# ORDINARY DIFFERENTIAL EQUATIONS

## PT7.1 MOTIVATION

In the first chapter of this book, we derived the following equation based on Newton's second law to compute the velocity $v$ of a falling parachutist as a function of time $t$ [recall Eq. (1.9)]:

$$\frac{dv}{dt} = g - \frac{c}{m}v \tag{PT7.1}$$

where $g$ is the gravitational constant, $m$ is the mass, and $c$ is a drag coefficient. Such equations, which are composed of an unknown function and its derivatives, are called *differential equations.* Equation (PT7.1) is sometimes referred to as a *rate equation* because it expresses the rate of change of a variable as a function of variables and parameters. Such equations play a fundamental role in engineering because many physical phenomena are best formulated mathematically in terms of their rate of change.

In Eq. (PT7.1), the quantity being differentiated, $v$, is called the *dependent variable.* The quantity with respect to which $v$ is differentiated, $t$, is called the *independent variable.* When the function involves one independent variable, the equation is called an *ordinary differential equation* (or *ODE*). This is in contrast to a *partial differential equation* (or *PDE*) that involves two or more independent variables.

Differential equations are also classified as to their order. For example, Eq. (PT7.1) is called a *first-order equation* because the highest derivative is a first derivative. A *second-order equation* would include a second derivative. For example, the equation describing the position $x$ of a mass-spring system with damping is the second-order equation (recall Sec. 8.4),

$$m\frac{d^2x}{dt^2} + c\frac{dx}{dt} + kx = 0 \tag{PT7.2}$$

where $c$ is a damping coefficient and $k$ is a spring constant. Similarly, an $n$th-order equation would include an $n$th derivative.

Higher-order equations can be reduced to a system of first-order equations. For Eq. (PT7.2), this is done by defining a new variable $y$, where

$$y = \frac{dx}{dt} \tag{PT7.3}$$

which itself can be differentiated to yield

$$\frac{dy}{dt} = \frac{d^2x}{dt^2} \tag{PT7.4}$$

Equations (PT7.3) and (PT7.4) can then be substituted into Eq. (PT7.2) to give

$$m\frac{dy}{dt} + cy + kx = 0 \tag{PT7.5}$$

or

$$\frac{dy}{dt} = -\frac{cy + kx}{m} \tag{PT7.6}$$

Thus, Eqs. (PT7.3) and (PT7.6) are a pair of first-order equations that are equivalent to the original second-order equation. Because other *n*th-order differential equations can be similarly reduced, this part of our book focuses on the solution of first-order equations. Some of the engineering applications in Chap. 28 deal with the solution of second-order ODEs by reduction to a pair of first-order equations.

### PT7.1.1 Noncomputer Methods for Solving ODEs

Without computers, ODEs are usually solved with analytical integration techniques. For example, Eq. (PT7.1) could be multiplied by $dt$ and integrated to yield

$$v = \int \left(g - \frac{c}{m}v\right) dt \tag{PT7.7}$$

The right-hand side of this equation is called an *indefinite integral* because the limits of integration are unspecified. This is in contrast to the definite integrals discussed previously in Part Six [compare Eq. (PT7.7) with Eq. (PT6.6)].

An analytical solution for Eq. (PT7.7) is obtained if the indefinite integral can be evaluated exactly in equation form. For example, recall that for the falling parachutist problem, Eq. (PT7.7) was solved analytically by Eq. (1.10) (assuming $v = 0$ at $t = 0$):

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

The mechanics of deriving such analytical solutions will be discussed in Sec. PT7.2. For the time being, the important fact is that exact solutions for many ODEs of practical importance are not available. As is true for most situations discussed in other parts of this book, numerical methods offer the only viable alternative for these cases. Because these numerical methods usually require computers, engineers in the precomputer era were somewhat limited in the scope of their investigations.

One very important method that engineers and applied mathematicians developed to overcome this dilemma was *linearization.* A linear ordinary differential equation is one that fits the general form

$$a_n(x)y^{(n)} + \cdots + a_1(x)y' + a_0(x)y = f(x) \tag{PT7.8}$$

where $y^{(n)}$ is the *n*th derivative of $y$ with respect to $x$ and the $a$'s and $f$'s are specified functions of $x$. This equation is called *linear* because there are no products or nonlinear functions of the dependent variable $y$ and its derivatives. The practical importance of linear ODEs is that they can be solved analytically. In contrast, most nonlinear equations cannot

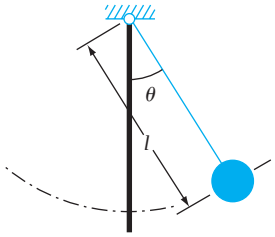



**FIGURE PT7.1**
The swinging pedulum.

be solved exactly. Thus, in the precomputer era, one tactic for solving nonlinear equations was to linearize them.

A simple example is the application of ODEs to predict the motion of a swinging pendulum (Fig. PT7.1). In a manner similar to the derivation of the falling parachutist problem, Newton's second law can be used to develop the following differential equation (see Sec. 28.4 for the complete derivation):

$$\frac{d^2\theta}{dt^2} + \frac{g}{l}\sin\theta = 0 \tag{PT7.9}$$

where $\theta$ is the angle of displacement of the pendulum, $g$ is the gravitational constant, and $l$ is the pendulum length. This equation is nonlinear because of the term $\sin\theta$. One way to obtain an analytical solution is to realize that for small displacements of the pendulum from equilibrium (that is, for small values of $\theta$),

$$\sin\theta \cong \theta \tag{PT7.10}$$

Thus, if it is assumed that we are interested only in cases where $\theta$ is small, Eq. (PT7.10) can be substituted into Eq. (PT7.9) to give

$$\frac{d^2\theta}{dt^2} + \frac{g}{l}\theta = 0 \tag{PT7.11}$$

We have, therefore, transformed Eq. (PT7.9) into a linear form that is easy to solve analytically.

Although linearization remains a very valuable tool for engineering problem solving, there are cases where it cannot be invoked. For example, suppose that we were interested in studying the behavior of the pendulum for large displacements from equilibrium. In such instances, numerical methods offer a viable option for obtaining solutions. Today, the widespread availability of computers places this option within reach of all practicing engineers.

### PT7.1.2 ODEs and Engineering Practice

The fundamental laws of physics, mechanics, electricity, and thermodynamics are usually based on empirical observations that explain variations in physical properties and states of systems. Rather than describing the state of physical systems directly, the laws are usually couched in terms of spatial and temporal changes.

Several examples are listed in Table PT7.1. These laws define mechanisms of change. When combined with continuity laws for energy, mass, or momentum, differential equations result. Subsequent integration of these differential equations results in mathematical functions that describe the spatial and temporal state of a system in terms of energy, mass, or velocity variations.

The falling parachutist problem introduced in Chap. 1 is an example of the derivation of an ordinary differential equation from a fundamental law. Recall that Newton's second law was used to develop an ODE describing the rate of change of velocity of a falling parachutist. By integrating this relationship, we obtained an equation to predict fall velocity as a function of time (Fig. PT7.2). This equation could be utilized in a number of different ways, including design purposes.

**TABLE PT7.1** Examples of fundamental laws that are written in terms of the rate of change of variables ($t$ = time and $x$ = position).

| Law | Mathematical Expression | Variables and Parameters |
|---|---|---|
| Newton's second law of motion | $\frac{dv}{dt} = \frac{F}{m}$ | Velocity ($v$), force ($F$), and mass ($m$) |
| Fourier's heat law | $q = -k' \frac{dT}{dx}$ | Heat flux ($q$), thermal conductivity ($k'$) and temperature ($T$) |
| Fick's law of diffusion | $J = -D \frac{dc}{dx}$ | Mass flux ($J$), diffusion coefficient ($D$), and concentration ($c$) |
| Faraday's law (voltage drop across an inductor) | $\Delta V_L = L \frac{di}{dt}$ | Voltage drop ($\Delta V_L$), inductance ($L$), and current ($i$) |

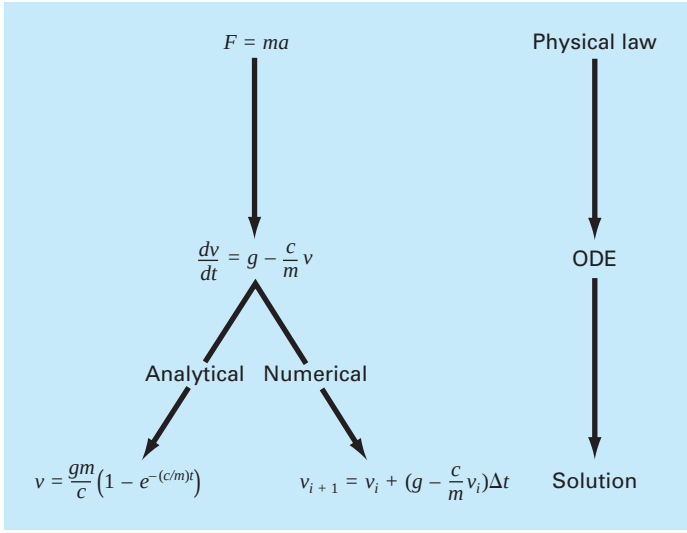



**FIGURE PT7.2**
The sequence of events in the application of ODEs for engineering problem solving. The example shown is the velocity of a falling parachutist.

In fact, such mathematical relationships are the basis of the solution for a great number of engineering problems. However, as described in the previous section, many of the differential equations of practical significance cannot be solved using the analytical methods of calculus. Thus, the methods discussed in the following chapters are extremely important in all fields of engineering.

## PT7.2 MATHEMATICAL BACKGROUND

A solution of an ordinary differential equation is a specific function of the independent variable and parameters that satisfies the original differential equation. To illustrate this concept, let us start with a given function

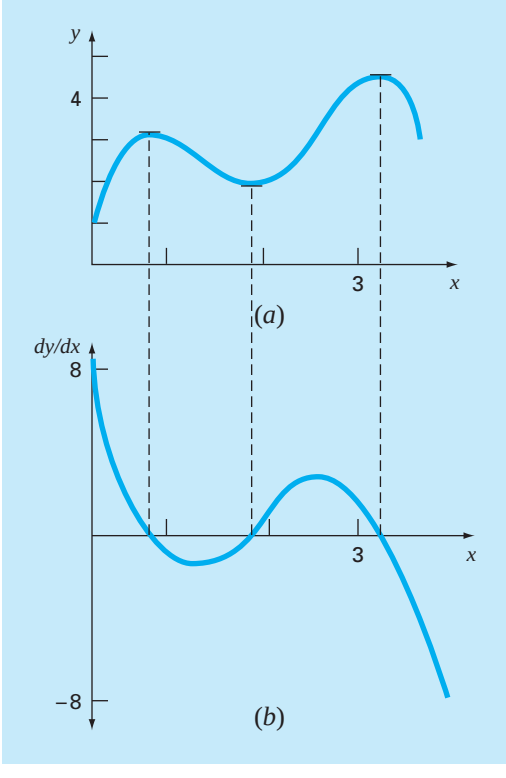
$$y = -0.5x^4 + 4x^3 - 10x^2 + 8.5x + 1 \tag{PT7.12}$$

which is a fourth-order polynomial (Fig. PT7.3*a*). Now, if we differentiate Eq. (PT7.12), we obtain an ODE:

$$\frac{dy}{dx} = -2x^3 + 12x^2 - 20x + 8.5 \tag{PT7.13}$$

This equation also describes the behavior of the polynomial, but in a manner different from Eq. (PT7.12). Rather than explicitly representing the values of $y$ for each value of $x$, Eq. (PT7.13) gives the rate of change of $y$ with respect to $x$ (that is, the slope) at every value of $x$. Figure PT7.3 shows both the function and the derivative plotted versus $x$. Notice how

**FIGURE PT7.3**
Plots of (*a*) $y$ versus $x$ and (*b*) $dy/dx$ versus $x$ for the function $y = -0.5x^4 + 4x^3 - 10x^2 + 8.5x + 1$.



the zero values of the derivatives correspond to the point at which the original function is flat—that is, has a zero slope. Also, the maximum absolute values of the derivatives are at the ends of the interval where the slopes of the function are greatest.

Although, as just demonstrated, we can determine a differential equation given the original function, the object here is to determine the original function given the differential equation. The original function then represents the solution. For the present case, we can determine this solution analytically by integrating Eq. (PT7.13):

$$y = \int (-2x^3 + 12x^2 - 20x + 8.5)\, dx$$

Applying the integration rule (recall Table PT6.2)

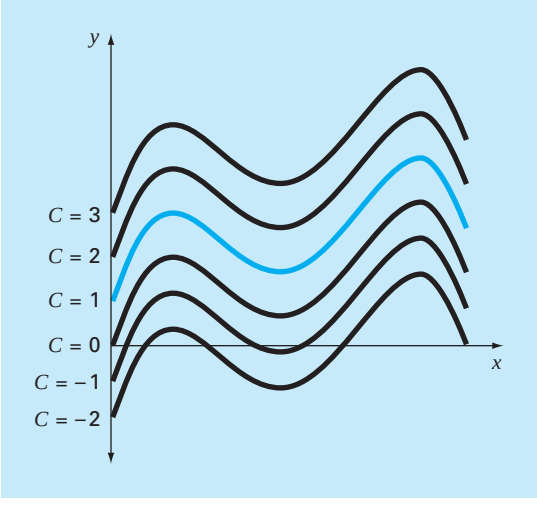
$$\int u^n\, du = \frac{u^{n+1}}{n+1} + C \qquad n \neq -1$$

to each term of the equation gives the solution

$$y = -0.5x^4 + 4x^3 - 10x^2 + 8.5x + C \tag{PT7.14}$$

which is identical to the original function with one notable exception. In the course of differentiating and then integrating, we lost the constant value of 1 in the original equation and gained the value $C$. This $C$ is called a *constant of integration.* The fact that such an arbitrary constant appears indicates that the solution is not unique. In fact, it is but one of an infinite number of possible functions (corresponding to an infinite number of possible values of $C$) that satisfy the differential equation. For example, Fig. PT7.4 shows six possible functions that satisfy Eq. (PT7.14).

