gives the probability that any one of a series of trials will lie within x units of the mean, assuming that the trials have a normal distribution with mean 0 and standard deviation $\sqrt{2}/2$. This integral cannot be evaluated in terms of elementary functions, so an approximating technique must be used.

a. Integrate the Maclaurin series for e^{-x^2} to show that

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \sum_{k=0}^{\infty} \frac{(-1)^k x^{2k+1}}{(2k+1)k!}.$$

b. The error function can also be expressed in the form

$$\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} e^{-x^2} \sum_{k=0}^{\infty} \frac{2^k x^{2k+1}}{1 \cdot 3 \cdot 5 \cdots (2k+1)}.$$

Verify that the two series agree for $k = 1, 2, 3$, and 4. [*Hint:* Use the Maclaurin series for e^{-x^2}.]

c. Use the series in part (a) to approximate erf(1) to within 10^{-7}.

d. Use the same number of terms as in part (c) to approximate erf(1) with the series in part (b).

e. Explain why difficulties occur using the series in part (b) to approximate erf(x).

27. A function $f : [a, b] \to \mathbb{R}$ is said to satisfy a *Lipschitz condition* with Lipschitz constant L on $[a, b]$ if, for every $x, y \in [a, b]$, we have $|f(x) - f(y)| \le L|x - y|$.

a. Show that if f satisfies a Lipschitz condition with Lipschitz constant L on an interval $[a, b]$, then $f \in C[a, b]$.

b. Show that if f has a derivative that is bounded on $[a, b]$ by L, then f satisfies a Lipschitz condition with Lipschitz constant L on $[a, b]$.

c. Give an example of a function that is continuous on a closed interval but does not satisfy a Lipschitz condition on the interval.

28. Suppose $f \in C[a, b]$, that x_1 and x_2 are in $[a, b]$.

a. Show that a number ξ exists between x_1 and x_2 with

$$f(\xi) = \frac{f(x_1) + f(x_2)}{2} = \frac{1}{2} f(x_1) + \frac{1}{2} f(x_2).$$

b. Suppose that c_1 and c_2 are positive constants. Show that a number ξ exists between x_1 and x_2 with

$$f(\xi) = \frac{c_1 f(x_1) + c_2 f(x_2)}{c_1 + c_2}.$$

c. Give an example to show that the result in part **b.** does not necessarily hold when c_1 and c_2 have opposite signs with $c_1 \ne -c_2$.

29. Let $f \in C[a, b]$, and let p be in the open interval (a, b).

a. Suppose $f(p) \ne 0$. Show that a $\delta > 0$ exists with $f(x) \ne 0$, for all x in $[p - \delta, p + \delta]$, with $[p - \delta, p + \delta]$ a subset of $[a, b]$.

b. Suppose $f(p) = 0$ and $k > 0$ is given. Show that a $\delta > 0$ exists with $|f(x)| \le k$, for all x in $[p - \delta, p + \delta]$, with $[p - \delta, p + \delta]$ a subset of $[a, b]$.

1.2 Round-off Errors and Computer Arithmetic

The arithmetic performed by a calculator or computer is different from the arithmetic in algebra and calculus courses. You would likely expect that we always have as true statements things such as $2+2 = 4$, $4 \cdot 8 = 32$, and $(\sqrt{3})^2 = 3$. However, with *computer* arithmetic we expect exact results for $2+2 = 4$ and $4 \cdot 8 = 32$, but we will not have precisely $(\sqrt{3})^2 = 3$. To understand why this is true we must explore the world of finite-digit arithmetic.

In our traditional mathematical world we permit numbers with an infinite number of digits. The arithmetic we use in this world *defines* $\sqrt{3}$ as that unique positive number that when multiplied by itself produces the integer 3. In the computational world, however, each representable number has only a fixed and finite number of digits. This means, for example, that only rational numbers—and not even all of these—can be represented exactly. Since $\sqrt{3}$ is not rational, it is given an approximate representation, one whose square will not be precisely 3, although it will likely be sufficiently close to 3 to be acceptable in most situations. In most cases, then, this machine arithmetic is satisfactory and passes without notice or concern, but at times problems arise because of this discrepancy.

Error due to rounding should be expected whenever computations are performed using numbers that are not powers of 2. Keeping this error under control is extremely important when the number of calculations is large.

