CHAPTER 1

Mathematical Modeling and Engineering Problem Solving

Knowledge and understanding are prerequisites for the effective implementation of any tool. No matter how impressive your tool chest, you will be hard-pressed to repair a car if you do not understand how it works.

This is particularly true when using computers to solve engineering problems. Although they have great potential utility, computers are practically useless without a fundamental understanding of how engineering systems work.

This understanding is initially gained by empirical means—that is, by observation and experiment. However, while such empirically derived information is essential, it is only half the story. Over years and years of observation and experiment, engineers and scientists have noticed that certain aspects of their empirical studies occur repeatedly. Such general behavior can then be expressed as fundamental laws that essentially embody the cumulative wisdom of past experience. Thus, most engineering problem solving employs the two-pronged approach of empiricism and theoretical analysis (Fig. 1.1).

It must be stressed that the two prongs are closely coupled. As new measurements are taken, the generalizations may be modified or new ones developed. Similarly, the generalizations can have a strong influence on the experiments and observations. In particular, generalizations can serve as organizing principles that can be employed to synthesize observations and experimental results into a coherent and comprehensive framework from which conclusions can be drawn. From an engineering problem-solving perspective, such a framework is most useful when it is expressed in the form of a mathematical model.

The primary objective of this chapter is to introduce you to mathematical modeling and its role in engineering problem solving. We will also illustrate how numerical methods figure in the process.

1.1 A SIMPLE MATHEMATICAL MODEL

A mathematical model can be broadly defined as a formulation or equation that expresses the essential features of a physical system or process in mathematical terms. In a very general sense, it can be represented as a functional relationship of the form

$$\begin{matrix}\text{Dependent}\\ \text{variable}\end{matrix} = f\left(\begin{matrix}\text{independent}\\ \text{variables}\end{matrix}, \text{ parameters, } \begin{matrix}\text{forcing}\\ \text{functions}\end{matrix}\right) \tag{1.1}$$

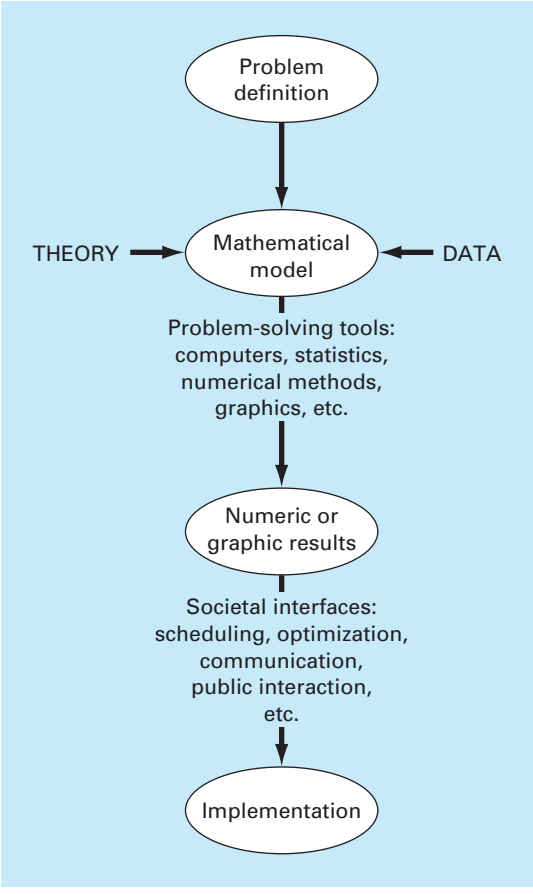


FIGURE 1.1
The engineering problem-solving process.

where the *dependent variable* is a characteristic that usually reflects the behavior or state of the system; the *independent variables* are usually dimensions, such as time and space, along which the system's behavior is being determined; the *parameters* are reflective of the system's properties or composition; and the *forcing functions* are external influences acting upon the system.

The actual mathematical expression of Eq. (1.1) can range from a simple algebraic relationship to large complicated sets of differential equations. For example, on the basis of his observations, Newton formulated his second law of motion, which states that the time rate of change of momentum of a body is equal to the resultant force acting on it. The mathematical expression, or model, of the second law is the well-known equation

$$F = ma \tag{1.2}$$

where F = net force acting on the body (N, or kg m/s^2), m = mass of the object (kg), and a = its acceleration (m/s^2).

FIGURE 1.2
Schematic diagram of the forces acting on a falling parachutist. F_D is the downward force due to gravity. F_U is the upward force due to air resistance.

The second law can be recast in the format of Eq. (1.1) by merely dividing both sides by m to give

$$a = \frac{F}{m} \tag{1.3}$$

where $a =$ the dependent variable reflecting the system's behavior, $F =$ the forcing function, and $m =$ a parameter representing a property of the system. Note that for this simple case there is no independent variable because we are not yet predicting how acceleration varies in time or space.

Equation (1.3) has several characteristics that are typical of mathematical models of the physical world:

1. It describes a natural process or system in mathematical terms.
2. It represents an idealization and simplification of reality. That is, the model ignores negligible details of the natural process and focuses on its essential manifestations. Thus, the second law does not include the effects of relativity that are of minimal importance when applied to objects and forces that interact on or about the earth's surface at velocities and on scales visible to humans.
3. Finally, it yields reproducible results and, consequently, can be used for predictive purposes. For example, if the force on an object and the mass of an object are known, Eq. (1.3) can be used to compute acceleration.

Because of its simple algebraic form, the solution of Eq. (1.2) can be obtained easily. However, other mathematical models of physical phenomena may be much more complex, and either cannot be solved exactly or require more sophisticated mathematical techniques than simple algebra for their solution. To illustrate a more complex model of this kind, Newton's second law can be used to determine the terminal velocity of a free-falling body near the earth's surface. Our falling body will be a parachutist (Fig. 1.2). A model for this case can be derived by expressing the acceleration as the time rate of change of the velocity (dv/dt) and substituting it into Eq. (1.3) to yield

$$\frac{dv}{dt} = \frac{F}{m} \tag{1.4}$$

where v is velocity (m/s) and t is time (s). Thus, the mass multiplied by the rate of change of the velocity is equal to the net force acting on the body. If the net force is positive, the object will accelerate. If it is negative, the object will decelerate. If the net force is zero, the object's velocity will remain at a constant level.

Next, we will express the net force in terms of measurable variables and parameters. For a body falling within the vicinity of the earth (Fig. 1.2), the net force is composed of two opposing forces: the downward pull of gravity F_D and the upward force of air resistance F_U:

$$F = F_D + F_U \tag{1.5}$$

If the downward force is assigned a positive sign, the second law can be used to formulate the force due to gravity, as

$$F_D = mg \tag{1.6}$$

where $g =$ the gravitational constant, or the acceleration due to gravity, which is approximately equal to 9.8 m/s^2.

Air resistance can be formulated in a variety of ways. A simple approach is to assume that it is linearly proportional to velocity[1] and acts in an upward direction, as in

$$F_U = -cv \tag{1.7}$$

where c = a proportionality constant called the *drag coefficient* (kg/s). Thus, the greater the fall velocity, the greater the upward force due to air resistance. The parameter c accounts for properties of the falling object, such as shape or surface roughness, that affect air resistance. For the present case, c might be a function of the type of jumpsuit or the orientation used by the parachutist during free-fall.

The net force is the difference between the downward and upward force. Therefore, Eqs. (1.4) through (1.7) can be combined to yield

$$\frac{dv}{dt} = \frac{mg - cv}{m} \tag{1.8}$$

or simplifying the right side,

$$\frac{dv}{dt} = g - \frac{c}{m}v \tag{1.9}$$

Equation (1.9) is a model that relates the acceleration of a falling object to the forces acting on it. It is a *differential equation* because it is written in terms of the differential rate of change (dv/dt) of the variable that we are interested in predicting. However, in contrast to the solution of Newton's second law in Eq. (1.3), the exact solution of Eq. (1.9) for the velocity of the falling parachutist cannot be obtained using simple algebraic manipulation. Rather, more advanced techniques such as those of calculus, must be applied to obtain an exact or analytical solution. For example, if the parachutist is initially at rest ($v = 0$ at $t = 0$), calculus can be used to solve Eq. (1.9) for

$$v(t) = \frac{gm}{c}\left(1 - e^{-(c/m)t}\right) \tag{1.10}$$

Note that Eq. (1.10) is cast in the general form of Eq. (1.1), where $v(t)$ = the dependent variable, t = the independent variable, c and m = parameters, and g = the forcing function.

EXAMPLE 1.1 Analytical Solution to the Falling Parachutist Problem

Problem Statement. A parachutist of mass 68.1 kg jumps out of a stationary hot air balloon. Use Eq. (1.10) to compute velocity prior to opening the chute. The drag coefficient is equal to 12.5 kg/s.

Solution. Inserting the parameters into Eq. (1.10) yields

$$v(t) = \frac{9.8(68.1)}{12.5}\left(1 - e^{-(12.5/68.1)t}\right) = 53.39\left(1 - e^{-0.18355t}\right)$$

which can be used to compute

[1]In fact, the relationship is actually nonlinear and might better be represented by a power relationship such as $F_U = -cv^2$. We will explore how such nonlinearities affect the model in a problem at the end of this chapter.

t, s	*v*, m/s
0	0.00
2	16.40
4	27.77
6	35.64
8	41.10
10	44.87
12	47.49
∞	53.39

According to the model, the parachutist accelerates rapidly (Fig. 1.3). A velocity of 44.87 m/s (100.4 mi/h) is attained after 10 s. Note also that after a sufficiently long time, a constant velocity, called the *terminal velocity*, of 53.39 m/s (119.4 mi/h) is reached. This velocity is constant because, eventually, the force of gravity will be in balance with the air resistance. Thus, the net force is zero and acceleration has ceased.

Equation (1.10) is called an *analytical*, or *exact*, *solution* because it exactly satisfies the original differential equation. Unfortunately, there are many mathematical models that cannot be solved exactly. In many of these cases, the only alternative is to develop a numerical solution that approximates the exact solution.

As mentioned previously, *numerical methods* are those in which the mathematical problem is reformulated so it can be solved by arithmetic operations. This can be illustrated

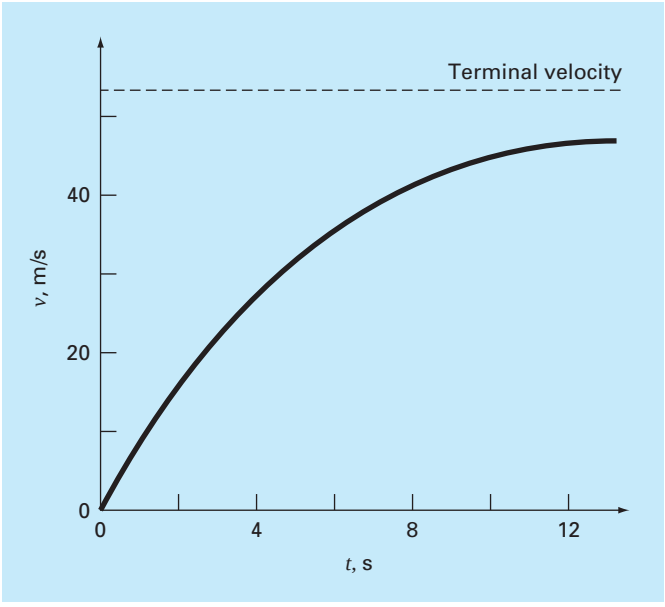


FIGURE 1.3
The analytical solution to the falling parachutist problem as computed in Example 1.1. Velocity increases with time and asymptotically approaches a terminal velocity.

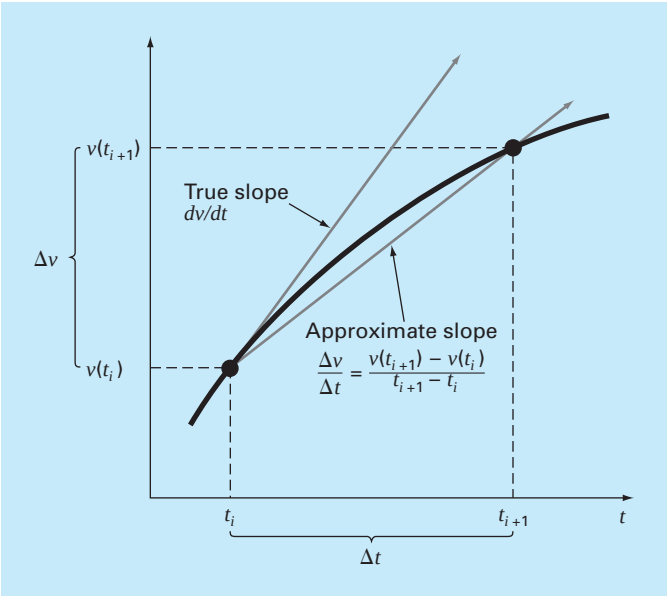


FIGURE 1.4
The use of a finite difference to approximate the first derivative of v with respect to t.

for Newton's second law by realizing that the time rate of change of velocity can be approximated by (Fig. 1.4):

$$\frac{dv}{dt} \cong \frac{\Delta v}{\Delta t} = \frac{v(t_{i+1}) - v(t_i)}{t_{i+1} - t_i} \tag{1.11}$$

where Δv and Δt = differences in velocity and time, respectively, computed over finite intervals, $v(t_i)$ = velocity at an initial time t_i, and $v(t_{i+1})$ = velocity at some later time t_{i+1}. Note that $dv/dt \cong \Delta v/\Delta t$ is approximate because Δt is finite. Remember from calculus that

$$\frac{dv}{dt} = \lim_{\Delta t \to 0} \frac{\Delta v}{\Delta t}$$

Equation (1.11) represents the reverse process.

Equation (1.11) is called a *finite divided difference* approximation of the derivative at time t_i. It can be substituted into Eq. (1.9) to give

$$\frac{v(t_{i+1}) - v(t_i)}{t_{i+1} - t_i} = g - \frac{c}{m} v(t_i)$$

This equation can then be rearranged to yield

$$v(t_{i+1}) = v(t_i) + \left[g - \frac{c}{m} v(t_i) \right] (t_{i+1} - t_i) \tag{1.12}$$

Notice that the term in brackets is the right-hand side of the differential equation itself [Eq. (1.9)]. That is, it provides a means to compute the rate of change or slope of v. Thus, the differential equation has been transformed into an equation that can be used to determine the velocity algebraically at t_{i+1} using the slope and previous values of v and t. If you are given an initial value for velocity at some time t_i, you can easily compute velocity at a

later time t_{i+1}. This new value of velocity at t_{i+1} can in turn be employed to extend the computation to velocity at t_{i+2} and so on. Thus, at any time along the way,

$$\text{New value} = \text{old value} + \text{slope} \times \text{step size}$$

Note that this approach is formally called *Euler's method.*

EXAMPLE 1.2 Numerical Solution to the Falling Parachutist Problem

Problem Statement. Perform the same computation as in Example 1.1 but use Eq. (1.12) to compute the velocity. Employ a step size of 2 s for the calculation.

Solution. At the start of the computation ($t_i = 0$), the velocity of the parachutist is zero. Using this information and the parameter values from Example 1.1, Eq. (1.12) can be used to compute velocity at $t_{i+1} = 2$ s:

$$v = 0 + \left[9.8 - \frac{12.5}{68.1}(0)\right] 2 = 19.60 \text{ m/s}$$

For the next interval (from $t = 2$ to 4 s), the computation is repeated, with the result

$$v = 19.60 + \left[9.8 - \frac{12.5}{68.1}(19.60)\right] 2 = 32.00 \text{ m/s}$$

The calculation is continued in a similar fashion to obtain additional values:

t, s	*v*, m/s
0	0.00
2	19.60
4	32.00
6	39.85
8	44.82
10	47.97
12	49.96
∞	53.39

The results are plotted in Fig. 1.5 along with the exact solution. It can be seen that the numerical method captures the essential features of the exact solution. However, because we have employed straight-line segments to approximate a continuously curving function, there is some discrepancy between the two results. One way to minimize such discrepancies is to use a smaller step size. For example, applying Eq. (1.12) at l-s intervals results in a smaller error, as the straight-line segments track closer to the true solution. Using hand calculations, the effort associated with using smaller and smaller step sizes would make such numerical solutions impractical. However, with the aid of the computer, large numbers of calculations can be performed easily. Thus, you can accurately model the velocity of the falling parachutist without having to solve the differential equation exactly.

As in the previous example, a computational price must be paid for a more accurate numerical result. Each halving of the step size to attain more accuracy leads to a doubling

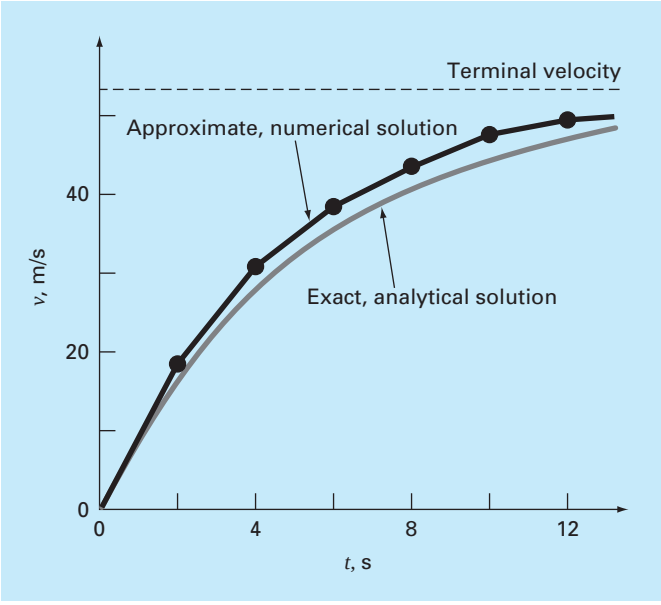


FIGURE 1.5
Comparison of the numerical and analytical solutions for the falling parachutist problem.

of the number of computations. Thus, we see that there is a trade-off between accuracy and computational effort. Such trade-offs figure prominently in numerical methods and constitute an important theme of this book. Consequently, we have devoted the Epilogue of Part One to an introduction to more of these trade-offs.

1.2 CONSERVATION LAWS AND ENGINEERING

Aside from Newton's second law, there are other major organizing principles in engineering. Among the most important of these are the conservation laws. Although they form the basis for a variety of complicated and powerful mathematical models, the great conservation laws of science and engineering are conceptually easy to understand. They all boil down to

$$\text{Change} = \text{increases} - \text{decreases} \tag{1.13}$$

This is precisely the format that we employed when using Newton's law to develop a force balance for the falling parachutist [Eq. (1.8)].

Although simple, Eq. (1.13) embodies one of the most fundamental ways in which conservation laws are used in engineering—that is, to predict changes with respect to time. We give Eq. (1.13) the special name *time-variable* (or *transient*) computation.

Aside from predicting changes, another way in which conservation laws are applied is for cases where change is nonexistent. If change is zero, Eq. (1.13) becomes

$$\text{Change} = 0 = \text{increases} - \text{decreases}$$

or

$$\text{Increases} = \text{decreases} \tag{1.14}$$

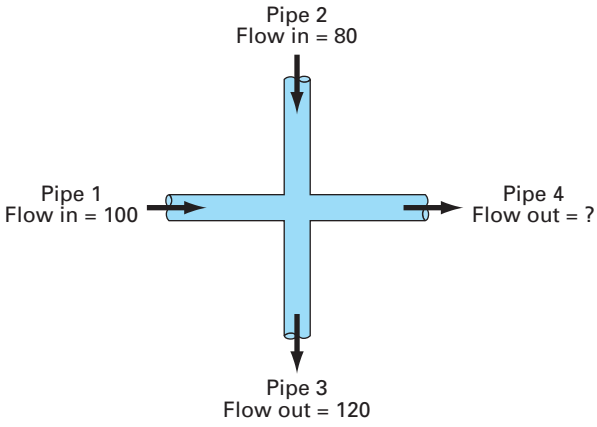


FIGURE 1.6
A flow balance for steady incompressible fluid flow at the junction of pipes.

Thus, if no change occurs, the increases and decreases must be in balance. This case, which is also given a special name—the *steady-state* computation—has many applications in engineering. For example, for steady-state incompressible fluid flow in pipes, the flow into a junction must be balanced by flow going out, as in

$$\text{Flow in} = \text{flow out}$$

For the junction in Fig. 1.6, the balance can be used to compute that the flow out of the fourth pipe must be 60.

For the falling parachutist, steady-state conditions would correspond to the case where the net force was zero, or [Eq. (1.8) with $dv/dt = 0$]

$$mg = cv \tag{1.15}$$

Thus, at steady state, the downward and upward forces are in balance, and Eq. (1.15) can be solved for the terminal velocity

$$v = \frac{mg}{c}$$

Although Eqs. (1.13) and (1.14) might appear trivially simple, they embody the two fundamental ways that conservation laws are employed in engineering. As such, they will form an important part of our efforts in subsequent chapters to illustrate the connection between numerical methods and engineering. Our primary vehicles for making this connection are the engineering applications that appear at the end of each part of this book.

Table 1.1 summarizes some of the simple engineering models and associated conservation laws that will form the basis for many of these engineering applications. Most of the chemical engineering applications will focus on mass balances for reactors. The mass balance is derived from the conservation of mass. It specifies that the change of mass of a chemical in the reactor depends on the amount of mass flowing in minus the mass flowing out.

Both the civil and mechanical engineering applications will focus on models developed from the conservation of momentum. For civil engineering, force balances are utilized to analyze structures such as the simple truss in Table 1.1. The same principles are employed for the mechanical engineering applications to analyze the transient up-and-down motion or vibrations of an automobile.

TABLE 1.1 Devices and types of balances that are commonly used in the four major areas of engineering. For each case, the conservation law upon which the balance is based is specified.

