. The internal representation of numbers and their storage is **standard floating-point form**, which is used in almost all computers. For simplicity, we have left out a discussion of some of the details and features. Fortunately, one need not know all the details of the floating-point arithmetic system used in a computer to use it intelligently. Nevertheless, it is generally helpful in debugging a program to have a basic understanding of the representation of numbers in your computer.

By **single-precision floating-point numbers**, we mean all acceptable numbers in a computer using the standard single-precision floating-point arithmetic format. (In this discussion, we are assuming that such a computer stores these numbers in 32-bit words.) This set is a finite subset of the real numbers. It consists of ± 0, $\pm\infty$, normal and subnormal single-precision floating-point numbers, and even NotaNumber (`NaN`) values. (More detail on these subjects are in Appendix B and in the references.)

Recall that most real numbers *cannot* be represented exactly as floating-point numbers, since they have infinite decimal or binary expansions (all irrational numbers and some rational numbers); for example, π, e, $\frac{1}{3}$, 0.1, and so on.

Because of the 32-bit word-length, as much as possible of the normalized floating-point number

$$\pm q \times 2^m$$

must be contained in those 32 bits. One way of allocating the 32 bits is as follows:

sign of q	1 bit
integer $\lvert m \rvert$	8 bits
number q	23 bits

Information on the sign of m is contained in the 8 bits allocated for the integer $|m|$. In such a scheme, we can represent real numbers with $|m|$ as large as $2^7 - 1 = 127$. The exponent represents numbers from -127 through 128.

Single-Precision Floating-Point Form

We now describe a machine number of the following form in **standard single-precision floating-point** representation:

$$(-1)^{\mathrm{s}} \times 2^{\mathrm{c}-127} \times (1.\mathrm{f})_2$$

The leftmost bit is used for the sign of the mantissa, where $\mathrm{s} = 0$ corresponds to $+$ and $\mathrm{s} = 1$ corresponds to $-$. The next 8 bits are used to represent the number c in the exponent of $2^{\mathrm{c}-127}$, which is interpreted as an *excess-127 code*. Finally, the last 23 bits represent f from the fractional part of the mantissa in the 1-plus form: $(1.\mathrm{f})_2$. Each floating-point single-precision word is partitioned as in Figure 1.8.

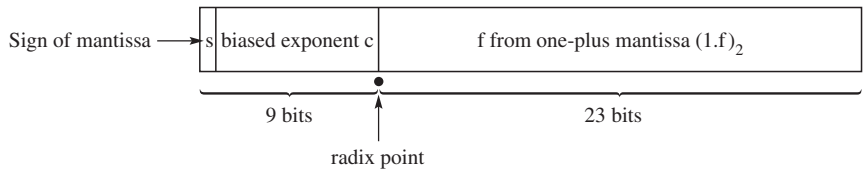


FIGURE 1.8 Partitioned floating-point single-precision computer word

In the normalized representation of a nonzero floating-point number, the first bit in the mantissa is *always* 1 so that this bit does not have to be stored. This can be accomplished by shifting the binary point to a "1-plus" form $(1.\mathrm{f})_2$. The mantissa is the rightmost 23 bits and contains f with an understood binary point as in Figure 1.8. So the mantissa **(significand)** *actually* corresponds to 24 binary digits since there is a **hidden bit**. (An important exception is the number ± 0.)

We now outline the procedure for determining the representation of a real number x. If x is zero, it is represented by a full word of zero bits with the possible exception of the sign

bit. For a nonzero x, first assign the sign bit for x and consider $|x|$. Then convert both the integer and fractional parts of $|x|$ from decimal to binary. Next one-plus normalize $(|x|)_2$ by shifting the binary point so that the first bit to the left of the binary point is a 1 and all bits to the left of this 1 are 0. To compensate for this shift of the binary point, adjust the exponent of 2; that is, multiply by the appropriate power of 2. The 24-bit one-plus-normalized mantissa in binary is thus found. Now the current exponent of 2 should be set equal to $\mathrm{c} - 127$ to determine c, which is then converted from decimal to binary. The sign bit of the mantissa is combined with $(\mathrm{c})_2$ and $(\mathrm{f})_2$. Finally, write the 32-bit representation of x as eight hexadecimal digits.

The value of c in the representation of a floating-point number in single precision is restricted by the inequality

$$0 < \mathrm{c} < (11\,111\,111)_2 = 255$$

The values 0 and 255 are reserved for special cases, including ± 0 and $\pm\infty$, respectively. Hence, the actual exponent of the number is restricted by the inequality

$$-126 \leqq \mathrm{c} - 127 \leqq 127$$

Likewise, we find that the mantissa of each nonzero number is restricted by the inequality

$$1 \leqq (1.\mathrm{f})_2 \leqq (1.111\,111\,111\,111\,111\,111\,111\,11)_2 = 2 - 2^{-23}$$

The largest number representable is therefore $(2 - 2^{-23})2^{127} \approx 2^{128} \approx 3.4 \times 10^{38}$. The smallest positive number is $2^{-126} \approx 1.2 \times 10^{-38}$.

The binary machine floating-point number $\varepsilon = 2^{-24}$ is called the **machine epsilon** when using single precision. It is the smallest positive machine number ε such that $1 + \varepsilon \neq 1$. Because $2^{-24} \approx 5.96 \times 10^{-8}$, we infer that in a simple computation, approximately **7 significant decimal digits** of accuracy may be obtained in single precision. Recall that 23 bits are allocated for the mantissa plus the hidden bit.

Double-Precision Floating-Point Form

When more precision is needed, **double precision** can be used, in which case each double-precision floating-point number is stored in two computer words in memory. In double precision, there are 52 bits allocated for the mantissa. The double precision **machine epsilon** is $2^{-53} \approx 1.11 \times 10^{-16}$, so approximately **15 significant decimal digits** of precision are available. There are 11 bits allowed for the exponent, which is biased by 1023. The exponent represents numbers from -1022 through 1023. A machine number in **standard double-precision floating-point form** corresponds to

$$(-1)^{\mathrm{s}} \times 2^{\mathrm{c}-1023} \times (1.\mathrm{f})_2$$

The leftmost bit is used for the sign of the mantissa with $\mathrm{s} = 0$ for $+$ and $\mathrm{s} = 1$ for $-$. The next 11 bits are used to represent the exponent c corresponding to $2^{\mathrm{c}-1023}$. Finally, 52 bits represent f from the fractional part of the mantissa in the one-plus form: $(1.\mathrm{f})_2$.

The value of c in the representation of a floating-point number in double precision is restricted by the inequality

$$0 < \mathrm{c} < (1\,111\,111\,111)_2 = 2047$$

As in single precision, the values at the ends of this interval are reserved for special cases. Hence, the actual exponent of the number is restricted by the inequality

$$-1022 \leqq \mathrm{c} - 1023 \leqq 1023$$

We find that the mantissa of each nonzero number is restricted by the inequality

$$1 \leqq (1.\mathrm{f})_2 \leqq (1.111\,111\,111\,\cdots\,111\,111\,111\,1)_2 = 2 - 2^{-52}$$

Because $2^{-52} \approx 1.2 \times 10^{-16}$, we infer that in a simple computation approximately 15 significant decimal digits of accuracy may be obtained in double precision.

Recall that 52 bits are allocated for the mantissa. The largest double-precision machine number is $(2 - 2^{-52})2^{1023} \approx 2^{1024} \approx 1.8 \times 10^{308}$. The smallest double-precision positive machine number is $2^{-1022} \approx 2.2 \times 10^{-308}$.

Single precision on a 64-bit computer is comparable to double precision on a 32-bit computer, whereas double precision on a 64-bit computer gives four times the precision available on a 32-bit computer.

In single precision, 31 bits are available for an integer because only 1 bit is needed for the sign. Consequently, the range for integers is from $-(2^{31}-1)$ to $(2^{31}-1) = 2147483647$. In double precision, 63 bits are used for integers, giving integers in the range $-(2^{63} - 1)$ to $(2^{63} - 1)$. In using integer arithmetic, accurate calculations can result in only approximately 9 digits in single precision and 18 digits in double precision! For high accuracy, most computations should be done by using double-precision floating-point arithmetic.

At this point, some students may wish to read Appendix B for a review of representing numbers in different bases.

EXAMPLE 2 Determine the machine representation of the decimal number -52.234375 in both single precision and double precision.

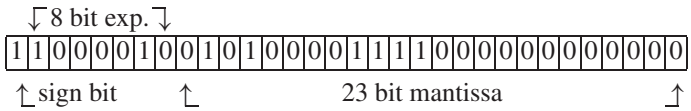
Solution Converting the integer part to binary, we have $(52.)_{10} = (64.)_8 = (110\,100.)_2$. Next, converting the fractional part, we have $(.234375)_{10} = (.17)_8 = (.001\,111)_2$. Now

$$(52.234375)_{10} = (110\,100.001\,111)_2 = (1.101\,000\,011\,110)_2 \times 2^5$$

is the corresponding one-plus form in base 2, and $(.101\,000\,011\,110)_2$ is the stored mantissa. Next the exponent is $(5)_{10}$, and since $c - 127 = 5$, we immediately see that $(132)_{10} = (204)_8 = (10\,000\,100)_2$ is the stored exponent. Thus, the single-precision machine representation of -52.234375 is

$$[1\,10\,000\,100\,101\,000\,011\,110\,000\,000\,000\,00]_2 =$$
$$[1100\,0010\,0101\,0000\,1111\,0000\,0000\,0000]_2 = [\mathrm{C250F000}]_{16}$$

Here is the bit pattern for -52.234375 in single-precision floating-point using 32 bits:

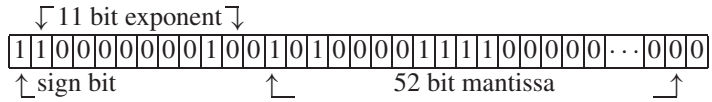


In double precision, for the exponent $(5)_{10}$, we let $c - 1023 = 5$, and we have $(1028)_{10} = (2004)_8 = (10\,000\,000\,100)_2$, which is the stored exponent. Thus, the double-precision machine representation of -52.234375 is

$$[1\,10\,000\,000\,100\,101\,000\,011\,110\,000\,\cdots\,00]_2 =$$
$$[1100\,0000\,0100\,1010\,0001\,1110\,0000\,\cdots\,0000]_2 = [\mathrm{C04A1E0000000000}]_{16}$$

Here $[\cdots]_k$ is the bit pattern of the machine word(s) that represents floating-point numbers, which is displayed in base-k. Here is the bit pattern for -52.234375 in double-precision

floating-point using 64 bits:



Mathematical software can be used to display the range of the numbers in a computer. In MATLAB, commands `realmin('single')` and `realmax('single')` return the smallest and largest finite floating-point numbers in single precision—double precision is similar. Here is a table of the range of floating-point normalized numbers with the hole at zero. Because this number range is not a continuum, there are lots of holes or gaps (*discontinuities*) throughout.