The error that is produced when a calculator or computer is used to perform real-number calculations is called **round-off error**. It occurs because the arithmetic performed in a machine involves numbers with only a finite number of digits, with the result that calculations are performed with only approximate representations of the actual numbers. In a computer, only a relatively small subset of the real number system is used for the representation of all the real numbers. This subset contains only rational numbers, both positive and negative, and stores the fractional part, together with an exponential part.

Binary Machine Numbers

In 1985, the IEEE (Institute for Electrical and Electronic Engineers) published a report called *Binary Floating Point Arithmetic Standard 754–1985*. An updated version was published in 2008 as *IEEE 754-2008*. This provides standards for binary and decimal floating point numbers, formats for data interchange, algorithms for rounding arithmetic operations, and for the handling of exceptions. Formats are specified for single, double, and extended precisions, and these standards are generally followed by all microcomputer manufacturers using floating-point hardware.

A 64-bit (binary digit) representation is used for a real number. The first bit is a sign indicator, denoted s. This is followed by an 11-bit exponent, c, called the **characteristic**, and a 52-bit binary fraction, f, called the **mantissa**. The base for the exponent is 2.

Since 52 binary digits correspond to between 16 and 17 decimal digits, we can assume that a number represented in this system has at least 16 decimal digits of precision. The exponent of 11 binary digits gives a range of 0 to $2^{11}-1 = 2047$. However, using only positive integers for the exponent would not permit an adequate representation of numbers with small magnitude. To ensure that numbers with small magnitude are equally representable, 1023 is subtracted from the characteristic, so the range of the exponent is actually from -1023 to 1024.

To save storage and provide a unique representation for each floating-point number, a normalization is imposed. Using this system gives a floating-point number of the form

$$(-1)^s 2^{c-1023}(1+f).$$

Illustration Consider the machine number

$$0\ 10000000011\ 1011100100010000000000000000000000000000000000000000.$$

The leftmost bit is $s = 0$, which indicates that the number is positive. The next 11 bits, 10000000011, give the characteristic and are equivalent to the decimal number

$$c = 1 \cdot 2^{10} + 0 \cdot 2^9 + \cdots + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 1024 + 2 + 1 = 1027.$$

The exponential part of the number is, therefore, $2^{1027-1023} = 2^4$. The final 52 bits specify that the mantissa is

$$f = 1 \cdot \left(\frac{1}{2}\right)^1 + 1 \cdot \left(\frac{1}{2}\right)^3 + 1 \cdot \left(\frac{1}{2}\right)^4 + 1 \cdot \left(\frac{1}{2}\right)^5 + 1 \cdot \left(\frac{1}{2}\right)^8 + 1 \cdot \left(\frac{1}{2}\right)^{12}.$$

As a consequence, this machine number precisely represents the decimal number

$$\begin{aligned}(-1)^s 2^{c-1023}(1+f) &= (-1)^0 \cdot 2^{1027-1023}\left(1 + \left(\frac{1}{2} + \frac{1}{8} + \frac{1}{16} + \frac{1}{32} + \frac{1}{256} + \frac{1}{4096}\right)\right) \\ &= 27.56640625.\end{aligned}$$

However, the next smallest machine number is

0 10000000011 1011100100001111111111111111111111111111111111111111,

and the next largest machine number is

0 10000000011 1011100100010000000000000000000000000000000000000001.

This means that our original machine number represents not only 27.56640625, but also half of the real numbers that are between 27.56640625 and the next smallest machine number, as well as half the numbers between 27.56640625 and the next largest machine number. To be precise, it represents any real number in the interval

[27.5664062499999982236431605997495353221893310546875,

27.5664062500000017763568394002504646778106689453125). □

The smallest normalized positive number that can be represented has $s = 0$, $c = 1$, and $f = 0$ and is equivalent to

$$2^{-1022} \cdot (1+0) \approx 0.22251 \times 10^{-307},$$

and the largest has $s = 0$, $c = 2046$, and $f = 1 - 2^{-52}$ and is equivalent to

$$2^{1023} \cdot (2 - 2^{-52}) \approx 0.17977 \times 10^{309}.$$

Numbers occurring in calculations that have a magnitude less than

$$2^{-1022} \cdot (1+0)$$

result in **underflow** and are generally set to zero. Numbers greater than

$$2^{1023} \cdot (2 - 2^{-52})$$

result in **overflow** and typically cause the computations to stop (unless the program has been designed to detect this occurrence). Note that there are two representations for the number zero; a positive 0 when $s = 0$, $c = 0$ and $f = 0$, and a negative 0 when $s = 1$, $c = 0$ and $f = 0$.