**FIGURE PT7.4**
Six possible solutions for the integral of $-2x^3 + 12x^2 - 20x + 8.5$. Each conforms to a different value of the constant of integration $C$.



Therefore, to specify the solution completely, a differential equation is usually accompanied by *auxiliary conditions.* For first-order ODEs, a type of auxiliary condition called an *initial value* is required to determine the constant and obtain a unique solution. For example, Eq. (PT7.13) could be accompanied by the initial condition that at $x = 0$, $y = 1$. These values could be substituted into Eq. (PT7.14):

$$1 = -0.5(0)^4 + 4(0)^3 - 10(0)^2 + 8.5(0) + C \tag{PT7.15}$$

to determine $C = 1$. Therefore, the unique solution that satisfies both the differential equation and the specified initial condition is obtained by substituting $C = 1$ into Eq. (PT7.14) to yield

$$y = -0.5x^4 + 4x^3 - 10x^2 + 8.5x + 1 \tag{PT7.16}$$

Thus, we have "pinned down'' Eq. (PT7.14) by forcing it to pass through the initial condition, and in so doing, we have developed a unique solution to the ODE and have come full circle to the original function [Eq. (PT7.12)].

Initial conditions usually have very tangible interpretations for differential equations derived from physical problem settings. For example, in the falling parachutist problem, the initial condition was reflective of the physical fact that at time zero the vertical velocity was zero. If the parachutist had already been in vertical motion at time zero, the solution would have been modified to account for this initial velocity.

When dealing with an *n*th-order differential equation, *n* conditions are required to obtain a unique solution. If all conditions are specified at the same value of the independent variable (for example, at $x$ or $t = 0$), then the problem is called an *initial-value problem.* This is in contrast to *boundary-value problems* where specification of conditions occurs at different values of the independent variable. Chapters 25 and 26 will focus on initial-value problems. Boundary-value problems are covered in Chap. 27 along with eigenvalues.

## PT7.3 ORIENTATION

Before proceeding to numerical methods for solving ordinary differential equations, some orientation might be helpful. The following material is intended to provide you with an overview of the material discussed in Part Seven. In addition, we have formulated objectives to focus your studies of the subject area.

### PT7.3.1 Scope and Preview

Figure PT7.5 provides an overview of Part Seven. Two broad categories of numerical methods for initial-value problems will be discussed in this part of this book. One-step methods, which are covered in Chap. 25, permit the calculation of $y_{i+1}$, given the differential equation and $y_i$. Multistep methods, which are covered in Chap. 26, require additional values of $y$ other than at $i$.

With all but a minor exception, the *one-step methods* in *Chap. 25* belong to what are called Runge-Kutta techniques. Although the chapter might have been organized around this theoretical notion, we have opted for a more graphical, intuitive approach to introduce the methods. Thus, we begin the chapter with *Euler's method,* which has a very straightforward graphical interpretation. Then, we use visually oriented arguments to develop two

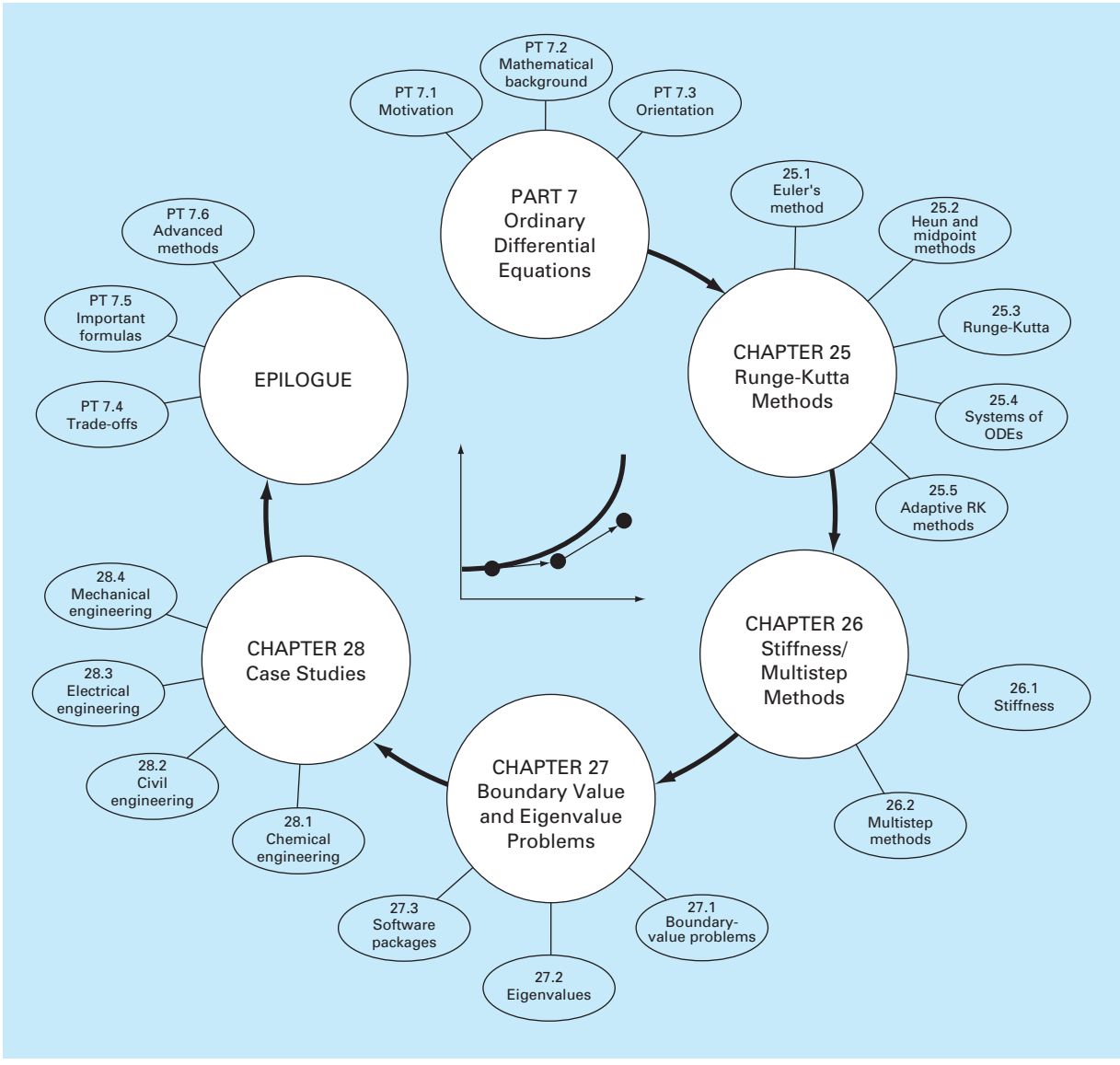



**FIGURE PT7.5**
Schematic representation of the organization of Part Seven: Ordinary Differential Equations.

improved versions of Euler's method—the *Heun* and the *midpoint* techniques. After this introduction, we formally develop the concept of *Runge-Kutta* (or *RK*) approaches and demonstrate how the foregoing techniques are actually first- and second-order RK methods. This is followed by a discussion of the higher-order RK formulations that are frequently used for engineering problem solving. In addition, we cover the application of one-step methods to *systems of ODEs*. Finally, the chapter ends with a discussion of *adaptive RK methods* that automatically adjust the step size in response to the truncation error of the computation.

*Chapter 26* starts with a description of *stiff ODEs*. These are both individual and systems of ODEs that have both fast and slow components to their solution. We introduce the idea of an *implicit solution* technique as one commonly used remedy for this problem.

Next, we discuss *multistep methods*. These algorithms retain information of previous steps to more effectively capture the trajectory of the solution. They also yield the truncation error estimates that can be used to implement step-size control. In this section, we initially take a visual, intuitive approach by using a simple method—the *non-self-starting Heun*—to introduce all the essential features of the multistep approaches.

In *Chap. 27* we turn to *boundary-value* and *eigenvalue* problems. For the former, we introduce both *shooting* and *finite-difference methods*. For the latter, we discuss several approaches, including the *polynomial* and the *power methods*. Finally, the chapter concludes with a description of the application of several *software packages* and *libraries* for solution of ODEs and eigenvalues.

*Chapter 28* is devoted to applications from all the fields of engineering. Finally, a short review section is included at the end of Part Seven. This epilogue summarizes and compares the important formulas and concepts related to ODEs. The comparison includes a discussion of trade-offs that are relevant to their implementation in engineering practice. The epilogue also summarizes important formulas and includes references for advanced topics.

### PT7.3.2 Goals and Objectives

Study Objectives. After completing Part Seven, you should have greatly enhanced your capability to confront and solve ordinary differential equations and eigenvalue problems. General study goals should include mastering the techniques, having the capability to assess the reliability of the answers, and being able to choose the "best" method (or methods) for any particular problem. In addition to these general objectives, the specific study objectives in Table PT7.2 should be mastered.

Computer Objectives. Algorithms are provided for many of the methods in Part Seven. This information will allow you to expand your software library. For example, you may find it useful from a professional viewpoint to have software that employs the fourth-order Runge-Kutta method for more than five equations and to solve ODEs with an adaptive step-size approach.

In addition, one of your most important goals should be to master several of the general-purpose software packages that are widely available. In particular, you should become adept at using these tools to implement numerical methods for engineering problem solving.

**TABLE PT7.2** Specific study objectives for Part Seven.

1. Understand the visual representations of Euler's, Heun's, and the midpoint methods
2. Know the relationship of Euler's method to the Taylor series expansion and the insight it provides regarding the error of the method
3. Understand the difference between local and global truncation errors and how they relate to the choice of a numerical method for a particular problem
4. Know the order and the step-size dependency of the global truncation errors for all the methods described in Part Seven; understand how these errors bear on the accuracy of the techniques
5. Understand the basis of predictor-corrector methods; in particular, realize that the efficiency of the corrector is highly dependent on the accuracy of the predictor
6. Know the general form of the Runge-Kutta methods; understand the derivation of the second-order RK method and how it relates to the Taylor series expansion; realize that there are an infinite number of possible versions for second- and higher-order RK methods
7. Know how to apply any of the RK methods to systems of equations; be able to reduce an *n*th-order ODE to a system of *n* first-order ODEs
8. Recognize the type of problem context where step size adjustment is important
9. Understand how adaptive step size control is integrated into a fourth-order RK method
10. Recognize how the combination of slow and fast components makes an equation or a system of equations stiff
11. Understand the distinction between explicit and implicit solution schemes for ODEs; in particular, recognize how the latter (1) ameliorates the stiffness problem and (2) complicates the solution mechanics
12. Understand the difference between initial-value and boundary-value problems
13. Know the difference between multistep and one-step methods; realize that all multistep methods are predictor-correctors but that not all predictor-correctors are multistep methods
14. Understand the connection between integration formulas and predictor-corrector methods
15. Recognize the fundamental difference between Newton-Cotes and Adams integration formulas
16. Know the rationale behind the polynomial and the power methods for determining eigenvalues; in particular, recognize their strengths and limitations
17. Understand how Hoteller's deflation allows the power method to be used to compute intermediate eigenvalues
18. Know how to use software packages and/or libraries to integrate ODEs and evaluate eigenvalues

# PART SEVEN

CHAPTER 25

# Runge-Kutta Methods

This chapter is devoted to solving ordinary differential equations of the form

$$\frac{dy}{dx} = f(x, y)$$

In Chap. 1, we used a numerical method to solve such an equation for the velocity of the falling parachutist. Recall that the method was of the general form

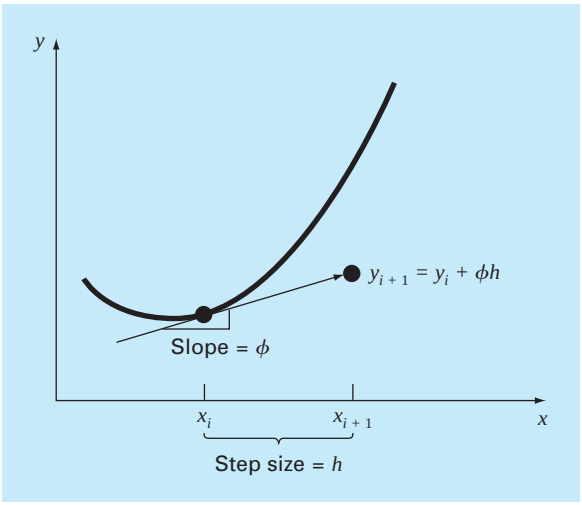
$$\text{New value} = \text{old value} + \text{slope} \times \text{step size}$$

or, in mathematical terms,

$$y_{i+1} = y_i + \phi h \tag{25.1}$$

According to this equation, the slope estimate of $\phi$ is used to extrapolate from an old value $y_i$ to a new value $y_{i+1}$ over a distance $h$ (Fig. 25.1). This formula can be applied step by step to compute out into the future and, hence, trace out the trajectory of the solution.