	Range
Single Precision	$-2^{128} \approx -3.4 \times 10^{38} \leqq x \leqq -1.2 \times 10^{-38} \approx -2^{-126}$
	0
	$2^{-126} \approx 1.2 \times 10^{-38} \leqq x \leqq 3.4 \times 10^{38} \approx 2^{128}$
Double Precision	$-2^{1024} \approx -1.8 \times 10^{318} \leqq x \leqq -2.2 \times 10^{-308} \approx 2^{-1022}$
	0
	$2^{-1022} \approx 2.2 \times 10^{-308} \leqq x \leqq 1.8 \times 10^{308} \approx 2^{1024}$

EXAMPLE 3 Determine the decimal numbers that correspond to these machine words:

$$[45DE4000]_{16} \qquad [BA390000]_{16}$$

Solution The first number in binary is

$$[0100\,0101\,1101\,1110\,0100\,0000\,0000\,0000]_2$$

The stored exponent is $(10\,001\,011)_2 = (213)_8 = (139)_{10}$, so $139 - 127 = 12$. The mantissa is positive and represents the number

$$\begin{aligned}(1.101\,111\,001)_2 \times 2^{12} &= (1\,101\,111\,001\,000.)_2 \\ &= (15710.)_8 \\ &= 0 \times 1 + 1 \times 8 + 7 \times 8^2 + 5 \times 8^3 + 1 \times 8^4 \\ &= 8(1 + 8(7 + 8(5 + 8(1)))) \\ &= 7112\end{aligned}$$

Similarly, the second word in binary is

$$[1011\,1010\,0011\,1001\,0000\,0000\,0000\,0000]_2$$

The exponential part of the word is $(01\,110\,100)_2 = (164)_8 = 116$, so the exponent is $116 - 127 = -11$. The mantissa is negative and corresponds to the following floating-point

number:

$$\begin{aligned} -(1.011\,100\,100)_2 \times 2^{-11} &= -(0.000\,000\,000\,010\,111\,001)_2 \\ &= -(0.00027\,1)_8 \\ &= -2 \times 8^{-4} - 7 \times 8^{-5} - 1 \times 8^{-6} \\ &= -8^{-6}(1 + 8(7 + 8(2))) \\ &= -\frac{185}{262144} \approx -7.05718\,99 \times 10^{-4} \end{aligned}$$

■

Computer Errors in Representing Numbers

We turn now to the errors that can occur when we attempt to represent a given real number x in the computer. We use a model computer with a 32-bit word length. Suppose first that we let $x = 2^{53\,21697}$ or $x = 2^{-32591}$. The exponents of these numbers far exceed the limitations of the machine (as described above). These numbers would overflow and underflow, respectively, and the relative error in replacing x by the closest machine number would be very large. Such numbers are *outside the range* of a 32-bit word-length computer.

Consider next a positive real number x in normalized floating-point form

$$x = q \times 2^m \qquad \left(\tfrac{1}{2} \leqq q < 1,\ -125 \leqq m \leqq 128\right)$$

The process of replacing x by its nearest machine number is called **correct rounding**, and the error involved is called **roundoff error**. We want to know how large it can be. We suppose that q is expressed in normalized binary notation, so

$$x = (0.1b_2b_3b_4 \ldots b_{24}b_{25}b_{26} \ldots)_2 \times 2^m$$

One nearby machine number can be obtained by *rounding down* or by simply dropping the excess bits $b_{25}b_{26} \ldots$, since only 23 bits have been allocated to the stored mantissa. This machine number is

$$x_- = (0.1b_2b_3b_4 \ldots b_{24})_2 \times 2^m$$

It lies to the left of x on the real-number axis. Another machine number, x_+, is just to the right of x on the real axis and is obtained by *rounding up*. It is found by adding one unit to b_{24} in the expression for x_-. Thus, we have

$$x_+ = \left[(0.1b_2b_3b_4 \ldots b_{24})_2 + 2^{-24}\right] \times 2^m$$

FIGURE 1.9
A possible relationship between x_-, x_+, and x.

The closer of these machine numbers is the one chosen to represent x.

The two situations are illustrated by the simple diagrams in Figure 1.9. If x lies closer to x_- than to x_+, then

$$|x - x_-| \leqq \tfrac{1}{2}|x_+ - x_-| = 2^{-25+m}$$

In this case, the relative error is bounded as follows:

$$\left|\frac{x - x_-}{x}\right| \leqq \frac{2^{-25+m}}{(0.1b_2b_3b_4 \ldots)_2 \times 2^m} \leqq \frac{2^{-25}}{1/2} = 2^{-24} = u$$

where $u = 2^{-24}$ is the **unit roundoff error** for a 32-bit binary computer with standard floating-point arithmetic.

Mathematical software can be used to display the precision of a computer. For example in MATLAB, command `eps('single')` returns the distance from 1.0 to the next largest single-precision floating-point number—double precision is similar.

	Precision
Single Precision	$2^{-23} \approx 1.2 \times 10^{-7}$
Double Precision	$2^{-52} \approx 2.2 \times 10^{-16}$

Recall that machine epsilon is $\varepsilon = 2^{-24}$, so $u = \varepsilon$. Moreover, $u = 2^{-k}$, where k is the number of binary digits used in the mantissa, including the hidden bit ($k = 24$ in single precision and $k = 53$ in double precision). On the other hand, if x lies closer to x_+ than to x_-, then

$$|x - x_+| \leqq \tfrac{1}{2}|x_+ - x_-|$$

and the same analysis shows that the relative error is no greater than $2^{-24} = u$. So in the case of rounding to the nearest machine number, the relative error is bounded by u. We note in passing that when *all* excess digits or bits are discarded, the process is called **chopping**. If a 32-bit word-length computer has been designed to chop numbers, the relative error bound would be twice as large as above.

The terms **machine epsilon** (ϵ) and **unit roundoff error** (u) are used interchangeably. Machine epsilon is used to study the effect of rounding errors because the actual errors of machine arithmetic are extremely complicated. Program libraries may provide precomputed values for these and other standard numerical quantities.

Often students are assigned the textbook exercise to compute an approximate value for machine epsilon. It is done in the sense of the spacing of the floating-point numbers at 1 rather than in the sense of the unit roundoff error. The following pseudocode produces an approximation to machine epsilon (within a factor of 2).

Machine Epsilon Pseudocode

```
epsi ← 1.0
while (1.0 + eps ≧ 1.0)
    epsi ← epsi/2.0
end for
epsi ← 2.0 * epsi
```

As with any computational results, it depends on the particular computer platform used as well as the programming language, the floating-point format (float, double, long double, etc.), and the runtime library.

Notation fl(x) and Backward Error Analysis

Next let us turn to the errors that are produced in the course of elementary arithmetic operations. To illustrate the principles, suppose that we are working with a five-place decimal machine and wish to add numbers. Two typical machine numbers in normalized

floating-point form are

$$x = 0.37218 \times 10^4, \qquad y = 0.71422 \times 10^{-1}$$

Many computers perform arithmetic operations in a double-length work area, so assume that our computer has a ten-place accumulator. First, the exponent of the smaller number is adjusted so that both exponents are the same. Then the numbers are added in the accumulator, and the rounded result is placed in a computer word:

$$\begin{aligned} x &= 0.37218\,00000 \times 10^4 \\ y &= 0.00000\,71422 \times 10^4 \\ \hline x + y &= 0.37218\,71422 \times 10^4 \end{aligned}$$

The nearest machine number is $z = 0.37219 \times 10^4$, and the relative error involved in this machine addition is

$$\frac{|x + y - z|}{|x + y|} = \frac{0.00000\,28578 \times 10^4}{0.37218\,71422 \times 10^4} \approx 0.77 \times 10^{-5}$$

This relative error would be regarded as acceptable on a machine of such low precision.

To facilitate the analysis of such errors, it is convenient to introduce the notation $\text{fl}(x)$ to denote the **floating-point machine number** that corresponds to the real number x. Of course, the function fl depends on the particular computer involved. Our hypothetical five-decimal-digit machine would give

$$\text{fl}(0.37218\,71422 \times 10^4) = 0.37219 \times 10^4$$

For a 32-bit word-length computer, we established previously that if x is any real number within the range of the computer, then

$$\frac{|x - \text{fl}(x)|}{|x|} \leqq u \qquad \left(u = 2^{-24}\right) \tag{1}$$

Here and throughout, we assume that correct rounding is used. This inequality can also be expressed in the more useful form

$$\text{fl}(x) = x(1 + \delta) \qquad \left(|\delta| \leqq 2^{-24}\right)$$

To see that these two inequalities are equivalent, simply let $\delta = [\text{fl}(x) - x]/x$. Then, by Inequality (1), we have $|\delta| \leqq 2^{-24}$ and solving for $\text{fl}(x)$ yields $\text{fl}(x) = x(1 + \delta)$.

By considering the details in the addition $1+\varepsilon$, we see that if $\varepsilon \geqq 2^{-23}$, then $\text{fl}(1+\varepsilon) > 1$, whereas if $\varepsilon < 2^{-23}$, then $\text{fl}(1 + \varepsilon) = 1$. Consequently, if **machine epsilon** is the smallest positive machine number ε such that

Machine Epsilon

$$\text{fl}(1 + \varepsilon) > 1$$

then $\varepsilon = 2^{-23}$. Sometimes it is necessary to furnish the machine epsilon to a program. Because it is a machine-dependent constant, it can be found by either calling a system routine or by writing a simple program that finds the smallest positive number $x = 2^m$ such that $1 + x > 1$ in the machine.

Now let the symbol $\odot$ denote any one of the arithmetic operations $+$, $-$, $\times$, or $\div$. Suppose a 32-bit word-length computer has been designed so that whenever two *machine numbers* x and y are to be combined arithmetically, the computer produces $\text{fl}(x \odot y)$ instead of $x \odot y$. We can imagine that $x \odot y$ is first *correctly* formed, then normalized, and finally rounded to become a machine number. Under this assumption, the relative error does not

exceed 2^{-24} by the previous analysis:

$$\text{fl}(x \odot y) = (x \odot y)(1 + \delta) \qquad \left(|\delta| \leqq 2^{-24}\right)$$

Special cases of this are, of course,

fl Properties

$$\text{fl}(x \pm y) = (x \pm y)(1 + \delta)$$
$$\text{fl}(xy) = xy(1 + \delta)$$
$$\text{fl}\left(\frac{x}{y}\right) = \left(\frac{x}{y}\right)(1 + \delta)$$

In these equations, δ is variable but satisfies $-2^{-24} \leqq \delta \leqq 2^{-24}$. The assumptions that we have made about a model 32-bit word-length computer are not quite true for a real computer. For example, it is possible for x and y to be machine numbers and for $x \odot y$ to overflow or underflow. Nevertheless, the assumptions should be realistic for most computing machines.

The equations given can be written in a variety of ways, some of which suggest alternative interpretations of roundoff. For example, we have

$$\text{fl}(x + y) = x(1 + \delta) + y(1 + \delta)$$

This says that the result of adding machine numbers x and y is not in general $x + y$ but is the true sum of $x(1 + \delta)$ and $y(1 + \delta)$. We can think of $x(1 + \delta)$ as the result of slightly perturbing x. Thus, the machine version of $x + y$, which is $\text{fl}(x + y)$, is the *exact* sum of a slightly perturbed x and a slightly perturbed y. The reader can supply similar interpretations in the examples given in the exercises.

This interpretation is an example of **backward error analysis**. It attempts to determine what perturbation of the original data would cause the *computer results* to be the exact results for a perturbed problem. In contrast, a **direct error analysis** attempts to determine how computed answers differ from exact answers based on the same data. In this aspect of scientific computing, computers have stimulated a new way of looking at computational errors.

Because the set of machine numbers is finite, some of the basic mathematical operations are not well defined and may breakdown in floating-point arithmetic. For example, we have

$$\text{fl}(x) = x(1 + \epsilon)$$

in which $|\epsilon| \leqq \epsilon_m$ where ϵ_m is **machine epsilon**, which is the smallest number such that $\text{fl}(1 + \epsilon_m) \neq 1$. Moreover, we obtain

$$\text{fl}(x \odot y) = (x \odot y)(1 + \epsilon_{\odot})$$

where $|\epsilon_{\odot}| \leqq \epsilon_m$ for the operations $\odot = +, -, *, /$ and

$$\text{fl}(x \odot y) = \text{fl}(y \odot x)$$

for operations $\odot = +, *$.

EXAMPLE 4 If x, y, and z are machine numbers in a 32-bit word-length computer, what upper bound can be given for the relative roundoff error in computing $z(x + y)$?