Decimal Machine Numbers

The use of binary digits tends to conceal the computational difficulties that occur when a finite collection of machine numbers is used to represent all the real numbers. To examine these problems, we will use more familiar decimal numbers instead of binary representation. Specifically, we assume that machine numbers are represented in the normalized *decimal* floating-point form

$$\pm 0.d_1d_2\ldots d_k \times 10^n, \quad 1 \le d_1 \le 9, \quad \text{and} \quad 0 \le d_i \le 9,$$

for each $i = 2, \ldots, k$. Numbers of this form are called k-digit *decimal machine numbers*.

Any positive real number within the numerical range of the machine can be normalized to the form

$$y = 0.d_1d_2\ldots d_kd_{k+1}d_{k+2}\ldots \times 10^n.$$

The error that results from replacing a number with its floating-point form is called **round-off error** regardless of whether the rounding or chopping method is used.

The floating-point form of y, denoted $fl(y)$, is obtained by terminating the mantissa of y at k decimal digits. There are two common ways of performing this termination. One method, called **chopping**, is to simply chop off the digits $d_{k+1}d_{k+2}\ldots$. This produces the floating-point form

$$fl(y) = 0.d_1d_2\ldots d_k \times 10^n.$$

The other method, called **rounding**, adds $5 \times 10^{n-(k+1)}$ to y and then chops the result to obtain a number of the form

$$fl(y) = 0.\delta_1\delta_2\ldots\delta_k \times 10^n.$$

For rounding, when $d_{k+1} \ge 5$, we add 1 to d_k to obtain $fl(y)$; that is, we *round up*. When $d_{k+1} < 5$, we simply chop off all but the first k digits; so we *round down*. If we round down, then $\delta_i = d_i$, for each $i = 1, 2, \ldots, k$. However, if we round up, the digits (and even the exponent) might change.

Example 1 Determine the five-digit **(a)** chopping and **(b)** rounding values of the irrational number π.

Solution The number π has an infinite decimal expansion of the form $\pi = 3.14159265\ldots$. Written in normalized decimal form, we have

$$\pi = 0.314159265\ldots \times 10^1.$$

The relative error is generally a better measure of accuracy than the absolute error because it takes into consideration the size of the number being approximated.

(a) The floating-point form of π using five-digit chopping is

$$fl(\pi) = 0.31415 \times 10^1 = 3.1415.$$

(b) The sixth digit of the decimal expansion of π is a 9, so the floating-point form of π using five-digit rounding is

$$fl(\pi) = (0.31415 + 0.00001) \times 10^1 = 3.1416.$$ ■

The following definition describes two methods for measuring approximation errors.

Definition 1.15 Suppose that p^* is an approximation to p. The **absolute error** is $|p - p^*|$, and the **relative error** is $\dfrac{|p - p^*|}{|p|}$, provided that $p \ne 0$. ■

Consider the absolute and relative errors in representing p by p^* in the following example.

Example 2 Determine the absolute and relative errors when approximating p by p^* when

(a) $p = 0.3000 \times 10^1$ and $p^* = 0.3100 \times 10^1$;

(b) $p = 0.3000 \times 10^{-3}$ and $p^* = 0.3100 \times 10^{-3}$;

(c) $p = 0.3000 \times 10^4$ and $p^* = 0.3100 \times 10^4$.

Solution

(a) For $p = 0.3000 \times 10^1$ and $p^* = 0.3100 \times 10^1$ the absolute error is 0.1, and the relative error is $0.333\overline{3} \times 10^{-1}$.

(b) For $p = 0.3000 \times 10^{-3}$ and $p^* = 0.3100 \times 10^{-3}$ the absolute error is 0.1×10^{-4}, and the relative error is $0.333\overline{3} \times 10^{-1}$.

(c) For $p = 0.3000 \times 10^4$ and $p^* = 0.3100 \times 10^4$, the absolute error is 0.1×10^3, and the relative error is again $0.333\overline{3} \times 10^{-1}$.

We often cannot find an accurate value for the true error in an approximation. Instead we find a bound for the error, which gives us a "worst-case" error.