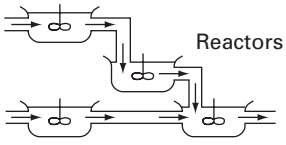
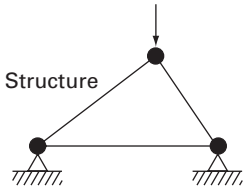
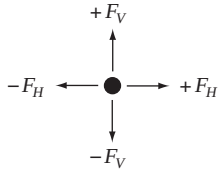
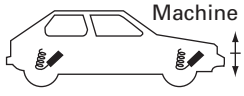
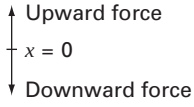
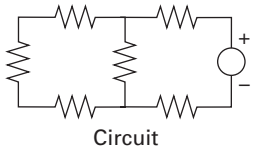
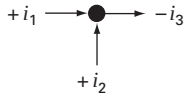
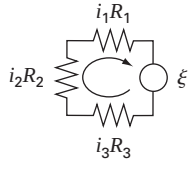
Field	Device	Organizing Principle	Mathematical Expression
Chemical engineering		Conservation of mass	Mass balance:  Over a unit of time period Δmass = inputs – outputs
Civil engineering		Conservation of momentum	Force balance:  At each node Σ horizontal forces (F_H) = 0 Σ vertical forces (F_V) = 0
Mechanical engineering		Conservation of momentum	Force balance:  $m\frac{d^2x}{dt^2}$ = downward force – upward force
Electrical engineering		Conservation of charge	Current balance:  For each node Σ current (i) = 0
		Conservation of energy	Voltage balance:  Around each loop Σ emf's – Σ voltage drops for resistors = 0 $\Sigma\,\xi - \Sigma\, iR = 0$

TABLE 1.2 Some practical issues that will be explored in the engineering applications at the end of each part of this book.

1. *Nonlinear versus linear.* Much of classical engineering depends on linearization to permit analytical solutions. Although this is often appropriate, expanded insight can often be gained if nonlinear problems are examined.
2. *Large versus small systems.* Without a computer, it is often not feasible to examine systems with over three interacting components. With computers and numerical methods, more realistic multicomponent systems can be examined.
3. *Nonideal versus ideal.* Idealized laws abound in engineering. Often there are nonidealized alternatives that are more realistic but more computationally demanding. Approximate numerical approaches can facilitate the application of these nonideal relationships.
4. *Sensitivity analysis.* Because they are so involved, many manual calculations require a great deal of time and effort for successful implementation. This sometimes discourages the analyst from implementing the multiple computations that are necessary to examine how a system responds under different conditions. Such sensitivity analyses are facilitated when numerical methods allow the computer to assume the computational burden.
5. *Design.* It is often a straightforward proposition to determine the performance of a system as a function of its parameters. It is usually more difficult to solve the inverse problem—that is, determining the parameters when the required performance is specified. Numerical methods and computers often permit this task to be implemented in an efficient manner.

Finally, the electrical engineering applications employ both current and energy balances to model electric circuits. The current balance, which results from the conservation of charge, is similar in spirit to the flow balance depicted in Fig. 1.6. Just as flow must balance at the junction of pipes, electric current must balance at the junction of electric wires. The energy balance specifies that the changes of voltage around any loop of the circuit must add up to zero. The engineering applications are designed to illustrate how numerical methods are actually employed in the engineering problem-solving process. As such, they will permit us to explore practical issues (Table 1.2) that arise in real-world applications. Making these connections between mathematical techniques such as numerical methods and engineering practice is a critical step in tapping their true potential. Careful examination of the engineering applications will help you to take this step.

PROBLEMS

1.1 Use calculus to solve Eq. (1.9) for the case where the initial velocity, $v(0)$ is nonzero.

1.2 Repeat Example 1.2. Compute the velocity to t = 10 s, with a step size of **(a)** 1 and **(b)** 0.5 s. Can you make any statement regarding the errors of the calculation based on the results?

1.3 Rather than the linear relationship of Eq. (1.7), you might choose to model the upward force on the parachutist as a second-order relationship,

$$F_U = -c'v^2$$

where c' = a second-order drag coefficient (kg/m).

(a) Using calculus, obtain the closed-form solution for the case where the jumper is initially at rest ($v = 0$ at $t = 0$).

(b) Repeat the numerical calculation in Example 1.2 with the same initial condition and parameter values. Use a value of 0.225 kg/m for c'.

1.4 For the free-falling parachutist with linear drag, assume a first jumper is 70 kg and has a drag coefficient of 12 kg/s. If a second jumper has a drag coefficient of 15 kg/s and a mass of 75 kg, how long will it take him to reach the same velocity the first jumper reached in 10 s?

1.5 Compute the velocity of a free-falling parachutist using Euler's method for the case where $m = 80$ kg and $c = 10$ kg/s. Perform the

calculation from $t = 0$ to 20 s with a step size of 1 s. Use an initial condition that the parachutist has an upward velocity of 20 m/s at $t = 0$. At $t = 10$ s, assume that the chute is instantaneously deployed so that the drag coefficient jumps to 50 kg/s.

1.6 The amount of a uniformly distributed radioactive contaminant contained in a closed reactor is measured by its concentration c (becquerel/liter or Bq/L). The contaminant decreases at a decay rate proportional to its concentration—that is

$$\text{decay rate} = -kc$$

where k is a constant with units of day^{-1}. Therefore, according to Eq. (1.13), a mass balance for the reactor can be written as

$$\begin{pmatrix}\text{change}\\ \text{in mass}\end{pmatrix} = \begin{pmatrix}\text{decrease}\\ \text{by decay}\end{pmatrix}$$

$$\frac{dc}{dt} = -kc$$

(a) Use Euler's method to solve this equation from $t = 0$ to 1 d with $k = 0.2\,\text{d}^{-1}$. Employ a step size of $\Delta t = 0.1$. The concentration at $t = 0$ is 10 Bq/L.

(b) Plot the solution on a semilog graph (i.e., ln c versus t) and determine the slope. Interpret your results.

1.7 A storage tank contains a liquid at depth y where $y = 0$ when the tank is half full. Liquid is withdrawn at a constant flow rate Q to meet demands. The contents are resupplied at a sinusoidal rate $3Q\sin^2(t)$.

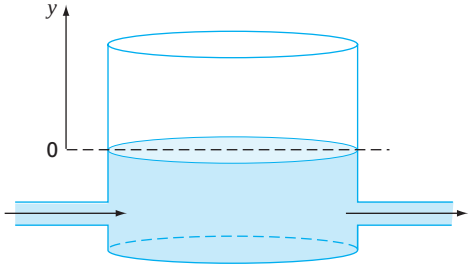


Figure P1.7

Equation (1.13) can be written for this system as

$$\begin{pmatrix}\text{change in}\\ \text{volume}\end{pmatrix} = \begin{pmatrix}\text{(inflow)}\end{pmatrix} - \begin{pmatrix}\text{(outflow)}\end{pmatrix}$$

$$\frac{d(Ay)}{dx} = 3Q\sin^2(t) - Q$$

or, since the surface area A is constant

$$\frac{dy}{dx} = 3\frac{Q}{A}\sin^2(t) - \frac{Q}{A}$$

Use Euler's method to solve for the depth y from $t = 0$ to 10 d with a step size of 0.5 d. The parameter values are $A = 1200$ m^2 and $Q = 500$ m^3/d. Assume that the initial condition is $y = 0$.

1.8 For the same storage tank described in Prob. 1.7, suppose that the outflow is not constant but rather depends on the depth. For this case, the differential equation for depth can be written as

$$\frac{dy}{dx} = 3\frac{Q}{A}\sin^2(t) - \frac{\alpha(1+y)^{1.5}}{A}$$

Use Euler's method to solve for the depth y from $t = 0$ to 10 d with a step size of 0.5 d. The parameter values are $A = 1200$ m^2, $Q = 500$ m^3/d, and $\alpha = 300$. Assume that the initial condition is $y = 0$.

1.9 The volume flow rate through a pipe is given by $Q = vA$, where v is the average velocity and A is the cross-sectional area. Use volume-continuity to solve for the required area in pipe 3.

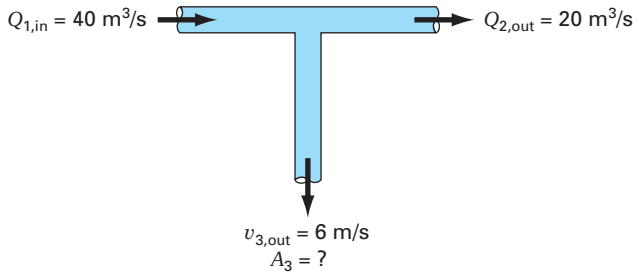


Figure P1.9

1.10 A group of 35 students attend a class in a room that measures 10 m by 8 m by 3 m. Each student takes up about 0.075 m^3 and gives out about 80 W of heat (1 W = 1 J/s). Calculate the air temperature rise during the first 15 minutes of the class if the room is completely sealed and insulated. Assume the heat capacity, C_v, for air is 0.718 kJ/(kg K). Assume air is an ideal gas at 20°C and 101.325 kPa. Note that the heat absorbed by the air Q is related to the mass of the air m, the heat capacity, and the change in temperature by the following relationship:

$$Q = m\int_{T_1}^{T_2} C_v dT = mC_v(T_2 - T_1)$$

The mass of air can be obtained from the ideal gas law:

$$PV = \frac{m}{\text{Mwt}}RT$$

where P is the gas pressure, V is the volume of the gas, Mwt is the molecular weight of the gas (for air, 28.97 kg/kmol), and R is the ideal gas constant [8.314 kPa m^3/(kmol K)].

1.11 Figure P1.11 depicts the various ways in which an average man gains and loses water in one day. One liter is ingested as food, and the body metabolically produces 0.3 L. In breathing air, the exchange is 0.05 L while inhaling, and 0.4 L while exhaling over a one-day period. The body will also lose 0.2, 1.4, 0.2, and 0.35 L through sweat, urine, feces, and through the skin, respectively. In order to maintain steady-state condition, how much water must be drunk per day?

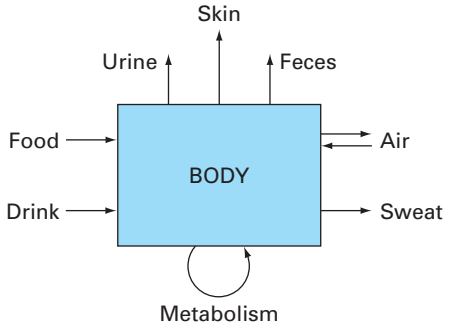


Figure P1.11

1.12 In our example of the free-falling parachutist, we assumed that the acceleration due to gravity was a constant value of 9.8 m/s^2. Although this is a decent approximation when we are examining falling objects near the surface of the earth, the gravitational force decreases as we move above sea level. A more general representation based on Newton's inverse square law of gravitational attraction can be written as

$$g(x) = g(0)\frac{R^2}{(R+x)^2}$$

where $g(x)$ = gravitational acceleration at altitude x (in m) measured upwards from the earth's surface (m/s^2), $g(0)$ = gravitational acceleration at the earth's surface ($\cong$ 9.8 m/s^2), and R = the earth's radius ($\cong 6.37 \times 10^6$ m).

(a) In a fashion similar to the derivation of Eq. (1.9) use a force balance to derive a differential equation for velocity as a function of time that utilizes this more complete representation of gravitation. However, for this derivation, assume that upward velocity is positive.

(b) For the case where drag is negligible, use the chain rule to express the differential equation as a function of altitude rather than time. Recall that the chain rule is

$$\frac{dv}{dt} = \frac{dv}{dx}\frac{dx}{dt}$$

(c) Use calculus to obtain the closed form solution where $v = v_0$ at $x = 0$.

(d) Use Euler's method to obtain a numerical solution from $x = 0$ to 100,000 m using a step of 10,000 m where the initial velocity is 1400 m/s upwards. Compare your result with the analytical solution.

1.13 Suppose that a spherical droplet of liquid evaporates at a rate that is proportional to its surface area.

$$\frac{dV}{dt} = -kA$$

where V = volume (mm^3), t = time (min), k = the evaporation rate (mm/min), and A = surface area (mm^2). Use Euler's method to compute the volume of the droplet from $t = 0$ to 10 min using a step size of 0.25 min. Assume that $k = 0.1$ mm/min and that the droplet initially has a radius of 3 mm. Assess the validity of your results by determining the radius of your final computed volume and verifying that it is consistent with the evaporation rate.

1.14 Newton's law of cooling says that the temperature of a body changes at a rate proportional to the difference between its temperature and that of the surrounding medium (the ambient temperature),

$$\frac{dT}{dt} = -k(T - T_a)$$

where T = the temperature of the body (°C), t = time (min), k = the proportionality constant (per minute), and T_a = the ambient temperature (°C). Suppose that a cup of coffee originally has a temperature of 68°C. Use Euler's method to compute the temperature from $t = 0$ to 10 min using a step size of 1 min if $T_a = 21$°C and $k =$ 0.1/min.

1.15 Water accounts for roughly 60% of total body weight. Assuming it can be categorized into six regions, the percentages go as follows. Plasma claims 4.5% of the body weight and is 7.5% of the total body water. Dense connective tissue and cartilage occupies 4.5% of the total body weight and 7.5% of the total body water. Interstitial lymph is 12% of the body weight, which is 20% of the total body water. Inaccessible bone water is roughly 7.5% of the total body water and 4.5% total body weight. If intracellular water is 33% of the total body weight and transcellular water is 2.5% of the total body water, what percent of total body weight must the transcellular water be and what percent of total body water must the intracellular water be?

1.16 Cancer cells grow exponentially with a doubling time of 20 h when they have an unlimited nutrient supply. However, as the cells start to form a solid spherical tumor without a blood supply, growth at the center of the tumor becomes limited, and eventually cells start to die.

(a) Exponential growth of cell number N can be expressed as shown, where μ is the growth rate of the cells. For cancer cells, find the value of μ.

$$\frac{dN}{dt} = \mu N$$

(b) Write an equation that will describe the rate of change of tumor volume during exponential growth given that the diameter of an individual cell is 20 microns.

(c) After a particular type of tumor exceeds 500 microns in diameter, the cells at the center of the tumor die (but continue to take up space in the tumor). Determine how long it will take for the tumor to exceed this critical size.

1.17 A fluid is pumped into the network shown in Fig. P1.17. If $Q_2 = 0.7$, $Q_3 = 0.5$, $Q_7 = 0.1$, and $Q_8 = 0.3$ m^3/s, determine the other flows.

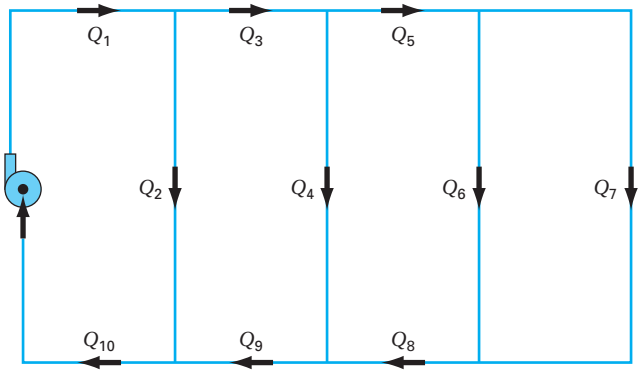


Figure P1.17

1.18 The following information is available for a bank account:

Date	Deposits	Withdrawals	Interest	Balance
5/1				1512.33
	220.13	327.26		
6/1				
	216.80	378.61		
7/1				
	450.25	106.80		
8/1				
	127.31	350.61		
9/1				

Note that the money earns interest which is computed as

$$\text{Interest} = i B_i$$

where i = the interest rate expressed as a fraction per month, and B_i the initial balance at the beginning of the month.

(a) Use the conservation of cash to compute the balance on 6/1, 7/1, 8/1, and 9/1 if the interest rate is 1% per month (i = 0.01/month). Show each step in the computation.

(b) Write a differential equation for the cash balance in the form

$$\frac{dB}{dt} = f(D(t), W(t), i)$$

where t = time (months), $D(t)$ = deposits as a function of time (\$/month), $W(t)$ = withdrawals as a function of time (\$/month). For this case, assume that interest is compounded continuously; that is, interest = $i B$.

(c) Use Euler's method with a time step of 0.5 month to simulate the balance. Assume that the deposits and withdrawals are applied uniformly over the month.

(d) Develop a plot of balance versus time for **(a)** and **(c)**.

1.19 The velocity is equal to the rate of change of distance x (m),

$$\frac{dx}{dt} = v(t) \tag{P1.19}$$

(a) Substitute Eq. (1.10) and develop an analytical solution for distance as a function of time. Assume that $x(0) = 0$.

(b) Use Euler's method to numerically integrate Eqs. (P1.19) and (1.9) in order to determine both the velocity and distance fallen as a function of time for the first 10 s of free fall using the same parameters as in Example 1.2.

(c) Develop a plot of your numerical results together with the analytical solutions.

CHAPTER 3

Approximations and Round-Off Errors

Because so many of the methods in this book are straightforward in description and application, it would be very tempting at this point for us to proceed directly to the main body of the text and teach you how to use these techniques. However, understanding the concept of error is so important to the effective use of numerical methods that we have chosen to devote the next two chapters to this topic.

The importance of error was introduced in our discussion of the falling parachutist in Chap. 1. Recall that we determined the velocity of a falling parachutist by both analytical and numerical methods. Although the numerical technique yielded estimates that were close to the exact analytical solution, there was a discrepancy, or *error*, because the numerical method involved an approximation. Actually, we were fortunate in that case because the availability of an analytical solution allowed us to compute the error exactly. For many applied engineering problems, we cannot obtain analytical solutions. Therefore, we cannot compute exactly the errors associated with our numerical methods. In these cases, we must settle for approximations or estimates of the errors.

Such errors are characteristic of most of the techniques described in this book. This statement might at first seem contrary to what one normally conceives of as sound engineering. Students and practicing engineers constantly strive to limit errors in their work. When taking examinations or doing homework problems, you are penalized, not rewarded, for your errors. In professional practice, errors can be costly and sometimes catastrophic. If a structure or device fails, lives can be lost.

Although perfection is a laudable goal, it is rarely, if ever, attained. For example, despite the fact that the model developed from Newton's second law is an excellent approximation, it would never in practice exactly predict the parachutist's fall. A variety of factors such as winds and slight variations in air resistance would result in deviations from the prediction. If these deviations are systematically high or low, then we might need to develop a new model. However, if they are randomly distributed and tightly grouped around the prediction, then the deviations might be considered negligible and the model deemed adequate. Numerical approximations also introduce similar discrepancies into the analysis. Again, the question is: How much the next error is present in our calculations and is it tolerable?

This chapter and Chap. 4 cover basic topics related to the identification, quantification, and minimization of these errors. In this chapter, general information concerned with the quantification of error is reviewed in the first sections. This is followed by a section on one

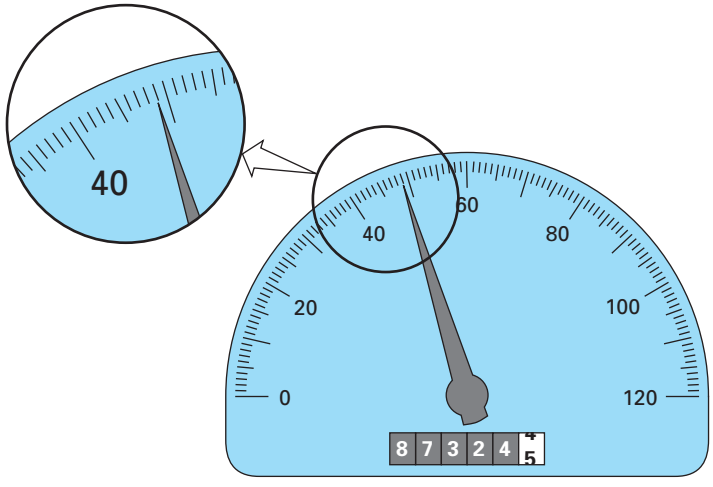
of the two major forms of numerical error: round-off error. *Round-off error* is due to the fact that computers can represent only quantities with a finite number of digits. Then Chap. 4 deals with the other major form: truncation error. *Truncation error* is the discrepancy introduced by the fact that numerical methods may employ approximations to represent exact mathematical operations and quantities. Finally, we briefly discuss errors not directly connected with the numerical methods themselves. These include blunders, formulation or model errors, and data uncertainty.

3.1 SIGNIFICANT FIGURES

This book deals extensively with approximations connected with the manipulation of numbers. Consequently, before discussing the errors associated with numerical methods, it is useful to review basic concepts related to approximate representation of the numbers themselves.

Whenever we employ a number in a computation, we must have assurance that it can be used with confidence. For example, Fig. 3.1 depicts a speedometer and odometer from an automobile. Visual inspection of the speedometer indicates that the car is traveling between 48 and 49 km/h. Because the indicator is higher than the midpoint between the markers on the gauge, we can say with assurance that the car is traveling at approximately 49 km/h. We have confidence in this result because two or more reasonable individuals reading this gauge would arrive at the same conclusion. However, let us say that we insist that the speed be estimated to one decimal place. For this case, one person might say 48.8, whereas another might say 48.9 km/h. Therefore, because of the limits of this instrument,

FIGURE 3.1
An automobile speedometer and odometer illustrating the concept of a significant figure.



only the first two digits can be used with confidence. Estimates of the third digit (or higher) must be viewed as approximations. It would be ludicrous to claim, on the basis of this speedometer, that the automobile is traveling at 48.8642138 km/h. In contrast, the odometer provides up to six certain digits. From Fig. 3.1, we can conclude that the car has traveled slightly less than 87,324.5 km during its lifetime. In this case, the seventh digit (and higher) is uncertain.

The concept of a significant figure, or digit, has been developed to formally designate the reliability of a numerical value. The *significant digits* of a number are those that can be used with confidence. They correspond to the number of certain digits plus one estimated digit. For example, the speedometer and the odometer in Fig. 3.1 yield readings of three and seven significant figures, respectively. For the speedometer, the two certain digits are 48. It is conventional to set the estimated digit at one-half of the smallest scale division on the measurement device. Thus the speedometer reading would consist of the three significant figures: 48.5. In a similar fashion, the odometer would yield a seven-significant-figure reading of 87,324.45.