Solution In the computer, the calculation of $x + y$ would be done first. This arithmetic operation produces the machine number $\text{fl}(x + y)$, which differs from $x + y$ because of roundoff. By the principles established above, there is a δ_1 such that

$$\text{fl}(x + y) = (x + y)(1 + \delta_1) \qquad \left(|\delta_1| \leqq 2^{-24}\right)$$

When z multiplies the machine number $\text{fl}(x+y)$, the result is the machine number $\text{fl}[z\ \text{fl}(x+y)]$ because z is already a machine number. This, too, differs from its exact counterpart, and we have, for some δ_2,

$$\text{fl}[z\ \text{fl}(x+y)] = z\ \text{fl}(x+y)(1+\delta_2) \qquad \left(|\delta_2| \leqq 2^{-24}\right)$$

Putting both of our equations together, we have

$$\begin{aligned}\text{fl}[z\ \text{fl}(x+y)] &= z(x+y)(1+\delta_1)(1+\delta_2)\\ &= z(x+y)(1+\delta_1+\delta_2+\delta_1\delta_2)\\ &\approx z(x+y)(1+\delta_1+\delta_2)\\ &= z(x+y)(1+\delta) \qquad \left(|\delta| \leqq 2^{-23}\right)\end{aligned}$$

In this calculation, $|\delta_1\delta_2| \leqq 2^{-48}$, and so we ignore it. Also, we put $\delta = \delta_1 + \delta_2$ and then reason that $|\delta| = |\delta_1 + \delta_2| \leqq |\delta_1| + |\delta_2| \leqq 2^{-24} + 2^{-24} = 2^{-23}$. ■

EXAMPLE 5 Critique the following attempt to estimate the relative roundoff error in computing the sum of two real numbers, x and y. In a 32-bit word-length computer, the calculation yields

$$\begin{aligned}z &= \text{fl}[\text{fl}(x) + \text{fl}(y)]\\ &= [x(1+\delta) + y(1+\delta)](1+\delta)\\ &= (x+y)(1+\delta)^2\\ &\approx (x+y)(1+2\delta)\end{aligned}$$

Therefore, the relative error is bounded as follows:

$$\left|\frac{(x+y)-z}{(x+y)}\right| = \left|\frac{2\delta(x+y)}{(x+y)}\right| = |2\delta| \leqq 2^{-23}$$

Why is this calculation *not* correct?

Solution The quantities δ that occur in such calculations are not, in general, equal to each other. The correct calculation is

$$\begin{aligned}z &= \text{fl}[\text{fl}(x) + \text{fl}(y)]\\ &= [x(1+\delta_1) + y(1+\delta_2)](1+\delta_3)\\ &= [(x+y) + \delta_1 x + \delta_2 y](1+\delta_3)\\ &= (x+y) + \delta_1 x + \delta_2 y + \delta_3 x + \delta_3 y + \delta_1\delta_3 x + \delta_2\delta_3 y\\ &\approx (x+y) + x(\delta_1+\delta_3) + y(\delta_2+\delta_3)\end{aligned}$$

Therefore, the relative roundoff error is

$$\begin{aligned}\left|\frac{(x+y)-z}{(x+y)}\right| &= \left|\frac{x(\delta_1+\delta_3) + y(\delta_2+\delta_3)}{(x+y)}\right|\\ &= \left|\frac{(x+y)\delta_3 + x\delta_1 + y\delta_2}{(x+y)}\right|\\ &= \left|\delta_3 + \frac{x\delta_1 + y\delta_2}{(x+y)}\right|\end{aligned}$$

This cannot be bounded, because the second term has a denominator that can be zero or close to zero. Notice that if x and y are machine numbers, then δ_1 and δ_2 are zero, and a

useful bound results—namely, δ_3. But we do not need this calculation to know that! It has been assumed that when machine numbers are combined with any of the four arithmetic operations, the relative roundoff error would not exceed 2^{-24} in magnitude (on a 32-bit word-length computer). ■

Historical Notes

Examples of Numerical Computer Failures

In the 1991 Gulf War, a failure of the Patriot missile defense system was the result of a software conversion error. The system clock measured time in tenths of a second, but it was stored as a 24-bit floating-point number, resulting in rounding errors. The system failed to intercept an incoming Iraqi Scud missile, which resulted in the death of 28 American soldiers in a barracks in Dhahran, Saudi Arabia. (Field data had shown that the system would fail to track and intercept an incoming missile after being on for 20 consecutive hours and would need to be rebooted—it had been on for 100 hours!)

In 1994, Professor Thomas R. Nicely discovered that the Intel Pentium floating-point processor returned erroneous results for certain division operations. For example, 0001/824633702441.0 was calculated incorrectly for all digits beyond the eighth significant digit. After initially declaring that it would not impact many users, Intel eventually set aside $420 million dollars to fix the problem and replaced the chip for anyone that requested it!

In 1996, the Ariane 5 rocket launched by the European Space Agency exploded 40 seconds after lift-off from Kourou, French Guiana. An investigation determined that the horizontal velocity required the conversion of a 64-bit floating-point number to a 16-bit signed integer. It failed because the number was larger than 32,767, which was the largest integer of this type that could be stored in memory. The rocket and its cargo were valued at $500 million.

Further details about these disasters can be found by searching the World Wide Web on the Internet. There are other interesting accounts of calamities that could have been averted by more careful computer programming, especially in using floating-point arithmetic.

Summary 1.3

- A **single-precision floating-point number** in a 32-bit word-length computer with standard floating-point representation is stored in a single word with the bit pattern

$$\boxed{b_1 b_2 b_3 \cdots b_9 b_{10} b_{11} \cdots b_{32}}$$

which is interpreted as the real number

$$(-1)^{b_1} \times 2^{(b_2 b_3 \ldots b_9)_2} \times 2^{-127} \times (1.b_{10} b_{11} \ldots b_{32})_2$$

- A **double-precision floating-point number** in a 32-bit word-length computer with standard floating-point representation is stored in two words with the bit pattern

$$\boxed{b_1 b_2 b_3 \cdots b_9 b_{10} b_{11} b_{12} b_{13} \cdots b_{32} b_{33} b_{34} b_{35} \cdots\cdots b_{64}}$$

which is interpreted as the real number

$$(-1)^{b_1} \times 2^{(b_2 b_3 \ldots b_{12})_2} \times 2^{-1023} \times (1.b_{13} b_{14} \ldots b_{64})_2$$

- The relationship between a real number x and the **floating-point machine number** $\text{fl}(x)$ can be written as

$$\text{fl}(x) = x(1+\delta) \qquad (|\delta| \leqq 2^{-24})$$

If $\odot$ denotes any one of the arithmetic operations, then we write

$$\text{fl}(x \odot y) = (x \odot y)(1+\delta)$$

In these equations, δ depends on x and y.

Exercises 1.3

1. Determine the machine representation in single precision on a 32-bit word-length computer for the following decimal numbers.

a. 2^{-30} **b.** 64.015625 [a]**c.** -8×2^{-24}

2. Determine the single-precision and double-precision machine representation in a 32-bit word-length computer of the following decimal numbers:

a. 0.5, −0.5 **b.** 0.125, −0.125

c. e0.0625, −0.0625 [a]**d.** 0.03125, −0.03125

3. Which of these are machine numbers?

a. 10^{403} **b.** $1 + 2^{-32}$ **c.** 1/5

d. 1/10 **e.** 1/256

4. Determine the single-precision and double-precision machine representation of the following decimal numbers:

a. 1.0, −1.0 **b.** +0.0, −0.0 **c.** −9876.54321

[a]**d.** 0.2343750 [a]**e.** 492.78125 **f.** 64.37109375

g. −285.75 **h.** 10^{-2}

5. Identify the floating-point numbers corresponding to the following bit strings:

a. 0 00000000 00000000000000000000000

b. 1 00000000 00000000000000000000000

c. 0 11111111 00000000000000000000000

[a]**d.** 1 11111111 00000000000000000000000

e. 0 00000001 00000000000000000000000

f. 0 10000001 01100000000000000000000

g. 0 01111111 00000000000000000000000

h. 0 01111011 00110011001100110011001100

6. What are the bit-string machine representations for the following subnormal numbers?

a. $2^{-127} + 2^{-128}$ **b.** $2^{-127} + 2^{-150}$

c. $2^{-127} + 2^{-130}$ **d.** $\sum_{k=127}^{150} 2^{-k}$

7. Determine the decimal numbers that have the following machine representations:

a. $[3F27E520]_{16}$ **b.** $[3BCDCA00]_{16}$

c. $[BF4F9680]_{16}$ **d.** $[CB187ABC]_{16}$

8. Determine the decimal numbers that have the following machine representations:

[a]**a.** $[CA3F2900]_{16}$ **b.** $[C705A700]_{16}$

c. $[494F96A0]_{16}$ [a]**d.** $[4B187ABC]_{16}$

e. $[45223000]_{16}$ **f.** $[45607000]_{16}$

[a]**g.** $[C553E000]_{16}$ **h.** $[437F0000]_{16}$

9. Are these machine representations? Why or why not?

a. $[4BAB2BEB]_{16}$ **b.** $[1A1AIA1A]_{16}$

c. $[FADEDEAD]_{16}$ **d.** $[CABE6G94]_{16}$

10. The computer word associated with the variable Δ appears as $[7F7FFFFF]_{16}$, which is the largest representable floating-point single-precision number. What is the decimal value of Δ? The variable ε appears as $[00800000]_{16}$, which is the smallest positive number. What is the decimal value of ε?

11. Enumerate the set of numbers in the floating-point number system that have binary representations of the form $\pm(0.b_1b_2) \times 2^k$, where

a. $k \in \{-1, 0\}$ **b.** $k \in \{-1, 1\}$

[a]**c.** $k \in \{-1, 0, 1\}$

12. What are the machine numbers immediately to the right and left of 2^m? How far is each from 2^m?

13. Generally, when a list of floating-point numbers is added, less roundoff error will occur if the numbers are added in order of increasing magnitude. Give some examples to illustrate this principle.

14. (Continuation) The principle of the preceding exercise is *not universally* valid. Consider a decimal machine with two decimal digits allocated to the mantissa. Show that the four numbers 0.25, 0.0034, 0.00051, and 0.061 can be added with less roundoff error if *not* added in ascending order.

[a] **15.** In the case of machine underflow, what is the relative error involved in replacing a number x by zero?

16. Consider a computer that operates in base β and carries n digits in the mantissa of its floating-point number system. Show that the rounding of a real number x to the nearest machine number $\tilde{x}$ involves a relative error of at most $\frac{1}{2}\beta^{1-n}$.
Hint: Imitate the argument in the text.

[a] **17.** Consider a decimal machine in which five decimal digits are allocated to the mantissa. Give an example, avoiding overflow or underflow, of a real number x whose closest machine number $\tilde{x}$ involves the greatest possible relative error.

[a] **18.** In a five-decimal machine that correctly rounds numbers to the nearest machine number, what real numbers x have the property $\text{fl}(1.0 + x) = 1.0$?

[a] **19.** Consider a computer operating in base β. Suppose that it chops numbers instead of correctly rounding them. If its floating-point numbers have a mantissa of n digits, how large is the relative error in storing a real number in machine format?

20. What is the roundoff error when we represent $2^{-1} + 2^{-25}$ by a machine number?
Note: This refers to absolute error, not relative error.

[a] **21.** (Continuation) What is the relative roundoff error when we round off $2^{-1} + 2^{-26}$ to get the closest machine number?

22. If x is a real number within the range of a 32-bit word-length computer that is rounded and stored, what can happen when x^2 is computed? Explain the difference between $\text{fl}[\text{fl}(x)\text{fl}(x)]$ and $\text{fl}(x\,x)$.

23. A binary machine that carries 30 bits in the fractional part of each floating-point number is designed to round a number up or down correctly to get the closest floating-point number. What simple upper bound can be given for the relative error in this rounding process?

24. A decimal machine that carries 15 decimal places in its floating-point numbers is designed to chop numbers. If x is a real number in the range of this machine and $\hat{x}$ is its machine representation, what upper bound can be given for $|x - \hat{x}|/|x|$?

[a] **25.** If x and y are real numbers within the range of a 32-bit word-length computer and if xy is also within the range, what relative error can there be in the machine computation of xy?
Hint: The machine produces $\text{fl}[\text{fl}(x)\text{fl}(y)]$.

[a] **26.** Let x and y be positive real numbers that are not machine numbers but are within the exponent range of a 32-bit word-length computer. What is the largest possible relative error in the machine representation of $x + y^2$? Include errors made to get the numbers in the machine as well as errors in the arithmetic.

27. Show that if x and y are positive real numbers that have the same first n digits in their decimal representations, then y approximates x with relative error less than 10^{1-n}. Is the converse true?