This example shows that the same relative error, $0.333\overline{3} \times 10^{-1}$, occurs for widely varying absolute errors. As a measure of accuracy, the absolute error can be misleading and the relative error more meaningful, because the relative error takes into consideration the size of the value. ■

The following definition uses relative error to give a measure of significant digits of accuracy for an approximation.

Definition 1.16 The number p^* is said to approximate p to t **significant digits** (or figures) if t is the largest nonnegative integer for which

$$\frac{|p - p^*|}{|p|} \leq 5 \times 10^{-t}. \quad ■$$

The term significant digits is often used to loosely describe the number of decimal digits that appear to be accurate. The definition is more precise, and provides a continuous concept.

Table 1.1 illustrates the continuous nature of significant digits by listing, for the various values of p, the least upper bound of $|p - p^*|$, denoted max $|p - p^*|$, when p^* agrees with p to four significant digits.

Table 1.1

p	0.1	0.5	100	1000	5000	9990	10000
max $\lvert p - p^*\rvert$	0.00005	0.00025	0.05	0.5	2.5	4.995	5.

Returning to the machine representation of numbers, we see that the floating-point representation $fl(y)$ for the number y has the relative error

$$\left| \frac{y - fl(y)}{y} \right|.$$

If k decimal digits and chopping are used for the machine representation of

$$y = 0.d_1 d_2 \ldots d_k d_{k+1} \ldots \times 10^n,$$

then

$$\left|\frac{y - fl(y)}{y}\right| = \left|\frac{0.d_1d_2\ldots d_kd_{k+1}\ldots \times 10^n - 0.d_1d_2\ldots d_k \times 10^n}{0.d_1d_2\ldots \times 10^n}\right|$$
$$= \left|\frac{0.d_{k+1}d_{k+2}\ldots \times 10^{n-k}}{0.d_1d_2\ldots \times 10^n}\right| = \left|\frac{0.d_{k+1}d_{k+2}\ldots}{0.d_1d_2\ldots}\right| \times 10^{-k}.$$

Since $d_1 \neq 0$, the minimal value of the denominator is 0.1. The numerator is bounded above by 1. As a consequence,

$$\left|\frac{y - fl(y)}{y}\right| \leq \frac{1}{0.1} \times 10^{-k} = 10^{-k+1}.$$

In a similar manner, a bound for the relative error when using k-digit rounding arithmetic is $0.5 \times 10^{-k+1}$. (See Exercise 24.)

Note that the bounds for the relative error using k-digit arithmetic are independent of the number being represented. This result is due to the manner in which the machine numbers are distributed along the real line. Because of the exponential form of the characteristic, the same number of decimal machine numbers is used to represent each of the intervals $[0.1, 1]$, $[1, 10]$, and $[10, 100]$. In fact, within the limits of the machine, the number of decimal machine numbers in $[10^n, 10^{n+1}]$ is constant for all integers n.

Finite-Digit Arithmetic

In addition to inaccurate representation of numbers, the arithmetic performed in a computer is not exact. The arithmetic involves manipulating binary digits by various shifting, or logical, operations. Since the actual mechanics of these operations are not pertinent to this presentation, we shall devise our own approximation to computer arithmetic. Although our arithmetic will not give the exact picture, it suffices to explain the problems that occur. (For an explanation of the manipulations actually involved, the reader is urged to consult more technically oriented computer science texts, such as [Ma], *Computer System Architecture*.)

Assume that the floating-point representations $fl(x)$ and $fl(y)$ are given for the real numbers x and y and that the symbols $\oplus$, $\ominus$, $\otimes$, $\oplus\!\!\!\!\div$ represent machine addition, subtraction, multiplication, and division operations, respectively. We will assume a finite-digit arithmetic given by

$$x \oplus y = fl(fl(x) + fl(y)), \quad x \otimes y = fl(fl(x) \times fl(y)),$$
$$x \ominus y = fl(fl(x) - fl(y)), \quad x \oplus\!\!\!\!\div\, y = fl(fl(x) \div fl(y)).$$

This arithmetic corresponds to performing exact arithmetic on the floating-point representations of x and y and then converting the exact result to its finite-digit floating-point representation.

Rounding arithmetic is easily implemented in Maple. For example, the command

$Digits := 5$

causes all arithmetic to be rounded to 5 digits. To ensure that Maple uses approximate rather than exact arithmetic we use the *evalf*. For example, if $x = \pi$ and $y = \sqrt{2}$ then

$evalf(x);\ evalf(y)$

produces 3.1416 and 1.4142, respectively. Then $fl(fl(x) + fl(y))$ is performed using 5-digit rounding arithmetic with

$evalf(evalf(x) + evalf(y))$

which gives 4.5558. Implementing finite-digit chopping arithmetic is more difficult and requires a sequence of steps or a procedure. Exercise 27 explores this problem.