Although it is usually a straightforward procedure to ascertain the significant figures of a number, some cases can lead to confusion. For example, zeros are not always significant figures because they may be necessary just to locate a decimal point. The numbers 0.00001845, 0.0001845, and 0.001845 all have four significant figures. Similarly, when trailing zeros are used in large numbers, it is not clear how many, if any, of the zeros are significant. For example, at face value the number 45,300 may have three, four, or five significant digits, depending on whether the zeros are known with confidence. Such uncertainty can be resolved by using scientific notation, where 4.53×10^4, 4.530×10^4, 4.5300×10^4 designate that the number is known to three, four, and five significant figures, respectively.

The concept of significant figures has two important implications for our study of numerical methods:

1. As introduced in the falling parachutist problem, numerical methods yield approximate results. We must, therefore, develop criteria to specify how confident we are in our approximate result. One way to do this is in terms of significant figures. For example, we might decide that our approximation is acceptable if it is correct to four significant figures.
2. Although quantities such as π, e, or $\sqrt{7}$ represent specific quantities, they cannot be expressed exactly by a limited number of digits. For example,

   $$\pi = 3.14159265358979323846264 3\ldots$$

   ad infinitum. Because computers retain only a finite number of significant figures, such numbers can never be represented exactly. The omission of the remaining significant figures is called round-off error.

Both round-off error and the use of significant figures to express our confidence in a numerical result will be explored in detail in subsequent sections. In addition, the concept of significant figures will have relevance to our definition of accuracy and precision in the next section.

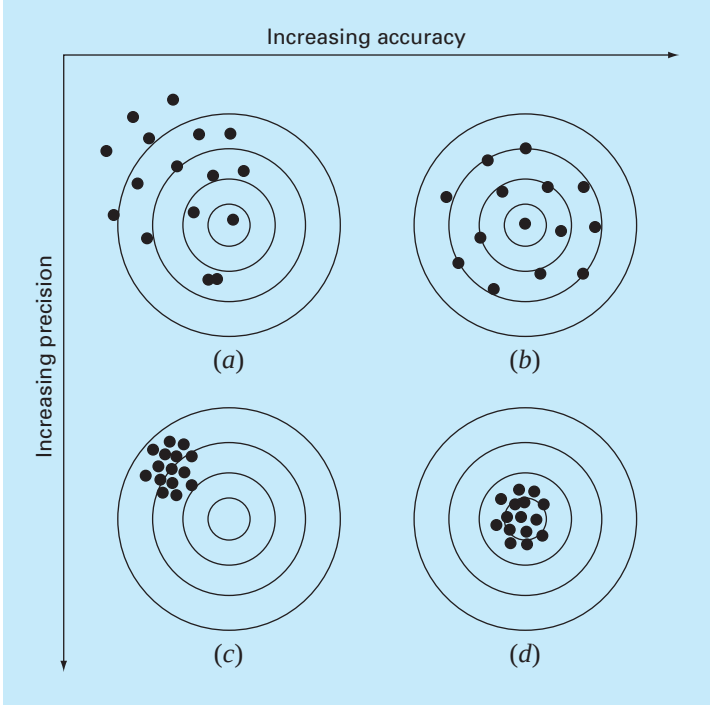
3.2 ACCURACY AND PRECISION

The errors associated with both calculations and measurements can be characterized with regard to their accuracy and precision. *Accuracy* refers to how closely a computed or measured value agrees with the true value. *Precision* refers to how closely individual computed or measured values agree with each other.

These concepts can be illustrated graphically using an analogy from target practice. The bullet holes on each target in Fig. 3.2 can be thought of as the predictions of a numerical technique, whereas the bull's-eye represents the truth. *Inaccuracy* (also called *bias*) is defined as systematic deviation from the truth. Thus, although the shots in Fig. 3.2*c* are more tightly grouped than those in Fig. 3.2*a*, the two cases are equally biased because they are both centered on the upper left quadrant of the target. *Imprecision* (also called *uncertainty*), on the other hand, refers to the magnitude of the scatter. Therefore, although Fig. 3.2*b* and *d* are equally accurate (that is, centered on the bull's-eye), the latter is more precise because the shots are tightly grouped.

Numerical methods should be sufficiently accurate or unbiased to meet the requirements of a particular engineering problem. They also should be precise enough for adequate

FIGURE 3.2
An example from marksmanship illustrating the concepts of accuracy and precision. (*a*) Inaccurate and imprecise; (*b*) accurate and imprecise; (*c*) inaccurate and precise; (*d*) accurate and precise.



engineering design. In this book, we will use the collective term *error* to represent both the inaccuracy and the imprecision of our predictions. With these concepts as background, we can now discuss the factors that contribute to the error of numerical computations.

3.3 ERROR DEFINITIONS

Numerical errors arise from the use of approximations to represent exact mathematical operations and quantities. These include *truncation errors*, which result when approximations are used to represent exact mathematical procedures, and *round-off errors*, which result when numbers having limited significant figures are used to represent exact numbers. For both types, the relationship between the exact, or true, result and the approximation can be formulated as

$$\text{True value} = \text{approximation} + \text{error} \tag{3.1}$$

By rearranging Eq. (3.1), we find that the numerical error is equal to the discrepancy between the truth and the approximation, as in

$$E_t = \text{true value} - \text{approximation} \tag{3.2}$$

where E_t is used to designate the exact value of the error. The subscript t is included to designate that this is the "true" error. This is in contrast to other cases, as described shortly, where an "approximate" estimate of the error must be employed.

A shortcoming of this definition is that it takes no account of the order of magnitude of the value under examination. For example, an error of a centimeter is much more significant if we are measuring a rivet rather than a bridge. One way to account for the magnitudes of the quantities being evaluated is to normalize the error to the true value, as in

$$\text{True fractional relative error} = \frac{\text{true error}}{\text{true value}}$$

where, as specified by Eq. (3.2), error = true value − approximation. The relative error can also be multiplied by 100 percent to express it as

$$\varepsilon_t = \frac{\text{true error}}{\text{true value}} 100\% \tag{3.3}$$

where ε_t designates the true percent relative error.

EXAMPLE 3.1 Calculation of Errors

Problem Statement. Suppose that you have the task of measuring the lengths of a bridge and a rivet and come up with 9999 and 9 cm, respectively. If the true values are 10,000 and 10 cm, respectively, compute (*a*) the true error and (*b*) the true percent relative error for each case.

Solution.

(a) The error for measuring the bridge is [Eq. (3.2)]

$$E_t = 10{,}000 - 9999 = 1 \text{ cm}$$

and for the rivet it is

$$E_t = 10 - 9 = 1 \text{ cm}$$

(b) The percent relative error for the bridge is [Eq. (3.3)]

$$\varepsilon_t = \frac{1}{10{,}000}100\% = 0.01\%$$

and for the rivet it is

$$\varepsilon_t = \frac{1}{10}100\% = 10\%$$

Thus, although both measurements have an error of 1 cm, the relative error for the rivet is much greater. We would conclude that we have done an adequate job of measuring the bridge, whereas our estimate for the rivet leaves something to be desired.

Notice that for Eqs. (3.2) and (3.3), E and ε are subscripted with a t to signify that the error is normalized to the true value. In Example 3.1, we were provided with this value. However, in actual situations such information is rarely available. For numerical methods, the true value will be known only when we deal with functions that can be solved analytically. Such will typically be the case when we investigate the theoretical behavior of a particular technique for simple systems. However, in real-world applications, we will obviously not know the true answer a priori. For these situations, an alternative is to normalize the error using the best available estimate of the true value, that is, to the approximation itself, as in

$$\varepsilon_a = \frac{\text{approximate error}}{\text{approximation}}100\% \tag{3.4}$$

where the subscript a signifies that the error is normalized to an approximate value. Note also that for real-world applications, Eq. (3.2) cannot be used to calculate the error term for Eq. (3.4). One of the challenges of numerical methods is to determine error estimates in the absence of knowledge regarding the true value. For example, certain numerical methods use an *iterative approach* to compute answers. In such an approach, a present approximation is made on the basis of a previous approximation. This process is performed repeatedly, or iteratively, to successively compute (we hope) better and better approximations. For such cases, the error is often estimated as the difference between previous and current approximations. Thus, percent relative error is determined according to

$$\varepsilon_a = \frac{\text{current approximation} - \text{previous approximation}}{\text{current approximation}}100\% \tag{3.5}$$

This and other approaches for expressing errors will be elaborated on in subsequent chapters.

The signs of Eqs. (3.2) through (3.5) may be either positive or negative. If the approximation is greater than the true value (or the previous approximation is greater than the current approximation), the error is negative; if the approximation is less than the true value, the error is positive. Also, for Eqs. (3.3) to (3.5), the denominator may be less than

zero, which can also lead to a negative error. Often, when performing computations, we may not be concerned with the sign of the error, but we are interested in whether the percent absolute value is lower than a prespecified percent tolerance ε_s. Therefore, it is often useful to employ the absolute value of Eqs. (3.2) through (3.5). For such cases, the computation is repeated until

$$|\varepsilon_a| < \varepsilon_s \tag{3.6}$$

If this relationship holds, our result is assumed to be within the prespecified acceptable level ε_s. Note that for the remainder of this text, we will almost exclusively employ absolute values when we use relative errors.

It is also convenient to relate these errors to the number of significant figures in the approximation. It can be shown (Scarborough, 1966) that if the following criterion is met, we can be assured that the result is correct to *at least n* significant figures.

$$\varepsilon_s = (0.5 \times 10^{2-n})\% \tag{3.7}$$

EXAMPLE 3.2 Error Estimates for Iterative Methods

Problem Statement. In mathematics, functions can often be represented by infinite series. For example, the exponential function can be computed using

$$e^x = 1 + x + \frac{x^2}{2} + \frac{x^3}{3!} + \cdots + \frac{x^n}{n!} \tag{E3.2.1}$$

Thus, as more terms are added in sequence, the approximation becomes a better and better estimate of the true value of e^x. Equation (E3.2.1) is called a *Maclaurin series expansion.*

Starting with the simplest version, $e^x = 1$, add terms one at a time to estimate $e^{0.5}$. After each new term is added, compute the true and approximate percent relative errors with Eqs. (3.3) and (3.5), respectively. Note that the true value is $e^{0.5} = 1.648721\ldots$. Add terms until the absolute value of the approximate error estimate ε_a falls below a prespecified error criterion ε_s conforming to three significant figures.

Solution. First, Eq. (3.7) can be employed to determine the error criterion that ensures a result is correct to at least three significant figures:

$$\varepsilon_s = (0.5 \times 10^{2-3})\% = 0.05\%$$

Thus, we will add terms to the series until ε_a falls below this level.

The first estimate is simply equal to Eq. (E3.2.1) with a single term. Thus, the first estimate is equal to 1. The second estimate is then generated by adding the second term, as in

$$e^x = 1 + x$$

or for $x = 0.5$,

$$e^{0.5} = 1 + 0.5 = 1.5$$

This represents a true percent relative error of [Eq. (3.3)]

$$\varepsilon_t = \frac{1.648721 - 1.5}{1.648721} 100\% = 9.02\%$$

Equation (3.5) can be used to determine an approximate estimate of the error, as in

$$\varepsilon_a = \frac{1.5 - 1}{1.5} 100\% = 33.3\%$$

Because ε_a is not less than the required value of ε_s,we would continue the computation by adding another term, $x^2/2!$, and repeating the error calculations. The process is continued until $\varepsilon_a < \varepsilon_s$. The entire computation can be summarized as

Terms	Result	ε_t (%)	ε_a (%)
1	1	39.3	
2	1.5	9.02	33.3
3	1.625	1.44	7.69
4	1.645833333	0.175	1.27
5	1.648437500	0.0172	0.158
6	1.648697917	0.00142	0.0158

Thus, after six terms are included, the approximate error falls below $\varepsilon_s = 0.05\%$ and the computation is terminated. However, notice that, rather than three significant figures, the result is accurate to five! This is because, for this case, both Eqs. (3.5) and (3.7) are conservative. That is, they ensure that the result is at least as good as they specify. Although, as discussed in Chap. 6, this is not always the case for Eq. (3.5), it is true most of the time.

3.3.1 Computer Algorithm for Iterative Calculations

Many of the numerical methods described in the remainder of this text involve iterative calculations of the sort illustrated in Example 3.2. These all entail solving a mathematical problem by computing successive approximations to the solution starting from an initial guess.

The computer implementation of such iterative solutions involves loops. As we saw in Sec. 2.1.1, these come in two basic flavors: count-controlled and decision loops. Most iterative solutions use decision loops. Thus, rather than employing a prespecified number of iterations, the process typically is repeated until an approximate error estimate falls below a stopping criterion as in Example 3.2.

A pseudocode for a generic iterative calculation is presented in Fig. 3.3. The function is passed a value (`val`) along with a stopping error criterion (`es`) and a maximum allowable number of iterations (`maxit`). The value is typically either (1) an initial value or (2) the value for which the iterative calculation is to be made.

The function first initializes three variables. These include (1) a variable `iter` that keeps track of the number of iterations, (2) a variable `sol` that holds the current estimate of the solution, and (3) a variable `ea` that holds the approximate percent relative error. Note that `ea` is initially set to a value of 100 to ensure that the loop executes at least once.

These initializations are followed by the decision loop that actually implements the iterative calculation. Prior to generating a new solution, `sol` is first assigned to `solold`. Then a new value of `sol` is computed and the iteration counter is incremented. If the new value of `sol` is nonzero, the percent relative error `ea` is determined. The stopping criteria

```
FUNCTION IterMeth(val, es, maxit)
iter = 1
sol = val
ea = 100
DO
  solold = sol
  sol = ...
  iter = iter + 1
  IF sol ≠ 0 ea=abs((sol − solold)/sol)*100
  IF ea ≤ es OR iter ≥ maxit EXIT
END DO
IterMeth = sol
END IterMeth
```

FIGURE 3.3
Pseudocode for a generic iterative calculation.

are then tested. If both are false, the loop repeats. If either are true, the loop terminates and the final solution is sent back to the function call. The following example illustrates how the generic algorithm can be applied to a specific iterative calculation.

EXAMPLE 3.3

Computer Implementation of an Iterative Calculation

Problem Statement. Develop a computer program based on the pseudocode from Fig. 3.3 to implement the calculation from Example 3.2.

Solution. A function to implement the Maclaurin series expansion for e^x can be based on the general scheme in Fig. 3.3. To do this, we first formulate the series expansion as a formula:

$$e^x \cong \sum_{i=0}^{n} \frac{x^n}{n!}$$

Figure 3.4 shows functions to implement this series written in VBA and MATLAB. Similar codes could be developed in other languages such a C++ or Fortran 95. Notice that whereas MATLAB has a built-in `factorial` function, it is necessary to compute the factorial as part of the VBA implementation with a simple product accumulator `fac`.

When the programs are run, they generate an estimate for the exponential function. For the MATLAB version, the answer is returned along with the approximate error and the number of iterations. For example, e^1 can be evaluated as

```
>> format long
>> [val, ea, iter] = IterMeth(1,1e-6,100)

val =
   2.718281826198493
ea =
    9.216155641522974e-007
iter =
    12
```

We can see that after 12 iterations, we obtain a result of 2.7182818 with an approximate error estimate of $= 9.2162 \times 10^{-7}\%$. The result can be verified by using the built-in `exp` function to directly calculate the exact value and the true percent relative error,

```
>> trueval=exp(1)

trueval =
   2.718281828459046

>> et=abs((trueval-val)/trueval)*100
et =
    8.316108397236229e-008
```

As was the case with Example 3.2, we obtain the desirable outcome that the true error is less than the approximate error.

(*a*) VBA/Excel

```
Function IterMeth(x, es, maxit)
‘ initialization
iter = 1
sol = 1
ea = 100
fac = 1
‘ iterative calculation
Do
  solold = sol
  fac = fac * iter
  sol = sol + x ^ iter / fac
  iter = iter + 1
  If sol <> 0 Then
    ea = Abs((sol - solold) / sol) * 100
  End If
  If ea <= es Or iter >= maxit Then Exit Do
Loop
IterMeth = sol
End Function
```

(*b*) MATLAB

```
function [v,ea,iter] = IterMeth(x,es,maxit)
% initialization
iter = 1;
sol = 1;
ea = 100;

% iterative calculation
while (1)
  solold = sol;

  sol = sol + x ^ iter / factorial(iter);
  iter = iter + 1;
  if sol~=0
    ea=abs((sol - solold)/sol)*100;
  end
  if ea<=es | iter>=maxit,break,end
end
v = sol;
end
```

FIGURE 3.4
(*a*) VBA/Excel and (*b*) MATLAB functions based on the pseudocode from Fig. 3.3.

With the preceding definitions as background, we can now proceed to the two types of error connected directly with numerical methods: round-off errors and truncation errors.

3.4 ROUND-OFF ERRORS

As mentioned previously, round-off errors originate from the fact that computers retain only a fixed number of significant figures during a calculation. Numbers such as π, e, or $\sqrt{7}$ cannot be expressed by a fixed number of significant figures. Therefore, they cannot be represented exactly by the computer. In addition, because computers use a base-2 representation, they cannot precisely represent certain exact base-10 numbers. The discrepancy introduced by this omission of significant figures is called *round-off error.*

3.4.1 Computer Representation of Numbers

Numerical round-off errors are directly related to the manner in which numbers are stored in a computer. The fundamental unit whereby information is represented is called a *word.* This is an entity that consists of a string of *b*inary dig*its,* or *bits.* Numbers are typically stored in one or more words. To understand how this is accomplished, we must first review some material related to number systems.

Number Systems. A *number system* is merely a convention for representing quantities. Because we have 10 fingers and 10 toes, the number system that we are most familiar with is the *decimal,* or *base*-10, number system. A base is the number used as the reference for constructing the system. The base-10 system uses the 10 digits—0, 1, 2, 3, 4, 5, 6, 7, 8, 9—to represent numbers. By themselves, these digits are satisfactory for counting from 0 to 9.

For larger quantities, combinations of these basic digits are used, with the position or *place value* specifying the magnitude. The right-most digit in a whole number represents a number from 0 to 9. The second digit from the right represents a multiple of 10. The third digit from the right represents a multiple of 100 and so on. For example, if we have the number 86,409 then we have eight groups of 10,000, six groups of 1000, four groups of 100, zero groups of 10, and nine more units, or

$$(8 \times 10^4) + (6 \times 10^3) + (4 \times 10^2) + (0 \times 10^1) + (9 \times 10^0) = 86{,}409$$

Figure 3.5*a* provides a visual representation of how a number is formulated in the base-10 system. This type of representation is called *positional notation.*

Because the decimal system is so familiar, it is not commonly realized that there are alternatives. For example, if human beings happened to have had eight fingers and eight toes, we would undoubtedly have developed an *octal,* or *base*-8, representation. In the same sense, our friend the computer is like a two-fingered animal who is limited to two states—either 0 or 1. This relates to the fact that the primary logic units of digital computers are on/off electronic components. Hence, numbers on the computer are represented with a *binary,* or *base*-2, system. Just as with the decimal system, quantities can be represented using positional notation. For example, the binary number 11 is equivalent to $(1 \times 2^1) + (1 \times 2^0) = 2 + 1 = 3$ in the decimal system. Figure 3.5*b* illustrates a more complicated example.

Integer Representation. Now that we have reviewed how base-10 numbers can be represented in binary form, it is simple to conceive of how integers are represented on a computer. The most straightforward approach, called the *signed magnitude method,* employs the first bit of a word to indicate the sign, with a 0 for positive and a 1 for negative. The

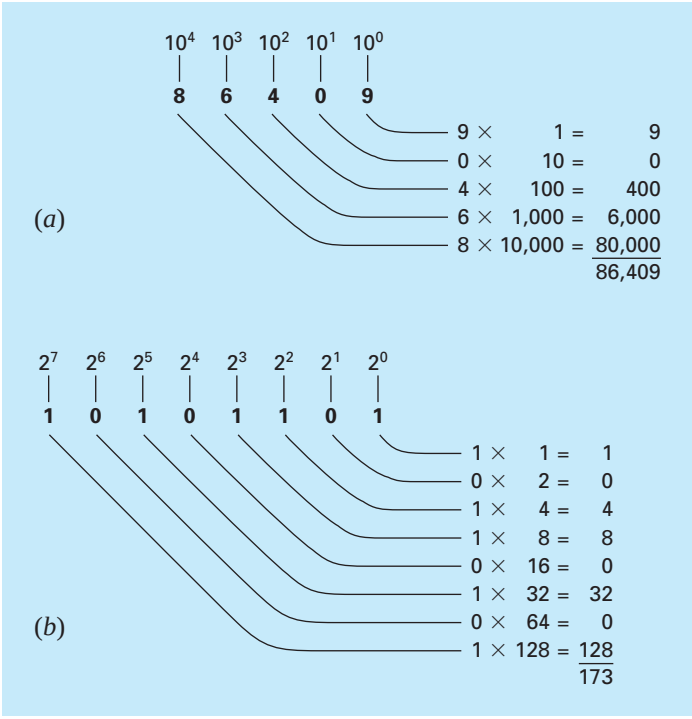


FIGURE 3.5
How the (*a*) decimal (base 10) and the (*b*) binary (base 2) systems work. In (*b*), the binary number 10101101 is equivalent to the decimal number 173.

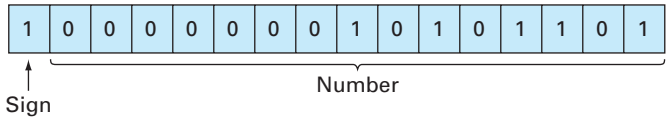


FIGURE 3.6
The representation of the decimal integer −173 on a 16-bit computer using the signed magnitude method.

remaining bits are used to store the number. For example, the integer value of −173 would be stored on a 16-bit computer, as in Fig. 3.6.

EXAMPLE 3.4 Range of Integers

Problem Statement. Determine the range of integers in base-10 that can be represented on a 16-bit computer.