**FIGURE 25.1**
Graphical depiction of a one-step method.



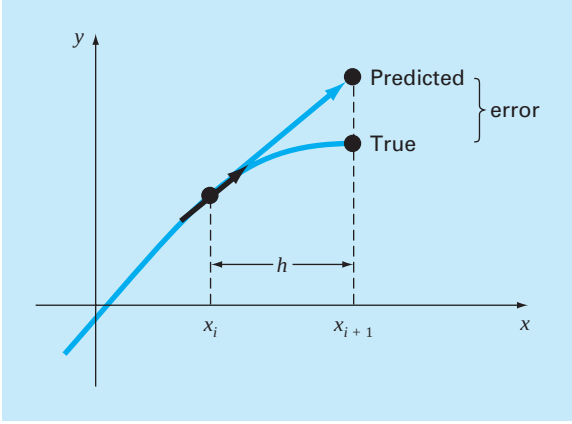



**FIGURE 25.2**
Euler's method.

All one-step methods can be expressed in this general form, with the only difference being the manner in which the slope is estimated. As in the falling parachutist problem, the simplest approach is to use the differential equation to estimate the slope in the form of the first derivative at $x_i$. In other words, the slope at the beginning of the interval is taken as an approximation of the average slope over the whole interval. This approach, called *Euler's method*, is discussed in the first part of this chapter. This is followed by other one-step methods that employ alternative slope estimates that result in more accurate predictions. All these techniques are generally called *Runge-Kutta* methods.

## 25.1 EULER'S METHOD

The first derivative provides a direct estimate of the slope at $x_i$ (Fig. 25.2):

$$\phi = f(x_i, y_i)$$

where $f(x_i, y_i)$ is the differential equation evaluated at $x_i$ and $y_i$. This estimate can be substituted into Eq. (25.1):

$$y_{i+1} = y_i + f(x_i, y_i)h \tag{25.2}$$

This formula is referred to as *Euler's* (or the *Euler-Cauchy* or the *point-slope*) *method*. A new value of $y$ is predicted using the slope (equal to the first derivative at the original value of $x$) to extrapolate linearly over the step size $h$ (Fig. 25.2).

EXAMPLE 25.1 Euler's Method

Problem Statement. Use Euler's method to numerically integrate Eq. (PT7.13):

$$\frac{dy}{dx} = -2x^3 + 12x^2 - 20x + 8.5$$

from $x = 0$ to $x = 4$ with a step size of 0.5. The initial condition at $x = 0$ is $y = 1$. Recall that the exact solution is given by Eq. (PT7.16):

$$y = -0.5x^4 + 4x^3 - 10x^2 + 8.5x + 1$$

Solution. Equation (25.2) can be used to implement Euler's method:

$$y(0.5) = y(0) + f(0, 1)0.5$$

where $y(0) = 1$ and the slope estimate at $x = 0$ is

$$f(0, 1) = -2(0)^3 + 12(0)^2 - 20(0) + 8.5 = 8.5$$

Therefore,

$$y(0.5) = 1.0 + 8.5(0.5) = 5.25$$

The true solution at $x = 0.5$ is

$$y = -0.5(0.5)^4 + 4(0.5)^3 - 10(0.5)^2 + 8.5(0.5) + 1 = 3.21875$$

Thus, the error is

$$E_t = \text{true} - \text{approximate} = 3.21875 - 5.25 = -2.03125$$

or, expressed as percent relative error, $\varepsilon_t = -63.1\%$. For the second step,

$$\begin{aligned} y(1) &= y(0.5) + f(0.5, 5.25)0.5 \\ &= 5.25 + [-2(0.5)^3 + 12(0.5)^2 - 20(0.5) + 8.5]0.5 \\ &= 5.875 \end{aligned}$$

The true solution at $x = 1.0$ is 3.0, and therefore, the percent relative error is $-95.8\%$. The computation is repeated, and the results are compiled in Table 25.1 and Fig. 25.3. Note that,

**TABLE 25.1** Comparison of true and approximate values of the integral of $y' = -2x^3 + 12x^2 - 20x + 8.5$, with the initial condition that $y = 1$ at $x = 0$. The approximate values were computed using Euler's method with a step size of 0.5. The local error refers to the error incurred over a single step. It is calculated with a Taylor series expansion as in Example 25.2. The global error is the total discrepancy due to past as well as present steps.

| | | | Percent Relative Error | |
|---|---|---|---|---|
| $x$ | $y_{true}$ | $y_{Euler}$ | Global | Local |
| 0.0 | 1.00000 | 1.00000 | | |
| 0.5 | 3.21875 | 5.25000 | −63.1 | −63.1 |
| 1.0 | 3.00000 | 5.87500 | −95.8 | −28.0 |
| 1.5 | 2.21875 | 5.12500 | 131.0 | −1.41 |
| 2.0 | 2.00000 | 4.50000 | −125.0 | 20.5 |
| 2.5 | 2.71875 | 4.75000 | −74.7 | 17.3 |
| 3.0 | 4.00000 | 5.87500 | 46.9 | 4.0 |
| 3.5 | 4.71875 | 7.12500 | −51.0 | −11.3 |
| 4.0 | 3.00000 | 7.00000 | −133.3 | −53.0 |

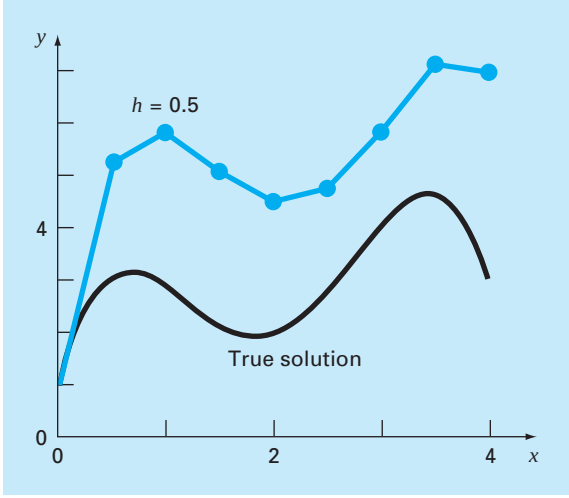



**FIGURE 25.3**
Comparison of the true solution with a numerical solution using Euler's method for the integral of $y' = -2x^3 + 12x^2 - 20x + 8.5$ from $x = 0$ to $x = 4$ with a step size of 0.5. The initial condition at $x = 0$ is $y = 1$.

although the computation captures the general trend of the true solution, the error is considerable. As discussed in the next section, this error can be reduced by using a smaller step size.

The preceding example uses a simple polynomial for the differential equation to facilitate the error analyses that follow. Thus,

$$\frac{dy}{dx} = f(x)$$

Obviously, a more general (and more common) case involves ODEs that depend on both $x$ and $y$,

$$\frac{dy}{dx} = f(x, y)$$

As we progress through this part of the text, our examples will increasingly involve ODEs that depend on both the independent and the dependent variables.

### 25.1.1 Error Analysis for Euler's Method

The numerical solution of ODEs involves two types of error (recall Chaps. 3 and 4):

1. *Truncation*, or discretization, errors caused by the nature of the techniques employed to approximate values of $y$.

**2.** *Round-off* errors caused by the limited numbers of significant digits that can be retained by a computer.

The truncation errors are composed of two parts. The first is a *local truncation error* that results from an application of the method in question over a single step. The second is a *propagated truncation error* that results from the approximations produced during the previous steps. The sum of the two is the total, or *global truncation, error.*

Insight into the magnitude and properties of the truncation error can be gained by deriving Euler's method directly from the Taylor series expansion. To do this, realize that the differential equation being integrated will be of the general form

$$y' = f(x, y) \tag{25.3}$$

where $y' = dy/dx$ and $x$ and $y$ are the independent and the dependent variables, respectively. If the solution—that is, the function describing the behavior of $y$—has continuous derivatives, it can be represented by a Taylor series expansion about a starting value $(x_i, y_i)$, as in [recall Eq. (4.7)]

$$y_{i+1} = y_i + y_i' h + \frac{y_i''}{2!}h^2 + \cdots + \frac{y_i^{(n)}}{n!}h^n + R_n \tag{25.4}$$

where $h = x_{i+1} - x_i$ and $R_n =$ the remainder term, defined as

$$R_n = \frac{y^{(n+1)}(\xi)}{(n+1)!}h^{n+1} \tag{25.5}$$

where $\xi$ lies somewhere in the interval from $x_i$ to $x_{i+1}$. An alternative form can be developed by substituting Eq. (25.3) into Eqs. (25.4) and (25.5) to yield

$$y_{i+1} = y_i + f(x_i, y_i)h + \frac{f'(x_i, y_i)}{2!}h^2 + \cdots + \frac{f^{(n-1)}(x_i, y_i)}{n!}h^n + O(h^{n+1}) \tag{25.6}$$

where $O(h^{n+1})$ specifies that the local truncation error is proportional to the step size raised to the $(n + 1)$th power.

By comparing Eqs. (25.2) and (25.6), it can be seen that Euler's method corresponds to the Taylor series up to and including the term $f(x_i, y_i)h$. Additionally, the comparison indicates that a truncation error occurs because we approximate the true solution using a finite number of terms from the Taylor series. We thus truncate, or leave out, a part of the true solution. For example, the truncation error in Euler's method is attributable to the remaining terms in the Taylor series expansion that were not included in Eq. (25.2). Subtracting Eq. (25.2) from Eq. (25.6) yields

$$E_t = \frac{f'(x_i, y_i)}{2!}h^2 + \cdots + O(h^{n+1}) \tag{25.7}$$

where $E_t =$ the true local truncation error. For sufficiently small $h$, the errors in the terms in Eq. (25.7) usually decrease as the order increases (recall Example 4.2 and the accompanying discussion), and the result is often represented as

$$E_a = \frac{f'(x_i, y_i)}{2!}h^2 \tag{25.8}$$

or

$$E_a = O(h^2) \tag{25.9}$$

where $E_a$ = the approximate local truncation error.

## EXAMPLE 25.2 Taylor Series Estimate for the Error of Euler's Method

**Problem Statement.** Use Eq. (25.7) to estimate the error of the first step of Example 25.1. Also use it to determine the error due to each higher-order term of the Taylor series expansion.

**Solution.** Because we are dealing with a polynomial, we can use the Taylor series to obtain exact estimates of the errors in Euler's method. Equation (25.7) can be written as

$$E_t = \frac{f'(x_i, y_i)}{2!}h^2 + \frac{f''(x_i, y_i)}{3!}h^3 + \frac{f^{(3)}(x_i, y_i)}{4!}h^4 \tag{E25.2.1}$$

where $f'(x_i, y_i)$ = the first derivative of the differential equation (that is, the second derivative of the solution). For the present case, this is

$$f'(x_i, y_i) = -6x^2 + 24x - 20 \tag{E25.2.2}$$

and $f''(x_i, y_i)$ is the second derivative of the ODE

$$f''(x_i, y_i) = -12x + 24 \tag{E25.2.3}$$

and $f^{(3)}(x_i, y_i)$ is the third derivative of the ODE

$$f^{(3)}(x_i, y_i) = -12 \tag{E25.2.4}$$

We can omit additional terms (that is, fourth derivatives and higher) from Eq. (E25.2.1) because for this particular case they equal zero. It should be noted that for other functions (for example, transcendental functions such as sinusoids or exponentials) this would not necessarily be true, and higher-order terms would have nonzero values. However, for the present case, Eqs. (E25.2.1) through (E25.2.4) completely define the truncation error for a single application of Euler's method.

For example, the error due to truncation of the second-order term can be calculated as

$$E_{t,2} = \frac{-6(0.0)^2 + 24(0.0) - 20}{2}(0.5)^2 = -2.5 \tag{E25.2.5}$$

For the third-order term:

$$E_{t,3} = \frac{-12(0.0) + 24}{6}(0.5)^3 = 0.5$$

and the fourth-order term:

$$E_{t,4} = \frac{-12}{24}(0.5)^4 = -0.03125$$

These three results can be added to yield the total truncation error:

$$E_t = E_{t,2} + E_{t,3} + E_{t,4} = -2.5 + 0.5 - 0.03125 = -2.03125$$

which is exactly the error that was incurred in the initial step of Example 25.1. Note how $E_{t,2} > E_{t,3} > E_{t,4}$, which supports the approximation represented by Eq. (25.8).

As illustrated in Example 25.2, the Taylor series provides a means of quantifying the error in Euler's method. However, there are limitations associated with its use for this purpose:

1. The Taylor series provides only an estimate of the local truncation error—that is, the error created during a single step of the method. It does not provide a measure of the propagated and, hence, the global truncation error. In Table 25.1, we have included the local and global truncation errors for Example 25.1. The local error was computed for each time step with Eq. (25.2) but using the true value of $y_i$ (the second column of the table) to compute each $y_{i+1}$ rather than the approximate value (the third column), as is done in the Euler method. As expected, the average absolute local truncation error (25 percent) is less than the average global error (90 percent). The only reason that we can make these exact error calculations is that we know the true value a priori. Such would not be the case in an actual problem. Consequently, as discussed below, you must usually apply techniques such as Euler's method using a number of different step sizes to obtain an indirect estimate of the errors involved.
2. As mentioned above, in actual problems we usually deal with functions that are more complicated than simple polynomials. Consequently, the derivatives that are needed to evaluate the Taylor series expansion would not always be easy to obtain.

Although these limitations preclude exact error analysis for most practical problems, the Taylor series still provides valuable insight into the behavior of Euler's method. According to Eq. (25.9), we see that the local error is proportional to the square of the step size and the first derivative of the differential equation. It can also be demonstrated that the global truncation error is $O(h)$, that is, it is proportional to the step size (Carnahan et al. 1969). These observations lead to some useful conclusions:

1. The error can be reduced by decreasing the step size.
2. The method will provide error-free predictions if the underlying function (that is, the solution of the differential equation) is linear, because for a straight line the second derivative would be zero.