28. Show that a rough bound on the relative roundoff error when n machine numbers are multiplied in a 32-bit word-length computer is $(n - 1)2^{-24}$.

29. Show that $\text{fl}(x + y) = y$ on a 32-bit word-length computer if x and y are positive machine numbers and $x < y \times 2^{-25}$.

[a] **30.** If 1000 nonzero machine numbers are added in a 32-bit word-length computer, what upper bound can be given for the relative roundoff error in the result? How many decimal digits in the answer can be trusted?

31. Suppose that $x = \sum_{i=1}^{n} a_i 2^{-i}$, where $a_i \in \{-1, 0, 1\}$ is a positive number. Show that x can also be written in the form $\sum_{i=1}^{n} b_i 2^{-i}$, where $b_i \in \{0, 1\}$.

32. If x and y are machine numbers in a 32-bit word-length computer and if $\text{fl}(x/y) = x/[y(1 + \delta)]$, what upper bound can be placed on $|\delta|$?

33. How big is the hole at zero in a 32-bit word-length computer?

34. How many machine numbers are there in a 32-bit length computer? (Consider only normalized floating-point numbers.)

35. How many normalized floating-point numbers are available in a binary machine if n bits are allocated to the mantissa and m bits are allocated to the exponent? Assume that two additional bits are used for signs, as in a 32-bit length computer.

36. Show by an example that in computer arithmetic $a + (b + c)$ may differ from $(a + b) + c$.

[a] **37.** Consider a decimal machine in which floating-point numbers have 13 decimal places. Suppose that numbers are correctly rounded up or down to the nearest machine number. Give the best bound for the roundoff error, assuming

no underflow or overflow. Use relative error, of course. What if the numbers are always chopped?

[a] **38.** Consider a computer that uses five-decimal-digit numbers. Let $\text{fl}(x)$ denote the floating-point machine number closest to x. Show that if $x = 0.53214\,87513$ and $y = 0.53213\,04421$, then the operation $\text{fl}(x) - \text{fl}(y)$ involves a large relative error. Compute it.

[a] **39.** Two numbers x and y that are not machine numbers are read into a 32-bit word-length computer. The machine computes xy^2. What sort of relative error can be expected? Assume no underflow or overflow.

40. Let x, y, and z be three machine numbers in a 32-bit word-length computer. By analyzing the relative error in the worst case, determine how much roundoff error should be expected in forming $(xy)z$.

41. Let x and y be machine numbers in a 32-bit word-length computer. What relative roundoff error should be expected in the computation of $x + y$? If x is around 30 and y is around 250, what absolute error should be expected in the computation of $x + y$?

[a] **42.** Every machine number in a 32-bit word-length computer can be interpreted as the correct machine representation of an entire *interval* of real numbers. Describe this interval for the machine number $q \times 2^m$.

43. Is every machine number on a 32-bit word-length computer the average of two other machine numbers? If not, describe those that are not averages.

44. Let x and y be machine numbers in a 32-bit word-length computer. Let u and v be real numbers in the range of a 32-bit word-length computer but not machine numbers. Find a realistic upper bound on the relative roundoff error when u and v are read into the computer and then used to compute $(x + y)/(uv)$. As usual, ignore products of two or more numbers having magnitudes as small as 2^{-24}. Assume that no overflow or underflow occurs in this calculation.

45. Interpret the following:

a. $\text{fl}(x) = x(1 - \delta)$

b. $\text{fl}(xy) = [x(1 + \delta)]y$

c. $\text{fl}(xy) = x[y(1 + \delta)]$

d. $\text{fl}(xy) = \left(x\sqrt{1+\delta}\right)\left(y\sqrt{1+\delta}\right)$

e. $\text{fl}\left(\dfrac{x}{y}\right) = \dfrac{x(1+\delta)}{y}$

f. $\text{fl}\left(\dfrac{x}{y}\right) = \dfrac{x\sqrt{1+\delta}}{y/\sqrt{1+\delta}}$

g. $\text{fl}\left(\dfrac{x}{y}\right) \approx \dfrac{x}{y(1-\delta)}$

46. Let x and y be real numbers that are not machine numbers for a 32-bit word-length computer and have to be rounded to get them into the machine. Assume that there is no overflow or underflow in getting their (rounded) values into the machine. (Thus, the numbers are within the *range* of a 32-bit word-length computer, although they are not machine numbers.) Find a rough upper bound on the relative error in computing x^2y^3.
Hint: We say *rough upper bound* because you may use $(1 + \delta_1)(1 + \delta_2) \approx 1 + \delta_1 + \delta_2$ and similar approximations. Be sure to include errors involved in getting the numbers into the machine as well as errors that arise from the arithmetic operations.

47. (**Student Research Project**) Write a research paper on the standard floating-point number system, providing additional details on

a. Types of rounding

b. Subnormal floating-point numbers

c. Long-double precision

d. Handling exceptional situations

Computer Exercises 1.3

1. Print several numbers, both integers and reals, in octal format and try to explain the machine representation used in your computer. For example, examine $(0.1)_{10}$ and compare to the results given at the beginning of this chapter.

2. Use your computer to construct a table of three functions f, g, and h defined as follows. For each integer n in the range 1 to 50, let $f(n) = 1/n$. Then $g(n)$ is computed by adding $f(n)$ to itself $n - 1$ times. Finally, set $h(n) = nf(n)$. We want to see the effects of roundoff error in these computations. Use the function real(n) to convert an integer variable n to its real (floating-point) form. Print the table with all the precision of which your computer is capable (in single-precision mode).

[a] **3.** Predict and then show what value your computer will print for $\sqrt{2}$ computed in single precision. Repeat for double or long-double precision. Explain.

4. Write a program to determine the machine epsilon ε within a factor of 2 for single, double, and long-double precision.

5. Let $\mathcal{A}$ denote the set of positive integers whose decimal representation does not contain the digit 0. The sum of the reciprocals of the elements in $\mathcal{A}$ is known to be 23.10345. Can you verify this numerically?

6. Write a computer code

integer function *nDigit*(n, x)

which returns the nth nonzero digit in the decimal expression for the real number x.

[a]**7.** The **harmonic series** $1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots$ is known to diverge to $+\infty$. The nth partial sum approaches $+\infty$ at the same rate as $\ln(n)$. **Euler's constant** is defined to be

$$\gamma = \lim_{n\to\infty}\left[\sum_{k=1}^{n}\frac{1}{k} - \ln(n)\right] \approx 0.57721$$

If your computer ran a program for a week based on the pseudocode

real s, x
$x \leftarrow 1.0$;
$s \leftarrow 1.0$
repeat
  $x \leftarrow x + 1.0$;
  $s \leftarrow s + 1.0/x$
end repeat

what is the largest value of s it would obtain? Write and test a program that uses a loop of 5000 steps to estimate Euler's constant. Print intermediate answers at every 100 steps.

[a]**8.** (Continuation) Prove that **Euler's constant**, γ, can also be represented by

$$\gamma = \lim_{m\to\infty}\left[\sum_{k=1}^{m}\frac{1}{k} - \ln\left(m + \frac{1}{2}\right)\right]$$

Write and test a program that uses $m = 1, 2, 3, \ldots, 5000$ to compute γ by this formula. The convergence should be more rapid than that in the preceding computer exercise. (See the article by de Temple [1993].)

9. Determine the binary form of $\frac{1}{3}$. What is the correctly rounded machine representation in single precision on a 32-bit word-length computer? Check your answer on an actual machine with the instructions

$x \leftarrow 1.0/3.0$; **output** x

using a long format of 16 digits for the output statement.

10. Owing to its gravitational pull, the Earth gains weight and volume slowly over time from space dust, meteorites, and comets. Suppose the Earth is a sphere. Let the radius be $r_a = 7000$ kilometers at the beginning of the year 1900, and let r_b be its radius at the end of the year 2000. Assume that $r_b = r_a + 0.000001$, an increase of 1 millimeter. Using a computer, calculate how much the Earth's volume and surface area has increased during the last century by the following three procedures (exactly as given):

a. Difference in **spherical volume**

$$V_a = \frac{4}{3}\pi r_a^3, \quad V_b = \frac{4}{3}\pi r_b^3, \delta_1 = V_b - V_a$$

b. Difference in **spherical volume**

$$\delta_2 = \frac{4}{3}\pi(r_b - r_a)(r_b^2 + r_b r_a + r_a^2)$$

c. Difference in **spherical surface area**

$$h = r_b - r_a, \quad \delta_3 = 4\pi r_a^2 h$$

First use single precision and then double precision. Compare and analyze your results.

11. **(Student Research Project)** Explore recent developments in floating-point arithmetic. In particular, learn about long-double precision for both real numbers and integers as well as for complex numbers.

12. What is the largest integer your computer can handle?

13. Program each of the following pseudocodes in MATLAB or using some other mathematical software package or programming language. What is the purpose of each of them? What happens when each program is run on a computer? Explain what (if anything) is significant about the output. For example, how many iterations until the program stops, and what are the final values.

a. $x = 0.0$
while $x \neq 0$
  $x = x + 0.1$
end while
output x

b. $x = 1.0$
while $(x + 1.0) > 1.0$
  $x = x/2.0$
end while
$y = 2.0^{*}x$
output y

c.
```
x = 1.0
while x > 0.0
    y = x
    x = x/2.0
end while
output y
```

14. MATLAB `dec2bin` and `bin2dec` for converting numbers from decimal to binary and vise versa as well as `hex2dec` and `dec2hex` for converting a hexadecimal numbers to double-precision and vise versa. Test these commands with these numbers:

a. $(11)_{10}$ **c.** $(0.625)_{10}$

b. $(197)_{10}$ **d.** $(0.2)_{10}$

1.4 Loss of Significance

In this section, we show how loss of significance in subtraction can often be reduced or eliminated by various techniques, such as the use of rationalization, Taylor series, trigonometric identities, logarithmic properties, double precision, and/or range reduction. These are some of the techniques that can be used when one wants to guard against the degradation of precision in a calculation. Of course, we cannot always know when a loss of significance has occurred in a long computation, but we should be alert to the possibility and take steps to avoid it, if possible.

Significant Digits

We first address the elusive concept of **significant digits** in a number. Suppose that x is a real number expressed in normalized scientific notation in the decimal system

$$x = \pm r \times 10^n \qquad \left(\tfrac{1}{10} \leqq r < 1\right)$$

An Example of Significant Digits

For example, x might be

$$x = 0.37214\,98 \times 10^{-5}$$

The digits 3, 7, 2, 1, 4, 9, 8 used to express r do not all have the same significance because they represent different powers of 10. Thus, we say that 3 is the *most* significant digit, and the significance of the digits diminishes from left to right. In this example, 8 is the *least* significant digit.

If x is a *mathematically exact* real number, then its approximate decimal form can be given with as many significant digits as we wish. Thus, we may write

$$\frac{\pi}{10} \approx 0.31415\,92653\,58979$$

Examples of Measured Quantities

and all the digits given are correct. If x is a *measured quantity*, however, the situation is quite different. Every measured quantity involves an error whose magnitude depends on the nature of the measuring device. Thus, if a meter stick is used, it is not reasonable to measure any length with precision better than 1 millimeter. Therefore, the result of measuring, say, a plate glass window with a meter stick should not be reported as 2.73594 meters. That would be misleading! Only digits that are believed to be correct or in error by at most a few units should be reported. It is a scientific convention that the least significant digit given in a measured quantity should be in error by at most five units; that is, the result is rounded correctly.

Similar remarks pertain to quantities computed from measured quantities. For example, if the side of a square is reported to be $s = 0.736$ meter, then one can assume that the error

does not exceed a few units in the third decimal place. The diagonal of that square is then

$$s\sqrt{2} \approx 0.10408\,61182 \times 10^{1}$$

but should be reported as 0.1041×10^{1} or (more conservatively) 0.104×10^{1}. The infinite precision available in $\sqrt{2}$,

$$\sqrt{2} = 1.41421\,35623\,73095\ldots$$

does *not* convey any more precision to $s\sqrt{2}$ than was already present in s.