Solution. Of the 16 bits, the first bit holds the sign. The remaining 15 bits can hold binary numbers from 0 to 111111111111111. The upper limit can be converted to a decimal integer, as in

$$(1 \times 2^{14}) + (1 \times 2^{13}) + \cdots + (1 \times 2^{1}) + (1 \times 2^{0})$$

which equals 32,767 (note that this expression can be simply evaluated as $2^{15} - 1$). Thus, a 16-bit computer word can store decimal integers ranging from −32,767 to 32,767. In addition, because zero is already defined as 0000000000000000, it is redundant to use the number 1000000000000000 to define a "minus zero." Therefore, it is usually employed to represent an additional negative number: −32,768, and the range is from −32,768 to 32,767.

Note that the signed magnitude method described above is not used to represent integers on conventional computers. A preferred approach called the *2's complement* technique directly incorporates the sign into the number's magnitude rather than providing a separate bit to represent plus or minus (see Chapra and Canale 1994). However, Example 3.4 still serves to illustrate how all digital computers are limited in their capability to represent integers. That is, numbers above or below the range cannot be represented. A more serious limitation is encountered in the storage and manipulation of fractional quantities as described next.

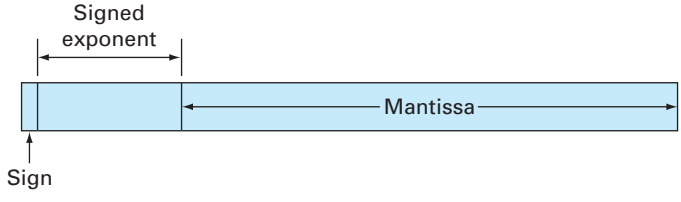
Floating-Point Representation. Fractional quantities are typically represented in computers using floating-point form. In this approach, the number is expressed as a fractional part, called a *mantissa* or *significand*, and an integer part, called an *exponent* or *characteristic*, as in

$$m \cdot b^{e}$$

where m = the mantissa, b = the base of the number system being used, and e = the exponent. For instance, the number 156.78 could be represented as 0.15678×10^{3} in a floating-point base-10 system.

Figure 3.7 shows one way that a floating-point number could be stored in a word. The first bit is reserved for the sign, the next series of bits for the signed exponent, and the last bits for the mantissa.

FIGURE 3.7
The manner in which a floating-point number is stored in a word.



Note that the mantissa is usually *normalized* if it has leading zero digits. For example, suppose the quantity $1/34 = 0.029411765\ldots$ was stored in a floating-point base-10 system that allowed only four decimal places to be stored. Thus, $1/34$ would be stored as

$$0.0294 \times 10^0$$

However, in the process of doing this, the inclusion of the useless zero to the right of the decimal forces us to drop the digit 1 in the fifth decimal place. The number can be normalized to remove the leading zero by multiplying the mantissa by 10 and lowering the exponent by 1 to give

$$0.2941 \times 10^{-1}$$

Thus, we retain an additional significant figure when the number is stored.

The consequence of normalization is that the absolute value of m is limited. That is,

$$\frac{1}{b} \le m < 1 \tag{3.8}$$

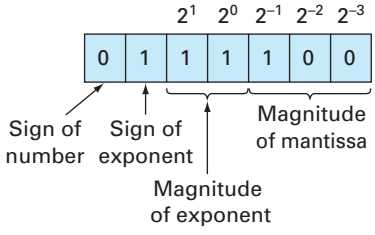
where $b =$ the base. For example, for a base-10 system, m would range between 0.1 and 1, and for a base-2 system, between 0.5 and 1.

Floating-point representation allows both fractions and very large numbers to be expressed on the computer. However, it has some disadvantages. For example, floating-point numbers take up more room and take longer to process than integer numbers. More significantly, however, their use introduces a source of error because the mantissa holds only a finite number of significant figures. Thus, a round-off error is introduced.

EXAMPLE 3.5 Hypothetical Set of Floating-Point Numbers

Problem Statement. Create a hypothetical floating-point number set for a machine that stores information using 7-bit words. Employ the first bit for the sign of the number, the next three for the sign and the magnitude of the exponent, and the last three for the magnitude of the mantissa (Fig. 3.8).

FIGURE 3.8
The smallest possible positive floating-point number from Example 3.5.



Solution. The smallest possible positive number is depicted in Fig. 3.8. The initial 0 indicates that the quantity is positive. The 1 in the second place designates that the exponent has a negative sign. The 1's in the third and fourth places give a maximum value to the exponent of

$$1 \times 2^1 + 1 \times 2^0 = 3$$

Therefore, the exponent will be -3. Finally, the mantissa is specified by the 100 in the last three places, which conforms to

$$1 \times 2^{-1} + 0 \times 2^{-2} + 0 \times 2^{-3} = 0.5$$

Although a smaller mantissa is possible (e.g., 000, 001, 010, 011), the value of 100 is used because of the limit imposed by normalization [Eq. (3.8)]. Thus, the smallest possible positive number for this system is $+0.5 \times 2^{-3}$, which is equal to 0.0625 in the base-10 system. The next highest numbers are developed by increasing the mantissa, as in

$$0111101 = (1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3}) \times 2^{-3} = (0.078125)_{10}$$
$$0111110 = (1 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3}) \times 2^{-3} = (0.093750)_{10}$$
$$0111111 = (1 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3}) \times 2^{-3} = (0.109375)_{10}$$

Notice that the base-10 equivalents are spaced evenly with an interval of 0.015625.

At this point, to continue increasing, we must decrease the exponent to 10, which gives a value of

$$1 \times 2^1 + 0 \times 2^0 = 2$$

The mantissa is decreased back to its smallest value of 100. Therefore, the next number is

$$0110100 = (1 \times 2^{-1} + 0 \times 2^{-2} + 0 \times 2^{-3}) \times 2^{-2} = (0.125000)_{10}$$

This still represents a gap of $0.125000 - 0.109375 = 0.015625$. However, now when higher numbers are generated by increasing the mantissa, the gap is lengthened to 0.03125,

$$0110101 = (1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3}) \times 2^{-2} = (0.156250)_{10}$$
$$0110110 = (1 \times 2^{-1} + 1 \times 2^{-2} + 0 \times 2^{-3}) \times 2^{-2} = (0.187500)_{10}$$
$$0110111 = (1 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3}) \times 2^{-2} = (0.218750)_{10}$$

This pattern is repeated as each larger quantity is formulated until a maximum number is reached,

$$0011111 = (1 \times 2^{-1} + 1 \times 2^{-2} + 1 \times 2^{-3}) \times 2^3 = (7)_{10}$$

The final number set is depicted graphically in Fig. 3.9.

Figure 3.9 manifests several aspects of floating-point representation that have significance regarding computer round-off errors:

1. *There Is a Limited Range of Quantities That May Be Represented.* Just as for the integer case, there are large positive and negative numbers that cannot be represented. Attempts to employ numbers outside the acceptable range will result in what is called

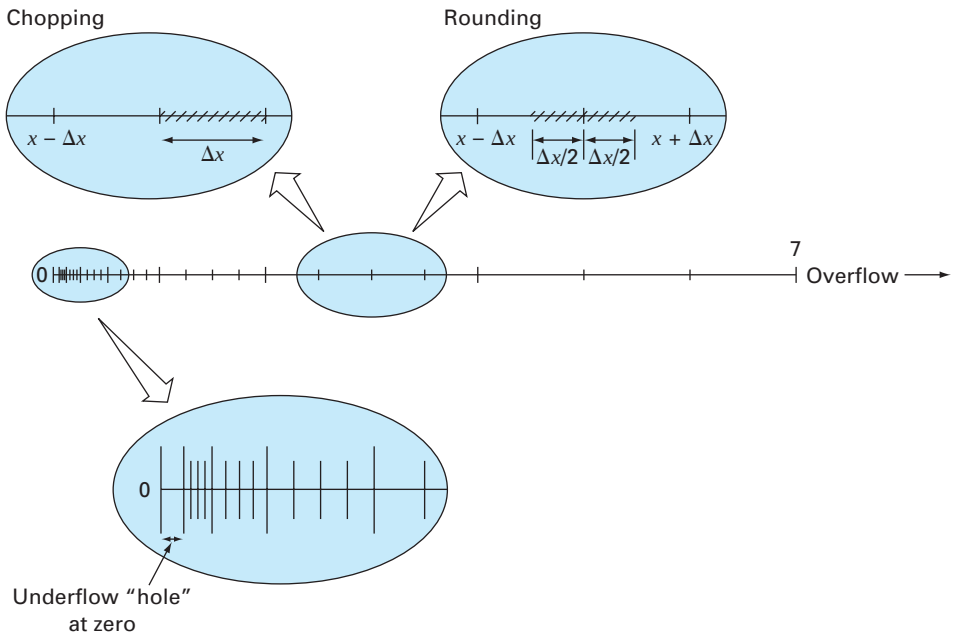


FIGURE 3.9
The hypothetical number system developed in Example 3.5. Each value is indicated by a tick mark. Only the positive numbers are shown. An identical set would also extend in the negative direction.

an *overflow error*. However, in addition to large quantities, the floating-point representation has the added limitation that very small numbers cannot be represented. This is illustrated by the *underflow* "hole" between zero and the first positive number in Fig. 3.9. It should be noted that this hole is enlarged because of the normalization constraint of Eq. (3.8).

2. *There Are Only a Finite Number of Quantities That Can Be Represented within the Range*. Thus, the degree of precision is limited. Obviously, irrational numbers cannot be represented exactly. Furthermore, rational numbers that do not exactly match one of the values in the set also cannot be represented precisely. The errors introduced by approximating both these cases are referred to as *quantizing* errors. The actual approximation is accomplished in either of two ways: chopping or rounding. For example, suppose that the value of $\pi = 3.14159265358\ldots$ is to be stored on a base-10 number system carrying seven significant figures. One method of approximation would be to merely omit, or "chop off," the eighth and higher terms, as in $\pi = 3.141592$, with the introduction of an associated error of [Eq. (3.2)]

$$E_t = 0.00000065\ldots$$

This technique of retaining only the significant terms was originally dubbed "truncation" in computer jargon. We prefer to call it *chopping* to distinguish it from the truncation errors discussed in Chap. 4. Note that for the base-2 number system in

Fig. 3.9, chopping means that any quantity falling within an interval of length Δx will be stored as the quantity at the lower end of the interval. Thus, the upper error bound for chopping is Δx. Additionally, a bias is introduced because all errors are positive. The shortcomings of chopping are attributable to the fact that the higher terms in the complete decimal representation have no impact on the shortened version. For instance, in our example of π, the first discarded digit is 6. Thus, the last retained digit should be rounded up to yield 3.141593. Such *rounding* reduces the error to

$$E_t = -0.00000035\ldots$$

Consequently, rounding yields a lower absolute error than chopping. Note that for the base-2 number system in Fig. 3.9, rounding means that any quantity falling within an interval of length Δx will be represented as the nearest allowable number. Thus, the upper error bound for rounding is $\Delta x/2$. Additionally, no bias is introduced because some errors are positive and some are negative. Some computers employ rounding. However, this adds to the computational overhead, and, consequently, many machines use simple chopping. This approach is justified under the supposition that the number of significant figures is large enough that resulting round-off error is usually negligible.

3. *The Interval between Numbers, Δx, Increases as the Numbers Grow in Magnitude.* It is this characteristic, of course, that allows floating-point representation to preserve significant digits. However, it also means that quantizing errors will be proportional to the magnitude of the number being represented. For normalized floating-point numbers, this proportionality can be expressed, for cases where chopping is employed, as

$$\frac{|\Delta x|}{|x|} \leq \mathscr{E} \tag{3.9}$$

and, for cases where rounding is employed, as

$$\frac{|\Delta x|}{|x|} \leq \frac{\mathscr{E}}{2} \tag{3.10}$$

where $\mathscr{E}$ is referred to as the *machine epsilon*, which can be computed as

$$\mathscr{E} = b^{1-t} \tag{3.11}$$

where b is the number base and t is the number of significant digits in the mantissa. Notice that the inequalities in Eqs. (3.9) and (3.10) signify that these are error bounds. That is, they specify the worst cases.

EXAMPLE 3.6 Machine Epsilon

Problem Statement. Determine the machine epsilon and verify its effectiveness in characterizing the errors of the number system from Example 3.5. Assume that chopping is used.

Solution. The hypothetical floating-point system from Example 3.5 employed values of the base $b = 2$, and the number of mantissa bits $t = 3$. Therefore, the machine epsilon would be [Eq. (3.11)]

$$\mathscr{E} = 2^{1-3} = 0.25$$

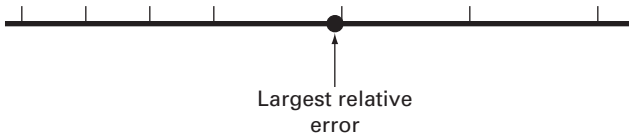


FIGURE 3.10
The largest quantizing error will occur for those values falling just below the upper bound of the first of a series of equispaced intervals.

Consequently, the relative quantizing error should be bounded by 0.25 for chopping. The largest relative errors should occur for those quantities that fall just below the upper bound of the first interval between successive equispaced numbers (Fig. 3.10). Those numbers falling in the succeeding higher intervals would have the same value of Δx but a greater value of x and, hence, would have a lower relative error. An example of a maximum error would be a value falling just below the upper bound of the interval between $(0.125000)_{10}$ and $(0.156250)_{10}$. For this case, the error would be less than

$$\frac{0.03125}{0.125000} = 0.25$$

Thus, the error is as predicted by Eq. (3.9).

The magnitude dependence of quantizing errors has a number of practical applications in numerical methods. Most of these relate to the commonly employed operation of testing whether two numbers are equal. This occurs when testing convergence of quantities as well as in the stopping mechanism for iterative processes (recall Example 3.2). For these cases, it should be clear that, rather than test whether the two quantities are equal, it is advisable to test whether their difference is less than an acceptably small tolerance. Further, it should also be evident that normalized rather than absolute difference should be compared, particularly when dealing with numbers of large magnitude. In addition, the machine epsilon can be employed in formulating stopping or convergence criteria. This ensures that programs are portable—that is, they are not dependent on the computer on which they are implemented. Figure 3.11 lists pseudocode to automatically determine the machine epsilon of a binary computer.

```
epsilon = 1
DO
  IF (epsilon+1 ≤ 1)EXIT
  epsilon = epsilon/2
END DO
epsilon = 2 × epsilon
```

FIGURE 3.11
Pseudocode to determine machine epsilon for a binary computer.

Extended Precision. It should be noted at this point that, although round-off errors can be important in contexts such as testing convergence, the number of significant digits carried on most computers allows most engineering computations to be performed with more than acceptable precision. For example, the hypothetical number system in Fig. 3.9 is a gross exaggeration that was employed for illustrative purposes. Commercial computers use much larger words and, consequently, allow numbers to be expressed with more than adequate precision. For example, computers that use IEEE format allow 24 bits to be used for the mantissa, which translates into about seven significant base-10 digits of precision[1] with a range of about 10^{-38} to 10^{39}.

[1]Note that only 23 bits are actually used to store the mantissa. However, because of normalization, the first bit of the mantissa is always 1 and is, therefore, not stored. Thus, this first bit together with the 23 stored bits gives the 24 total bits of precision for the mantissa.

With this acknowledged, there are still cases where round-off error becomes critical. For this reason most computers allow the specification of extended precision. The most common of these is double precision, in which the number of words used to store floating-point numbers is doubled. It provides about 15 to 16 decimal digits of precision and a range of approximately 10^{-308} to 10^{308}.

In many cases, the use of double-precision quantities can greatly mitigate the effect of round-off errors. However, a price is paid for such remedies in that they also require more memory and execution time. The difference in execution time for a small calculation might seem insignificant. However, as your programs become larger and more complicated, the added execution time could become considerable and have a negative impact on your effectiveness as a problem solver. Therefore, extended precision should not be used frivolously. Rather, it should be selectively employed where it will yield the maximum benefit at the least cost in terms of execution time. In the following sections, we will look closer at how round-off errors affect computations, and in so doing provide a foundation of understanding to guide your use of the double-precision capability.

Before proceeding, it should be noted that some of the commonly used software packages (for example, Excel, Mathcad) routinely use double precision to represent numerical quantities. Thus, the developers of these packages decided that mitigating round-off errors would take precedence over any loss of speed incurred by using extended precision. Others, like MATLAB software, allow you to use extended precision, if you desire.

3.4.2 Arithmetic Manipulations of Computer Numbers

Aside from the limitations of a computer's number system, the actual arithmetic manipulations involving these numbers can also result in round-off error. In the following section, we will first illustrate how common arithmetic operations affect round-off errors. Then we will investigate a number of particular manipulations that are especially prone to round-off errors.

Common Arithmetic Operations. Because of their familiarity, normalized base-10 numbers will be employed to illustrate the effect of round-off errors on simple addition, subtraction, multiplication, and division. Other number bases would behave in a similar fashion. To simplify the discussion, we will employ a hypothetical decimal computer with a 4-digit mantissa and a 1-digit exponent. In addition, chopping is used. Rounding would lead to similar though less dramatic errors.

When two floating-point numbers are added, the mantissa of the number with the smaller exponent is modified so that the exponents are the same. This has the effect of aligning the decimal points. For example, suppose we want to add $0.1557 \cdot 10^1 + 0.4381 \cdot 10^{-1}$. The decimal of the mantissa of the second number is shifted to the left a number of places equal to the difference of the exponents $[1 - (-1) = 2]$, as in

$$0.4381 \cdot 10^{-1} \rightarrow 0.004381 \cdot 10^1$$

Now the numbers can be added,

$$\begin{array}{ll} 0.1557 & \cdot 10^1 \\ 0.004381 & \cdot 10^1 \\ \hline 0.160081 & \cdot 10^1 \end{array}$$

and the result chopped to $0.1600 \cdot 10^1$. Notice how the last two digits of the second number that were shifted to the right have essentially been lost from the computation.

Subtraction is performed identically to addition except that the sign of the subtrahend is reversed. For example, suppose that we are subtracting 26.86 from 36.41. That is,

$$\begin{array}{r} 0.3641 \cdot 10^2 \\ -\ 0.2686 \cdot 10^2 \\ \hline 0.0955 \cdot 10^2 \end{array}$$

For this case the result is not normalized, and so we must shift the decimal one place to the right to give $0.9550 \cdot 10^1 = 9.550$. Notice that the zero added to the end of the mantissa is not significant but is merely appended to fill the empty space created by the shift. Even more dramatic results would be obtained when the numbers are very close, as in

$$\begin{array}{r} 0.7642 \cdot 10^3 \\ -\ 0.7641 \cdot 10^3 \\ \hline 0.0001 \cdot 10^3 \end{array}$$

which would be converted to $0.1000 \cdot 10^0 = 0.1000$. Thus, for this case, three nonsignificant zeros are appended. This introduces a substantial computational error because subsequent manipulations would act as if these zeros were significant. As we will see in a later section, the loss of significance during the subtraction of nearly equal numbers is among the greatest source of round-off error in numerical methods.

Multiplication and division are somewhat more straightforward than addition or subtraction. The exponents are added and the mantissas multiplied. Because multiplication of two n-digit mantissas will yield a $2n$-digit result, most computers hold intermediate results in a double-length register. For example,

$$0.1363 \cdot 10^3 \times 0.6423 \cdot 10^{-1} = 0.08754549 \cdot 10^2$$

If, as in this case, a leading zero is introduced, the result is normalized,

$$0.08754549 \cdot 10^2 \rightarrow 0.8754549 \cdot 10^1$$

and chopped to give

$$0.8754 \cdot 10^1$$

Division is performed in a similar manner, but the mantissas are divided and the exponents are subtracted. Then the results are normalized and chopped.

Large Computations. Certain methods require extremely large numbers of arithmetic manipulations to arrive at their final results. In addition, these computations are often interdependent. That is, the later calculations are dependent on the results of earlier ones. Consequently, even though an individual round-off error could be small, the cumulative effect over the course of a large computation can be significant.

EXAMPLE 3.7 Large Numbers of Interdependent Computations

Problem Statement. Investigate the effect of round-off error on large numbers of interdependent computations. Develop a program to sum a number 100,000 times. Sum the number 1 in single precision, and 0.00001 in single and double precision.

Solution. Figure 3.12 shows a Fortran 90 program that performs the summation. Whereas the single-precision summation of 1 yields the expected result, the single-precision

FIGURE 3.12
Fortran 90 program to sum a number 10^5 times. The case sums the number 1 in single precision and the number 10^{-5} in single and double precision.

```
PROGRAM fig0312
IMPLICIT none
INTEGER::i
REAL::sum1, sum2, x1, x2
DOUBLE PRECISION::sum3, x3
sum1=0.
sum2=0.
sum3=0.
x1=1.
x2=1.e-5
x3=1.d-5
DO i=1,100000
  sum1=sum1+x1
  sum2=sum2+x2
  sum3=sum3+x3
END DO
PRINT *, sum1
PRINT *, sum2
PRINT *, sum3
END
output:
100000.000000
       1.000990
  9.999999999980838E-001
```

summation of 0.00001 yields a large discrepancy. This error is reduced significantly when 0.00001 is summed in double precision.

Quantizing errors are the source of the discrepancies. Because the integer 1 can be represented exactly within the computer, it can be summed exactly. In contrast, 0.00001 cannot be represented exactly and is quantized by a value that is slightly different from its true value. Whereas this very slight discrepancy would be negligible for a small computation, it accumulates after repeated summations. The problem still occurs in double precision but is greatly mitigated because the quantizing error is much smaller.

Note that the type of error illustrated by the previous example is somewhat atypical in that all the errors in the repeated operation are of the same sign. In most cases the errors of a long computation alternate sign in a random fashion and, thus, often cancel out. However, there are also instances where such errors do not cancel but, in fact, lead to a spurious final result. The following sections are intended to provide insight into ways in which this may occur.