Example 3 Suppose that $x = \frac{5}{7}$ and $y = \frac{1}{3}$. Use five-digit chopping for calculating $x + y$, $x - y$, $x \times y$, and $x \div y$.

Solution Note that

$$x = \frac{5}{7} = 0.\overline{714285} \quad \text{and} \quad y = \frac{1}{3} = 0.\overline{3}$$

implies that the five-digit chopping values of x and y are

$$fl(x) = 0.71428 \times 10^0 \quad \text{and} \quad fl(y) = 0.33333 \times 10^0.$$

Thus

$$\begin{aligned} x \oplus y = fl(fl(x) + fl(y)) &= fl\left(0.71428 \times 10^0 + 0.33333 \times 10^0\right) \\ &= fl\left(1.04761 \times 10^0\right) = 0.10476 \times 10^1. \end{aligned}$$

The true value is $x + y = \frac{5}{7} + \frac{1}{3} = \frac{22}{21}$, so we have

$$\text{Absolute Error} = \left| \frac{22}{21} - 0.10476 \times 10^1 \right| = 0.190 \times 10^{-4}$$

and

$$\text{Relative Error} = \left| \frac{0.190 \times 10^{-4}}{22/21} \right| = 0.182 \times 10^{-4}.$$

Table 1.2 lists the values of this and the other calculations. ■

Table 1.2

Operation	Result	Actual value	Absolute error	Relative error
$x \oplus y$	0.10476×10^1	22/21	0.190×10^{-4}	0.182×10^{-4}
$x \ominus y$	0.38095×10^0	8/21	0.238×10^{-5}	0.625×10^{-5}
$x \otimes y$	0.23809×10^0	5/21	0.524×10^{-5}	0.220×10^{-4}
$x \oplus\!\!\!\!\div y$	0.21428×10^1	15/7	0.571×10^{-4}	0.267×10^{-4}

The maximum relative error for the operations in Example 3 is 0.267×10^{-4}, so the arithmetic produces satisfactory five-digit results. This is not the case in the following example.

Example 4 Suppose that in addition to $x = \frac{5}{7}$ and $y = \frac{1}{3}$ we have

$$u = 0.714251, \quad v = 98765.9, \quad \text{and} \quad w = 0.111111 \times 10^{-4},$$

so that

$$fl(u) = 0.71425 \times 10^0, \quad fl(v) = 0.98765 \times 10^5, \quad \text{and} \quad fl(w) = 0.11111 \times 10^{-4}.$$

Determine the five-digit chopping values of $x \ominus u$, $(x \ominus u) \oplus\!\!\!\!\div w$, $(x \ominus u) \otimes v$, and $u \oplus v$.

Solution These numbers were chosen to illustrate some problems that can arise with finite-digit arithmetic. Because x and u are nearly the same, their difference is small. The absolute error for $x \ominus u$ is

$$\begin{aligned}
|(x-u)-(x \ominus u)| &= |(x-u)-(fl(fl(x)-fl(u)))| \\
&= \left|\left(\frac{5}{7}-0.714251\right)-\left(fl\left(0.71428 \times 10^0 - 0.71425 \times 10^0\right)\right)\right| \\
&= \left|0.347143 \times 10^{-4} - fl\left(0.00003 \times 10^0\right)\right| = 0.47143 \times 10^{-5}.
\end{aligned}$$

This approximation has a small absolute error, but a large relative error

$$\left|\frac{0.47143 \times 10^{-5}}{0.347143 \times 10^{-4}}\right| \leq 0.136.$$

The subsequent division by the small number w or multiplication by the large number v magnifies the absolute error without modifying the relative error. The addition of the large and small numbers u and v produces large absolute error but not large relative error. These calculations are shown in Table 1.3. ■

Table 1.3

Operation	Result	Actual value	Absolute error	Relative error
$x \ominus u$	0.30000×10^{-4}	0.34714×10^{-4}	0.471×10^{-5}	0.136
$(x \ominus u) \oplus w$	0.27000×10^{1}	0.31242×10^{1}	0.424	0.136
$(x \ominus u) \otimes v$	0.29629×10^{1}	0.34285×10^{1}	0.465	0.136
$u \oplus v$	0.98765×10^{5}	0.98766×10^{5}	0.161×10^{1}	0.163×10^{-4}