This latter conclusion makes intuitive sense because Euler's method uses straight-line segments to approximate the solution. Hence, Euler's method is referred to as a *first-order method.*

It should also be noted that this general pattern holds for the higher-order one-step methods described in the following pages. That is, an $n$th-order method will yield perfect results if the underlying solution is an $n$th-order polynomial. Further, the local truncation error will be $O(h^{n+1})$ and the global error $O(h^n)$.

## EXAMPLE 25.3 Effect of Reduced Step Size on Euler's Method

**Problem Statement.** Repeat the computation of Example 25.1 but use a step size of 0.25.

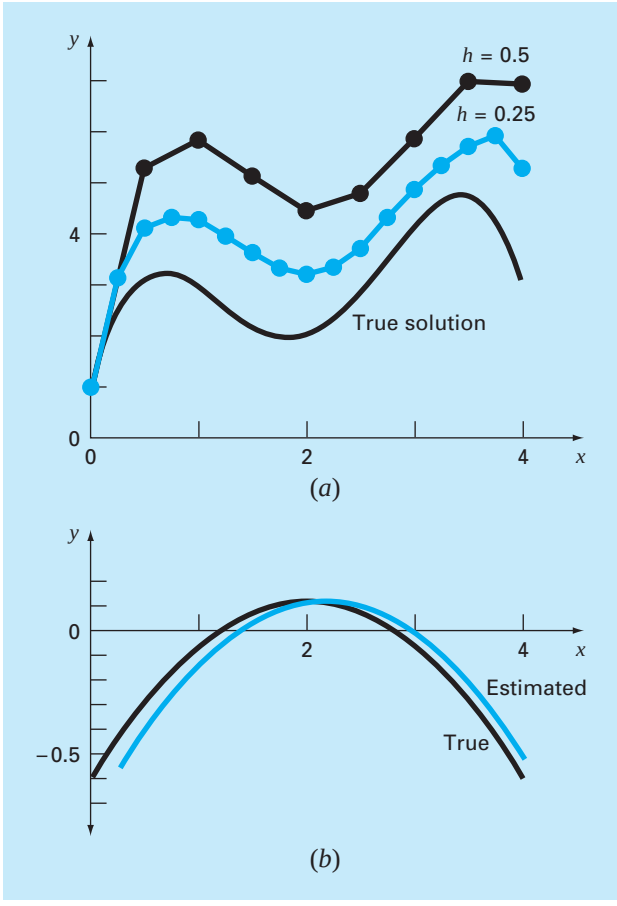



**FIGURE 25.4**
(*a*) Comparison of two numerical solutions with Euler's method using step sizes of 0.5 and 0.25. (*b*) Comparison of true and estimated local truncation error for the case where the step size is 0.5. Note that the "estimated" error is based on Eq. (E25.2.5).

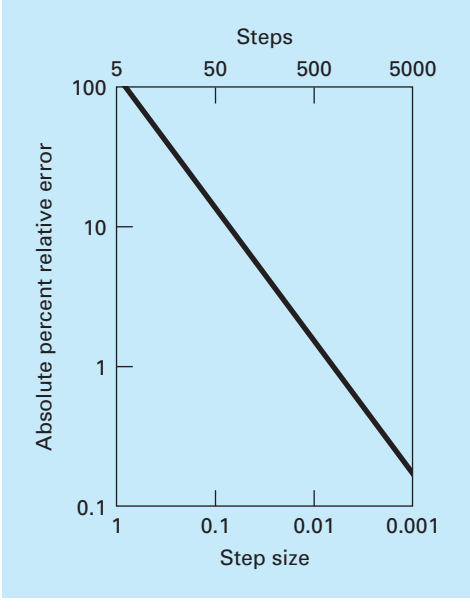
Solution. The computation is repeated, and the results are compiled in Fig. 25.4*a*. Halving the step size reduces the absolute value of the average global error to 40 percent and the absolute value of the local error to 6.4 percent. This is compared to global and local errors for Example 25.1 of 90 percent and 24.8 percent, respectively. Thus, as expected, the local error is quartered and the global error is halved.

Also, notice how the local error changes sign for intermediate values along the range. This is due primarily to the fact that the first derivative of the differential equation is a parabola that changes sign [recall Eq. (E25.2.2) and see Fig. 25.4*b*]. Because the local error is proportional to this function, the net effect of the oscillation in sign is to keep the global error from continuously growing as the calculation proceeds. Thus, from $x = 0$ to $x = 1.25$, the local errors are all negative, and consequently, the global error increases over this

interval. In the intermediate section of the range, positive local errors begin to reduce the global error. Near the end, the process is reversed and the global error again inflates. If the local error continuously changes sign over the computation interval, the net effect is usually to reduce the global error. However, where the local errors are of the same sign, the numerical solution may diverge farther and farther from the true solution as the computation proceeds. Such results are said to be *unstable*.

The effect of further step-size reductions on the global truncation error of Euler's method is illustrated in Fig. 25.5. This plot shows the absolute percent relative error at $x = 5$ as a function of step size for the problem we have been examining in Examples 25.1 through 25.3. Notice that even when $h$ is reduced to 0.001, the error still exceeds 0.1 percent. Because this step size corresponds to 5000 steps to proceed from $x = 0$ to $x = 5$, the plot suggests that a first-order technique such as Euler's method demands great computational effort to obtain acceptable error levels. Later in this chapter, we present higher-order techniques that attain much better accuracy for the same computational effort. However, it should be noted that, despite its inefficiency, the simplicity of Euler's method makes it an extremely attractive option for many engineering problems. Because it is very easy to program, the technique is particularly useful for quick analyses. In the next section, a computer algorithm for Euler's method is developed.

**FIGURE 25.5**
Effect of step size on the global truncation error of Euler's method for the integral of $y' = -2x^3 + 12x^2 - 20x + 8.5$. The plot shows the absolute percent relative global error at $x = 5$ as a function of step size.



### 25.1.2 Algorithm for Euler's Method

Algorithms for one-step techniques such as Euler's method are extremely simple to program. As specified previously at the beginning of this chapter, all one-step methods have the general form

$$\text{New value} = \text{old value} + \text{slope} \times \text{step size} \tag{25.10}$$

The only way in which the methods differ is in the calculation of the slope.

Suppose that you want to perform the simple calculation outlined in Table 25.1. That is, you would like to use Euler's method to integrate $y' = -2x^3 + 12x^2 - 20x + 8.5$, with the initial condition that $y = 1$ at $x = 0$. You would like to integrate out to $x = 4$ using a step size of 0.5, and display all the results. A simple pseudocode to accomplish this task could be written as in Fig. 25.6.

Although this program will "do the job" of duplicating the results of Table 25.1, it is not very well designed. First, and foremost, it is not very modular. Although this is not very important for such a small program, it would be critical if we desired to modify and improve the algorithm.

Further, there are a number of issues related to the way we have set up the iterations. For example, suppose that the step size were to be made very small to obtain better accuracy. In such cases, because every computed value is displayed, the number of output values might be very large. Further, the algorithm is predicated on the assumption that the calculation interval is evenly divisible by the step size. Finally, the accumulation of $x$ in the line $x = x + dx$ can be subject to quantizing errors of the sort previously discussed in

**FIGURE 25.6**
Pseudocode for a "dumb" version of Euler's method.

```
'set integration range
xi = 0
xf = 4
'initialize variables
x = xi
y = 1
'set step size and determine
'number of calculation steps
dx = 0.5
nc = (xf − xi)/dx
'output initial condition
PRINT x, y
'loop to implement Euler's method
'and display results
DOFOR i = 1, nc
  dydx = −2x^3 + 12x^2 − 20x + 8.5
  y = y + dydx · dx
  x = x + dx
  PRINT x, y
END DO
```

Sec. 3.4.1. For example, if $dx$ were changed to 0.01 and standard IEEE floating point representation were used (about seven significant digits), the result at the end of the calculation would be 3.999997 rather than 4. For $dx = 0.001$, it would be 3.999892!

A much more modular algorithm that avoids these difficulties is displayed in Fig. 25.7. The algorithm does not output all calculated values. Rather, the user specifies an output interval, *xout*, that dictates the interval at which calculated results are stored in arrays, $xp_m$ and $yp_m$. These values are stored in arrays so that they can be output in a variety of ways after the computation is completed (for example, printed, graphed, or written to a file).

The Driver Program takes big output steps and calls an Integrator routine that takes finer calculation steps. Note that the loops controlling both large and small steps exit on logical conditions. Thus, the intervals do not have to be evenly divisible by the step sizes.

The Integrator routine then calls an Euler routine that takes a single step with Euler's method. The Euler routine calls a Derivative routine that calculates the derivative value.

Whereas such modularization might seem like overkill for the present case, it will greatly facilitate modifying the program in later sections. For example, although the program in Fig. 25.7 is specifically designed to implement Euler's method, the Euler module is the only part that is method-specific. Thus, all that is required to apply this algorithm to the other one-step methods is to modify this routine.

**FIGURE 25.7**
Pseudocode for an "improved" modular version of Euler's method.

**(*a*) Main or "Driver" Program**

```
Assign values for
y = initial value dependent variable
xi = initial value independent variable
xf = final value independent variable
dx = calculation step size
xout = output interval

x = xi
m = 0
xp_m = x
yp_m = y
DO
  xend = x + xout
  IF (xend > xf) THEN xend = xf
  h = dx
  CALL Integrator (x, y, h, xend)
  m = m + 1
  xp_m = x
  yp_m = y
  IF (x ≥ xf) EXIT
END DO
DISPLAY RESULTS
END
```

**(*b*) Routine to Take One Output Step**

```
SUB Integrator (x, y, h, xend)
  DO
    IF (xend − x < h) THEN h = xend − x
    CALL Euler (x, y, h, ynew)
    y = ynew
    IF (x ≥ xend) EXIT
  END DO
END SUB
```

**(*c*) Euler's Method for a Single ODE**

```
SUB Euler (x, y, h, ynew)
  CALL Derivs(x, y, dydx)
  ynew = y + dydx * h
  x = x + h
END SUB
```

**(*d*) Routine to Determine Derivative**

```
SUB Derivs (x, y, dydx)
  dydx = ...
END SUB
```

## EXAMPLE 25.4 Solving ODEs with the Computer

**Problem Statement.** A computer program can be developed from the pseudocode in Fig. 25.7. We can use this software to solve another problem associated with the falling parachutist. You recall from Part One that our mathematical model for the velocity was based on Newton's second law in the form

$$\frac{dv}{dt} = g - \frac{c}{m}v \tag{E25.4.1}$$

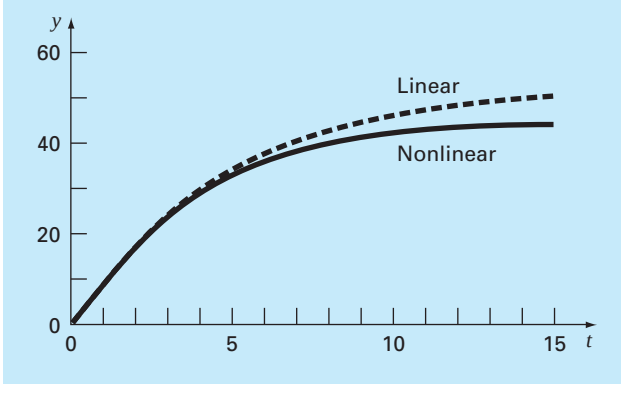
This differential equation was solved both analytically (Example 1.1) and numerically using Euler's method (Example 1.2). These solutions were for the case where $g = 9.8$, $c = 12.5$, $m = 68.1$, and $v = 0$ at $t = 0$.

The objective of the present example is to repeat these numerical computations employing a more complicated model for the velocity based on a more complete mathematical description of the drag force caused by wind resistance. This model is given by

$$\frac{dv}{dt} = g - \frac{c}{m}\left[v + a\left(\frac{v}{v_{\max}}\right)^b\right] \tag{E25.4.2}$$

where $g$, $m$, and $c$ are the same as for Eq. (E25.4.1), and $a$, $b$, and $v_{\max}$ are empirical constants, which for this case are equal to 8.3, 2.2, and 46, respectively. Note that this model is more capable of accurately fitting empirical measurements of drag forces versus velocity than is the simple linear model of Example 1.1. However, this increased flexibility is gained at the expense of evaluating three coefficients rather than one. Furthermore, the resulting mathematical model is more difficult to solve analytically. In this case, Euler's method provides a convenient alternative to obtain an approximate numerical solution.

**FIGURE 25.8**
Graphical results for the solution of the nonlinear ODE [Eq. (E25.4.2)]. Notice that the plot also shows the solution for the linear model [Eq. (E25.4.1)] for comparative purposes.



Solution. The results for both the linear and nonlinear model are displayed in Fig. 25.8 with an integration step size of 0.1 s. The plot in Fig. 25.8 also shows an overlay of the solution of the linear model for comparison purposes.

The results of the two simulations indicate how increasing the complexity of the formulation of the drag force affects the velocity of the parachutist. In this case, the terminal velocity is lowered because of resistance caused by the higher-order terms in Eq. (E25.4.2).

Alternative models could be tested in a similar fashion. The combination of a computer-generated solution makes this an easy and efficient task. This convenience should allow you to devote more of your time to considering creative alternatives and holistic aspects of the problem rather than to tedious manual computations.