Adding a Large and a Small Number. Suppose we add a small number, 0.0010, to a large number, 4000, using a hypothetical computer with the 4-digit mantissa and the 1-digit exponent. We modify the smaller number so that its exponent matches the larger,

$$\begin{array}{l} 0.4000 \quad \cdot 10^4 \\ \underline{0.0000001 \cdot 10^4} \\ 0.4000001 \cdot 10^4 \end{array}$$

which is chopped to $0.4000 \cdot 10^4$. Thus, we might as well have not performed the addition!

This type of error can occur in the computation of an infinite series. The initial terms in such series are often relatively large in comparison with the later terms. Thus, after a few terms have been added, we are in the situation of adding a small quantity to a large quantity.

One way to mitigate this type of error is to sum the series in reverse order—that is, in ascending rather than descending order. In this way, each new term will be of comparable magnitude to the accumulated sum (see Prob. 3.5).

Subtractive Cancellation. This term refers to the round-off induced when subtracting two nearly equal floating-point numbers.

One common instance where this can occur involves finding the roots of a quadratic equation or parabola with the quadratic formula,

$$\begin{matrix} x_1 \\ x_2 \end{matrix} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{3.12}$$

For cases where $b^2 \gg 4ac$, the difference in the numerator can be very small. In such cases, double precision can mitigate the problem. In addition, an alternative formulation can be used to minimize subtractive cancellation,

$$\begin{matrix} x_1 \\ x_2 \end{matrix} = \frac{-2c}{b \pm \sqrt{b^2 - 4ac}} \tag{3.13}$$

An illustration of the problem and the use of this alternative formula are provided in the following example.

EXAMPLE 3.8 Subtractive Cancellation

Problem Statement. Compute the values of the roots of a quadratic equation with $a = 1$, $b = 3000.001$, and $c = 3$. Check the computed values versus the true roots of $x_1 = -0.001$ and $x_2 = -3000$.

Solution. Figure 3.13 shows an Excel/VBA program that computes the roots x_1 and x_2 on the basis of the quadratic formula [(Eq. (3.12)]. Note that both single- and double-precision versions are given. Whereas the results for x_2 are adequate, the percent relative errors for x_1 are poor for the single-precision version, $\varepsilon_t = 2.4\%$. This level could be inadequate for many applied engineering problems. This result is particularly surprising because we are employing an analytical formula to obtain our solution!

The loss of significance occurs in the line of both programs where two relatively large numbers are subtracted. Similar problems do not occur when the same numbers are added.

On the basis of the above, we can draw the general conclusion that the quadratic formula will be susceptible to subtractive cancellation whenever $b^2 \gg 4ac$. One way to circumvent this problem is to use double precision. Another is to recast the quadratic formula in the format of Eq. (3.13). As in the program output, both options give a much smaller error because the subtractive cancellation is minimized or avoided.

```
Option Explicit

Sub fig0313()
Dim a As Single, b As Single
Dim c As Single, d As Single
Dim x1 As Single, x2 As Single
Dim x1r As Single
Dim aa As Double, bb As Double
Dim cc As Double, dd As Double
Dim x11 As Double, x22 As Double

'Single precision:
a = 1: b = 3000.001: c = 3
d = Sqr(b * b - 4 * a * c)
x1 = (-b + d) / (2 * a)
x2 = (-b - d) / (2 * a)

'Double precision:
aa = 1: bb = 3000.001: cc = 3
dd = Sqr(bb * bb - 4 * aa * cc)
x11 = (-bb + dd) / (2 * aa)
x22 = (-bb - dd) / (2 * aa)

'Modified formula for first root
'single precision:
x1r = -2 * c / (b + d)

'Display results
Sheets("sheet1").Select
Range("b2").Select
ActiveCell.Value = x1
ActiveCell.Offset(1, 0).Select
ActiveCell.Value = x2
ActiveCell.Offset(2, 0).Select
ActiveCell.Value = x11
ActiveCell.Offset(1, 0).Select
ActiveCell.Value = x22
ActiveCell.Offset(2, 0).Select
ActiveCell.Value = x1r
End Sub
```

OUTPUT:

	A	B	C
1	Single-precision results:		
2	x1	-0.000976563	
3	x2	-3000.00000000	
4	Double-precision results:		
5	x1	-0.00100000	
6	x2	-3000.00000000	
7	Modified formula for first root (single precision):		
8	x1	-0.00100000	

FIGURE 3.13
Excel/VBA program to determine the roots of a quadratic.

Note that, as in the foregoing example, there are times where subtractive cancellation can be circumvented by using a transformation. However, the only general remedy is to employ extended precision.

Smearing. Smearing occurs whenever the individual terms in a summation are larger than the summation itself. As in the following example, one case where this occurs is in series of mixed signs.

EXAMPLE 3.9 Evaluation of e^x using Infinite Series

Problem Statement. The exponential function $y = e^x$ is given by the infinite series

$$y = 1 + x + \frac{x^2}{2} + \frac{x^3}{3!} + \cdots$$

Evaluate this function for $x = 10$ and $x = -10$, and be attentive to the problems of round-off error.

Solution. Figure 3.14*a* gives an Excel/VBA program that uses the infinite series to evaluate e^x. The variable i is the number of terms in the series, *term* is the value of the

the sum are much larger than the final result of the sum. Furthermore, unlike the previous case, the individual terms vary in sign. Thus, in effect we are adding and subtracting large numbers (each with some small error) and placing great significance on the differences—that is, subtractive cancellation. Thus, we can see that the culprit behind this example of smearing is, in fact, subtractive cancellation. For such cases it is appropriate to seek some other computational strategy. For example, one might try to compute $y = e^{10}$ as $y = (e^{-1})^{10}$. Other than such a reformulation, the only general recourse is extended precision.

Inner Products. As should be clear from the last sections, some infinite series are particularly prone to round-off error. Fortunately, the calculation of series is not one of the more common operations in numerical methods. A far more ubiquitous manipulation is the calculation of inner products, as in

$$\sum_{i=1}^{n} x_i y_i = x_1 y_1 + x_2 y_2 + \cdots + x_n y_n$$

This operation is very common, particularly in the solution of simultaneous linear algebraic equations. Such summations are prone to round-off error. Consequently, it is often desirable to compute such summations in extended precision.

Although the foregoing sections should provide rules of thumb to mitigate round-off error, they do not provide a direct means beyond trial and error to actually determine the effect of such errors on a computation. In Chap. 4, we will introduce the Taylor series, which will provide a mathematical approach for estimating these effects.

PROBLEMS

3.1 Convert the following base-2 numbers to base-10: **(a)** 1011001, **(b)** 110.00101 and **(c)** 0.01011

3.2 Convert the following base-8 numbers to base 10: 71,563 and 3.14.

3.3 Compose your own program based on Fig. 3.11 and use it to determine your computer's machine epsilon.

3.4 In a fashion similar to that in Fig. 3.11, write a short program to determine the smallest number, x_{min}, used on the computer you will be employing along with this book. Note that your computer will be unable to reliably distinguish between zero and a quantity that is smaller than this number.

3.5 The infinite series

$$f(n) = \sum_{i=1}^{n} \frac{1}{i^4}$$

converges on a value of $f(n) = \pi^4/90$ as n approaches infinity. Write a program in single precision to calculate $f(n)$ for $n = 10{,}000$ by computing the sum from $i = 1$ to 10,000. Then repeat the calculation but in reverse order—that is, from $i = 10{,}000$ to 1 using increments of -1. In each case, compute the true percent relative error. Explain the results.

3.6 Evaluate e^{-5} using two approaches

$$e^{-x} = 1 - x + \frac{x^2}{2} - \frac{x^3}{3!} + \cdots$$

and

$$e^{-x} = \frac{1}{e^x} = \frac{1}{1 + x + \frac{x^2}{2} + \frac{x^3}{3!} + \cdots}$$

and compare with the true value of 6.737947×10^{-3}. Use 20 terms to evaluate each series and compute true and approximate relative errors as terms are added.

3.7 The derivative of $f(x) = 1/(1 - 3x^2)$ is given by

$$\frac{6x}{(1 - 3x^2)^2}$$

Do you expect to have difficulties evaluating this function at $x = 0.577$? Try it using 3- and 4-digit arithmetic with chopping.

3.8 **(a)** Evaluate the polynomial

$$y = x^3 - 7x^2 + 8x - 0.35$$

at $x = 1.37$. Use 3-digit arithmetic with chopping. Evaluate the percent relative error.

(b) Repeat **(a)** but express y as

$$y = ((x - 7)x + 8)x - 0.35$$

Evaluate the error and compare with part **(a)**.

3.9 Calculate the random access memory (RAM) in megabytes necessary to store a multidimensional array that is $20 \times 40 \times 120$. This array is double precision, and each value requires a 64-bit word. Recall that a 64-bit word $= 8$ bytes and 1 kilobyte $= 2^{10}$ bytes. Assume that the index starts at 1.

3.10 Determine the number of terms necessary to approximate $\cos x$ to 8 significant figures using the Maclaurin series approximation

$$\cos x = 1 - \frac{x^2}{2} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \cdots$$

Calculate the approximation using a value of $x = 0.3\pi$. Write a program to determine your result.

3.11 Use 5-digit arithmetic with chopping to determine the roots of the following equation with Eqs. (3.12) and (3.13)

$$x^2 - 5000.002x + 10$$

Compute percent relative errors for your results.

3.12 How can the machine epsilon be employed to formulate a stopping criterion ε_s for your programs? Provide an example.

3.13 The "divide and average" method, an old-time method for approximating the square root of any positive number a, can be formulated as

$$x = \frac{x + a/x}{2}$$

Write a well-structured function to implement this algorithm based on the algorithm outlined in Fig. 3.3.

CHAPTER 4

Truncation Errors and the Taylor Series

Truncation errors are those that result from using an approximation in place of an exact mathematical procedure. For example, in Chap. 1 we approximated the derivative of velocity of a falling parachutist by a finite-divided-difference equation of the form [Eq. (1.11)]

$$\frac{dv}{dt} \cong \frac{\Delta v}{\Delta t} = \frac{v(t_{i+1}) - v(t_i)}{t_{i+1} - t_i} \tag{4.1}$$

A truncation error was introduced into the numerical solution because the difference equation only approximates the true value of the derivative (recall Fig. 1.4). In order to gain insight into the properties of such errors, we now turn to a mathematical formulation that is used widely in numerical methods to express functions in an approximate fashion—the Taylor series.

4.1 THE TAYLOR SERIES

Taylor's theorem (Box 4.1) and its associated formula, the Taylor series, is of great value in the study of numerical methods. In essence, the *Taylor series* provides a means to predict a function value at one point in terms of the function value and its derivatives at another point. In particular, the theorem states that any smooth function can be approximated as a polynomial.

A useful way to gain insight into the Taylor series is to build it term by term. For example, the first term in the series is

$$f(x_{i+1}) \cong f(x_i) \tag{4.2}$$

This relationship, called the *zero-order approximation*, indicates that the value of f at the new point is the same as its value at the old point. This result makes intuitive sense because if x_i and x_{i+1} are close to each other, it is likely that the new value is probably similar to the old value.

Equation (4.2) provides a perfect estimate if the function being approximated is, in fact, a constant. However, if the function changes at all over the interval, additional terms

Box 4.1 Taylor's Theorem

Taylor's Theorem

If the function f and its first $n+1$ derivatives are continuous on an interval containing a and x, then the value of the function at x is given by

$$f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \frac{f^{(3)}(a)}{3!}(x-a)^3 + \cdots + \frac{f^{(n)}(a)}{n!}(x-a)^n + R_n \tag{B4.1.1}$$

where the remainder R_n is defined as

$$R_n = \int_a^x \frac{(x-t)^n}{n!} f^{(n+1)}(t)\,dt \tag{B4.1.2}$$

where t = a dummy variable. Equation (B4.1.1) is called the *Taylor series* or *Taylor's formula*. If the remainder is omitted, the right side of Eq. (B4.1.1) is the Taylor polynomial approximation to $f(x)$. In essence, the theorem states that any smooth function can be approximated as a polynomial.

Equation (B4.1.2) is but one way, called the *integral form*, by which the remainder can be expressed. An alternative formulation can be derived on the basis of the integral mean-value theorem.

First Theorem of Mean for Integrals

If the function g is continuous and integrable on an interval containing a and x, then there exists a point ξ between a and x such that

$$\int_a^x g(t)\,dt = g(\xi)(x-a) \tag{B4.1.3}$$

In other words, this theorem states that the integral can be represented by an average value for the function $g(\xi)$ times the interval length $x-a$. Because the average must occur between the minimum and maximum values for the interval, there is a point $x=\xi$ at which the function takes on the average value.

The first theorem is in fact a special case of a second mean-value theorem for integrals.

Second Theorem of Mean for Integrals

If the functions g and h are continuous and integrable on an interval containing a and x, and h does not change sign in the interval, then there exists a point ξ between a and x such that

$$\int_a^x g(t)h(t)\,dt = g(\xi)\int_a^x h(t)\,dt \tag{B4.1.4}$$

Thus, Eq. (B4.1.3) is equivalent to Eq. (B4.1.4) with $h(t)=1$.

The second theorem can be applied to Eq. (B4.1.2) with

$$g(t) = f^{(n+1)}(t) \qquad h(t) = \frac{(x-t)^n}{n!}$$

As t varies from a to x, $h(t)$ is continuous and does not change sign. Therefore, if $f^{(n+1)}(t)$ is continuous, then the integral mean-value theorem holds and

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

This equation is referred to as the *derivative* or *Lagrange form* of the remainder.

of the Taylor series are required to provide a better estimate. For example, the *first-order approximation* is developed by adding another term to yield

$$f(x_{i+1}) \cong f(x_i) + f'(x_i)(x_{i+1} - x_i) \tag{4.3}$$

The additional first-order term consists of a slope $f'(x_i)$ multiplied by the distance between x_i and x_{i+1}. Thus, the expression is now in the form of a straight line and is capable of predicting an increase or decrease of the function between x_i and x_{i+1}.

Although Eq. (4.3) can predict a change, it is exact only for a straight-line, or *linear*, trend. Therefore, a *second-order* term is added to the series to capture some of the curvature that the function might exhibit:

$$f(x_{i+1}) \cong f(x_i) + f'(x_i)(x_{i+1} - x_i) + \frac{f''(x_i)}{2!}(x_{i+1} - x_i)^2 \tag{4.4}$$

In a similar manner, additional terms can be included to develop the complete Taylor series expansion:

$$f(x_{i+1}) = f(x_i) + f'(x_i)(x_{i+1} - x_i) + \frac{f''(x_i)}{2!}(x_{i+1} - x_i)^2 + \frac{f^{(3)}(x_i)}{3!}(x_{i+1} - x_i)^3 + \cdots + \frac{f^{(n)}(x_i)}{n!}(x_{i+1} - x_i)^n + R_n \tag{4.5}$$

Note that because Eq. (4.5) is an infinite series, an equal sign replaces the approximate sign that was used in Eqs. (4.2) through (4.4). A remainder term is included to account for all terms from $n + 1$ to infinity:

$$R_n = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x_{i+1} - x_i)^{n+1} \tag{4.6}$$

where the subscript n connotes that this is the remainder for the nth-order approximation and ξ is a value of x that lies somewhere between x_i and x_{i+1}. The introduction of the ξ is so important that we will devote an entire section (Sec. 4.1.1) to its derivation. For the time being, it is sufficient to recognize that there is such a value that provides an exact determination of the error.

It is often convenient to simplify the Taylor series by defining a step size $h = x_{i+1} - x_i$ and expressing Eq. (4.5) as

$$f(x_{i+1}) = f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2!}h^2 + \frac{f^{(3)}(x_i)}{3!}h^3 + \cdots + \frac{f^{(n)}(x_i)}{n!}h^n + R_n \tag{4.7}$$

where the remainder term is now

$$R_n = \frac{f^{(n+1)}(\xi)}{(n+1)!}h^{n+1} \tag{4.8}$$

EXAMPLE 4.1 Taylor Series Approximation of a Polynomial

Problem Statement. Use zero- through fourth-order Taylor series expansions to approximate the function

$$f(x) = -0.1x^4 - 0.15x^3 - 0.5x^2 - 0.25x + 1.2$$

from $x_i = 0$ with $h = 1$. That is, predict the function's value at $x_{i+1} = 1$.

Solution. Because we are dealing with a known function, we can compute values for $f(x)$ between 0 and 1. The results (Fig. 4.1) indicate that the function starts at $f(0) = 1.2$ and then curves downward to $f(1) = 0.2$. Thus, the true value that we are trying to predict is 0.2.

The Taylor series approximation with $n = 0$ is [Eq. (4.2)]

$$f(x_{i+1}) \simeq 1.2$$

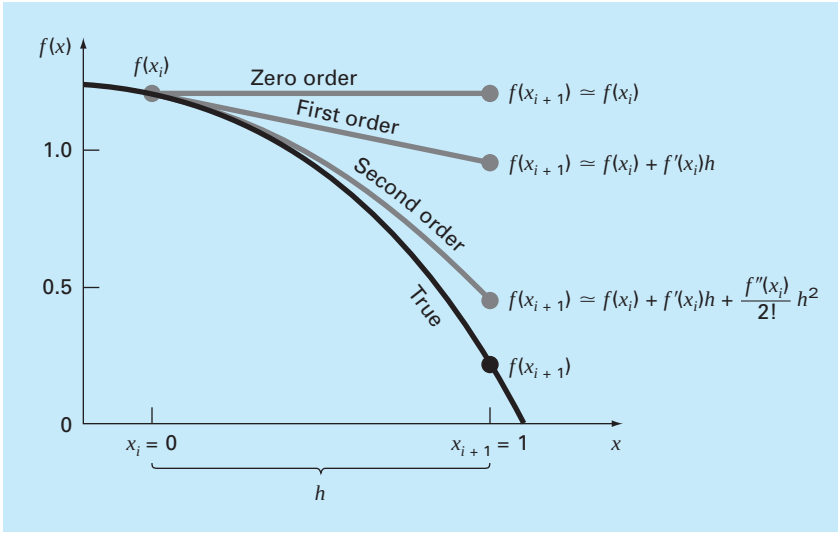


FIGURE 4.1
The approximation of $f(x) = -0.1x^4 - 0.15x^3 - 0.5x^2 - 0.25x + 1.2$ at $x = 1$ by zero-order, first-order, and second-order Taylor series expansions.

Thus, as in Fig. 4.1, the zero-order approximation is a constant. Using this formulation results in a truncation error [recall Eq. (3.2)] of

$$E_t = 0.2 - 1.2 = -1.0$$

at $x = 1$.

For $n = 1$, the first derivative must be determined and evaluated at $x = 0$:

$$f'(0) = -0.4(0.0)^3 - 0.45(0.0)^2 - 1.0(0.0) - 0.25 = -0.25$$

Therefore, the first-order approximation is [Eq. (4.3)]

$$f(x_{i+1}) \simeq 1.2 - 0.25h$$

which can be used to compute $f(1) = 0.95$. Consequently, the approximation begins to capture the downward trajectory of the function in the form of a sloping straight line (Fig. 4.1). This results in a reduction of the truncation error to

$$E_t = 0.2 - 0.95 = -0.75$$

For $n = 2$, the second derivative is evaluated at $x = 0$:

$$f''(0) = -1.2(0.0)^2 - 0.9(0.0) - 1.0 = -1.0$$

Therefore, according to Eq. (4.4),

$$f(x_{i+1}) \simeq 1.2 - 0.25h - 0.5h^2$$

and substituting $h = 1$, $f(1) = 0.45$. The inclusion of the second derivative now adds some downward curvature resulting in an improved estimate, as seen in Fig. 4.1. The truncation error is reduced further to $0.2 - 0.45 = -0.25$.

Additional terms would improve the approximation even more. In fact, the inclusion of the third and the fourth derivatives results in exactly the same equation we started with:

$$f(x) = 1.2 - 0.25h - 0.5h^2 - 0.15h^3 - 0.1h^4$$

where the remainder term is

$$R_4 = \frac{f^{(5)}(\xi)}{5!}h^5 = 0$$

because the fifth derivative of a fourth-order polynomial is zero. Consequently, the Taylor series expansion to the fourth derivative yields an exact estimate at $x_{i+1} = 1$:

$$f(1) = 1.2 - 0.25(1) - 0.5(1)^2 - 0.15(1)^3 - 0.1(1)^4 = 0.2$$

In general, the *n*th-order Taylor series expansion will be exact for an *n*th-order polynomial. For other differentiable and continuous functions, such as exponentials and sinusoids, a finite number of terms will not yield an exact estimate. Each additional term will contribute some improvement, however slight, to the approximation. This behavior will be demonstrated in Example 4.2. Only if an infinite number of terms are added will the series yield an exact result.

Although the above is true, the practical value of Taylor series expansions is that, in most cases, the inclusion of only a few terms will result in an approximation that is close enough to the true value for practical purposes. The assessment of how many terms are required to get "close enough" is based on the remainder term of the expansion. Recall that the remainder term is of the general form of Eq. (4.8). This relationship has two major drawbacks. First, ξ is not known exactly but merely lies somewhere between x_i and x_{i+1}. Second, to evaluate Eq. (4.8), we need to determine the $(n + 1)$th derivative of $f(x)$. To do this, we need to know $f(x)$. However, if we knew $f(x)$, there would be no need to perform the Taylor series expansion in the present context!

Despite this dilemma, Eq. (4.8) is still useful for gaining insight into truncation errors. This is because we *do* have control over the term h in the equation. In other words, we can choose how far away from x we want to evaluate $f(x)$, and we can control the number of terms we include in the expansion. Consequently, Eq. (4.8) is usually expressed as

$$R_n = O(h^{n+1})$$

where the nomenclature $O(h^{n+1})$ means that the truncation error is of the order of h^{n+1}. That is, the error is proportional to the step size h raised to the $(n + 1)$th power. Although this approximation implies nothing regarding the magnitude of the derivatives that multiply h^{n+1}, it is extremely useful in judging the comparative error of numerical methods based on Taylor series expansions. For example, if the error is $O(h)$, halving the step size will halve the error. On the other hand, if the error is $O(h^2)$, halving the step size will quarter the error.