Computer-Caused Loss of Significance

Perhaps it is surprising that a loss of significance can occur within a computer. It is essential to understand this process so that blind trust will not be placed in numerical output from a computer. One of the most common causes for a deterioration in precision is the subtraction of one quantity from another nearly equal quantity. This effect is potentially quite serious and can be catastrophic. The closer these two numbers are to each other, the more pronounced is the effect.

Loss of Significance
$x - \sin(x)$

To illustrate this phenomenon, consider the assignment statement

$$y \leftarrow x - \sin(x)$$

and suppose that at some point in a computer program this statement is executed with an x value of $\frac{1}{15}$. Assume further that our computer works with floating-point numbers that have ten decimal digits. Then

$$\begin{aligned} x &\leftarrow 0.66666\,66667 \times 10^{-1} \\ \sin(x) &\leftarrow 0.66617\,29492 \times 10^{-1} \\ x - \sin(x) &\leftarrow 0.00049\,37175 \times 10^{-1} \\ x - \sin(x) &\leftarrow 0.49371\,75000 \times 10^{-4} \end{aligned}$$

In the last step, the result has been shifted to normalized floating-point form. Three zeros have then been supplied by the computer in the three *least* significant decimal places. We refer to these as **spurious zeros**; they are *not* significant digits. In fact, the ten-decimal-digit correct value is

$$\frac{1}{15} - \sin\left(\frac{1}{15}\right) \approx 0.49371\,74327 \times 10^{-4}$$

Another way of interpreting this is to note that the final digit in $x - \sin(x)$ is derived from the tenth digits in x and $\sin(x)$. When the eleventh digit in either x or $\sin(x)$ is 5, 6, 7, 8, or 9, the numerical values are rounded up to ten digits so that their tenth digits may be altered by plus one unit. Since these tenth digits may be in error, the final digit in $x - \sin(x)$ may also be in error—which it is!

EXAMPLE 1 If $x = 0.37214\,48693$ and $y = 0.37202\,14371$, what is the relative error in the computation of $x - y$ in a computer that has five decimal digits of accuracy?

Solution The numbers would first be rounded to $\tilde{x} = 0.37214$ and $\tilde{y} = 0.37202$. Then we have $\tilde{x} - \tilde{y} = 0.00012$, while the correct answer is $x - y = 0.00012\,34322$. The relative error involved is

$$\frac{|(x - y) - (\tilde{x} - \tilde{y})|}{|x - y|} = \frac{0.00000\,34322}{0.00012\,34322} \approx 3 \times 10^{-2}$$

This magnitude of relative error must be judged quite large when compared with the relative error of $\tilde{x}$ and $\tilde{y}$. (They cannot exceed $\frac{1}{2} \times 10^{-4}$ by the coarsest estimates, and in this example, they are, in fact, approximately 1.3×10^{-5}.) ■

It should be emphasized that this discussion pertains not to the operation

$$\text{fl}(x - y) \leftarrow x - y$$

but rather to the operation

fl Notation

$$\text{fl}[\text{fl}(x) - \text{fl}(y)] \leftarrow x - y$$

Roundoff error in the former case is governed by the equation

$$\text{fl}(x - y) = (x - y)(1 + \delta)$$

where $|\delta| \leqq 2^{-24}$ on a 32-bit word-length computer, and on a five-decimal-digit computer in the preceding example $|\delta| \leqq \frac{1}{2} \times 10^{-4}$.

In Example 1 above, we observe that the computed difference of 0.00012 has only two significant figures of accuracy, whereas in general, one expects the numbers and calculations in this computer to have five significant figures of accuracy.

The remedy for this difficulty is first to anticipate that it may occur and then to reprogram. The simplest technique may be to carry out part of a computation in double- or long-double-precision arithmetic (that means roughly twice as many significant digits), but often a slight change in the formulas is required. Several illustrations of this will be given, and additional examples are found among the exercises.

Again consider Example 1, but imagine that the calculations to obtain x, y, and $x - y$ are being done in double precision. Suppose that single-precision arithmetic is used thereafter. In the computer, all ten digits of x, y, and $x - y$ are retained, but at the end, $x - y$ is rounded to its five-digit form, which is 0.12343×10^{-3}. This answer has five significant digits of accuracy, as we would like. Of course, the programmer or analyst must know in advance where double-precision arithmetic may be necessary in the computation. Programming everything in double precision is very wasteful if it is not needed. This approach has another drawback: There may be such serious cancellation of significant digits that even double precision might not help!

Theorem on Loss of Precision

Before considering other techniques for avoiding this problem, we ask the following question:

> *Exactly how many significant binary digits are lost in the subtraction $x - y$ when x is close to y?*

The closeness of x and y is conveniently measured by $|1 - (y/x)|$. Here is the result:

■ **Theorem 1**

Loss of Precision Theorem

Let x and y be normalized floating-point machine numbers, where $x > y > 0$. If $2^{-p} \leqq 1 - (y/x) \leqq 2^{-q}$ for some positive integers p and q, then at most p and at least q significant binary bits are lost in the subtraction $x - y$.

Proof We prove the second part of the theorem and leave the first as an exercise. To this end, let $x = r \times 2^n$ and $y = s \times 2^m$, where $\frac{1}{2} \leqq r, s < 1$. (This is the normalized binary floating-point form.) Since $y < x$, the computer may have to *shift* y before carrying out the subtraction. In any case, y must first be expressed with the same exponent as x. Hence, $y = (s2^{m-n}) \times 2^n$ and

$$x - y = (r - s2^{m-n}) \times 2^n$$

The mantissa of this number satisfies

$$r - s2^{m-n} = r\left(1 - \frac{s2^m}{r2^n}\right) = r\left(1 - \frac{y}{x}\right) < 2^{-q}$$

Hence, to normalize the representation of $x - y$, a shift of at least q bits to the left is necessary. Then at least q (spurious) zeros are supplied on the right-hand end of the mantissa. This means that at least q bits of precision have been lost. ■

EXAMPLE 2 In the subtraction 37.59362 1 − 37.58421 6, how many bits of significants are lost?

Solution Let x denote the first number and y the second. Then

$$1 - \frac{y}{x} = 0.00025\,01754$$

This lies between 2^{-12} and 2^{-11}. These two numbers are 0.00024 4 and 0.00048 8. Hence, at least 11 but not more than 12 bits are lost. ■

Here is an example in decimal form.

EXAMPLE 3 In the subtraction of $y = .6311$ from $x = .6353$, how many significance are lost?

Solution These numbers are close, and $1 - y/x = .00661 < 10^{-2}$. In the subtraction, we have $x - y = .0042$. There are two significant figures in the answer, although there were four significant figures in x and y. ■

Avoiding Loss of Significance in Subtraction

Now we take up various techniques that can be used to avoid the loss of significance that may occur in subtraction.

EXAMPLE 4 Explore the function

$$f(x) = \sqrt{x^2 + 1} - 1 \tag{1}$$

whose values may be required for x near zero.

Solution Since $\sqrt{x^2 + 1} \approx 1$ when $x \approx 0$, we see that there is a potential loss of significance in the subtraction. However, the function can be rewritten in the form

$$f(x) = \left(\sqrt{x^2 + 1} - 1\right)\left(\frac{\sqrt{x^2 + 1} + 1}{\sqrt{x^2 + 1} + 1}\right) = \frac{x^2}{\sqrt{x^2 + 1} + 1} \tag{2}$$

by **rationalizing** the numerator—that is, removing the radical in the numerator. This procedure allows terms to be canceled and thereby removes the subtraction. For example, if we use five-decimal-digit arithmetic and if $x = 10^{-3}$, then $f(x)$ is computed incorrectly as zero by the first formula, but as $\frac{1}{2} \times 10^{-6}$ by the second. If we use the first formula together with double precision, the difficulty is ameliorated, but *not* circumvented altogether. For example, in double precision, we have the same problem when $x = 10^{-6}$. ■

EXAMPLE 5 How can accurate values of the function

$$f(x) = x - \sin x \tag{3}$$

be computed near $x = 0$.

Solution A careless programmer might code this function just as indicated in Equation (3), not realizing that a serious loss of accuracy occurs. Recall from calculus that

$$\lim_{x \to 0} \frac{\sin x}{x} = 1$$

to see that $\sin x \approx x$ when $x \approx 0$. One cure for this problem is to use the Taylor series for $\sin x$:

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$$

This series is known to represent $\sin x$ for all real values of x. For x near zero, it converges quite rapidly. Using this series, we can write the function f as

$$f(x) = x - \left(x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} - \cdots\right) = \frac{x^3}{3!} - \frac{x^5}{5!} + \frac{x^7}{7!} - \cdots \tag{4}$$

We see in this equation where the original difficulty arose; namely, for small values of x, the term x in the sine series is much larger than $x^3/3!$ and thus more important. But when $f(x)$ is formed, this dominant x term disappears, leaving only the lesser terms. The series that starts with $x^3/3!$ is very effective for calculating $f(x)$ when x is small. ■

In Example 5, further analysis is needed to determine the range in which Series (4) should be used and the range in which Formula (3) can be used. Using the Theorem on Loss of Precision, we see that the loss of bits in the subtraction of Formula (3) can be limited to at most 1 bit by restricting x so that $\frac{1}{2} \leqq 1 - \sin x/x$. (Here we are considering only the case

when $\sin x > 0$.) With a calculator, it is easy to see that x must be at least 1.9. Thus, for $|x| < 1.9$, we use the first few terms in Series (4), and for $|x| \geqq 1.9$, we use $f(x) = x - \sin x$. We can verify that for the worst case ($x = 1.9$), ten terms in the series give $f(x)$ with an error of at most 10^{-16}. (That is good enough for double precision on a 32-bit word-length computer.)

Now we can construct a function procedure for computing $f(x) = x - \sin x$ and write its pseudocode. Notice that the terms in the series can be obtained inductively by the algorithm

$$\begin{cases} t_1 = \dfrac{x^3}{6} \\ t_{n+1} = \dfrac{-t_n x^2}{(2n+2)(2n+3)} \qquad (n \geqq 1) \end{cases}$$

Then the partial sums can be obtained inductively by

$$\begin{cases} s_1 = t_1 \\ s_{n+1} = s_n + t_{n+1} \qquad (n \geqq 1) \end{cases}$$

so that

$$s_n = \sum_{k=1}^{n} t_k = \sum_{k=1}^{n} (-1)^{k+1} \left[\frac{x^{2k+1}}{(2k+1)!} \right]$$

A suitable pseudocode for the function in Example 5 is given here:

$f(x) = x - \sin x$ Function Pseudocode

```
real function f(x)
integer i, n ← 10;   real s, t, x
if |x| ≧ 1.9 then
    s ← x − sin x
    else
    t ← x^3/6
    s ← t
    for i = 2 to n
        t ← −tx^2/[(2i + 2)(2i + 3)]
        s ← s + t
    end for
end if
f ← s
end function f
```

EXAMPLE 6 How can accurate values of the function

$$f(x) = e^x - e^{-2x}$$

be computed in the vicinity of $x = 0$?