### 25.1.3 Higher-Order Taylor Series Methods

One way to reduce the error of Euler's method would be to include higher-order terms of the Taylor series expansion in the solution. For example, including the second-order term from Eq. (25.6) yields

$$y_{i+1} = y_i + f(x_i, y_i)h + \frac{f'(x_i, y_i)}{2!}h^2 \tag{25.11}$$

with a local truncation error of

$$E_a = \frac{f''(x_i, y_i)}{6}h^3$$

Although the incorporation of higher-order terms is simple enough to implement for polynomials, their inclusion is not so trivial when the ODE is more complicated. In particular, ODEs that are a function of both the dependent and independent variable require chain-rule differentiation. For example, the first derivative of $f(x, y)$ is

$$f'(x_i, y_i) = \frac{\partial f(x, y)}{\partial x} + \frac{\partial f(x, y)}{\partial y}\frac{dy}{dx}$$

The second derivative is

$$f''(x_i, y_i) = \frac{\partial[\partial f/\partial x + (\partial f/\partial y)(dy/dx)]}{\partial x} + \frac{\partial[\partial f/\partial x + (\partial f/\partial y)(dy/dx)]}{\partial y}\frac{dy}{dx}$$

Higher-order derivatives become increasingly more complicated.

Consequently, as described in the following sections, alternative one-step methods have been developed. These schemes are comparable in performance to the higher-order Taylor-series approaches but require only the calculation of first derivatives.

## 25.2 IMPROVEMENTS OF EULER'S METHOD

A fundamental source of error in Euler's method is that the derivative at the beginning of the interval is assumed to apply across the entire interval. Two simple modifications are available to help circumvent this shortcoming. As will be demonstrated in Sec. 25.3, both modifications actually belong to a larger class of solution techniques called Runge-Kutta

methods. However, because they have a very straightforward graphical interpretation, we will present them prior to their formal derivation as Runge-Kutta methods.

### 25.2.1 Heun's Method

One method to improve the estimate of the slope involves the determination of two derivatives for the interval—one at the initial point and another at the end point. The two derivatives are then averaged to obtain an improved estimate of the slope for the entire interval. This approach, called *Heun's method*, is depicted graphically in Fig. 25.9.

Recall that in Euler's method, the slope at the beginning of an interval

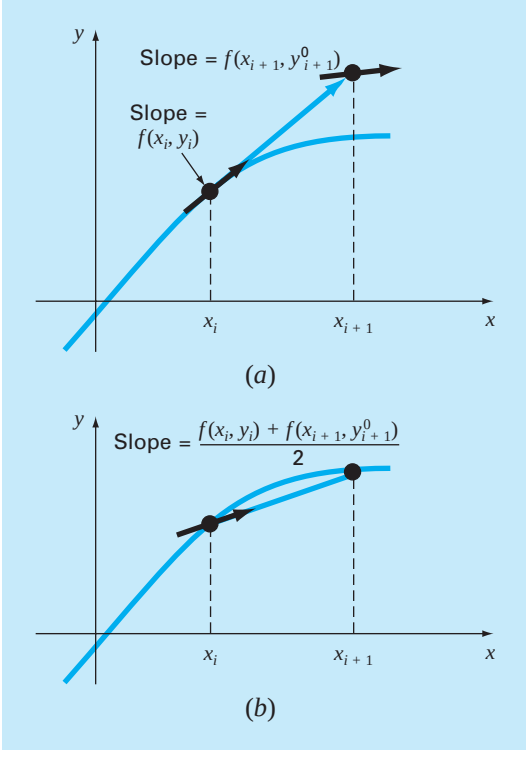
$$y_i' = f(x_i, y_i) \tag{25.12}$$

is used to extrapolate linearly to $y_{i+1}$:

$$y_{i+1}^0 = y_i + f(x_i, y_i)h \tag{25.13}$$

For the standard Euler method we would stop at this point. However, in Heun's method the $y_{i+1}^0$ calculated in Eq. (25.13) is not the final answer, but an intermediate prediction. This is why we have distinguished it with a superscript 0. Equation (25.13) is called a *predictor*

**FIGURE 25.9**
Graphical depiction of Heun's method. (*a*) Predictor and (*b*) corrector.



*equation.* It provides an estimate of $y_{i+1}$ that allows the calculation of an estimated slope at the end of the interval:

$$y'_{i+1} = f\left(x_{i+1}, y^0_{i+1}\right) \tag{25.14}$$

Thus, the two slopes [Eqs. (25.12) and (25.14)] can be combined to obtain an average slope for the interval:

$$\bar{y}' = \frac{y'_i + y'_{i+1}}{2} = \frac{f(x_i, y_i) + f\left(x_{i+1}, y^0_{i+1}\right)}{2}$$

This average slope is then used to extrapolate linearly from $y_i$ to $y_{i+1}$ using Euler's method:

$$y_{i+1} = y_i + \frac{f(x_i, y_i) + f\left(x_{i+1}, y^0_{i+1}\right)}{2} h$$

which is called a *corrector equation.*

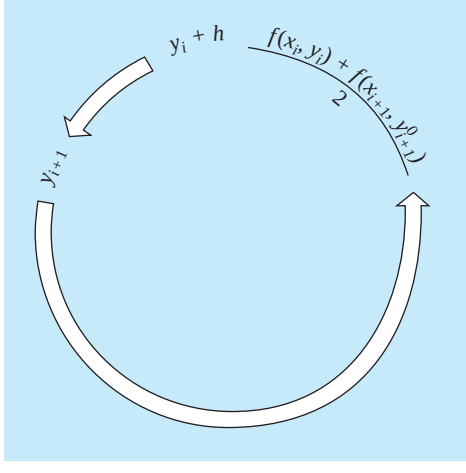
The Heun method is a *predictor-corrector approach.* All the multistep methods to be discussed subsequently in Chap. 26 are of this type. The Heun method is the only one-step predictor-corrector method described in this book. As derived above, it can be expressed concisely as

Predictor (Fig. 25.9*a*): $$y^0_{i+1} = y_i + f(x_i, y_i)h \tag{25.15}$$

Corrector (Fig. 25.9*b*): $$y_{i+1} = y_i + \frac{f(x_i, y_i) + f\left(x_{i+1}, y^0_{i+1}\right)}{2} h \tag{25.16}$$

Note that because Eq. (25.16) has $y_{i+1}$ on both sides of the equal sign, it can be applied in an iterative fashion. That is, an old estimate can be used repeatedly to provide an improved estimate of $y_{i+1}$. The process is depicted in Fig. 25.10. It should be understood that

**FIGURE 25.10**
Graphical representation of iterating the corrector of Heun's method to obtain an improved estimate.



this iterative process does not necessarily converge on the true answer but will converge on an estimate with a finite truncation error, as demonstrated in the following example.

As with similar iterative methods discussed in previous sections of the book, a termination criterion for convergence of the corrector is provided by [recall Eq. (3.5)]

$$|\varepsilon_a| = \left| \frac{y_{i+1}^j - y_{i+1}^{j-1}}{y_{i+1}^j} \right| 100\% \tag{25.17}$$

where $y_{i+1}^{j-1}$ and $y_{i+1}^j$ are the result from the prior and the present iteration of the corrector, respectively.

EXAMPLE 25.5 Heun's Method

**Problem Statement.** Use Heun's method to integrate $y' = 4e^{0.8x} - 0.5y$ from $x = 0$ to $x = 4$ with a step size of 1. The initial condition at $x = 0$ is $y = 2$.

**Solution.** Before solving the problem numerically, we can use calculus to determine the following analytical solution:

$$y = \frac{4}{1.3}(e^{0.8x} - e^{-0.5x}) + 2e^{-0.5x} \tag{E25.5.1}$$

This formula can be used to generate the true solution values in Table 25.2.

First, the slope at $(x_0, y_0)$ is calculated as

$$y_0' = 4e^0 - 0.5(2) = 3$$

This result is quite different from the actual average slope for the interval from 0 to 1.0, which is equal to 4.1946, as calculated from the differential equation using Eq. (PT6.4).

The numerical solution is obtained by using the predictor [Eq. (25.15)] to obtain an estimate of $y$ at 1.0:

$$y_1^0 = 2 + 3(1) = 5$$

**TABLE 25.2** Comparison of true and approximate values of the integral of $y' = 4e^{0.8x} - 0.5y$, with the initial condition that $y = 2$ at $x = 0$. The approximate values were computed using the Heun method with a step size of 1. Two cases, corresponding to different numbers of corrector iterations, are shown, along with the absolute percent relative error.

| | | Iterations of Heun's Method | | | |
|---|---|---|---|---|---|
| | | 1 | | 15 | |
| $x$ | $y_{true}$ | $y_{Heun}$ | $\lvert\varepsilon_t\rvert$ (%) | $y_{Heun}$ | $\lvert\varepsilon_t\rvert$ (%) |
| 0 | 2.0000000 | 2.0000000 | 0.00 | 2.0000000 | 0.00 |
| 1 | 6.1946314 | 6.7010819 | 8.18 | 6.3608655 | 2.68 |
| 2 | 14.8439219 | 16.3197819 | 9.94 | 15.3022367 | 3.09 |
| 3 | 33.6771718 | 37.1992489 | 10.46 | 34.7432761 | 3.17 |
| 4 | 75.3389626 | 83.3377674 | 10.62 | 77.7350962 | 3.18 |

Note that this is the result that would be obtained by the standard Euler method. The true value in Table 25.2 shows that it corresponds to a percent relative error of 19.3 percent.

Now, to improve the estimate for $y_{i+1}$, we use the value $y_1^0$ to predict the slope at the end of the interval

$$y_1' = f(x_1, y_1^0) = 4e^{0.8(1)} - 0.5(5) = 6.402164$$

which can be combined with the initial slope to yield an average slope over the interval from $x = 0$ to 1

$$y' = \frac{3 + 6.402164}{2} = 4.701082$$

which is closer to the true average slope of 4.1946. This result can then be substituted into the corrector [Eq. (25.16)] to give the prediction at $x = 1$

$$y_1 = 2 + 4.701082(1) = 6.701082$$

which represents a percent relative error of $-8.18$ percent. Thus, the Heun method without iteration of the corrector reduces the absolute value of the error by a factor of 2.4 as compared with Euler's method.

Now this estimate can be used to refine or correct the prediction of $y_1$ by substituting the new result back into the right-hand side of Eq. (25.16):

$$y_1 = 2 + \frac{\left[3 + 4e^{0.8(1)} - 0.5(6.701082)\right]}{2}1 = 6.275811$$

which represents an absolute percent relative error of 1.31 percent. This result, in turn, can be substituted back into Eq. (25.16) to further correct:

$$y_1 = 2 + \frac{\left[3 + 4e^{0.8(1)} - 0.5(6.275811)\right]}{2}1 = 6.382129$$

which represents an $|\varepsilon_t|$ of 3.03%. Notice how the errors sometimes grow as the iterations proceed. Such increases can occur, especially for large step sizes, and they prevent us from drawing the general conclusion that an additional iteration will always improve the result. However, for a sufficiently small step size, the iterations should eventually converge on a single value. For our case, 6.360865, which represents a relative error of 2.68 percent, is attained after 15 iterations. Table 25.2 shows results for the remainder of the computation using the method with 1 and 15 iterations per step.

In the previous example, the derivative is a function of both the dependent variable $y$ and the independent variable $x$. For cases such as polynomials, where the ODE is solely a function of the independent variable, the predictor step [Eq. (25.16)] is not required and the corrector is applied only once for each iteration. For such cases, the technique is expressed concisely as

$$y_{i+1} = y_i + \frac{f(x_i) + f(x_{i+1})}{2}h \tag{25.18}$$

Notice the similarity between the right-hand side of Eq. (25.18) and the trapezoidal rule [Eq. (21.3)]. The connection between the two methods can be formally demonstrated by starting with the ordinary differential equation

$$\frac{dy}{dx} = f(x)$$

This equation can be solved for $y$ by integration:

$$\int_{y_i}^{y_{i+1}} dy = \int_{x_i}^{x_{i+1}} f(x)\,dx \tag{25.19}$$

which yields

$$y_{i+1} - y_i = \int_{x_i}^{x_{i+1}} f(x)\,dx \tag{25.20}$$

or

$$y_{i+1} = y_i + \int_{x_i}^{x_{i+1}} f(x)\,dx \tag{25.21}$$

Now, recall from Chap. 21 that the trapezoidal rule [Eq. (21.3)] is defined as

$$\int_{x_i}^{x_{i+1}} f(x)\,dx \cong \frac{f(x_i) + f(x_{i+1})}{2}h \tag{25.22}$$

where $h = x_{i+1} - x_i$. Substituting Eq. (25.22) into Eq. (25.21) yields

$$y_{i+1} = y_i + \frac{f(x_i) + f(x_{i+1})}{2}h \tag{25.23}$$

which is equivalent to Eq. (25.18).

Because Eq. (25.23) is a direct expression of the trapezoidal rule, the local truncation error is given by [recall Eq. (21.6)]

$$E_t = -\frac{f''(\xi)}{12}h^3 \tag{25.24}$$

where $\xi$ is between $x_i$ and $x_{i+1}$. Thus, the method is second order because the second derivative of the ODE is zero when the true solution is a quadratic. In addition, the local and global errors are $O(h^3)$ and $O(h^2)$, respectively. Therefore, decreasing the step size decreases the error at a faster rate than for Euler's method. Figure 25.11, which shows the result of using Heun's method to solve the polynomial from Example 25.1 demonstrates this behavior.