In general, we can usually assume that the truncation error is decreased by the addition of terms to the Taylor series. In many cases, if h is sufficiently small, the first- and other lower-order terms usually account for a disproportionately high percent of the error. Thus, only a few terms are required to obtain an adequate estimate. This property is illustrated by the following example.

EXAMPLE 4.2 Use of Taylor Series Expansion to Approximate a Function with an Infinite Number of Derivatives

Problem Statement. Use Taylor series expansions with $n = 0$ to 6 to approximate $f(x) = \cos x$ at $x_{i+1} = \pi/3$ on the basis of the value of $f(x)$ and its derivatives at $x_i = \pi/4$. Note that this means that $h = \pi/3 - \pi/4 = \pi/12$.

Solution. As with Example 4.1, our knowledge of the true function means that we can determine the correct value $f(\pi/3) = 0.5$.

The zero-order approximation is [Eq. (4.3)]

$$f\left(\frac{\pi}{3}\right) \cong \cos\left(\frac{\pi}{4}\right) = 0.707106781$$

which represents a percent relative error of

$$\varepsilon_t = \frac{0.5 - 0.707106781}{0.5} 100\% = -41.4\%$$

For the first-order approximation, we add the first derivative term where $f'(x) = -\sin x$:

$$f\left(\frac{\pi}{3}\right) \cong \cos\left(\frac{\pi}{4}\right) - \sin\left(\frac{\pi}{4}\right)\left(\frac{\pi}{12}\right) = 0.521986659$$

which has $\varepsilon_t = -4.40$ percent.

For the second-order approximation, we add the second derivative term where $f''(x) = -\cos x$:

$$f\left(\frac{\pi}{3}\right) \cong \cos\left(\frac{\pi}{4}\right) - \sin\left(\frac{\pi}{4}\right)\left(\frac{\pi}{12}\right) - \frac{\cos(\pi/4)}{2}\left(\frac{\pi}{12}\right)^2 = 0.497754491$$

with $\varepsilon_t = 0.449$ percent. Thus, the inclusion of additional terms results in an improved estimate.

The process can be continued and the results listed, as in Table 4.1. Notice that the derivatives never go to zero as was the case with the polynomial in Example 4.1. Therefore, each additional term results in some improvement in the estimate. However, also notice how most of the improvement comes with the initial terms. For this case, by the time we

TABLE 4.1 Taylor series approximation of $f(x) = \cos x$ at $x_{i+1} = \pi/3$ using a base point of $\pi/4$. Values are shown for various orders (n) of approximation.

Order n	$f^{(n)}(x)$	$f(\pi/3)$	ε_t
0	$\cos x$	0.707106781	−41.4
1	$-\sin x$	0.521986659	−4.4
2	$-\cos x$	0.497754491	0.449
3	$\sin x$	0.499869147	2.62×10^{-2}
4	$\cos x$	0.500007551	-1.51×10^{-3}
5	$-\sin x$	0.500000304	-6.08×10^{-5}
6	$-\cos x$	0.499999988	2.44×10^{-6}

have added the third-order term, the error is reduced to 2.62×10^{-2} percent, which means that we have attained 99.9738 percent of the true value. Consequently, although the addition of more terms will reduce the error further, the improvement becomes negligible.

4.1.1 The Remainder for the Taylor Series Expansion

Before demonstrating how the Taylor series is actually used to estimate numerical errors, we must explain why we included the argument ξ in Eq. (4.8). A mathematical derivation is presented in Box 4.1. We will now develop an alternative exposition based on a somewhat more visual interpretation. Then we can extend this specific case to the more general formulation.

Suppose that we truncated the Taylor series expansion [Eq. (4.7)] after the zero-order term to yield

$$f(x_{i+1}) \cong f(x_i)$$

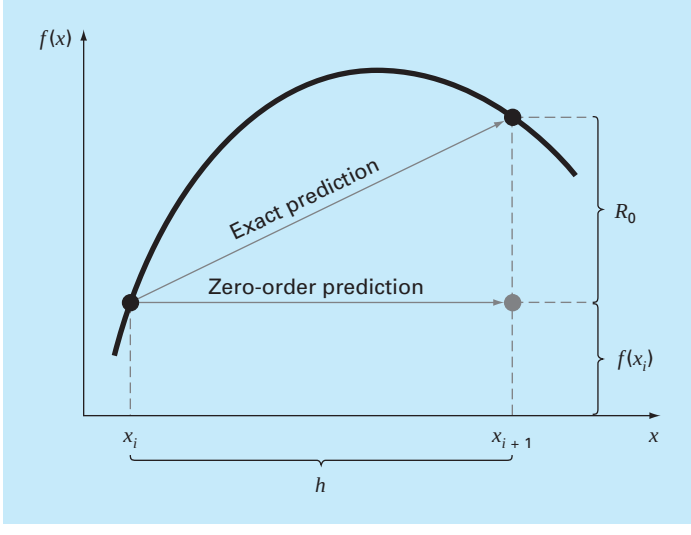
A visual depiction of this zero-order prediction is shown in Fig. 4.2. The remainder, or error, of this prediction, which is also shown in the illustration, consists of the infinite series of terms that were truncated:

$$R_0 = f'(x_i)h + \frac{f''(x_i)}{2!}h^2 + \frac{f^{(3)}(x_i)}{3!}h^3 + \cdots$$

It is obviously inconvenient to deal with the remainder in this infinite series format. One simplification might be to truncate the remainder itself, as in

$$R_0 \cong f'(x_i)h \tag{4.9}$$

FIGURE 4.2
Graphical depiction of a zero-order Taylor series prediction and remainder.



Although, as stated in the previous section, lower-order derivatives usually account for a greater share of the remainder than the higher-order terms, this result is still inexact because of the neglected second- and higher-order terms. This "inexactness" is implied by the approximate equality symbol ($\cong$) employed in Eq. (4.9).

An alternative simplification that transforms the approximation into an equivalence is based on a graphical insight. As in Fig. 4.3, the *derivative mean-value theorem* states that if a function $f(x)$ and its first derivative are continuous over an interval from x_i to x_{i+1}, then there exists at least one point on the function that has a slope, designated by $f'(\xi)$, that is parallel to the line joining $f(x_i)$ and $f(x_{i+1})$. The parameter ξ marks the x value where this slope occurs (Fig. 4.3). A physical illustration of this theorem is that, if you travel between two points with an average velocity, there will be at least one moment during the course of the trip when you will be moving at that average velocity.

By invoking this theorem it is simple to realize that, as illustrated in Fig. 4.3, the slope $f'(\xi)$ is equal to the rise R_0 divided by the run h, or

$$f'(\xi) = \frac{R_0}{h}$$

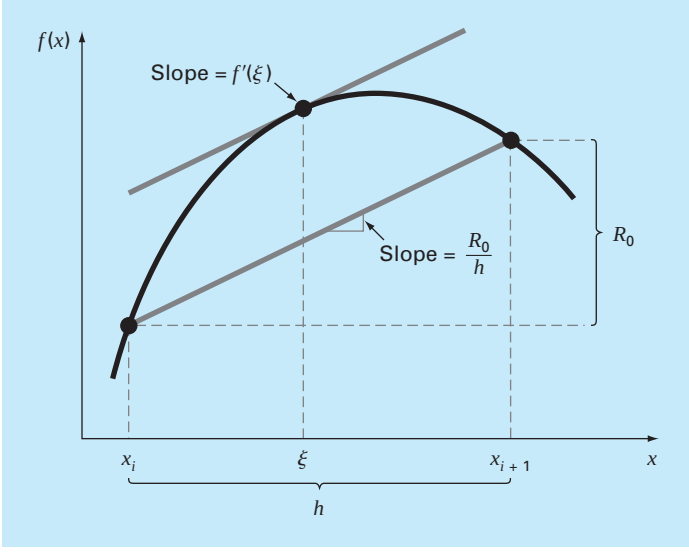
which can be rearranged to give

$$R_0 = f'(\xi)h \tag{4.10}$$

Thus, we have derived the zero-order version of Eq. (4.8). The higher-order versions are merely a logical extension of the reasoning used to derive Eq. (4.10). The first-order version is

$$R_1 = \frac{f''(\xi)}{2!}h^2 \tag{4.11}$$

FIGURE 4.3
Graphical depiction of the derivative mean-value theorem.



For this case, the value of ξ conforms to the x value corresponding to the second derivative that makes Eq. (4.11) exact. Similar higher-order versions can be developed from Eq. (4.8).

4.1.2 Using the Taylor Series to Estimate Truncation Errors

Although the Taylor series will be extremely useful in estimating truncation errors throughout this book, it may not be clear to you how the expansion can actually be applied to numerical methods. In fact, we have already done so in our example of the falling parachutist. Recall that the objective of both Examples 1.1 and 1.2 was to predict velocity as a function of time. That is, we were interested in determining $v(t)$. As specified by Eq. (4.5), $v(t)$ can be expanded in a Taylor series:

$$v(t_{i+1}) = v(t_i) + v'(t_i)(t_{i+1} - t_i) + \frac{v''(t_i)}{2!}(t_{i+1} - t_i)^2 + \cdots + R_n \tag{4.12}$$

Now let us truncate the series after the first derivative term:

$$v(t_{i+1}) = v(t_i) + v'(t_i)(t_{i+1} - t_i) + R_1 \tag{4.13}$$

Equation (4.13) can be solved for

$$v'(t_i) = \underbrace{\frac{v(t_{i+1}) - v(t_i)}{t_{i+1} - t_i}}_{\substack{\text{First-order} \\ \text{approximation}}} - \underbrace{\frac{R_1}{t_{i+1} - t_i}}_{\substack{\text{Truncation} \\ \text{error}}} \tag{4.14}$$

The first part of Eq. (4.14) is exactly the same relationship that was used to approximate the derivative in Example 1.2 [Eq. (1.11)]. However, because of the Taylor series approach, we have now obtained an estimate of the truncation error associated with this approximation of the derivative. Using Eqs. (4.6) and (4.14) yields

$$\frac{R_1}{t_{i+1} - t_i} = \frac{v''(\xi)}{2!}(t_{i+1} - t_i) \tag{4.15}$$

or

$$\frac{R_1}{t_{i+1} - t_i} = O(t_{i+1} - t_i) \tag{4.16}$$

Thus, the estimate of the derivative [Eq. (1.11) or the first part of Eq. (4.14)] has a truncation error of order $t_{i+1} - t_i$. In other words, the error of our derivative approximation should be proportional to the step size. Consequently, if we halve the step size, we would expect to halve the error of the derivative.

EXAMPLE 4.3 The Effect of Nonlinearity and Step Size on the Taylor Series Approximation

Problem Statement. Figure 4.4 is a plot of the function

$$f(x) = x^m \tag{E4.3.1}$$

for $m = 1$, 2, 3, and 4 over the range from $x = 1$ to 2. Notice that for $m = 1$ the function is linear, and as m increases, more curvature or nonlinearity is introduced into the function.

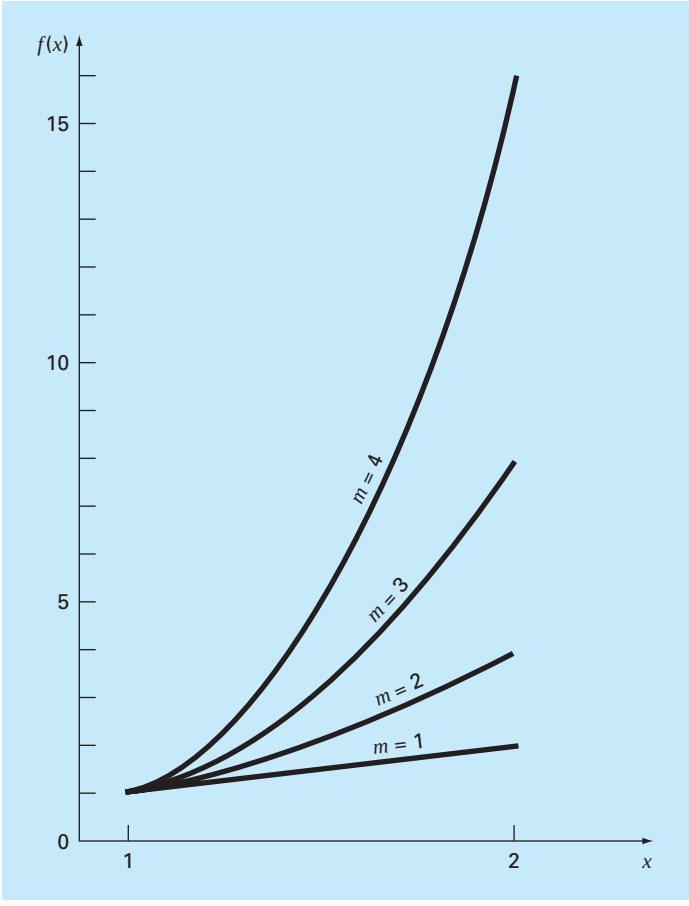


FIGURE 4.4
Plot of the function $f(x) = x^m$ for $m = 1$, 2, 3, and 4. Notice that the function becomes more nonlinear as m increases.

Employ the first-order Taylor series to approximate this function for various values of the exponent m and the step size h.

Solution. Equation (E4.3.1) can be approximated by a first-order Taylor series expansion, as in

$$f(x_{i+1}) = f(x_i) + mx_i^{m-1}h \tag{E4.3.2}$$

which has a remainder

$$R_1 = \frac{f''(x_i)}{2!}h^2 + \frac{f^{(3)}(x_i)}{3!}h^3 + \frac{f^{(4)}(x_i)}{4!}h^4 + \cdots$$

First, we can examine how the approximation performs as m increases—that is, as the function becomes more nonlinear. For $m = 1$, the actual value of the function at $x = 2$ is 2.

The Taylor series yields

$$f(2) = 1 + 1(1) = 2$$

and

$$R_1 = 0$$

The remainder is zero because the second and higher derivatives of a linear function are zero. Thus, as expected, the first-order Taylor series expansion is perfect when the underlying function is linear.

For $m = 2$, the actual value is $f(2) = 2^2 = 4$. The first-order Taylor series approximation is

$$f(2) = 1 + 2(1) = 3$$

and

$$R_1 = \frac{2}{2}(1)^2 + 0 + 0 + \cdots = 1$$

Thus, because the function is a parabola, the straight-line approximation results in a discrepancy. Note that the remainder is determined exactly.

For $m = 3$, the actual value is $f(2) = 2^3 = 8$. The Taylor series approximation is

$$f(2) = 1 + 3(1)^2(1) = 4$$

and

$$R_1 = \frac{6}{2}(1)^2 + \frac{6}{6}(1)^3 + 0 + 0 + \cdots = 4$$

Again, there is a discrepancy that can be determined exactly from the Taylor series.

For $m = 4$, the actual value is $f(2) = 2^4 = 16$. The Taylor series approximation is

$$f(2) = 1 + 4(1)^3(1) = 5$$

and

$$R_1 = \frac{12}{2}(1)^2 + \frac{24}{6}(1)^3 + \frac{24}{24}(1)^4 + 0 + 0 + \cdots = 11$$

On the basis of these four cases, we observe that R_1 increases as the function becomes more nonlinear. Furthermore, R_1 accounts exactly for the discrepancy. This is because Eq. (E4.3.1) is a simple monomial with a finite number of derivatives. This permits a complete determination of the Taylor series remainder.

Next, we will examine Eq. (E4.3.2) for the case $m = 4$ and observe how R_1 changes as the step size h is varied. For $m = 4$, Eq. (E4.3.2) is

$$f(x + h) = f(x) + 4x_i^3 h$$

If $x = 1$, $f(1) = 1$ and this equation can be expressed as

$$f(1 + h) = 1 + 4h$$

with a remainder of

$$R_1 = 6h^2 + 4h^3 + h^4$$

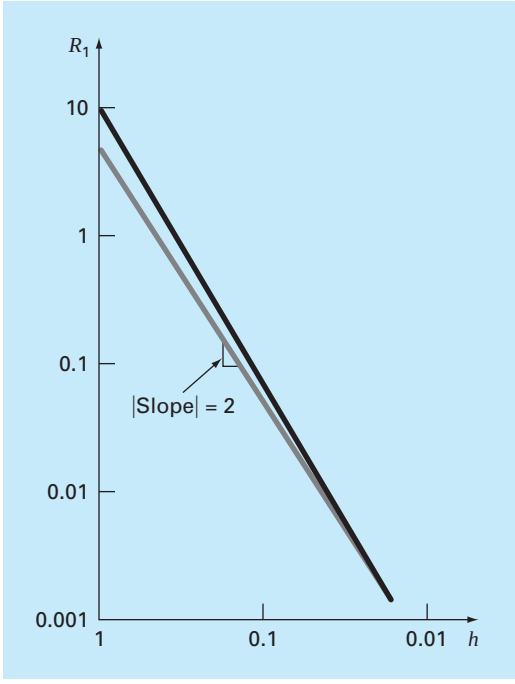


FIGURE 4.5
Log-log plot of the remainder R_1 of the first-order Taylor series approximation of the function $f(x) = x^4$ versus step size h. A line with a slope of 2 is also shown to indicate that as h decreases, the error becomes proportional to h^2.

TABLE 4.2 Comparison of the exact value of the function $f(x) = x^4$ with the first-order Taylor series approximation. Both the function and the approximation are evaluated at $x + h$, where $x = 1$.

h	True	First-Order Approximation	R_1
1	16	5	11
0.5	5.0625	3	2.0625
0.25	2.441406	2	0.441406
0.125	1.601807	1.5	0.101807
0.0625	1.274429	1.25	0.024429
0.03125	1.130982	1.125	0.005982
0.015625	1.063980	1.0625	0.001480

This leads to the conclusion that the discrepancy will decrease as h is reduced. Also, at sufficiently small values of h, the error should become proportional to h^2. That is, as h is halved, the error will be quartered. This behavior is confirmed by Table 4.2 and Fig. 4.5.

Thus, we conclude that the error of the first-order Taylor series approximation decreases as m approaches 1 and as h decreases. Intuitively, this means that the Taylor series

becomes more accurate when the function we are approximating becomes more like a straight line over the interval of interest. This can be accomplished either by reducing the size of the interval or by "straightening" the function by reducing m. Obviously, the latter option is usually not available in the real world because the functions we analyze are typically dictated by the physical problem context. Consequently, we do not have control of their lack of linearity, and our only recourse is reducing the step size or including additional terms in the Taylor series expansion.

4.1.3 Numerical Differentiation

Equation (4.14) is given a formal label in numerical methods—it is called a *finite divided difference.* It can be represented generally as

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} + O(x_{i+1} - x_i) \tag{4.17}$$

or

$$f'(x_i) = \frac{\Delta f_i}{h} + O(h) \tag{4.18}$$

where Δf_i is referred to as the *first forward difference* and h is called the step size, that is, the length of the interval over which the approximation is made. It is termed a "forward" difference because it utilizes data at i and $i + 1$ to estimate the derivative (Fig. 4.6*a*). The entire term $\Delta f/h$ is referred to as a *first finite divided difference.*

This forward divided difference is but one of many that can be developed from the Taylor series to approximate derivatives numerically. For example, backward and centered difference approximations of the first derivative can be developed in a fashion similar to the derivation of Eq. (4.14). The former utilizes values at x_{i-1} and x_i (Fig. 4.6*b*), whereas the latter uses values that are equally spaced around the point at which the derivative is estimated (Fig. 4.6*c*). More accurate approximations of the first derivative can be developed by including higher-order terms of the Taylor series. Finally, all the above versions can also be developed for second, third, and higher derivatives. The following sections provide brief summaries illustrating how some of these cases are derived.

Backward Difference Approximation of the First Derivative. The Taylor series can be expanded backward to calculate a previous value on the basis of a present value, as in

$$f(x_{i-1}) = f(x_i) - f'(x_i)h + \frac{f''(x_i)}{2!}h^2 - \cdots \tag{4.19}$$

Truncating this equation after the first derivative and rearranging yields

$$f'(x_i) \cong \frac{f(x_i) - f(x_{i-1})}{h} = \frac{\nabla f_1}{h} \tag{4.20}$$

where the error is $O(h)$, and ∇f_i is referred to as the *first backward difference.* See Fig. 4.6*b* for a graphical representation.

(*a*) Program

```
Option Explicit

Sub fig0314()
Dim term As Single, test As Single
Dim sum As Single, x As Single
Dim i As Integer
i = 0: term = 1#: sum = 1#: test = 0#
Sheets("sheet1").Select
Range("b1").Select
x = ActiveCell.Value
Range("a3:c1003").ClearContents
Range("a3").Select
Do
  If sum = test Then Exit Do
  ActiveCell.Value = i
  ActiveCell.Offset(0, 1).Select
  ActiveCell.Value = term
  ActiveCell.Offset(0, 1).Select
  ActiveCell.Value = sum
  ActiveCell.Offset(1, -2).Select
  i = i + 1
  test = sum
  term = x ^ i / _
    Application.WorksheetFunction.Fact(i)
  sum = sum + term
Loop
ActiveCell.Offset(0, 1).Select
ActiveCell.Value = "Exact value = "
ActiveCell.Offset(0, 1).Select
ActiveCell.Value = Exp(x)
End Sub
```

(*b*) Evaluation of e^{10}

	A	B	C
1	x	10	
2	i	term	sum
3	0	1.000000	1.000000
4	1	10.000000	11.000000
5	2	50.000000	61.000000
6	3	166.666672	227.666672
7	4	416.666656	644.333313
8	5	833.333313	1477.666626
30	27	9.183690E-02	22026.416016
31	28	3.279889E-02	22026.449219
32	29	1.130996E-02	22026.460938
33	30	3.769988E-03	22026.464844
34	31	1.216125E-03	22026.466797
35		Exact value =	22026.465795

(*c*) Evaluation of e^{10}

	A	B	C
1	x	-10	
2	i	term	sum
3	0	1.000000	1.000000
4	1	-10.000000	-9.000000
5	2	50.000000	41.000000
6	3	-166.666672	-125.666672
7	4	416.666656	291.000000
8	5	-833.333313	-542.333313
44	41	-2.989311E-09	1.103359E-04
45	42	7.117407E-10	1.103366E-04
46	43	-1.655211E-10	1.103365E-04
47	44	3.761843E-11	1.103365E-04
48	45	-8.359651E-12	1.103365E-04
49		Exact value =	4.539993E-05

FIGURE 3.14
(*a*) An Excel/VBA program to evaluate e^x using an infinite series. (*b*) Evaluation of e^x. (*c*) Evaluation of e^{-x}.

current term added to the series, and *sum* is the accumulative value of the series. The variable *test* is the preceding accumulative value of the series prior to adding *term.* The series is terminated when the computer cannot detect the difference between *test* and *sum.*

Figure 3.14*b* shows the results of running the program for $x = 10$. Note that this case is completely satisfactory. The final result is achieved in 31 terms with the series identical to the library function value within seven significant figures.