Solution Since e^x and e^{-2x} are both equal to 1 when $x = 0$, there is a loss of significance in the subtraction when x is close to zero. Inserting the appropriate Taylor series, we obtain

$$\begin{aligned} f(x) &= \left(1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots\right) - \left(1 - 2x + \frac{4x^2}{2!} - \frac{8x^3}{3!} + \cdots\right) \\ &= 3x - \frac{3}{2}x^2 + \frac{3}{2}x^3 - \cdots \end{aligned}$$

An alternative approach is to write

$$f(x) = e^{-2x}\left(e^{3x} - 1\right)$$
$$= e^{-2x}\left(3x + \frac{9}{2!}x^2 + \frac{27}{3!}x^3 + \cdots\right)$$

By using the Theorem on Loss of Precision, we find that at most 1 bit is lost in the subtraction $e^x - e^{-2x}$ when $x > 0$ and

$$\frac{1}{2} \leqq 1 - e^{-3x}$$

This inequality is valid when $x \geqq \frac{1}{3}\ln 2 = 0.23105$. Similar reasoning when $x < 0$ shows that for $x \leqq -0.23105$ and at most 1 bit is lost. Hence, the series should be used for $|x| < 0.23105$. ■

EXAMPLE 7 Criticize the assignment statement

$$y \leftarrow \cos^2(x) - \sin^2(x)$$

Solution When $\cos^2(x) - \sin^2(x)$ is computed, there is a loss of significance at $x = \pi/4$ (and at other points). The simple trigonometric identity

$$\cos 2\theta = \cos^2\theta - \sin^2\theta$$

can be used. Thus, the assignment statement can be replaced by

$$y \leftarrow \cos(2x)$$ ■

EXAMPLE 8 Criticize the assignment statement

$$y \leftarrow \ln(x) - 1$$

Solution If the expression $\ln x - 1$ is used for x near e, there is a cancellation of digits and a loss of accuracy. Use elementary facts about logarithms to overcome the difficulty. Thus, we have $y = \ln x - 1 = \ln x - \ln e = \ln(x/e)$. Here is a suitable assignment statement

$$y \leftarrow \ln\left(\frac{x}{e}\right)$$ ■

Range Reduction

Another cause of loss of significant figures is the evaluation of various library functions with large arguments. This problem is more subtle than those previously discussed. We illustrate with the sine function.

A basic property of the function $\sin x$ is its **periodicity**:

$$\sin x = \sin(x + 2n\pi)$$

for all real values of x and for all integer values of n. Because of this relationship, we need to know only the values of $\sin x$ in some fixed interval of length 2π to compute $\sin x$ for arbitrary x. This property can be used in the computer evaluation of $\sin x$ and is called **range reduction**.

EXAMPLE 9 Discuss how to evaluate sin(12532.14), by subtracting integer multiples of 2π. Show that it equals sin(3.47), if we retain only two decimal digits of accuracy.

Solution From $\sin(12532.14) = \sin(12532.14 - 2k\pi)$, we want $12532 = 2k\pi$ and $k = 3989/2\pi \approx 1994$. Consequently, we obtain $12532.14 - 2(1994)\pi = 3.49$ and $\sin(12532.14) \approx \sin(3.49)$. Thus, although our original argument 12532.14 had seven significant figures, the reduced argument has only three. The remaining digits disappeared in the subtraction of 3988π. Since 3.47 has only three significant figures, our computed value of $\sin(12532.14)$ has *no more than* three significant figures. This decrease in precision is unavoidable, if there is no way of increasing the precision of the original argument. If the original argument (12532.14) can be obtained with more significant figures, these additional figures are present in the *reduced* argument (3.47). In some cases, double- or long-double-precision programming may be helpful. ■

EXAMPLE 10 For $\sin x$, how many binary bits of significance are lost in range reduction to the interval $[0, 2\pi)$?

Solution Given an argument $x > 2\pi$, we determine an integer n that satisfies the inequality $0 \leqq x - 2n\pi < 2\pi$. Then in evaluating elementary trigonometric functions, we use $f(x) = f(x - 2n\pi)$. In the subtraction $x - 2n\pi$, there is a loss of significance. By the Theorem on Loss of Precision, at least q bits are lost if

$$1 - \frac{2n\pi}{x} \leqq 2^{-q}$$

Since

$$1 - \frac{2n\pi}{x} = \frac{x - 2n\pi}{x} < \frac{2\pi}{x}$$

we conclude that at least q bits are lost if $2\pi/x \leqq 2^{-q}$. Stated otherwise, at least q bits are lost if $2^q \leqq x/2\pi$. ■

Summary 1.4

- To avoid loss of significance in subtraction, one may be able to reformulate the expression using rationalizing, series expansions, or mathematical identities.
- If x and y are positive normalized floating-point machine numbers with

  $$2^{-p} \leqq 1 - \frac{y}{x} \leqq 2^{-q}$$

  then at most p and at least q significant binary bits are lost in computing $x - y$. (*Note*: It is permissible to leave out the hypothesis $x > y$ here.)

Exercises 1.4

1. How can values of the function $f(x) = \sqrt{x + 4} - 2$ be computed accurately when x is small?

2. Calculate $f(10^{-2})$ for the function

$$f(x) = e^x - x - 1$$

The answer should have five significant figures and can easily be obtained with pencil and paper. Contrast it with the straightforward evaluation of $f(10^{-2})$ using $e^{0.01} \approx 1.0101$.

3. What is a good way to compute values of the function $f(x) = e^x - e$ if full machine precision is needed? *Note*: There is some difficulty when $x = 1$.

[a]**4.** What difficulty could the following assignment cause?

$$y \leftarrow 1 - \sin x$$

Circumvent it without resorting to a Taylor series if possible.

5. The hyperbolic sine function is defined by $\sinh x = \frac{1}{2}(e^x - e^{-x})$. What drawback could there be in using this formula to obtain values of the function? How can values of $\sinh x$ be computed to full machine precision when $|x| \leqq \frac{1}{2}$?

[a]**6.** Determine the first two nonzero terms in the expansion about zero for the function

$$f(x) = \frac{\tan x - \sin x}{x - \sqrt{1 + x^2}}$$

Give an approximate value for $f(0.0125)$.

7. Find a method for computing

$$y \leftarrow \frac{1}{x}(\sinh x - \tanh x)$$

that avoids loss of significance when x is small. Find appropriate identities to solve this problem without using Taylor series.

[a]**8.** Find a way to calculate accurate values for

$$f(x) = \frac{\sqrt{1 + x^2} - 1}{x^2} - \frac{x^2 \sin x}{x - \tan x}$$

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

9. For some values of x, the assignment statement $y \leftarrow 1 - \cos x$ involves a difficulty. What is it, what values of x are involved, and what remedy do you propose?

[a]**10.** For some values of x, the function $f(x) = \sqrt{x^2 + 1} - x$ cannot be accurately computed by using this formula. Explain and find a way around the difficulty.

[a]**11.** The inverse hyperbolic sine is given by $f(x) = \ln\left(x + \sqrt{x^2 + 1}\right)$. Show how to avoid loss of significance in computing $f(x)$ when x is negative.
Hint: Find and exploit the relationship between $f(x)$ and $f(-x)$.

12. On most computers, a highly accurate routine for $\cos x$ is provided. It is proposed to base a routine for $\sin x$ on the formula $\sin x = \pm\sqrt{1 - \cos^2 x}$. From the standpoint of precision (not efficiency), what problems do you foresee and how can they be avoided if we insist on using the routine for $\cos x$?

[a]**13.** Criticize and recode the assignment statement

$$z \leftarrow \sqrt{x^4 + 4} - 2$$

assuming that z is sometimes needed for an x close to zero.

14. How can values of the function $f(x) = \sqrt{x + 2} - \sqrt{x}$ be computed accurately when x is large?

15. Write a function that computes accurate values of $f(x) = \sqrt[4]{x + 4} - \sqrt[4]{x}$ for positive x.

[a]**16.** Find a way to calculate

$$f(x) = (\cos x - e^{-x})/\sin x$$

correctly. Determine $f(0.008)$ correctly to ten decimal places (rounded).

17. Without using series, how could the function

$$f(x) = \frac{\sin x}{x - \sqrt{x^2 - 1}}$$

be computed to avoid loss of significance?

18. Write a function procedure that returns accurate values of the hyperbolic tangent function

$$\tanh x = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

for all values of x. Notice the difficulty when $|x| < \frac{1}{2}$.

19. Find a good way to compute $\sin x + \cos x - 1$ for x near zero.

[a]**20.** Find a good way to compute $\arctan x - x$ for x near zero.

21. Find a good bound for $|\sin x - x|$ using Taylor series and assuming that $|x| < \frac{1}{10}$.

[a]**22.** How would you compute $(e^{2x} - 1)/(2x)$ to avoid loss of significance near zero?

23. For any $x_0 > -1$, the sequence defined recursively by

$$x_{n+1} = 2^{n+1}\left(\sqrt{1 + 2^{-n}x_n} - 1\right) \qquad (n \geqq 0)$$

converges to $\ln(x_0 + 1)$. Arrange this formula in a way that avoids loss of significance.

24. Indicate how the following formulas may be useful for arranging computations to avoid loss of significant digits.

[a]**a.** $\sin x - \sin y = 2 \sin \frac{1}{2}(x - y) \cos \frac{1}{2}(x + y)$

b. $\log x - \log y = \log(x/y)$

c. $e^{x-y} = e^x/e^y$

d. $1 - \cos x = 2 \sin^2(x/2)$

e. $\arctan x - \arctan y = \arctan\left(\frac{x - y}{1 + xy}\right)$

25. What is a good way to compute $\tan x - x$ when x is near zero?

26. Find ways to compute these functions without serious loss of significant figures:

a. $e^x - \sin x - \cos x$ [a] **b.** $\ln(x) - 1$

c. $\log x - \log(1/x)$

[a] **d.** $x^{-2}(\sin x - e^x + 1)$ **e.** $x - \operatorname{arctanh} x$

27. Let

$$a(x) = \frac{1 - \cos x}{\sin x}$$

$$b(x) = \frac{\sin x}{1 + \cos x}$$

$$c(x) = \frac{x}{2} + \frac{x^3}{24}$$

Show that $b(x)$ is identical to $a(x)$ and that $c(x)$ approximates $a(x)$ in a neighborhood of zero.

[a] **28.** On your computer determine the range of x for which $(\sin x)/x \approx 1$ with full machine precision.
Hint: Use Taylor series.

[a] **29.** The familiar quadratic formula

$$x = \frac{1}{2a}\left(-b \pm \sqrt{b^2 - 4ac}\right)$$

will cause a problem when the quadratic equation $x^2 - 10^5 x + 1 = 0$ is solved with a machine that carries only eight decimal digits. Investigate the example, observe the difficulty, and propose a remedy.
Hint: An example in the text is similar.

[a] **30.** When accurate values for the roots of a quadratic equation are desired, some loss of significance may occur if $b^2 \approx 4ac$. What (if anything) can be done to overcome this when writing a computer routine?

31. Refer to the discussion of the function $f(x) = x - \sin x$ given in the text. Show that when $0 < x < 1.9$, there will be no undue loss of significance from subtraction in Equation (3).

32. Discuss the exercise of computing $\tan(10^{100})$. (See Gleick [1992], p. 178.)

33. Let x and y be two normalized binary floating-point machine numbers. Assume that $x = q \times 2^n$, $y = r \times 2^{n-1}$, $\frac{1}{2} \leqq r$, $q < 1$, and $2q - 1 \geqq r$. How much loss of significance occurs in subtracting $x - y$? Answer the same question when $2q - 1 < r$. Observe that the Theorem on Loss of Precision is not strong enough to solve this exercise precisely.

34. Prove the first part of the Theorem on Loss of Precision.

35. Show that if x is a machine number on a 32-bit computer that satisfies the inequality $x > \pi 2^{25}$, then $\sin x$ will be computed with *no* significant digits.

36. Let x and y be two positive normalized floating-point machine numbers in a 32-bit computer. Let $x = q \times 2^m$ and $y = r \times 2^n$ with $\frac{1}{2} \leqq r, q < 1$. Show that if $n = m$, then at least 1 bit of significance is lost in the subtraction $x - y$.

37. (**Student Research Project**) Read about and discuss the difference between **cancellation error**, a **bad algorithm**, and an **ill-conditioned problem**.
Suggestion: One example involves the quadratic equation. Read Stewart [1996].

38. On a three-significant-digit computer, calculate $\sqrt{9.01} - 3.00$, with as much accuracy as possible.

Computer Exercises 1.4

[a] **1.** Write a routine for computing the two roots x_1 and x_2 of the quadratic equation $f(x) = ax^2 + bx + c = 0$ with real constants a, b, and c and for evaluating $f(x_1)$ and $f(x_2)$. Use formulas that reduce roundoff errors and write efficient code. Test your routine on the following (a, b, c) values: (0, 0, 1); (0, 1, 0); (1, 0, 0); (0, 0, 0); (1, 1, 0); (2, 10, 1); (1, −4, 3.99999); (1, −8.01, 16.004); $(2 \times 10^{17}, 10^{18}, 10^{17})$; and $(10^{-17}, -10^{17}, 10^{17})$.