### 25.2.2 The Midpoint (or Improved Polygon) Method

Figure 25.12 illustrates another simple modification of Euler's method. Called the *midpoint method* (or the *improved polygon* or the *modified Euler*), this technique uses Euler's method to predict a value of $y$ at the midpoint of the interval (Fig. 25.12*a*):

$$y_{i+1/2} = y_i + f(x_i, y_i)\frac{h}{2} \tag{25.25}$$

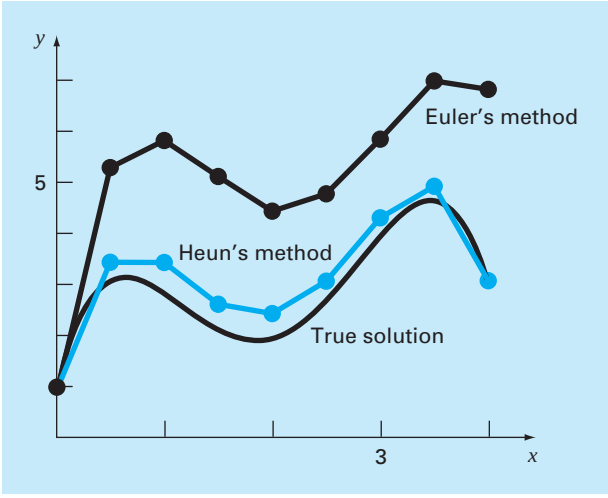



**FIGURE 25.11**
Comparison of the true solution with a numerical solution using Euler's and Heun's methods for the integral of $y' = -2x^3 + 12x^2 - 20x + 8.5$.

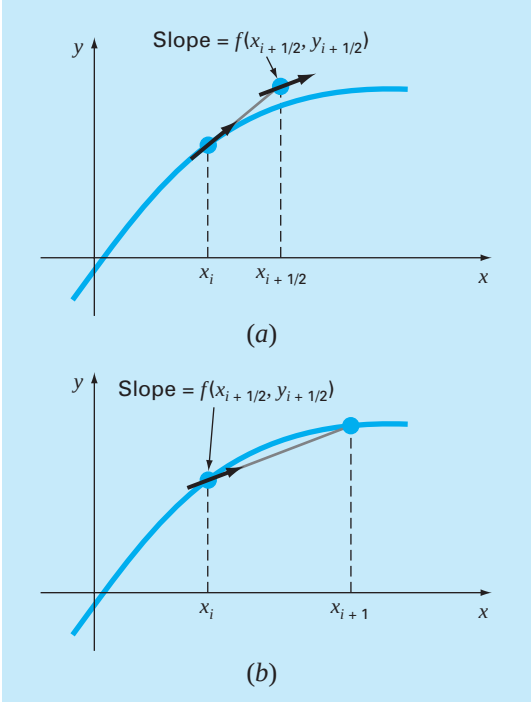



**FIGURE 25.12**
Graphical depiction of the midpoint method. (*a*) Eq. (25.25) and (*b*) Eq. (25.27).

Then, this predicted value is used to calculate a slope at the midpoint:

$$y'_{i+1/2} = f(x_{i+1/2}, y_{i+1/2}) \tag{25.26}$$

which is assumed to represent a valid approximation of the average slope for the entire interval. This slope is then used to extrapolate linearly from $x_i$ to $x_{i+1}$ (Fig. 25.12*b*):

$$y_{i+1} = y_i + f(x_{i+1/2}, y_{i+1/2})h \tag{25.27}$$

Observe that because $y_{i+1}$ is not on both sides, the corrector [Eq. (25.27)] cannot be applied iteratively to improve the solution.

As in the previous section, this approach can also be linked to Newton-Cotes integration formulas. Recall from Table 21.4, that the simplest Newton-Cotes open integration formula, which is called the midpoint method, can be represented as

$$\int_a^b f(x)\,dx \cong (b-a)\,f(x_1)$$

where $x_1$ is the midpoint of the interval $(a, b)$. Using the nomenclature for the present case, it can be expressed as

$$\int_{x_i}^{x_{i+1}} f(x)\,dx \cong h f(x_{i+1/2})$$

Substitution of this formula into Eq. (25.21) yields Eq. (25.27). Thus, just as the Heun method can be called the trapezoidal rule, the *midpoint method* gets its name from the underlying integration formula upon which it is based.

The midpoint method is superior to Euler's method because it utilizes a slope estimate at the midpoint of the prediction interval. Recall from our discussion of numerical differentiation in Sec. 4.1.3 that centered finite divided differences are better approximations of derivatives than either forward or backward versions. In the same sense, a centered approximation such as Eq. (25.26) has a local truncation error of $O(h^2)$ in comparison with the forward approximation of Euler's method, which has an error of $O(h)$. Consequently, the local and global errors of the midpoint method are $O(h^3)$ and $O(h^2)$, respectively.

### 25.2.3 Computer Algorithms for Heun and Midpoint Methods

Both the Heun method with a single corrector and the midpoint method can be easily programmed using the general structure depicted in Fig. 25.7. As in Fig. 25.13*a* and *b*, simple routines can be written to take the place of the Euler routine in Fig. 25.7.

However, when the iterative version of the Heun method is to be implemented, the modifications are a bit more involved (although they are still localized within a single module). We have developed pseudocode for this purpose in Fig. 25.13*c*. This algorithm can be combined with Fig. 25.7 to develop software for the iterative Heun method.

### 25.2.4 Summary

By tinkering with Euler's method, we have derived two new second-order techniques. Even though these versions require more computational effort to determine the slope, the accompanying reduction in error will allow us to conclude in a subsequent section

**(a) Simple Heun without Corrector**

```
SUB Heun (x, y, h, ynew)
  CALL Derivs (x, y, dy1dx)
  ye = y + dy1dx · h
  CALL Derivs(x + h, ye, dy2dx)
  Slope = (dy1dx + dy2dx)/2
  ynew = y + Slope · h
  x = x + h
END SUB
```

**(b) Midpoint Method**

```
SUB Midpoint (x, y, h, ynew)
  CALL Derivs(x, y, dydx)
  ym = y + dydx · h/2
  CALL Derivs (x + h/2, ym, dymdx)
  ynew = y + dymdx · h
  x = x + h
END SUB
```

**(c) Heun with Corrector**

```
SUB HeunIter (x, y, h, ynew)
  es = 0.01
  maxit = 20
  CALL Derivs(x, y, dy1dx)
  ye = y + dy1dx · h
  iter = 0
  DO
    yeold = ye
    CALL Derivs(x + h, ye, dy2dx)
    slope = (dy1dx + dy2dx)/2
    ye = y + slope · h
    iter = iter + 1
    ea = |(ye − yeold)/ye| 100%
    IF (ea ≤ es OR iter > maxit) EXIT
  END DO
  ynew = ye
  x = x + h
END SUB
```

**FIGURE 25.13**
Pseudocode to implement the (*a*) simple Heun, (*b*) midpoint, and (*c*) iterative Heun methods.

(Sec. 25.3.4) that the improved accuracy is usually worth the effort. Although there are certain cases where easily programmable techniques such as Euler's method can be applied to advantage, the Heun and midpoint methods are generally superior and should be implemented if they are consistent with the problem objectives.

As noted at the beginning of this section, the Heun (without iterations), the midpoint method, and in fact, the Euler technique itself are versions of a broader class of one-step approaches called Runge-Kutta methods. We now turn to a formal derivation of these techniques.

## 25.3 RUNGE-KUTTA METHODS

*Runge-Kutta* (*RK*) methods achieve the accuracy of a Taylor series approach without requiring the calculation of higher derivatives. Many variations exist but all can be cast in the generalized form of Eq. (25.1):

$$y_{i+1} = y_i + \phi(x_i, y_i, h)h \tag{25.28}$$

where $\phi(x_i, y_i, h)$ is called an *increment function*, which can be interpreted as a representative slope over the interval. The increment function can be written in general form as

$$\phi = a_1k_1 + a_2k_2 + \cdots + a_nk_n \tag{25.29}$$

where the $a$'s are constants and the $k$'s are

$$k_1 = f(x_i, y_i) \tag{25.29a}$$

$$k_2 = f(x_i + p_1 h, y_i + q_{11} k_1 h) \tag{25.29b}$$

$$k_3 = f(x_i + p_2 h, y_i + q_{21} k_1 h + q_{22} k_2 h) \tag{25.29c}$$

$$\vdots$$

$$k_n = f(x_i + p_{n-1} h, y_i + q_{n-1,1} k_1 h + q_{n-1,2} k_2 h + \cdots + q_{n-1,n-1} k_{n-1} h) \tag{25.29d}$$

where the $p$'s and $q$'s are constants. Notice that the $k$'s are recurrence relationships. That is, $k_1$ appears in the equation for $k_2$, which appears in the equation for $k_3$, and so forth. Because each $k$ is a functional evaluation, this recurrence makes RK methods efficient for computer calculations.

Various types of Runge-Kutta methods can be devised by employing different numbers of terms in the increment function as specified by $n$. Note that the first-order RK method with $n = 1$ is, in fact, Euler's method. Once $n$ is chosen, values for the $a$'s, $p$'s, and $q$'s are evaluated by setting Eq. (25.28) equal to terms in a Taylor series expansion (Box 25.1). Thus, at least for the lower-order versions, the number of terms, $n$, usually represents the order of the approach. For example, in the next section, second-order RK methods use an increment function with two terms ($n = 2$). These second-order methods will be exact if the solution to the differential equation is quadratic. In addition, because terms with $h^3$ and higher are dropped during the derivation, the local truncation error is $O(h^3)$ and the global error is $O(h^2)$. In subsequent sections, the third- and fourth-order RK methods ($n = 3$ and 4, respectively) are developed. For these cases, the global truncation errors are $O(h^3)$ and $O(h^4)$, respectively.

### 25.3.1 Second-Order Runge-Kutta Methods

The second-order version of Eq. (25.28) is

$$y_{i+1} = y_i + (a_1 k_1 + a_2 k_2) h \tag{25.30}$$

where

$$k_1 = f(x_i, y_i) \tag{25.30a}$$

$$k_2 = f(x_i + p_1 h, y_i + q_{11} k_1 h) \tag{25.30b}$$

As described in Box 25.1, values for $a_1$, $a_2$, $p_1$, and $q_{11}$ are evaluated by setting Eq. (25.30) equal to a Taylor series expansion to the second-order term. By doing this, we derive three equations to evaluate the four unknown constants. The three equations are

$$a_1 + a_2 = 1 \tag{25.31}$$

$$a_2 p_1 = \frac{1}{2} \tag{25.32}$$

$$a_2 q_{11} = \frac{1}{2} \tag{25.33}$$

## Box 25.1 Derivation of the Second-Order Runge-Kutta Methods

The second-order version of Eq. (25.28) is

$$y_{i+1} = y_i + (a_1 k_1 + a_2 k_2)h \tag{B25.1.1}$$

where

$$k_1 = f(x_i, y_i) \tag{B25.1.2}$$

and

$$k_2 = f(x_i + p_1 h, y_i + q_{11} k_1 h) \tag{B25.1.3}$$

To use Eq. (B25.1.1) we have to determine values for the constants $a_1$, $a_2$, $p_1$, and $q_{11}$. To do this, we recall that the second-order Taylor series for $y_{i+1}$ in terms of $y_i$ and $f(x_i, y_i)$ is written as [Eq. (25.11)]

$$y_{i+1} = y_i + f(x_i, y_i)h + \frac{f'(x_i, y_i)}{2!}h^2 \tag{B25.1.4}$$

where $f'(x_i, y_i)$ must be determined by chain-rule differentiation (Sec. 25.1.3):

$$f'(x_i, y_i) = \frac{\partial f(x, y)}{\partial x} + \frac{\partial f(x, y)}{\partial y}\frac{dy}{dx} \tag{B25.1.5}$$

Substituting Eq. (B25.1.5) into (B25.1.4) gives

$$y_{i+1} = y_i + f(x_i, y_i)h + \left(\frac{\partial f}{\partial x} + \frac{\partial f}{\partial y}\frac{dy}{dx}\right)\frac{h^2}{2!} \tag{B25.1.6}$$

The basic strategy underlying Runge-Kutta methods is to use algebraic manipulations to solve for values of $a_1$, $a_2$, $p_1$, and $q_{11}$ that make Eqs. (B25.1.1) and (B25.1.6) equivalent.

To do this, we first use a Taylor series to expand Eq. (25.1.3). The Taylor series for a two-variable function is defined as [recall Eq. (4.26)]

$$g(x + r, y + s) = g(x, y) + r\frac{\partial g}{\partial x} + s\frac{\partial g}{\partial y} + \cdots$$

Applying this method to expand Eq. (B25.1.3) gives

$$f(x_i + p_1 h, y_i + q_{11} k_1 h) = f(x_i, y_i) + p_1 h\frac{\partial f}{\partial x} + q_{11} k_1 h\frac{\partial f}{\partial y} + O(h^2)$$

This result can be substituted along with Eq. (B25.1.2) into Eq. (B25.1.1) to yield

$$y_{i+1} = y_i + a_1 h f(x_i, y_i) + a_2 h f(x_i, y_i) + a_2 p_1 h^2\frac{\partial f}{\partial x} + a_2 q_{11} h^2 f(x_i, y_i)\frac{\partial f}{\partial y} + O(h^3)$$

or, by collecting terms,

$$y_{i+1} = y_i + [a_1 f(x_i, y_i) + a_2 f(x_i, y_i)]h + \left[a_2 p_1\frac{\partial f}{\partial x} + a_2 q_{11} f(x_i, y_i)\frac{\partial f}{\partial y}\right]h^2 + O(h^3) \tag{B25.1.7}$$

Now, comparing like terms in Eqs. (B25.1.6) and (B25.1.7), we determine that for the two equations to be equivalent, the following must hold:

$$a_1 + a_2 = 1$$

$$a_2 p_1 = \frac{1}{2}$$

$$a_2 q_{11} = \frac{1}{2}$$

These three simultaneous equations contain the four unknown constants. Because there is one more unknown than the number of equations, there is no unique set of constants that satisfy the equations. However, by assuming a value for one of the constants, we can determine the other three. Consequently, there is a family of second-order methods rather than a single version.