Figure 3.14*c* shows similar results for $x = -10$. However, for this case, the results of the series calculation are not even the same sign as the true result. As a matter of fact, the negative results are open to serious question because e^x can never be less than zero. The problem here is caused by round-off error. Note that many of the terms that make up

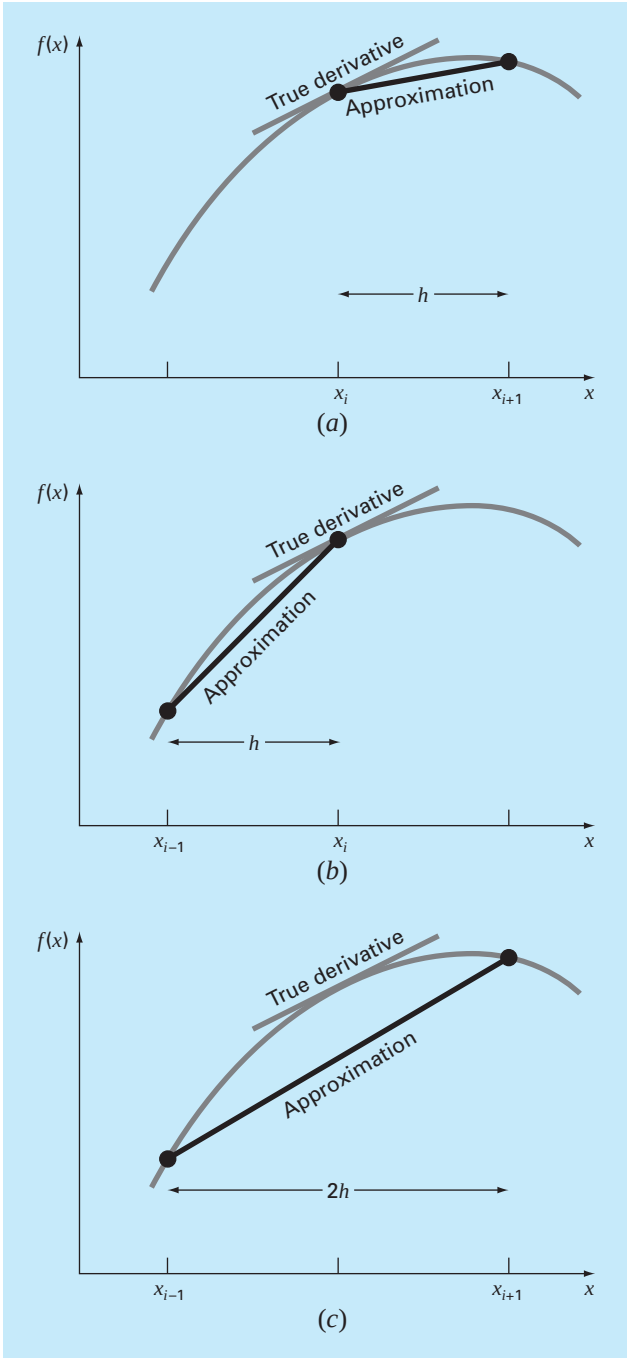


FIGURE 4.6
Graphical depiction of (*a*) forward, (*b*) backward, and (*c*) centered finite-divided-difference approximations of the first derivative.

Centered Difference Approximation of the First Derivative. A third way to approximate the first derivative is to subtract Eq. (4.19) from the forward Taylor series expansion:

$$f(x_{i+1}) = f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2!}h^2 + \cdots \tag{4.21}$$

to yield

$$f(x_{i+1}) = f(x_{i-1}) + 2f'(x_i)h + \frac{2f^{(3)}(x_i)}{3!}h^3 + \cdots$$

which can be solved for

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} - \frac{f^{(3)}(x_i)}{6}h^2 - \cdots$$

or

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} - O(h^2) \tag{4.22}$$

Equation (4.22) is a *centered difference* representation of the first derivative. Notice that the truncation error is of the order of h^2 in contrast to the forward and backward approximations that were of the order of h. Consequently, the Taylor series analysis yields the practical information that the centered difference is a more accurate representation of the derivative (Fig. 4.6*c*). For example, if we halve the step size using a forward or backward difference, we would approximately halve the truncation error, whereas for the central difference, the error would be quartered.

EXAMPLE 4.4 Finite-Divided-Difference Approximations of Derivatives

Problem Statement. Use forward and backward difference approximations of $O(h)$ and a centered difference approximation of $O(h^2)$ to estimate the first derivative of

$$f(x) = -0.1x^4 - 0.15x^3 - 0.5x^2 - 0.25x + 1.2$$

at $x = 0.5$ using a step size $h = 0.5$. Repeat the computation using $h = 0.25$. Note that the derivative can be calculated directly as

$$f'(x) = -0.4x^3 - 0.45x^2 - 1.0x - 0.25$$

and can be used to compute the true value as $f'(0.5) = -0.9125$.

Solution. For $h = 0.5$, the function can be employed to determine

$$\begin{aligned} x_{i-1} &= 0 & f(x_{i-1}) &= 1.2 \\ x_i &= 0.5 & f(x_i) &= 0.925 \\ x_{i+1} &= 1.0 & f(x_{i+1}) &= 0.2 \end{aligned}$$

These values can be used to compute the forward divided difference [Eq. (4.17)],

$$f'(0.5) \cong \frac{0.2 - 0.925}{0.5} = -1.45 \qquad |\varepsilon_t| = 58.9\%$$

the backward divided difference [Eq. (4.20)],

$$f'(0.5) \cong \frac{0.925 - 1.2}{0.5} = -0.55 \qquad |\varepsilon_t| = 39.7\%$$

and the centered divided difference [Eq. (4.22)],

$$f'(0.5) \cong \frac{0.2 - 1.2}{1.0} = -1.0 \qquad |\varepsilon_t| = 9.6\%$$

For $h = 0.25$,

$$\begin{aligned} x_{i-1} &= 0.25 & f(x_{i-1}) &= 1.10351563 \\ x_i &= 0.5 & f(x_i) &= 0.925 \\ x_{i+1} &= 0.75 & f(x_{i+1}) &= 0.63632813 \end{aligned}$$

which can be used to compute the forward divided difference,

$$f'(0.5) \cong \frac{0.63632813 - 0.925}{0.25} = -1.155 \qquad |\varepsilon_t| = 26.5\%$$

the backward divided difference,

$$f'(0.5) \cong \frac{0.925 - 1.10351563}{0.25} = -0.714 \qquad |\varepsilon_t| = 21.7\%$$

and the centered divided difference,

$$f'(0.5) \cong \frac{0.63632813 - 1.10351563}{0.5} = -0.934 \qquad |\varepsilon_t| = 2.4\%$$

For both step sizes, the centered difference approximation is more accurate than forward or backward differences. Also, as predicted by the Taylor series analysis, halving the step size approximately halves the error of the backward and forward differences and quarters the error of the centered difference.

Finite Difference Approximations of Higher Derivatives. Besides first derivatives, the Taylor series expansion can be used to derive numerical estimates of higher derivatives. To do this, we write a forward Taylor series expansion for $f(x_{i+2})$ in terms of $f(x_i)$:

$$f(x_{i+2}) = f(x_i) + f'(x_i)(2h) + \frac{f''(x_i)}{2!}(2h)^2 + \cdots \tag{4.23}$$

Equation (4.21) can be multiplied by 2 and subtracted from Eq. (4.23) to give

$$f(x_{i+2}) - 2f(x_{i+1}) = -f(x_i) + f''(x_i)h^2 + \cdots$$

which can be solved for

$$f''(x_i) = \frac{f(x_{i+2}) - 2f(x_{i+1}) + f(x_i)}{h^2} + O(h) \tag{4.24}$$

This relationship is called the *second forward finite divided difference.* Similar manipulations can be employed to derive a backward version

$$f''(x_i) = \frac{f(x_i) - 2f(x_{i-1}) + f(x_{i-2})}{h^2} + O(h)$$

and a centered version

$$f''(x_i) = \frac{f(x_{i+1}) - 2f(x_i) + f(x_{i-1})}{h^2} + O(h^2)$$

As was the case with the first-derivative approximations, the centered case is more accurate. Notice also that the centered version can be alternatively expressed as

$$f''(x_i) \cong \frac{\dfrac{f(x_{i+1}) - f(x_i)}{h} - \dfrac{f(x_i) - f(x_{i-1})}{h}}{h}$$

Thus, just as the second derivative is a derivative of a derivative, the second divided difference approximation is a difference of two first divided differences.

We will return to the topic of numerical differentiation in Chap. 23. We have introduced you to the topic at this point because it is a very good example of why the Taylor series is important in numerical methods. In addition, several of the formulas introduced in this section will be employed prior to Chap. 23.

4.2 ERROR PROPAGATION

The purpose of this section is to study how errors in numbers can propagate through mathematical functions. For example, if we multiply two numbers that have errors, we would like to estimate the error in the product.

4.2.1 Functions of a Single Variable

Suppose that we have a function $f(x)$ that is dependent on a single independent variable x. Assume that $\tilde{x}$ is an approximation of x. We, therefore, would like to assess the effect of the discrepancy between x and $\tilde{x}$ on the value of the function. That is, we would like to estimate

$$\Delta f(\tilde{x}) = |f(x) - f(\tilde{x})|$$

The problem with evaluating $\Delta f(\tilde{x})$ is that $f(x)$ is unknown because x is unknown. We can overcome this difficulty if $\tilde{x}$ is close to x and $f(\tilde{x})$ is continuous and differentiable. If these conditions hold, a Taylor series can be employed to compute $f(x)$ near $f(\tilde{x})$, as in

$$f(x) = f(\tilde{x}) + f'(\tilde{x})(x - \tilde{x}) + \frac{f''(\tilde{x})}{2}(x - \tilde{x})^2 + \cdots$$

Dropping the second- and higher-order terms and rearranging yields

$$f(x) - f(\tilde{x}) \cong f'(\tilde{x})(x - \tilde{x})$$

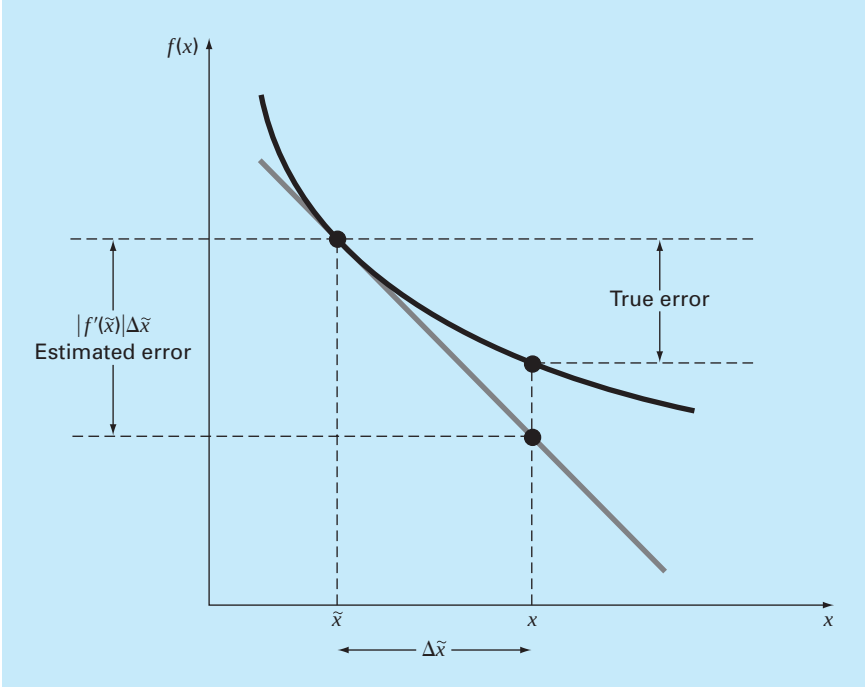


FIGURE 4.7
Graphical depiction of first-order error propagation.

or

$$\Delta f(\tilde{x}) = |f'(\tilde{x})|\Delta\tilde{x} \tag{4.25}$$

where $\Delta f(\tilde{x}) = |f(x) - f(\tilde{x})|$ represents an estimate of the error of the function and $\Delta\tilde{x} = |x - \tilde{x}|$ represents an estimate of the error of x. Equation (4.25) provides the capability to approximate the error in $f(x)$ given the derivative of a function and an estimate of the error in the independent variable. Figure 4.7 is a graphical illustration of the operation.

EXAMPLE 4.5 Error Propagation in a Function of a Single Variable

Problem Statement. Given a value of $\tilde{x} = 2.5$ with an error of $\Delta\tilde{x} = 0.01$, estimate the resulting error in the function, $f(x) = x^3$.

Solution. Using Eq. (4.25),

$$\Delta f(\tilde{x}) \cong 3(2.5)^2(0.01) = 0.1875$$

Because $f(2.5) = 15.625$, we predict that

$$f(2.5) = 15.625 \pm 0.1875$$

or that the true value lies between 15.4375 and 15.8125. In fact, if x were actually 2.49, the function could be evaluated as 15.4382, and if x were 2.51, it would be 15.8132. For this case, the first-order error analysis provides a fairly close estimate of the true error.

4.2.2 Functions of More than One Variable

The foregoing approach can be generalized to functions that are dependent on more than one independent variable. This is accomplished with a multivariable version of the Taylor series. For example, if we have a function of two independent variables u and v, the Taylor series can be written as

$$f(u_{i+1}, v_{i+1}) = f(u_i, v_i) + \frac{\partial f}{\partial u}(u_{i+1} - u_i) + \frac{\partial f}{\partial v}(v_{i+1} - v_i) + \frac{1}{2!}\left[\frac{\partial^2 f}{\partial u^2}(u_{i+1} - u_i)^2 + 2\frac{\partial^2 f}{\partial u \partial v}(u_{i+1} - u_i)(v_{i+1} - v_i) + \frac{\partial^2 f}{\partial v^2}(v_{i+1} - v_i)^2\right] + \cdots \tag{4.26}$$

where all partial derivatives are evaluated at the base point i. If all second-order and higher terms are dropped, Eq. (4.26) can be solved for

$$\Delta f(\tilde{u}, \tilde{v}) = \left|\frac{\partial f}{\partial u}\right| \Delta\tilde{u} + \left|\frac{\partial f}{\partial v}\right| \Delta\tilde{v}$$

where $\Delta\tilde{u}$ and $\Delta\tilde{v}$ = estimates of the errors in u and v, respectively.

For n independent variables $\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n$ having errors $\Delta\tilde{x}_1, \Delta\tilde{x}_2, \ldots, \Delta x_n$, the following general relationship holds:

$$\Delta f(\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n) \cong \left|\frac{\partial f}{\partial x_1}\right| \Delta\tilde{x}_1 + \left|\frac{\partial f}{\partial x_2}\right| \Delta\tilde{x}_2 + \cdots + \left|\frac{\partial f}{\partial x_n}\right| \Delta\tilde{x}_n \tag{4.27}$$

EXAMPLE 4.6 Error Propagation in a Multivariable Function

Problem Statement. The deflection y of the top of a sailboat mast is

$$y = \frac{FL^4}{8EI}$$

where F = a uniform side loading (N/m), L = height (m), E = the modulus of elasticity (N/m^2), and I = the moment of inertia (m^4). Estimate the error in y given the following data:

$\tilde{F} = 750$ N/m	$\Delta\tilde{F} = 30$ N/m
$\tilde{L} = 9$ m	$\Delta\tilde{L} = 0.03$ m
$\tilde{E} = 7.5 \times 10^9$ N/m^2	$\Delta\tilde{E} = 5 \times 10^7$ N/m^2
$\tilde{I} = 0.0005$ m^4	$\Delta\tilde{I} = 0.000005$ m^4

Solution. Employing Eq. (4.27) gives

$$\Delta y(\tilde{F}, \tilde{L}, \tilde{E}, \tilde{I}) = \left|\frac{\partial y}{\partial F}\right| \Delta\tilde{F} + \left|\frac{\partial y}{\partial L}\right| \Delta\tilde{L} + \left|\frac{\partial y}{\partial E}\right| \Delta\tilde{E} + \left|\frac{\partial y}{\partial I}\right| \Delta\tilde{I}$$

or

$$\Delta y(\tilde{F}, \tilde{L}, \tilde{E}, \tilde{I}) \cong \frac{\tilde{L}^4}{8\tilde{E}\tilde{I}}\Delta\tilde{F} + \frac{\tilde{F}\tilde{L}^3}{2\tilde{E}\tilde{I}}\Delta\tilde{L} + \frac{\tilde{F}\tilde{L}^4}{8\tilde{E}^2\tilde{I}}\Delta\tilde{E} + \frac{\tilde{F}\tilde{L}^4}{8\tilde{E}\tilde{I}^2}\Delta\tilde{I}$$

Substituting the appropriate values gives

$$\Delta y = 0.006561 + 0.002187 + 0.001094 + 0.00164 = 0.011482$$

Therefore, $y = 0.164025 \pm 0.011482$. In other words, y is between 0.152543 and 0.175507 m. The validity of these estimates can be verified by substituting the extreme values for the variables into the equation to generate an exact minimum of

$$y_{\min} = \frac{720(8.97)^4}{8(7.55 \times 10^9)0.000505} = 0.152818$$

and

$$y_{\max} = \frac{780(9.03)^4}{8(7.45 \times 10^9)0.000495} = 0.175790$$

Thus, the first-order estimates are reasonably close to the exact values.

Equation (4.27) can be employed to define error propagation relationships for common mathematical operations. The results are summarized in Table 4.3. We will leave the derivation of these formulas as a homework exercise.

4.2.3 Stability and Condition

The *condition* of a mathematical problem relates to its sensitivity to changes in its input values. We say that a computation is *numerically unstable* if the uncertainty of the input values is grossly magnified by the numerical method.

These ideas can be studied using a first-order Taylor series

$$f(x) = f(\tilde{x}) + f'(\tilde{x})(x - \tilde{x})$$

This relationship can be employed to estimate the *relative error* of $f(x)$ as in

$$\frac{f(x) - f(\tilde{x})}{f(x)} \cong \frac{f'(\tilde{x})(x - \tilde{x})}{f(\tilde{x})}$$

The *relative error* of x is given by

$$\frac{x - \tilde{x}}{\tilde{x}}$$

TABLE 4.3 Estimated error bounds associated with common mathematical operations using inexact numbers $\tilde{u}$ and $\tilde{v}$.

Operation		Estimated Error
Addition	$\Delta(\tilde{u} + \tilde{v})$	$\Delta\tilde{u} + \Delta\tilde{v}$
Subtraction	$\Delta(\tilde{u} - \tilde{v})$	$\Delta\tilde{u} + \Delta\tilde{v}$
Multiplication	$\Delta(\tilde{u} \times \tilde{v})$	$\lvert\tilde{u}\rvert\Delta\tilde{v} + \lvert\tilde{v}\rvert\Delta\tilde{u}$
Division	$\Delta\left(\frac{\tilde{u}}{\tilde{v}}\right)$	$\frac{\lvert\tilde{u}\rvert\Delta\tilde{v} + \lvert\tilde{v}\rvert\Delta\tilde{u}}{\lvert\tilde{v}\rvert^2}$

A *condition number* can be defined as the ratio of these relative errors

$$\text{Condition number} = \frac{\tilde{x}\, f'(\tilde{x})}{f(\tilde{x})} \tag{4.28}$$

The condition number provides a measure of the extent to which an uncertainty in x is magnified by $f(x)$. A value of 1 tells us that the function's relative error is identical to the relative error in x. A value greater than 1 tells us that the relative error is amplified, whereas a value less than 1 tells us that it is attenuated. Functions with very large values are said to be *ill-conditioned*. Any combination of factors in Eq. (4.28) that increases the numerical value of the condition number will tend to magnify uncertainties in the computation of $f(x)$.

EXAMPLE 4.7 Condition Number

Problem Statement. Compute and interpret the condition number for

$$f(x) = \tan x \qquad \text{for } \tilde{x} = \frac{\pi}{2} + 0.1\left(\frac{\pi}{2}\right)$$

$$f(x) = \tan x \qquad \text{for } \tilde{x} = \frac{\pi}{2} + 0.01\left(\frac{\pi}{2}\right)$$

Solution. The condition number is computed as

$$\text{Condition number} = \frac{\tilde{x}(1/\cos^2 x)}{\tan \tilde{x}}$$

For $\tilde{x} = \pi/2 + 0.1(\pi/2)$,

$$\text{Condition number} = \frac{1.7279(40.86)}{-6.314} = -11.2$$

Thus, the function is ill-conditioned. For $\tilde{x} = \pi/2 + 0.01(\pi/2)$, the situation is even worse:

$$\text{Condition number} = \frac{1.5865(4053)}{-63.66} = -101$$

For this case, the major cause of ill conditioning appears to be the derivative. This makes sense because in the vicinity of $\pi/2$, the tangent approaches both positive and negative infinity.