2. (Continuation) Write and test a routine for solving a quadratic equation that may have complex roots.

3. Alter and test the pseudocode in the text for computing $x - \sin x$ by using nested multiplication to evaluate the series.

4. Write a routine for the function $f(x) = e^x - e^{-2x}$ using the examples in the text for guidance.

5. Write code using double or extended precision to evaluate $f(x) = \cos(10^4 x)$ on the interval [0, 1]. Determine how many significant figures the values of $f(x)$ will have.

6. Write a procedure to compute $f(x) = \sin x - 1 + \cos x$. The routine should produce nearly full machine precision for all x in the interval $[0, \pi/4]$.
Hint: The trigonometric identity $\sin^2\theta = \frac{1}{2}(1 - \cos 2\theta)$ may be useful.

7. Write a procedure to compute $f(x, y) = \int_1^x t^y\, dt$ for arbitrary x and y.

Note: Notice the exceptional case $y = -1$ and the numerical problem *near* the exceptional case.

8. Suppose that we wish to evaluate the function $f(x) = (x - \sin x)/x^3$ for values of x close to zero.

a. Write a routine for this function. Evaluate $f(x)$ 16 times. Initially, let $x \leftarrow 1$, and then let $x \leftarrow \frac{1}{10}x$ 15 times. Explain the results.
Note: L'Hôpital's Rule indicates that $f(x)$ should tend to $\frac{1}{6}$. Test this code.

b. Write a function procedure that produces more accurate values of $f(x)$ for all values of x. Test this code.

9. Write a program to print a table of the function $f(x) = 5 - \sqrt{25 + x^2}$ for $x = 0$ to 1 with steps of 0.01. Be sure that your program yields full machine precision, but do not program the exercise in double precision. Explain the results.

[a] **10.** Write a routine that computes e^x by summing n terms of the Taylor series until the $n + 1$st term t is such that $|t| < \varepsilon = 10^{-6}$. Use the reciprocal of e^x for negative values of x. Test on the following data: 0, +1, −1, 0.5, −0.123, −25.5, −1776, 3.14159. Compute the relative error, the absolute error, and n for each case, using the exponential function on your computer system for the exact value. Sum no more than 25 terms.

11. (Continuation) The computation of e^x can be reduced to computing e^u for $|u| < (\ln 2)/2$ only. This algorithm removes powers of 2 and computes e^u in a range where the series converges very rapidly. It is given by

$$e^x = 2^m e^u$$

where m and u are computed by the steps

$$z \leftarrow x/\ln 2;\ m \leftarrow \text{integer}\ (z \pm \tfrac{1}{2})$$
$$w \leftarrow z - m;\ u \leftarrow w \ln 2$$

Here the minus sign is used if $x < 0$ because $z < 0$. Incorporate this range reduction technique into the code.

12. (Continuation) Write a routine that uses range reduction $e^x = 2^m e^u$ and computes e^u from the even part of the **Gaussian continued fraction**; that is,

$$e^u = \frac{s+u}{s-u}, \quad s = 2 + u^2\left(\frac{2520 + 28u^2}{15120 + 420u^2 + u^4}\right)$$

Test on the data given in Computer Exercise 1.4.10.
Note: Some of the computer exercises in this section contain rather complicated algorithms for computing various intrinsic functions that correspond to those actually used on a large mainframe computer system. Descriptions of these and other similar library functions are frequently found in the supporting documentation of your computer system.

13. Quite important in many numerical calculations is the accurate computation of the absolute value $|z|$ of a complex number $z = a + bi$. Design and carry out a computer experiment to compare the following three schemes:

a. $|z| = (a^2 + b^2)^{1/2}$ **b.** $|z| = v\left[1 + \left(\frac{w}{v}\right)^2\right]^{1/2}$

c. $|z| = 2v\left[\frac{1}{4} + \left(\frac{w}{2v}\right)^2\right]^{1/2}$

where $v = \max\{|a|, |b|\}$ and $w = \min\{|a|, |b|\}$. Use very small and large numbers for the experiment.

[a] **14.** For what range of x is the approximation $(e^x - 1)/2x \approx 0.5$ correct to 15 decimal digits of accuracy? Using this information, write a function procedure for $(e^x - 1)/2x$, producing 15 decimals of accuracy throughout the interval $[-10, 10]$.

[a] **15.** In the theory of Fourier series, some numbers known as **Lebesgue constants** play a role. A formula for them is

$$\rho_n = \frac{1}{2n+1} + \frac{2}{\pi}\sum_{k=1}^{n}\frac{1}{k}\tan\frac{\pi k}{2n+1}$$

Write and run a program to compute $\rho_1, \rho_2, \ldots, \rho_{100}$ with eight decimal digits of accuracy. Then test the validity of the inequality

$$0 \leqq \frac{4}{\pi^2}\ln(2n+1) + 1 - \rho_n \leqq 0.0106$$

16. Compute in double or extended precision the following number:

$$x = \left[\frac{1}{\pi}\ln(6\,40320^3 + 744)\right]^2$$

What is the point of this exercise?

17. Write a routine to compute $\sin x$ for x in radians as follows. First, using properties of the sine function, reduce the range so that $-\pi/2 \leqq x \leqq \pi/2$. Then if $|x| < 10^{-8}$, set $\sin x \approx x$; if $|x| > \pi/6$, set $u = x/3$, compute $\sin u$ by the formula below, and then set $\sin x \approx [3 - 4\sin^2 u]\sin u$; if $|x| \leqq \pi/6$, set $u = x$ and compute $\sin u$ as follows:

$$\sin u \approx u\left[\frac{1 - \left(\frac{29593}{2\,07636}\right)u^2 + \left(\frac{34911}{76\,13320}\right)u^4}{1 + \left(\frac{1671}{69212}\right)u^2 + \left(\frac{97}{3\,51384}\right)u^4} \right.$$
$$\left. \frac{- \left(\frac{4\,79249}{1\,15113\,39840}\right)u^6}{+ \left(\frac{2623}{16444\,77120}\right)u^6}\right]$$

Try to determine whether the sine function on your computer system uses this algorithm.
Note: This is the **Padé rational approximation** for sine.

18. Write a routine to compute the natural logarithm by the algorithm outlined here based on **telescoped rational** and **Gaussian continued fractions** for $\ln x$ and test for several values of x. First check whether $x = 1$ and return zero if so. Reduce the range of x by determining n and r such that $x = r \times 2^n$ with $\frac{1}{2} \leqq r < 1$. Next, set $u = (r - \sqrt{2}/2)/(r + \sqrt{2}/2)$, and compute $\ln[(1+u)/(1-u)]$ by the approximation

$$\ln\left(\frac{1+u}{1-u}\right) \approx u\left(\frac{20790 - 21545.27u^2}{10395 - 14237.635u^2} \right.$$
$$\left. \frac{+\ 4223.9187u^4}{+\ 4778.8377u^4 - 230.41913u^6}\right)$$

which is valid for $|u| < 3 - 2\sqrt{2}$. Finally, set

$$\ln x \approx \left(n - \frac{1}{2}\right)\ln 2 + \ln\left[\frac{1+u}{1-u}\right]$$

19. Write a routine to compute the tangent of x in radians, using the algorithm following. Test the resulting routine over a range of values of x. First, the argument x is reduced to $|x| \leqq \pi/2$ by adding or subtracting multiples of π. If we have $0 \leqq |x| \leqq 1.7 \times 10^{-9}$, set $\tan x \approx x$. If $|x| > \pi/4$, set $u = \pi/2 - x$; otherwise, set $u = x$. Now compute the approximation

$$\tan u \approx u\left(\frac{1\,35135 - 17336.106u^2}{+\ 379.23564u^4 - 1.01186\,25u^6} \right.$$
$$\left. \frac{+\ 1\,35135 - 62381.106u^2}{+\ 3154.9377u^4 - 28.17694u^6}\right)$$

Finally, if $|x| > \pi/4$, set $\tan x \approx 1/\tan u$; if $|x| \leqq \pi/4$, set $\tan x \approx \tan u$.
Note: This algorithm is obtained from the **telescoped rational** and **Gaussian continued fraction** for the tangent function.

20. Write a routine to compute $\arcsin x$ based on the following algorithm, using telescoped polynomials for the arcsine. If $|x| < 10^{-8}$, set $\arcsin x \approx x$. Otherwise, if $0 \leqq x \leqq \frac{1}{2}$, set $u = x$, $a = 0$, and $b = 1$; if $\frac{1}{2} < x \leqq \frac{1}{2}\sqrt{3}$, set $u = 2x^2 - 1$, $a = \pi/4$, and $b = \frac{1}{2}$; if $\frac{1}{2}\sqrt{3} < x \leqq \frac{1}{2}\sqrt{2+\sqrt{3}}$, set $u = 8x^4 - 8x^2 + 1$, $a = 3\pi/8$, and $b = \frac{1}{4}$; if $\frac{1}{2}\sqrt{2+\sqrt{3}} < x \leqq 1$, set $u = \sqrt{\frac{1}{2}(1-x)}$, $a = \pi/2$, and $b = -2$. Now compute the approximation

$$\arcsin u \approx u\left(1.0 + \frac{1}{6}u^2 + 0.075u^4 \right. \quad + 0.04464\,286u^6$$
$$+\ 0.03038\,182u^8 \quad + 0.02237\,5u^{10}$$
$$+\ 0.01731\,276u^{12} \quad + 0.01433\,124u^{14}$$
$$+\ 0.00934\,2806u^{16} + 0.01835\,667u^{18}$$
$$\left. -\ 0.01186\,224u^{20} \quad + 0.03162\,712u^{22}\right)$$

Finally, set $\arcsin x \approx a + b \arcsin u$. Test this routine for various values of x.

21. Write and test a routine to compute $\arctan x$ for x in radians as follows. If $0 \leqq x \leqq 1.7 \times 10^{-9}$, set $\arctan x \approx x$. If $1.7 \times 10^{-9} < x \leqq 2 \times 10^{-2}$, use the series approximation

$$\arctan x \approx x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7}$$

Otherwise, set $y = x$, $a = 0$, and $b = 1$ if $0 \leqq x \leqq 1$; set $y = 1/x$, $a = \pi/2$, and $b = -1$ if $1 < x$. Then set $c = \pi/16$ and $d = \tan c$ if $0 \leqq y \leqq \sqrt{2} - 1$ and $c = 3\pi/16$ and $d = \tan c$ if $\sqrt{2} - 1 < y \leqq 1$. Compute $u = (y - d)/(1 + dy)$ and the approximation

$$\arctan u \approx u\left(\frac{1\,35135 + 1\,71962.46u^2}{1\,35135 + 2\,17007.46u^2} \right.$$
$$\left. \frac{+52490.4832u^4 + 2218.1u^6}{+97799.3033u^4 + 10721.3745u^6}\right)$$

Finally, set $\arctan x \approx a + b(c + \arctan u)$.
Note: This algorithm uses **telescoped rational** and **Gaussian continued fractions**.

22. A fast algorithm for computing $\arctan x$ to n-bit precision for x in the interval $(0, 1]$ is as follows: Set $a = 2^{-n/2}$, $b = x/(1 + \sqrt{1+x^2})$, $c = 1$, and $d = 1$. Then repeatedly update these variables by these formulas (in order from left to right and top to bottom):

real a, b, c, d

$$c \leftarrow \frac{2c}{1+a}\ ; \quad d \leftarrow \frac{2ab}{1+b^2}\ ; \quad d \leftarrow \frac{d}{1+\sqrt{1-d^2}}$$
$$d \leftarrow \frac{b+d}{1-bd}; \quad b \leftarrow \frac{d}{1+\sqrt{1+d^2}}; \quad a \leftarrow \frac{2\sqrt{a}}{1+a}$$

After each sweep, print $f = c\ln[(1+b)/(1-b)]$. Stop when $1 - a \leqq 2^{-n}$. Write a double-precision routine to implement this algorithm and test it for various values of x. Compare the results to those obtained from the arctangent function on your computer system.
Note: This fast multiple-precision algorithm depends on the theory of **elliptic integrals**, using the arithmetic-geometric mean iteration and **ascending Landen transformations**. Other fast algorithms for trigonometric functions are discussed in Brent [1976].