One of the most common error-producing calculations involves the cancelation of significant digits due to the subtraction of nearly equal numbers. Suppose two nearly equal numbers x and y, with $x > y$, have the k-digit representations

$$fl(x) = 0.d_1 d_2 \ldots d_p \alpha_{p+1} \alpha_{p+2} \ldots \alpha_k \times 10^n,$$

and

$$fl(y) = 0.d_1 d_2 \ldots d_p \beta_{p+1} \beta_{p+2} \ldots \beta_k \times 10^n.$$

The floating-point form of $x - y$ is

$$fl(fl(x) - fl(y)) = 0.\sigma_{p+1}\sigma_{p+2} \ldots \sigma_k \times 10^{n-p},$$

where

$$0.\sigma_{p+1}\sigma_{p+2} \ldots \sigma_k = 0.\alpha_{p+1}\alpha_{p+2} \ldots \alpha_k - 0.\beta_{p+1}\beta_{p+2} \ldots \beta_k.$$

The floating-point number used to represent $x - y$ has at most $k - p$ digits of significance. However, in most calculation devices, $x - y$ will be assigned k digits, with the last p being either zero or randomly assigned. Any further calculations involving $x - y$ retain the problem of having only $k - p$ digits of significance, since a chain of calculations is no more accurate than its weakest portion.

If a finite-digit representation or calculation introduces an error, further enlargement of the error occurs when dividing by a number with small magnitude (or, equivalently, when multiplying by a number with large magnitude). Suppose, for example, that the number z has the finite-digit approximation $z + \delta$, where the error δ is introduced by representation or by previous calculation. Now divide by $\varepsilon = 10^{-n}$, where $n > 0$. Then

$$\frac{z}{\varepsilon} \approx fl\left(\frac{fl(z)}{fl(\varepsilon)}\right) = (z + \delta) \times 10^n.$$

The absolute error in this approximation, $|\delta| \times 10^n$, is the original absolute error, $|\delta|$, multiplied by the factor 10^n.

Example 5 Let $p = 0.54617$ and $q = 0.54601$. Use four-digit arithmetic to approximate $p - q$ and determine the absolute and relative errors using **(a)** rounding and **(b)** chopping.

Solution The exact value of $r = p - q$ is $r = 0.00016$.

(a) Suppose the subtraction is performed using four-digit rounding arithmetic. Rounding p and q to four digits gives $p^* = 0.5462$ and $q^* = 0.5460$, respectively, and $r^* = p^* - q^* = 0.0002$ is the four-digit approximation to r. Since

$$\frac{|r - r^*|}{|r|} = \frac{|0.00016 - 0.0002|}{|0.00016|} = 0.25,$$

the result has only one significant digit, whereas p^* and q^* were accurate to four and five significant digits, respectively.

(b) If chopping is used to obtain the four digits, the four-digit approximations to p, q, and r are $p^* = 0.5461$, $q^* = 0.5460$, and $r^* = p^* - q^* = 0.0001$. This gives

$$\frac{|r - r^*|}{|r|} = \frac{|0.00016 - 0.0001|}{|0.00016|} = 0.375,$$

which also results in only one significant digit of accuracy. ■

The loss of accuracy due to round-off error can often be avoided by a reformulation of the calculations, as illustrated in the next example.

Illustration The quadratic formula states that the roots of $ax^2 + bx + c = 0$, when $a \neq 0$, are

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \quad \text{and} \quad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}. \tag{1.1}$$

Consider this formula applied to the equation $x^2 + 62.10x + 1 = 0$, whose roots are approximately

$$x_1 = -0.01610723 \quad \text{and} \quad x_2 = -62.08390.$$

The roots x_1 and x_2 of a general quadratic equation are related to the coefficients by the fact that

$$x_1 + x_2 = -\frac{b}{a}$$

and

$$x_1 x_2 = \frac{c}{a}.$$

This is a special case of Vièta's Formulas for the coefficients of polynomials.