4.3 TOTAL NUMERICAL ERROR

The *total numerical error* is the summation of the truncation and round-off errors. In general, the only way to minimize round-off errors is to increase the number of significant figures of the computer. Further, we have noted that round-off error will *increase* due to subtractive cancellation or due to an increase in the number of computations in an analysis. In contrast, Example 4.4 demonstrated that the truncation error can be reduced by decreasing the step size. Because a decrease in step size can lead to subtractive cancellation or to an increase in computations, the truncation errors are *decreased* as the round-off errors are *increased*. Therefore, we are faced by the following dilemma: The strategy for decreasing

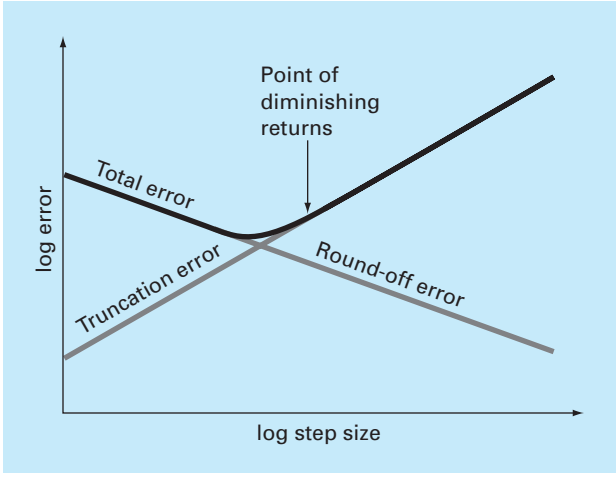


FIGURE 4.8
A graphical depiction of the trade-off between round-off and truncation error that sometimes comes into play in the course of a numerical method. The point of diminishing returns is shown, where round-off error begins to negate the benefits of step-size reduction.

one component of the total error leads to an increase of the other component. In a computation, we could conceivably decrease the step size to minimize truncation errors only to discover that in doing so, the round-off error begins to dominate the solution and the total error grows! Thus, our remedy becomes our problem (Fig. 4.8). One challenge that we face is to determine an appropriate step size for a particular computation. We would like to choose a large step size in order to decrease the amount of calculations and round-off errors without incurring the penalty of a large truncation error. If the total error is as shown in Fig. 4.8, the challenge is to identify the point of diminishing returns where round-off error begins to negate the benefits of step-size reduction.

In actual cases, however, such situations are relatively uncommon because most computers carry enough significant figures that round-off errors do not predominate. Nevertheless, they sometimes do occur and suggest a sort of "numerical uncertainty principle" that places an absolute limit on the accuracy that may be obtained using certain computerized numerical methods. We explore such a case in the following section.

4.3.1 Error Analysis of Numerical Differentiation

As described in the Sec. 4.1.3, a centered difference approximation of the first derivative can be written as (Eq. 4.22):

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} - \frac{f^{(3)}(\xi)}{6}h^2 \tag{4.29}$$

True value | Finite-difference approximation | Truncation error

Thus, if the two function values in the numerator of the finite-difference approximation have no round-off error, the only error is due to truncation.

However, because we are using digital computers, the function values do include round-off error as in

$$f(x_{i-1}) = \tilde{f}(x_{i-1}) + e_{i-1}$$
$$f(x_{i+1}) = \tilde{f}(x_{i+1}) + e_{i+1}$$

where the $\tilde{f}$'s are the rounded function values and the e's are the associated round-off errors. Substituting these values into Eq. (4.29) gives

$$f'(x_i) = \frac{\tilde{f}(x_{i+1}) - \tilde{f}(x_{i-1})}{2h} + \frac{e_{i+1} - e_{i-1}}{2h} - \frac{f^{(3)}(\xi)}{6}h^2$$

True value	Finite-difference approximation	Round-off error	Truncation error

We can see that the total error of the finite-difference approximation consists of a round-off error which increases with step size and a truncation error that decreases with step size.

Assuming that the absolute value of each component of the round-off error has an upper bound of ε, the maximum possible value of the difference $e_{i+1} - e_i$ will be 2ε. Further, assume that the third derivative has a maximum absolute value of M. An upper bound on the absolute value of the total error can therefore be represented as

$$\text{Total error} = \left| f'(x_i) - \frac{\tilde{f}(x_{i+1}) - \tilde{f}(x_{i-1})}{2h} \right| \le \frac{\varepsilon}{h} + \frac{h^2 M}{6} \tag{4.30}$$

An optimal step size can be determined by differentiating Eq. (4.30), setting the result equal to zero and solving for

$$h_{\text{opt}} = \sqrt[3]{\frac{3\varepsilon}{M}} \tag{4.31}$$

EXAMPLE 4.8 Round-off and Truncation Errors in Numerical Differentiation

Problem Statement. In Example 4.4, we used a centered difference approximation of $O(h^2)$ to estimate the first derivative of the following function at $x = 0.5$,

$$f(x) = -0.1x^4 - 0.15x^3 - 0.5x^2 - 0.25x + 1.2$$

Perform the same computation starting with $h = 1$. Then progressively divide the step size by a factor of 10 to demonstrate how round-off becomes dominant as the step size is reduced. Relate your results to Eq. (4.31). Recall that the true value of the derivative is -0.9125.

Solution. We can develop a program to perform the computations and plot the results. For the present example, we have done this with a MATLAB M-file. Notice that we pass both the function and its analytical derivative as arguments. In addition, the function generates a plot of the results.

```
function diffex(func,dfunc,x,n)
format long
dftrue=dfunc(x);
h=1;
H(1)=h;
D(1)=(func(x+h)-func(x-h))/(2*h);
E(1)=abs(dftrue-D(1));
for i=2:n
  h=h/10;
  H(i)=h;
  D(i)=(func(x+h)-func(x-h))/(2*h);
  E(i)=abs(dftrue-D(i));
end
L=[H' D' E'];
fprintf('   step size   finite difference    true error\n');
fprintf('%14.10f %16.14f %16.13f\n',L);
loglog(H,E),xlabel('Step Size'),ylabel('Error')
title('Plot of Error Versus Step Size')
format short
```

The M-file can then be run using the following commands:

```
>> ff=@(x) -0.1*x^4-0.15*x^3-0.5*x^2-0.25*x+1.2;
>> df=@(x) -0.4*x^3-0.45*x^2-x-0.25;
>> diffex(ff,df,0.5,11)
```

When the function is run, the following numeric output is generated along with the plot (Fig. 4.9):

```
   step size   finite difference    true error
 1.0000000000 -1.26250000000000  0.3500000000000
 0.1000000000 -0.91600000000000  0.0035000000000
 0.0100000000 -0.91253500000000  0.0000350000000
 0.0010000000 -0.91250035000001  0.0000003500000
 0.0001000000 -0.91250000349985  0.0000000034998
 0.0000100000 -0.91250000003318  0.0000000000332
 0.0000010000 -0.91250000000542  0.0000000000054
 0.0000001000 -0.91249999945031  0.0000000005497
 0.0000000100 -0.91250000333609  0.0000000033361
 0.0000000010 -0.91250001998944  0.0000000199894
 0.0000000001 -0.91250007550059  0.0000000755006
```

The results are as expected. At first, round-off is minimal and the estimate is dominated by truncation error. Hence, as in Eq. (4.30), the total error drops by a factor of 100 each time we divide the step by 10. However, starting at $h = 0.0001$, we see round-off error begin to creep in and erode the rate at which the error diminishes. A minimum error is reached at $h = 10^{-6}$. Beyond this point, the error increases as round-off dominates.

Because we are dealing with an easily differentiable function, we can also investigate whether these results are consistent with Eq. (4.31). First, we can estimate M by evaluating the function's third derivative as

$$M = |f^3(0.5)| = |-2.4(0.5) - 0.9| = 2.1$$

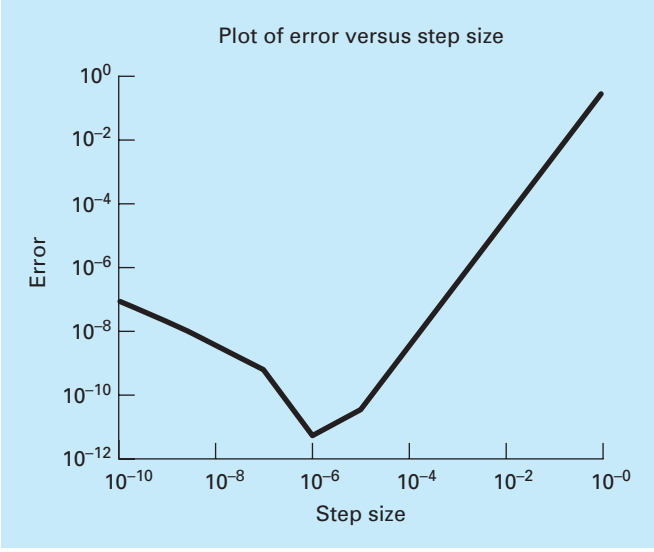


FIGURE 4.9
Plot of error versus step size.

Because MATLAB has a precision of about 15 to 16 base-10 digits, a rough estimate of the upper bound on round-off would be about $\varepsilon = 0.5 \times 10^{-16}$. Substituting these values into Eq. (4.31) gives

$$h_{\text{opt}} = \sqrt[3]{\frac{3(0.5 \times 10^{-16})}{2.1}} = 4.3 \times 10^{-6}$$

which is on the same order as the result of 1×10^{-6} obtained with our computer program.

4.3.2 Control of Numerical Errors

For most practical cases, we do not know the exact error associated with numerical methods. The exception, of course, is when we have obtained the exact solution that makes our numerical approximations unnecessary. Therefore, for most engineering applications we must settle for some estimate of the error in our calculations.

There are no systematic and general approaches to evaluating numerical errors for all problems. In many cases error estimates are based on the experience and judgment of the engineer.

Although error analysis is to a certain extent an art, there are several practical programming guidelines we can suggest. First and foremost, avoid subtracting two nearly equal numbers. Loss of significance almost always occurs when this is done. Sometimes you can rearrange or reformulate the problem to avoid subtractive cancellation. If this is not possible, you may want to use extended-precision arithmetic. Furthermore, when adding

and subtracting numbers, it is best to sort the numbers and work with the smallest numbers first. This avoids loss of significance.

Beyond these computational hints, one can attempt to predict total numerical errors using theoretical formulations. The Taylor series is our primary tool for analysis of both truncation and round-off errors. Several examples have been presented in this chapter. Prediction of total numerical error is very complicated for even moderately sized problems and tends to be pessimistic. Therefore, it is usually attempted for only small-scale tasks.

The tendency is to push forward with the numerical computations and try to estimate the accuracy of your results. This can sometimes be done by seeing if the results satisfy some condition or equation as a check. Or it may be possible to substitute the results back into the original equation to check that it is actually satisfied.

Finally you should be prepared to perform numerical experiments to increase your awareness of computational errors and possible ill-conditioned problems. Such experiments may involve repeating the computations with a different step size or method and comparing the results. We may employ sensitivity analysis to see how our solution changes when we change model parameters or input values. We may want to try different numerical algorithms that have different theoretical foundations, are based on different computational strategies, or have different convergence properties and stability characteristics.

When the results of numerical computations are extremely critical and may involve loss of human life or have severe economic ramifications, it is appropriate to take special precautions. This may involve the use of two or more independent groups to solve the same problem so that their results can be compared.

The roles of errors will be a topic of concern and analysis in all sections of this book. We will leave these investigations to specific sections.

4.4 BLUNDERS, FORMULATION ERRORS, AND DATA UNCERTAINTY

Although the following sources of error are not directly connected with most of the numerical methods in this book, they can sometimes have great impact on the success of a modeling effort. Thus, they must always be kept in mind when applying numerical techniques in the context of real-world problems.

4.4.1 Blunders

We are all familiar with gross errors, or blunders. In the early years of computers, erroneous numerical results could sometimes be attributed to malfunctions of the computer itself. Today, this source of error is highly unlikely, and most blunders must be attributed to human imperfection.

Blunders can occur at any stage of the mathematical modeling process and can contribute to all the other components of error. They can be avoided only by sound knowledge of fundamental principles and by the care with which you approach and design your solution to a problem.

Blunders are usually disregarded in discussions of numerical methods. This is no doubt due to the fact that, try as we may, mistakes are to a certain extent unavoidable. However, we believe that there are a number of ways in which their occurrence can be minimized. In

particular, the good programming habits that were outlined in Chap. 2 are extremely useful for mitigating programming blunders. In addition, there are usually simple ways to check whether a particular numerical method is working properly. Throughout this book, we discuss ways to check the results of numerical calculations.

4.4.2 Formulation Errors

Formulation, or *model*, *errors* relate to bias that can be ascribed to incomplete mathematical models. An example of a negligible formulation error is the fact that Newton's second law does not account for relativistic effects. This does not detract from the adequacy of the solution in Example 1.1 because these errors are minimal on the time and space scales associated with the falling parachutist problem.

However, suppose that air resistance is not linearly proportional to fall velocity, as in Eq. (1.7), but is a function of the square of velocity. If this were the case, both the analytical and numerical solutions obtained in the Chap. 1 would be erroneous because of formulation error. Further consideration of formulation error is included in some of the engineering applications in the remainder of the book. You should be cognizant of these problems and realize that, if you are working with a poorly conceived model, no numerical method will provide adequate results.

4.4.3 Data Uncertainty

Errors sometimes enter into an analysis because of uncertainty in the physical data upon which a model is based. For instance, suppose we wanted to test the falling parachutist model by having an individual make repeated jumps and then measuring his or her velocity after a specified time interval. Uncertainty would undoubtedly be associated with these measurements, since the parachutist would fall faster during some jumps than during others. These errors can exhibit both inaccuracy and imprecision. If our instruments consistently underestimate or overestimate the velocity, we are dealing with an inaccurate, or biased, device. On the other hand, if the measurements are randomly high and low, we are dealing with a question of precision.

Measurement errors can be quantified by summarizing the data with one or more well-chosen statistics that convey as much information as possible regarding specific characteristics of the data. These descriptive statistics are most often selected to represent (1) the location of the center of the distribution of the data and (2) the degree of spread of the data. As such, they provide a measure of the bias and imprecision, respectively. We will return to the topic of characterizing data uncertainty in Part Five.

Although you must be cognizant of blunders, formulation errors, and uncertain data, the numerical methods used for building models can be studied, for the most part, independently of these errors. Therefore, for most of this book, we will assume that we have not made gross errors, we have a sound model, and we are dealing with error-free measurements. Under these conditions, we can study numerical errors without complicating factors.

PROBLEMS

4.1 The Maclaurin series expansion for cos x is

$$\cos x = 1 - \frac{x^2}{2} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \cdots$$

Starting with the simplest version, cos $x = 1$, add terms one at a time to estimate $\cos(\pi/3)$. After each new term is added, compute the true and approximate percent relative errors. Use your pocket calculator to determine the true value. Add terms until the absolute value of the approximate error estimate falls below an error criterion conforming to two significant figures.

4.2 Perform the same computation as in Prob. 4.1, but use the Maclaurin series expansion for the sin x to estimate $\sin(\pi/3)$.

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

4.3 The following infinite series can be used to approximate e^x:

$$e^x = 1 + x + \frac{x^2}{2} + \frac{x^3}{3!} + \cdots + \frac{x^n}{n!}$$

(a) Prove that this Maclaurin series expansion is a special case of the Taylor series expansion [(Eq. (4.7)] with $x_i = 0$ and $h = x$.

(b) Use the Taylor series to estimate $f(x) = e^{-x}$ at $x_{i+1} = 1$ for $x_i = 0.2$. Employ the zero-, first-, second-, and third-order versions and compute the $|\varepsilon_t|$ for each case.

4.4 Use zero- through fourth-order Taylor series expansions to predict $f(2.5)$ for $f(x) = \ln x$ using a base point at $x = 1$. Compute the true percent relative error ε_t for each approximation. Discuss the meaning of the results.

4.5 Use zero- through third-order Taylor series expansions to predict $f(3)$ for

$$f(x) = 25x^3 - 6x^2 + 7x - 88$$

using a base point at $x = 1$. Compute the true percent relative error ε_t for each approximation.

4.6 Use forward and backward difference approximations of $O(h)$ and a centered difference approximation of $O(h^2)$ to estimate the first derivative of the function examined in Prob. 4.5. Evaluate the derivative at $x = 2$ using a step size of $h = 0.2$. Compare your results with the true value of the derivative. Interpret your results on the basis of the remainder term of the Taylor series expansion.

4.7 Use a centered difference approximation of $O(h^2)$ to estimate the second derivative of the function examined in Prob. 4.5. Perform the evaluation at $x = 2$ using step sizes of $h = 0.25$ and 0.125. Compare your estimates with the true value of the second derivative. Interpret your results on the basis of the remainder term of the Taylor series expansion.

4.8 The Stefan-Boltzmann law can be employed to estimate the rate of radiation of energy H from a surface, as in

$$H = Ae\sigma T^4$$

where H is in watts, A = the surface area (m^2), e = the emissivity that characterizes the emitting properties of the surface (dimensionless), σ = a universal constant called the Stefan-Boltzmann constant ($= 5.67 \times 10^{-8}$ W m^{-2} K^{-4}), and T = absolute temperature (K). Determine the error of H for a steel plate with $A = 0.15$ m^2, $e = 0.90$, and $T = 650 \pm 20$. Compare your results with the exact error. Repeat the computation but with $T = 650 \pm 40$. Interpret your results.

4.9 Repeat Prob. 4.8 but for a copper sphere with radius $= 0.15 \pm 0.01$ m, $e = 0.90 \pm 0.05$, and $T = 550 \pm 20$.

4.10 Recall that the velocity of the falling parachutist can be computed by [Eq. (1.10)],

$$v(t) = \frac{gm}{c}\left(1 - e^{-(c/m)t}\right)$$

Use a first-order error analysis to estimate the error of v at $t = 6$, if $g = 9.8$ and $m = 50$ but $c = 12.5 \pm 1.5$.

4.11 Repeat Prob. 4.10 with $g = 9.8$, $t = 6$, $c = 12.5 \pm 1.5$, and $m = 50 \pm 2$.

4.12 Evaluate and interpret the condition numbers for

(a) $f(x) = \sqrt{|x-1|} + 1$ for $x = 1.00001$

(b) $f(x) = e^{-x}$ for $x = 10$

(c) $f(x) = \sqrt{x^2+1} - x$ for $x = 300$

(d) $f(x) = \dfrac{e^{-x} - 1}{x}$ for $x = 0.001$

(e) $f(x) = \dfrac{\sin x}{1 + \cos x}$ for $x = 1.0001\pi$

4.13 Employing ideas from Sec. 4.2, derive the relationships from Table 4.3.

4.14 Prove that Eq. (4.4) is exact for all values of x if $f(x) = ax^2 + bx + c$.

4.15 Manning's formula for a rectangular channel can be written as

$$Q = \frac{1}{n}\frac{(BH)^{5/3}}{(B+2H)^{2/3}}\sqrt{S}$$

where Q = flow (m^3/s), n = a roughness coefficient, B = width (m), H = depth (m), and S = slope. You are applying this formula to a stream where you know that the width = 20 m and the depth = 0.3 m. Unfortunately, you know the roughness and the slope to only a $\pm$ 10% precision. That is, you know that the roughness is about 0.03 with a range from 0.027 to 0.033 and the slope is 0.0003 with a range from 0.00027 to 0.00033. Use a first-order

error analysis to determine the sensitivity of the flow prediction to each of these two factors. Which one should you attempt to measure with more precision?

4.16 If $|x| < 1$, it is known that

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

Repeat Prob. 4.1 for this series for $x = 0.1$.

4.17 A missile leaves the ground with an initial velocity v_0 forming an angle ϕ_0 with the vertical as shown in Fig. P4.17. The

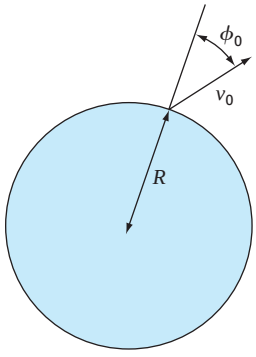


Figure P4.17

maximum desired altitude is αR where R is the radius of the earth. The laws of mechanics can be used to show that

$$\sin\phi_0 = (1+\alpha)\sqrt{1 - \frac{\alpha}{1+\alpha}\left(\frac{v_e}{v_0}\right)^2}$$

where v_e = the escape velocity of the missile. It is desired to fire the missile and reach the design maximum altitude within an accuracy of $\pm 2\%$. Determine the range of values for ϕ_0 if $v_e/v_0 = 2$ and $\alpha = 0.25$.

4.18 To calculate a planet's space coordinates, we have to solve the function

$$f(x) = x - 1 - 0.5\sin x$$

Let the base point be $a = x_i = \pi/2$ on the interval $[0, \pi]$. Determine the highest-order Taylor series expansion resulting in a maximum error of 0.015 on the specified interval. The error is equal to the absolute value of the difference between the given function and the specific Taylor series expansion. (Hint: Solve graphically.)

4.19 Consider the function $f(x) = x^3 - 2x + 4$ on the interval $[-2, 2]$ with $h = 0.25$. Use the forward, backward, and centered finite difference approximations for the first and second derivatives so as to graphically illustrate which approximation is most accurate. Graph all three first derivative finite difference approximations along with the theoretical, and do the same for the second derivative as well.

4.20 Derive Eq. (4.31).

4.21 Repeat Example 4.8, but for $f(x) = \cos(x)$ at $x = \pi/6$.

4.22 Repeat Example 4.8, but for the forward divided difference (Eq. 4.17).

4.23 Develop a well-structured program to compute the Maclaurin series expansion for the cosine function as described in Prob. 4.1. The function should have the following features:

- Iterate until the relative error falls below a stopping criterion (`es`) or exceeds a maximum number of iterations (`maxit`). Allow the user to specify values for these parameters.
- Include default values of `es` (= 0.000001) and `maxit` (= 100) in the event that they are not specified by the user.
- Return the estimate of cos(x), the approximate relative error, the number of iterations, and the true relative error (that you can calculate based on the built-in cosine function).