Because we have three equations with four unknowns, we must assume a value of one of the unknowns to determine the other three. Suppose that we specify a value for $a_2$. Then Eqs. (25.31) through (25.33) can be solved simultaneously for

$$a_1 = 1 - a_2 \tag{25.34}$$

$$p_1 = q_{11} = \frac{1}{2a_2} \tag{25.35}$$

Because we can choose an infinite number of values for $a_2$, there are an infinite number of second-order RK methods. Every version would yield exactly the same results if the solution to the ODE were quadratic, linear, or a constant. However, they yield different results when (as is typically the case) the solution is more complicated. We present three of the most commonly used and preferred versions:

**Heun Method with a Single Corrector ($a_2 = 1/2$).** If $a_2$ is assumed to be $1/2$, Eqs. (25.34) and (25.35) can be solved for $a_1 = 1/2$ and $p_1 = q_{11} = 1$. These parameters, when substituted into Eq. (25.30), yield

$$y_{i+1} = y_i + \left(\frac{1}{2}k_1 + \frac{1}{2}k_2\right)h \tag{25.36}$$

where

$$k_1 = f(x_i, y_i) \tag{25.36a}$$

$$k_2 = f(x_i + h, y_i + k_1 h) \tag{25.36b}$$

Note that $k_1$ is the slope at the beginning of the interval and $k_2$ is the slope at the end of the interval. Consequently, this second-order Runge-Kutta method is actually Heun's technique without iteration.

**The Midpoint Method ($a_2 = 1$).** If $a_2$ is assumed to be 1, then $a_1 = 0$, $p_1 = q_{11} = 1/2$, and Eq. (25.30) becomes

$$y_{i+1} = y_i + k_2 h \tag{25.37}$$

where

$$k_1 = f(x_i, y_i) \tag{25.37a}$$

$$k_2 = f\left(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1 h\right) \tag{25.37b}$$

This is the midpoint method.

**Ralston's Method ($a_2 = 2/3$).** Ralston (1962) and Ralston and Rabinowitz (1978) determined that choosing $a_2 = 2/3$ provides a minimum bound on the truncation error for the second-order RK algorithms. For this version, $a_1 = 1/3$ and $p_1 = q_{11} = 3/4$ and yields

$$y_{i+1} = y_i + \left(\frac{1}{3}k_1 + \frac{2}{3}k_2\right)h \tag{25.38}$$

where

$$k_1 = f(x_i, y_i) \tag{25.38a}$$

$$k_2 = f\left(x_i + \frac{3}{4}h, y_i + \frac{3}{4}k_1 h\right) \tag{25.38b}$$

EXAMPLE 25.6 Comparison of Various Second-Order RK Schemes

**Problem Statement.** Use the midpoint method [Eq. (25.37)] and Ralston's method [Eq. (25.38)] to numerically integrate Eq. (PT7.13)

$$f(x, y) = -2x^3 + 12x^2 - 20x + 8.5$$

from $x = 0$ to $x = 4$ using a step size of 0.5. The initial condition at $x = 0$ is $y = 1$. Compare the results with the values obtained using another second-order RK algorithm, that is, the Heun method without corrector iteration (Table 25.3).

**Solution.** The first step in the midpoint method is to use Eq. (25.37*a*) to compute

$$k_1 = -2(0)^3 + 12(0)^2 - 20(0) + 8.5 = 8.5$$

However, because the ODE is a function of $x$ only, this result has no bearing on the second step—the use of Eq. (25.37*b*) to compute

$$k_2 = -2(0.25)^3 + 12(0.25)^2 - 20(0.25) + 8.5 = 4.21875$$

Notice that this estimate of the slope is much closer to the average value for the interval (4.4375) than the slope at the beginning of the interval (8.5) that would have been used for Euler's approach. The slope at the midpoint can then be substituted into Eq. (25.37) to predict

$$y(0.5) = 1 + 4.21875(0.5) = 3.109375 \qquad \varepsilon_t = 3.4\%$$

The computation is repeated, and the results are summarized in Fig. 25.14 and Table 25.3.

**FIGURE 25.14**
Comparison of the true solution with numerical solutions using three second-order RK methods and Euler's method.

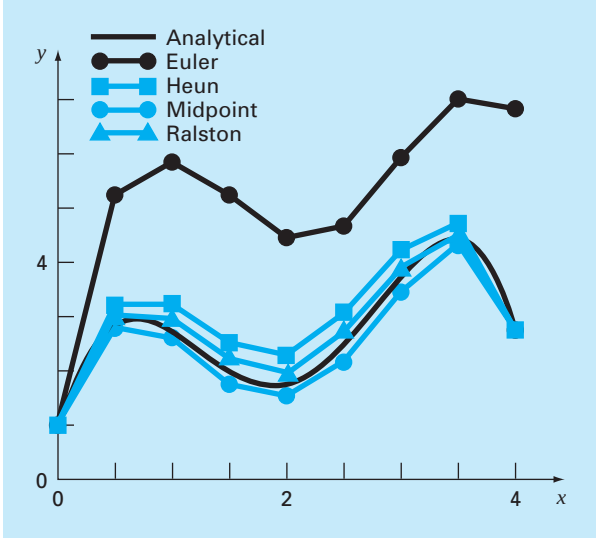


**TABLE 25.3** Comparison of true and approximate values of the integral of $y' = -2x^3 + 12x^2 - 20x + 8.5$, with the initial condition that $y = 1$ at $x = 0$. The approximate values were computed using three versions of second-order RK methods with a step size of 0.5.

| | | Heun | | Midpoint | | Second-Order Ralston RK | |
|---|---|---|---|---|---|---|---|
| $x$ | $y_{true}$ | $y$ | $\lvert\varepsilon_t\rvert$ (%) | $y$ | $\lvert\varepsilon_t\rvert$ (%) | $y$ | $\lvert\varepsilon_t\rvert$ (%) |
| 0.0 | 1.00000 | 1.00000 | 0 | 1.00000 | 0 | 1.00000 | 0 |
| 0.5 | 3.21875 | 3.43750 | 6.8 | 3.109375 | 3.4 | 3.277344 | 1.8 |
| 1.0 | 3.00000 | 3.37500 | 12.5 | 2.81250 | 6.3 | 3.101563 | 3.4 |
| 1.5 | 2.21875 | 2.68750 | 21.1 | 1.984375 | 10.6 | 2.347656 | 5.8 |
| 2.0 | 2.00000 | 2.50000 | 25.0 | 1.75 | 12.5 | 2.140625 | 7.0 |
| 2.5 | 2.71875 | 3.18750 | 17.2 | 2.484375 | 8.6 | 2.855469 | 5.0 |
| 3.0 | 4.00000 | 4.37500 | 9.4 | 3.81250 | 4.7 | 4.117188 | 2.9 |
| 3.5 | 4.71875 | 4.93750 | 4.6 | 4.609375 | 2.3 | 4.800781 | 1.7 |
| 4.0 | 3.00000 | 3.00000 | 0 | 3 | 0 | 3.031250 | 1.0 |

For Ralston's method, $k_1$ for the first interval also equals 8.5 and [Eq. (25.38*b*)]

$$k_2 = -2(0.375)^3 + 12(0.375)^2 - 20(0.375) + 8.5 = 2.58203125$$

The average slope is computed by

$$\phi = \frac{1}{3}(8.5) + \frac{2}{3}(2.58203125) = 4.5546875$$

which can be used to predict

$$y(0.5) = 1 + 4.5546875(0.5) = 3.27734375 \qquad \varepsilon_t = -1.82\%$$

The computation is repeated, and the results are summarized in Fig. 25.14 and Table 25.3. Notice how all the second-order RK methods are superior to Euler's method.

### 25.3.2 Third-Order Runge-Kutta Methods

For $n = 3$, a derivation similar to the one for the second-order method can be performed. The result of this derivation is six equations with eight unknowns. Therefore, values for two of the unknowns must be specified a priori in order to determine the remaining parameters. One common version that results is

$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 4k_2 + k_3)h \tag{25.39}$$

where

$$k_1 = f(x_i, y_i) \tag{25.39a}$$

$$k_2 = f\left(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1 h\right) \tag{25.39b}$$

$$k_3 = f(x_i + h, y_i - k_1 h + 2k_2 h) \tag{25.39c}$$

Note that if the derivative is a function of $x$ only, this third-order method reduces to Simpson's 1/3 rule. Ralston (1962) and Ralston and Rabinowitz (1978) have developed an alternative version that provides a minimum bound on the truncation error. In any case, the third-order RK methods have local and global errors of $O(h^4)$ and $O(h^3)$, respectively, and yield exact results when the solution is a cubic. When dealing with polynomials, Eq. (25.39) will also be exact when the differential equation is cubic and the solution is quartic. This is because Simpson's 1/3 rule provides exact integral estimates for cubics (recall Box 21.3).

### 25.3.3 Fourth-Order Runge-Kutta Methods

The most popular RK methods are fourth order. As with the second-order approaches, there are an infinite number of versions. The following is the most commonly used form, and we therefore call it the *classical fourth-order RK method:*

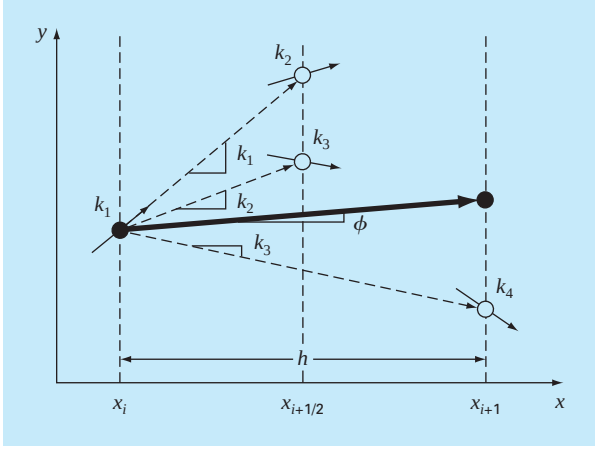
$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)h \tag{25.40}$$

where

$$k_1 = f(x_i, y_i) \tag{25.40a}$$

$$k_2 = f\left(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_1 h\right) \tag{25.40b}$$

**FIGURE 25.15**
Graphical depiction of the slope estimates comprising the fourth-order RK method.



$$k_3 = f\left(x_i + \frac{1}{2}h, y_i + \frac{1}{2}k_2 h\right) \tag{25.40c}$$

$$k_4 = f(x_i + h, y_i + k_3 h) \tag{25.40d}$$

Notice that for ODEs that are a function of $x$ alone, the classical fourth-order RK method is similar to Simpson's 1/3 rule. In addition, the fourth-order RK method is similar to the Heun approach in that multiple estimates of the slope are developed in order to come up with an improved average slope for the interval. As depicted in Fig. 25.15, each of the $k$'s represents a slope. Equation (25.40) then represents a weighted average of these to arrive at the improved slope.

EXAMPLE 25.7 Classical Fourth-Order RK Method

Problem Statement.

**(a)** Use the classical fourth-order RK method [Eq. (25.40)] to integrate

$$f(x, y) = -2x^3 + 12x^2 - 20x + 8.5$$

using a step size of $h = 0.5$ and an initial condition of $y = 1$ at $x = 0$.

**(b)** Similarly, integrate

$$f(x, y) = 4e^{0.8x} - 0.5y$$

using $h = 0.5$ with $y(0) = 2$ from $x = 0$ to 0.5.

Solution.

**(a)** Equations (25.40*a*) through (25.40*d*) are used to compute $k_1 = 8.5$, $k_2 = 4.21875$, $k_3 = 4.21875$ and $k_4 = 1.25$, which are substituted into Eq. (25.40) to yield

$$y(0.5) = 1 + \left\{\frac{1}{6}[8.5 + 2(4.21875) + 2(4.21875) + 1.25]\right\} 0.5$$
$$= 3.21875$$

which is exact. Thus, because the true solution is a quartic [Eq. (PT7.16)], the fourth-order method gives an exact result.

**(b)** For this case, the slope at the beginning of the interval is computed as

$$k_1 = f(0, 2) = 4e^{0.8(0)} - 0.5(2) = 3$$

This value is used to compute a value of $y$ and a slope at the midpoint,

$$y(0.25) = 2 + 3(0.25) = 2.75$$
$$k_2 = f(0.25, 2.75) = 4e^{0.8(0.25)} - 0.5(2.75) = 3.510611$$

This slope in turn is used to compute another value of $y$ and another slope at the midpoint,

$$y(0.25) = 2 + 3.510611(0.25) = 2.877653$$
$$k_3 = f(0.25, 2.877653) = 4e^{0.8(0.25)} - 0.5(2.877653) = 3.446785$$

Next, this slope is used to compute a value of $y$ and a slope at the end of the interval,

$$y(0.5) = 2 + 3.071785(0.5) = 3.723392$$
$$k_4 = f(0.5, 3.723392) = 4e^{0.8(0.5)} - 0.5(3.723392) = 4.105603$$

Finally, the four slope estimates are combined to yield an average slope. This average slope is then used to make the final prediction at the end of the interval.

$$\phi = \frac{1}{6}[3 + 2(3.510611) + 2(3.446785) + 4.105603] = 3.503399$$

$$y(0.5) = 2 + 3.503399(0.5) = 3.751699$$

which compares favorably with the true solution of 3.751521.