We will again use four-digit rounding arithmetic in the calculations to determine the root. In this equation, b^2 is much larger than $4ac$, so the numerator in the calculation for x_1 involves the *subtraction* of nearly equal numbers. Because

$$\sqrt{b^2 - 4ac} = \sqrt{(62.10)^2 - (4.000)(1.000)(1.000)}$$
$$= \sqrt{3856. - 4.000} = \sqrt{3852.} = 62.06,$$

we have

$$fl(x_1) = \frac{-62.10 + 62.06}{2.000} = \frac{-0.04000}{2.000} = -0.02000,$$

a poor approximation to $x_1 = -0.01611$, with the large relative error

$$\frac{|-0.01611 + 0.02000|}{|-0.01611|} \approx 2.4 \times 10^{-1}.$$

On the other hand, the calculation for x_2 involves the *addition* of the nearly equal numbers $-b$ and $-\sqrt{b^2 - 4ac}$. This presents no problem since

$$fl(x_2) = \frac{-62.10 - 62.06}{2.000} = \frac{-124.2}{2.000} = -62.10$$

has the small relative error

$$\frac{|-62.08 + 62.10|}{|-62.08|} \approx 3.2 \times 10^{-4}.$$

To obtain a more accurate four-digit rounding approximation for x_1, we change the form of the quadratic formula by *rationalizing the numerator*:

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}\left(\frac{-b - \sqrt{b^2 - 4ac}}{-b - \sqrt{b^2 - 4ac}}\right) = \frac{b^2 - (b^2 - 4ac)}{2a(-b - \sqrt{b^2 - 4ac})},$$

which simplifies to an alternate quadratic formula

$$x_1 = \frac{-2c}{b + \sqrt{b^2 - 4ac}}. \tag{1.2}$$

Using (1.2) gives

$$fl(x_1) = \frac{-2.000}{62.10 + 62.06} = \frac{-2.000}{124.2} = -0.01610,$$

which has the small relative error 6.2×10^{-4}.

The rationalization technique can also be applied to give the following alternative quadratic formula for x_2:

$$x_2 = \frac{-2c}{b - \sqrt{b^2 - 4ac}}. \tag{1.3}$$

This is the form to use if b is a negative number. In the Illustration, however, the mistaken use of this formula for x_2 would result in not only the subtraction of nearly equal numbers, but also the division by the small result of this subtraction. The inaccuracy that this combination produces,

$$fl(x_2) = \frac{-2c}{b - \sqrt{b^2 - 4ac}} = \frac{-2.000}{62.10 - 62.06} = \frac{-2.000}{0.04000} = -50.00,$$

has the large relative error 1.9×10^{-1}. □

- The lesson: Think before you compute!

Nested Arithmetic

Accuracy loss due to round-off error can also be reduced by rearranging calculations, as shown in the next example.

Example 6 Evaluate $f(x) = x^3 - 6.1x^2 + 3.2x + 1.5$ at $x = 4.71$ using three-digit arithmetic.

Solution Table 1.4 gives the intermediate results in the calculations.

Table 1.4

	x	x^2	x^3	$6.1x^2$	$3.2x$
Exact	4.71	22.1841	104.487111	135.32301	15.072
Three-digit (chopping)	4.71	22.1	104.	134.	15.0
Three-digit (rounding)	4.71	22.2	105.	135.	15.1

To illustrate the calculations, let us look at those involved with finding x^3 using three-digit rounding arithmetic. First we find

$$x^2 = 4.71^2 = 22.1841 \quad \text{which rounds to } 22.2.$$

Then we use this value of x^2 to find

$$x^3 = x^2 \cdot x = 22.2 \cdot 4.71 = 104.562 \quad \text{which rounds to } 105.$$

Also,

$$6.1x^2 = 6.1(22.2) = 135.42 \quad \text{which rounds to } 135,$$

and

$$3.2x = 3.2(4.71) = 15.072 \quad \text{which rounds to } 15.1.$$

The exact result of the evaluation is

$$\text{Exact:} \quad f(4.71) = 104.487111 - 135.32301 + 15.072 + 1.5 = -14.263899.$$

Using finite-digit arithmetic, the way in which we add the results can effect the final result. Suppose that we add left to right. Then for chopping arithmetic we have

$$\text{Three-digit (chopping):} \quad f(4.71) = ((104. - 134.) + 15.0) + 1.5 = -13.5,$$

and for rounding arithmetic we have

$$\text{Three-digit (rounding):} \quad f(4.71) = ((105. - 135.) + 15.1) + 1.5 = -13.4.$$

(You should carefully verify these results to be sure that your notion of finite-digit arithmetic is correct.) Note that the three-digit chopping values simply retain the leading three digits, with no rounding involved, and differ significantly from the three-digit rounding values.