23. On your computer, show that in single precision, you have only six decimal digits of accuracy if you enter 20 digits. Show that going to double precision is effective only if all work is done in double precision. For example, if you use

`pi = 3.14` or `pi = 22/7`, you will lose all the precision that you have gained by using double precision. Remember that the number of significant digits in the final results is what counts!

24. In some programming languages such as Java and C++, show that mixed-mode arithmetic can lead to results such as `(4/3)*pi=pi` when `pi` is a floating-point number because the fraction inside the parentheses is computed in integer mode.

25. (**Student Research Project**) Investigate interval arithmetic, which has the goal of obtaining results with a guaranteed precision.

26. The smallest root in absolute value of the equation $f(x) = x^2 + 2px - q = 0$ can be computed using either $y_1 = -p + \sqrt{p^2 + q}$ or $y_2 = q/[p + \sqrt{p^2 + q}]$. Show that severe cancellation may occur when p is positive and much greater than q. Program and test the following pseudocode for carrying out these computations step-by-step. Consider the case when $p = 10^4$ and $q = 1$ as well as when $p = -(1 + \frac{1}{2}10^{-8})$ and $q = -(1 + 10^8)$. Check the results by substitution back into the original equation. Explain your results.

$$
\begin{aligned}
s &= p^2 \\
t &= s + q \\
u &= \sqrt{t} \\
y_1 &= -p + u \\
v &= p + u \\
y_2 &= q/v
\end{aligned}
$$

Answers for Selected Exercises*

Exercises 1.1

2. $x = \dfrac{6032}{9990}$; $x = \dfrac{6032}{10010}$ **3.** 6×10^{-5}

4. Two other ways:
$pi \leftarrow 2.0 \arcsin(1.0)$ or $pi \leftarrow 2.0 \arccos(0.0)$

5a.
$sum \leftarrow 0$
for $i = 1$ **to** n
  for $j = 1$ **to** n
    $sum \leftarrow sum + a_{ij}$
  end for
end for

5d.
$sum \leftarrow 0.0$
for $i = 1$ **to** n
  $sum \leftarrow sum + a_{ii}$
end for
for $j = 2$ **to** n
  for $i = j$ **to** n
    $sum \leftarrow sum + a_{i,i-j+1} + a_{i-j+1,i}$
  end for
end for

6. n multiplications and n additions/subtractions

8a.
for $i = 1$ **to** 5
  $x \leftarrow x \cdot x$
end for
$p \leftarrow x$

8c.
$z \leftarrow x + 2$
$p \leftarrow z^3\left(6 + z^4\left(9 + z^8\left(3 - z^{16}\right)\right)\right)$

10.
$z \leftarrow a_n / b_n$
for $i = 1$ **to** $n - 1$
  $z \leftarrow a_{n-i}(z + 1/b_{n-i})$
end for

11b.
$z \leftarrow 1$
$v \leftarrow 1$
for $i = 1$ **to** $n - 1$
  $v \leftarrow vx$
  $z \leftarrow vz + 1$
end for
$z \leftarrow vxz$

12b. $v = \displaystyle\sum_{i=0}^{n} a_i x^i$

12e. $v = a_n x^n + x \displaystyle\sum_{i=1}^{n} a_{n-i} x^{n-i}$

13. $z = 1 + \displaystyle\sum_{i=2}^{n} \prod_{j=2}^{i} b_j$ **14.** $n(n+1)/2$

15b.
for $j = 1$ **to** n
  for $i = 1$ **to** n
    $a_{ij} \leftarrow 1.0/\text{real}(i + j - 1)$
  end for
end for

Computer Exercises 1.1

4. $\exp(1.0) \approx 2.71828\ 18284\ 6$

9. Computation deviates from theory; for example, when $a_1 = 10^{-12}, 10^{-8}, 10^{-4}, 10^{20}$.

10. x may underflow and be set to zero.

12. 40 different spellings

20a. The computation m/n may result in truncation so that $x \neq y$.

Exercises 1.2

4a. First derivative $+\infty$ at 0.

4b. First derivative not continuous.

*Answers to exercises marked in the text with the symbol ^{a} are given here and in the *Student Solution Manual* with more details.

4e. Function $-\infty$ at 0.

5. $\cosh x = \sum_{k=0}^{\infty} \frac{x^{2k}}{(2k)!}$; $\cosh 0.7 \approx 1.25517$

6a. $e^{\cos x} = e\left(1 - \frac{x^2}{2} + \cdots\right)$

6b. $\sin(\cos x) = (\sin 1) - (\cos 1)\left(\frac{x^2}{2}\right) + \cdots$

7. $m = 2$ **8.** At least 18 terms

9. Yes. By using this formula, we avoid the series for e^{-x} and use the series for e^x.

11. $\ln(1-x) = -\sum_{k=1}^{\infty} \frac{x^k}{k}$;

$$\ln\left(\frac{1+x}{1-x}\right) = 2\sum_{k=1}^{\infty} \frac{x^{2k-1}}{(2k-1)}$$

12. $x = \frac{1}{3}$; $\ln 2 = 0.69313$ (four terms). At least 10 terms.

15a. $\sin x + \cos x = 1 + x - \frac{x^2}{2} - \frac{x^3}{6} + \cdots$;
$\sin(0.001) + \cos(0.001) \approx 1.00099\,94998\,3$

15b. $(\sin x)(\cos x) = x - \frac{2}{3}x^3 + \frac{2}{15}x^5 - \frac{4}{315}x^7 + \cdots$;
$\sin(0.0006)\cos(0.0006) \approx 0.00059\,99998\,57$

16. $\ln(e+x) = 1 + \sum_{n=1}^{\infty} (-1)^{n-1}\frac{1}{n}\left(\frac{x}{e}\right)^n$

17. At least seven terms.

18. At least 100 terms. **20.** $-\frac{5}{8}h^4$

23. $\frac{1}{8}\left(x - \frac{17}{4}\right)$

24.
$s \leftarrow 0$
for $i = 2$ **to** n
  $s \leftarrow s + \log(i)$
  output i, s
end for

28. $\left|\cos x - \left(1 - \frac{x^2}{2}\right)\right| < \frac{1}{16 \times 24} = \frac{1}{384}$

32. Maclaurin series:
$f(x) = 3 + 7x - 1.33x^2 + 19.2x^4$;

$$f(x) = 318.88 + (x-2)616.08 + \frac{(x-2)^2}{2!}918.94 + \frac{(x-2)^3}{3!}921.6 + \frac{(x-2)^4}{4!}460.8$$

35. 400 terms.

38. $\cos\left(\frac{\pi}{3} + h\right) = \frac{1}{2}\sum_{k=0}^{\infty}(-1)^k\frac{h^{2k}}{(2k)!} + \frac{\sqrt{3}}{2}\sum_{k=1}^{\infty}(-1)^k\frac{h^{2k-1}}{(2k-1)!}$;
$\cos(60.001^\circ) \approx 0.49998\,488$

39. $\sin(45.0005^\circ) \approx 0.70711\,295$

42. $f(x-h) = (x-h)^m = x^m - mhx^{m-1} + m(m-1)\frac{h^2}{2!}x^{m-2} + \cdots$

47. $n = 16$ or $n = 17$

50b. $\lim_{x\to 0}\frac{\arctan x}{x} = 1$ **50c.** $\lim_{x\to\pi}\frac{\cos x + 1}{\sin x} = 0$

51. At least 38 terms.

52. $\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}}\left[x - \frac{x^3}{3} + \frac{x^5}{5(2!)} - \frac{x^7}{7(3!)} + \cdots\right]$;
$\operatorname{erf}(1) \approx 0.8382$

53. 10^{10} **54.** 10^5

Computer Exercises 1.2

1.

	$c = 1$	$c = 10^8$
x_1	0	-1
x_2	-10^8	-10^8

14. g converges faster (in five iterations)

16. $\lambda_{50} = 1\,25862\,69025$ **17.** $\alpha_{50} = 2\,81437\,53123$

Exercises 1.3

1c. $[\text{B5}\,000000]_{16}$

2d. $[\text{3FA}\,0000000000000]_{16}$; $[\text{BFA}\,0000000000000]_{16}$

4d. $[\text{3E7}\,00000]_{16}$, **4e.** $[\text{3FCE}\,0000000000000]_{16}$

5d. $-\infty$ **8a.** -3.131968×10^6

8d. 9.992892×10^6 **8g.** -3.39×10^3

11c. $m = -1, 0, 1$. Nonnegative machine numbers:
$0, \frac{1}{8}, \frac{1}{4}, \frac{3}{8}, \frac{1}{2}, \frac{3}{4}, 1, \frac{3}{2}$

15. 1 **17.** 1.00005; 1.0 **18.** $|x| < 5 \times 10^{-5}$

19. β^{1-n} **21.** $\approx 3 \times 2^{-25}$ **25.** $\approx 3 \times 2^{-24}$

26. $\approx 2^{-22}$ **30.** $\approx n \times 2^{-24}$; $n = 1000, \approx 2^{-14}$

37. $\frac{1}{2} \times 10^{-12}$ rounding; 10^{-12} chopping **38.** 9%

39. The relative error cannot exceed 5×2^{-24}.

42. $\left((q - 2^{-25})2^m,\ (q + 2^{-25})2^m\right)$

Computer Exercises 1.3

3. 1.41423 56237 30922 58894 75783 33318 23348 99902 34375

7. On 32-bit computer, $s \leqq 33.2710$

8. Two limits are equal.

Exercises 1.4

4. $y = \dfrac{\cos^2 x}{1 + \sin x}$

6. $f(x) = -\frac{1}{2}x^3 - \frac{1}{2}x^4$;
$f(0.0125) \approx -9.888 \times 10^{-7}$

8. $f(x) = \dfrac{1}{\sqrt{1 + x^2} + 1} + 3 - 1.7x^2$; $f(0) = 3.5$

10. $f(x) = \dfrac{1}{\sqrt{x^2 + 1} + x}$

11. $f(x) = \begin{cases} \ln\left(x + \sqrt{x^2 + 1}\right), & x > 0 \\ 0, & x = 0 \\ -\ln\left(-x + \sqrt{x^2 + 1}\right), & x < 0 \end{cases}$

13. $z = \dfrac{x^4}{\sqrt{x^4 + 4} + 2}$

16. $f(x) \approx 1 - x + \dfrac{x^2}{3} - \dfrac{x^3}{6}$; $f(0.008) \approx 0.99202\,0915$

20. $\arctan x - x \approx x^3\left(-\frac{1}{3} + x^2\left(\frac{1}{5} + x^2\left(-\frac{1}{7}\right)\right)\right)$

22. $\left(e^{2x} - 1\right)/2x \approx 1 + x(1 + (x/3)(2 + x))$

24a. Near $\pi/2$, sine curve is relatively flat.

26b. $\ln x - 1 = \ln(x/e)$

26d. $x^{-2}(\sin x - e^x + 1) \approx -\frac{1}{2} - \frac{x}{3}$ when $x \to 0$

28. $|x| < \sqrt{6\varepsilon}$, where ε machine precision

29. $x_1 \approx 10^5$, $x_2 \approx 10^{-5}$

30. Not much. Expect to compute $b^2 - 4ac$ in double precision.

Computer Exercises 1.4

1. No solution; (0, 0); (0, 0);
Any solution; (−1., 0.);
(−0.10208 42383, −4.89791 57617);
(4.00000 00001, 4.0009 99999);
(−0.10208 42383, −4.89791 57617);
(1.0000 00000, 1.00000 0000E34);
(1.99683 77223, 2.00316 22777)

10.

x	Series	n
0	1.0	1
1	2.71828 18285	10
−1	0.36787 94412	10
0.5	1.64872 12707	8
−0.123	0.88426 36626	5
−25.5	$8.42346\,37545 \times 10^{-12}$	25
−1776	0	25
3.14159	23.14063 12270	17

14. $|x| < 10^{-15}$ **15.** $\rho_{50} = 2.85987$