The relative errors for the three-digit methods are

$$\text{Chopping: } \left|\frac{-14.263899 + 13.5}{-14.263899}\right| \approx 0.05, \quad \text{and Rounding: } \left|\frac{-14.263899 + 13.4}{-14.263899}\right| \approx 0.06.$$

■

Illustration

Remember that chopping (or rounding) is performed after each calculation.

As an alternative approach, the polynomial $f(x)$ in Example 6 can be written in a **nested** manner as

$$f(x) = x^3 - 6.1x^2 + 3.2x + 1.5 = ((x - 6.1)x + 3.2)x + 1.5.$$

Using three-digit chopping arithmetic now produces

$$\begin{aligned} f(4.71) &= ((4.71 - 6.1)4.71 + 3.2)4.71 + 1.5 = ((-1.39)(4.71) + 3.2)4.71 + 1.5 \\ &= (-6.54 + 3.2)4.71 + 1.5 = (-3.34)4.71 + 1.5 = -15.7 + 1.5 = -14.2. \end{aligned}$$

In a similar manner, we now obtain a three-digit rounding answer of -14.3. The new relative errors are

$$\text{Three-digit (chopping):} \quad \left|\frac{-14.263899 + 14.2}{-14.263899}\right| \approx 0.0045;$$

$$\text{Three-digit (rounding):} \quad \left|\frac{-14.263899 + 14.3}{-14.263899}\right| \approx 0.0025.$$

Nesting has reduced the relative error for the chopping approximation to less than 10% of that obtained initially. For the rounding approximation the improvement has been even more dramatic; the error in this case has been reduced by more than 95%. □

Polynomials should *always* be expressed in nested form before performing an evaluation, because this form minimizes the number of arithmetic calculations. The decreased error in the Illustration is due to the reduction in computations from four multiplications and three additions to two multiplications and three additions. One way to reduce round-off error is to reduce the number of computations.

EXERCISE SET 1.2

1. Compute the absolute error and relative error in approximations of p by p^*.

a. $p = \pi,\ p^* = 22/7$ **b.** $p = \pi,\ p^* = 3.1416$

c. $p = e,\ p^* = 2.718$ **d.** $p = \sqrt{2},\ p^* = 1.414$

e. $p = e^{10},\ p^* = 22000$ **f.** $p = 10^{\pi},\ p^* = 1400$

g. $p = 8!,\ p^* = 39900$ **h.** $p = 9!,\ p^* = \sqrt{18\pi}\,(9/e)^9$

2. Find the largest interval in which p^* must lie to approximate p with relative error at most 10^{-4} for each value of p.

a. π **b.** e

c. $\sqrt{2}$ **d.** $\sqrt[3]{7}$

3. Suppose p^* must approximate p with relative error at most 10^{-3}. Find the largest interval in which p^* must lie for each value of p.

a. 150 **b.** 900

c. 1500 **d.** 90

4. Perform the following computations (i) exactly, (ii) using three-digit chopping arithmetic, and (iii) using three-digit rounding arithmetic. (iv) Compute the relative errors in parts (ii) and (iii).

a. $\frac{4}{5} + \frac{1}{3}$ **b.** $\frac{4}{5} \cdot \frac{1}{3}$

c. $\left(\frac{1}{3} - \frac{3}{11}\right) + \frac{3}{20}$ **d.** $\left(\frac{1}{3} + \frac{3}{11}\right) - \frac{3}{20}$

5. Use three-digit rounding arithmetic to perform the following calculations. Compute the absolute error and relative error with the exact value determined to at least five digits.

a. $133 + 0.921$ **b.** $133 - 0.499$

c. $(121 - 0.327) - 119$ **d.** $(121 - 119) - 0.327$

e. $\dfrac{\frac{13}{14} - \frac{6}{7}}{2e - 5.4}$ **f.** $-10\pi + 6e - \dfrac{3}{62}$

g. $\left(\frac{2}{9}\right) \cdot \left(\frac{9}{7}\right)$ **h.** $\dfrac{\pi - \frac{22}{7}}{\frac{1}{17}}$

6. Repeat Exercise 5 using four-digit rounding arithmetic.

7. Repeat Exercise 5 using three-digit chopping arithmetic.

8. Repeat Exercise 5 using four-digit chopping arithmetic.