### 25.3.4 Higher-Order Runge-Kutta Methods

Where more accurate results are required, *Butcher's* (1964) *fifth-order RK method* is recommended:

$$y_{i+1} = y_i + \frac{1}{90}(7k_1 + 32k_3 + 12k_4 + 32k_5 + 7k_6)h \tag{25.41}$$

where

$$k_1 = f(x_i, y_i) \tag{25.41a}$$

$$k_2 = f\left(x_i + \frac{1}{4}h, y_i + \frac{1}{4}k_1 h\right) \tag{25.41b}$$

$$k_3 = f\left(x_i + \frac{1}{4}h, y_i + \frac{1}{8}k_1 h + \frac{1}{8}k_2 h\right) \tag{25.41c}$$

$$k_4 = f\left(x_i + \frac{1}{2}h, y_i - \frac{1}{2}k_2 h + k_3 h\right) \tag{25.41d}$$

$$k_5 = f\left(x_i + \frac{3}{4}h, y_i + \frac{3}{16}k_1 h + \frac{9}{16}k_4 h\right) \tag{25.41e}$$

$$k_6 = f\left(x_i + h, y_i - \frac{3}{7}k_1 h + \frac{2}{7}k_2 h + \frac{12}{7}k_3 h - \frac{12}{7}k_4 h + \frac{8}{7}k_5 h\right) \tag{25.41f}$$

Note the similarity between Butcher's method and Boole's Rule in Table 21.2. Higher-order RK formulas such as Butcher's method are available, but in general, beyond fourth-order methods the gain in accuracy is offset by the added computational effort and complexity.

EXAMPLE 25.8 Comparison of Runge-Kutta Methods

Problem Statement. Use first- through fifth-order RK methods to solve

$$f(x, y) = 4e^{0.8x} - 0.5y$$

with $y(0) = 2$ from $x = 0$ to $x = 4$ with various step sizes. Compare the accuracy of the various methods for the result at $x = 4$ based on the exact answer of $y(4) = 75.33896$.

Solution. The computation is performed using Euler's, the noniterative Heun, the third-order RK [Eq. (25.39)], the classical fourth-order RK, and Butcher's fifth-order RK

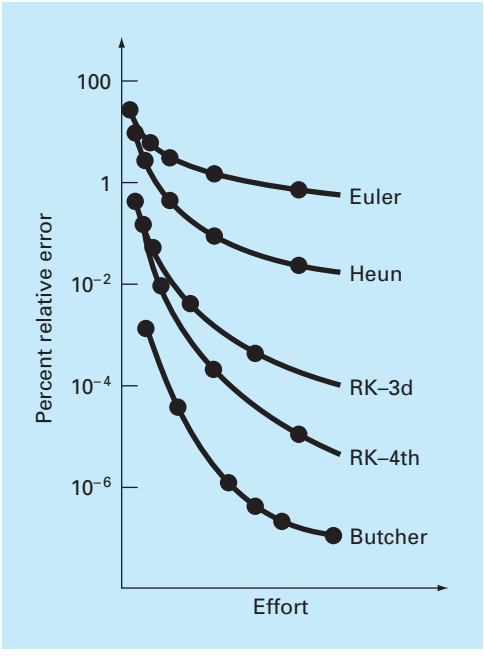



**FIGURE 25.16**
Comparison of percent relative error versus computational effort for first- through fifth-order RK methods.

methods. The results are presented in Fig. 25.16, where we have plotted the absolute value of the percent relative error versus the computational effort. This latter quantity is equivalent to the number of function evaluations required to attain the result, as in

$$\text{Effort} = n_f \frac{b-a}{h} \tag{E25.8.1}$$

where $n_f$ = the number of function evaluations involved in the particular RK computation. For orders $\leq 4$, $n_f$ is equal to the order of the method. However, note that Butcher's fifth-order technique requires six function evaluations [Eq. (25.41*a*) through (25.41*f*)]. The quantity $(b-a)/h$ is the total integration interval divided by the step size—that is, it is the number of applications of the RK technique required to obtain the result. Thus, because the function evaluations are usually the primary time-consuming steps, Eq. (E25.8.1) provides a rough measure of the run time required to attain the answer.

Inspection of Fig. 25.16 leads to a number of conclusions: first, that the higher-order methods attain better accuracy for the same computational effort and, second, that the gain in accuracy for the additional effort tends to diminish after a point. (Notice that the curves drop rapidly at first and then tend to level off.)

Example 25.8 and Fig. 25.16 might lead one to conclude that higher-order RK techniques are always the preferred methods. However, other factors such as programming

```
SUB RK4 (x, y, h, ynew)
  CALL Derivs(x, y, k1)
  ym = y + k1 · h/2
  CALL Derivs(x + h/2, ym, k2)
  ym = y + k2 · h/2
  CALL Derivs(x + h/2, ym, k3)
  ye = y + k3 · h
  CALL Derivs(x + h, ye, k4)
  slope = (k1 + 2(k2 + k3) + k4)/6
  ynew = y + slope · h
  x = x + h
END SUB
```

**FIGURE 25.17**
Pseudocode to determine a single step of the fourth-order RK method.

costs and the accuracy requirements of the problem also must be considered when choosing a solution technique. Such trade-offs will be explored in detail in the engineering applications in Chap. 28 and in the epilogue for Part Seven.

### 25.3.5 Computer Algorithms for Runge-Kutta Methods

As with all the methods covered in this chapter, the RK techniques fit nicely into the general algorithm embodied in Fig. 25.7. Figure 25.17 presents pseudocode to determine the slope of the classic fourth-order RK method [Eq. (25.40)]. Subroutines to compute slopes for all the other versions can be easily programmed in a similar fashion.

## 25.4 SYSTEMS OF EQUATIONS

Many practical problems in engineering and science require the solution of a system of simultaneous ordinary differential equations rather than a single equation. Such systems may be represented generally as

$$\begin{aligned}
\frac{dy_1}{dx} &= f_1(x, y_1, y_2, \ldots, y_n) \\
\frac{dy_2}{dx} &= f_2(x, y_1, y_2, \ldots, y_n) \\
&\vdots \\
\frac{dy_n}{dx} &= f_n(x, y_1, y_2, \ldots, y_n)
\end{aligned} \tag{25.42}$$

The solution of such a system requires that $n$ initial conditions be known at the starting value of $x$.

### 25.4.1 Euler's Method

All the methods discussed in this chapter for single equations can be extended to the system shown above. Engineering applications can involve thousands of simultaneous equations. In each case, the procedure for solving a system of equations simply involves applying the one-step technique for every equation at each step before proceeding to the next step. This is best illustrated by the following example for the simple Euler's method.

EXAMPLE 25.9 Solving Systems of ODEs Using Euler's Method

Problem Statement. Solve the following set of differential equations using Euler's method, assuming that at $x = 0$, $y_1 = 4$, and $y_2 = 6$. Integrate to $x = 2$ with a step size of 0.5.

$$\frac{dy_1}{dx} = -0.5y_1 \qquad \frac{dy_2}{dx} = 4 - 0.3y_2 - 0.1y_1$$

Solution. Euler's method is implemented for each variable as in Eq. (25.2):

$$y_1(0.5) = 4 + [-0.5(4)]0.5 = 3$$
$$y_2(0.5) = 6 + [4 - 0.3(6) - 0.1(4)]0.5 = 6.9$$

Note that $y_1(0) = 4$ is used in the second equation rather than the $y_1(0.5) = 3$ computed with the first equation. Proceeding in a like manner gives

| $x$ | $y_1$ | $y_2$ |
|---|---|---|
| 0 | 4 | 6 |
| 0.5 | 3 | 6.9 |
| 1.0 | 2.25 | 7.715 |
| 1.5 | 1.6875 | 8.44525 |
| 2.0 | 1.265625 | 9.094087 |

### 25.4.2 Runge-Kutta Methods

Note that any of the higher-order RK methods in this chapter can be applied to systems of equations. However, care must be taken in determining the slopes. Figure 25.15 is helpful in visualizing the proper way to do this for the fourth-order method. That is, we first develop slopes for all variables at the initial value. These slopes (a set of $k_1$'s) are then used to make predictions of the dependent variable at the midpoint of the interval. These midpoint values are in turn used to compute a set of slopes at the midpoint (the $k_2$'s). These new slopes are then taken back to the starting point to make another set of midpoint predictions that lead to new slope predictions at the midpoint (the $k_3$'s). These are then employed to make predictions at the end of the interval that are used to develop slopes at the end of the interval (the $k_4$'s). Finally, the $k$'s are combined into a set of increment functions [as in Eq. (25.40)] and brought back to the beginning to make the final prediction. The following example illustrates the approach.

EXAMPLE 25.10 Solving Systems of ODEs Using the Fourth-Order RK Method

**Problem Statement.** Use the fourth-order RK method to solve the ODEs from Example 25.9.

**Solution.** First, we must solve for all the slopes at the beginning of the interval:

$$k_{1,1} = f_1(0, 4, 6) = -0.5(4) = -2$$
$$k_{1,2} = f_2(0, 4, 6) = 4 - 0.3(6) - 0.1(4) = 1.8$$

where $k_{i,j}$ is the $i$th value of $k$ for the $j$th dependent variable. Next, we must calculate the first values of $y_1$ and $y_2$ at the midpoint:

$$y_1 + k_{1,1}\frac{h}{2} = 4 + (-2)\frac{0.5}{2} = 3.5$$
$$y_2 + k_{1,2}\frac{h}{2} = 6 + (1.8)\frac{0.5}{2} = 6.45$$

which can be used to compute the first set of midpoint slopes,

$$k_{2,1} = f_1(0.25, 3.5, 6.45) = -1.75$$
$$k_{2,2} = f_2(0.25, 3.5, 6.45) = 1.715$$

These are used to determine the second set of midpoint predictions,

$$y_1 + k_{2,1}\frac{h}{2} = 4 + (-1.75)\frac{0.5}{2} = 3.5625$$
$$y_2 + k_{2,2}\frac{h}{2} = 6 + (1.715)\frac{0.5}{2} = 6.42875$$

which can be used to compute the second set of midpoint slopes,

$$k_{3,1} = f_1(0.25, 3.5625, 6.42875) = -1.78125$$
$$k_{3,2} = f_2(0.25, 3.5625, 6.42875) = 1.715125$$

These are used to determine the predictions at the end of the interval

$$y_1 + k_{3,1}h = 4 + (-1.78125)(0.5) = 3.109375$$
$$y_2 + k_{3,2}h = 6 + (1.715125)(0.5) = 6.857563$$

which can be used to compute the endpoint slopes,

$$k_{4,1} = f_1(0.5, 3.109375, 6.857563) = -1.554688$$
$$k_{4,2} = f_2(0.5, 3.109375, 6.857563) = 1.631794$$

The values of $k$ can then be used to compute [Eq. (25.40)]:

$$y_1(0.5) = 4 + \frac{1}{6}[-2 + 2(-1.75 - 1.78125) - 1.554688]0.5 = 3.115234$$

$$y_2(0.5) = 6 + \frac{1}{6}[1.8 + 2(1.715 + 1.715125) + 1.631794]0.5 = 6.857670$$

Proceeding in a like manner for the remaining steps yields

| $x$ | $y_1$ | $y_2$ |
|---|---|---|
| 0 | 4 | 6 |
| 0.5 | 3.115234 | 6.857670 |
| 1.0 | 2.426171 | 7.632106 |
| 1.5 | 1.889523 | 8.326886 |
| 2.0 | 1.471577 | 8.946865 |

### 25.4.3 Computer Algorithm for Solving Systems of ODEs

The computer code for solving a single ODE with Euler's method (Fig. 25.7) can be easily extended to systems of equations. The modifications include:

1. Inputting the number of equations, $n$.
2. Inputting the initial values for each of the $n$ dependent variables.
3. Modifying the algorithm so that it computes slopes for each of the dependent variables.
4. Including additional equations to compute derivative values for each of the ODEs.
5. Including loops to compute a new value for each dependent variable.

Such an algorithm is outlined in Fig. 25.18 for the fourth-order RK method. Notice how similar it is in structure and organization to Fig. 25.7. Most of the differences relate to the fact that

1. There are $n$ equations.
2. The added detail of the fourth-order RK method.

EXAMPLE 25.11 Solving Systems of ODEs with the Computer

Problem Statement. A computer program to implement the fourth-order RK method for systems can be easily developed based on Fig. 25.18. Such software makes it convenient to compare different models of a physical system. For example, a linear model for a swinging pendulum is given by [recall Eq. (PT7.11)]

$$\frac{dy_1}{dx} = y_2 \qquad \frac{dy_2}{dx} = -16.1y_1$$

where $y_1$ and $y_2$ = angular displacement and velocity. A nonlinear model of the same system is [recall Eq. (PT7.9)]

$$\frac{dy_3}{dx} = y_4 \qquad \frac{dy_4}{dx} = -16.1\sin(y_3)$$

where $y_3$ and $y_4$ = angular displacement and velocity for the nonlinear case. Solve these systems for two cases: ($a$) a small initial displacement ($y_1 = y_3 = 0.1$ radians; $y_2 = y_4 = 0$) and ($b$) a large displacement ($y_1 = y_3 = \pi/4 = 0.785398$ radians; $y_2 = y_4 = 0$).

**(a) Main or "Driver" Program**

```
Assign values for
n = number of equations
yi = initial values of n dependent
     variables
xi = initial value independent
     variable
xf = final value independent variable
dx = calculation step size
xout = output interval

x = xi
m = 0
xp_m = x
DOFOR i = 1, n
  yp_i,m = yi_i
  y_i = yi_i
END DO
DO
  xend = x + xout
  IF (xend > xf) THEN xend = xf
  h = dx
  CALL Integrator (x, y, n, h, xend)
  m = m + 1
  xp_m = x
  DOFOR i = 1, n
    yp_i,m = y_i
  END DO
  IF (x ≥ xf) EXIT
END DO
DISPLAY RESULTS
END
```

**(b) Routine to Take One Output Step**

```
SUB Integrator (x, y, n, h, xend)
  DO
    IF (xend − x < h) THEN h = xend − x
    CALL RK4 (x, y, n, h)
    IF (x ≥ xend) EXIT
  END DO
END SUB
```

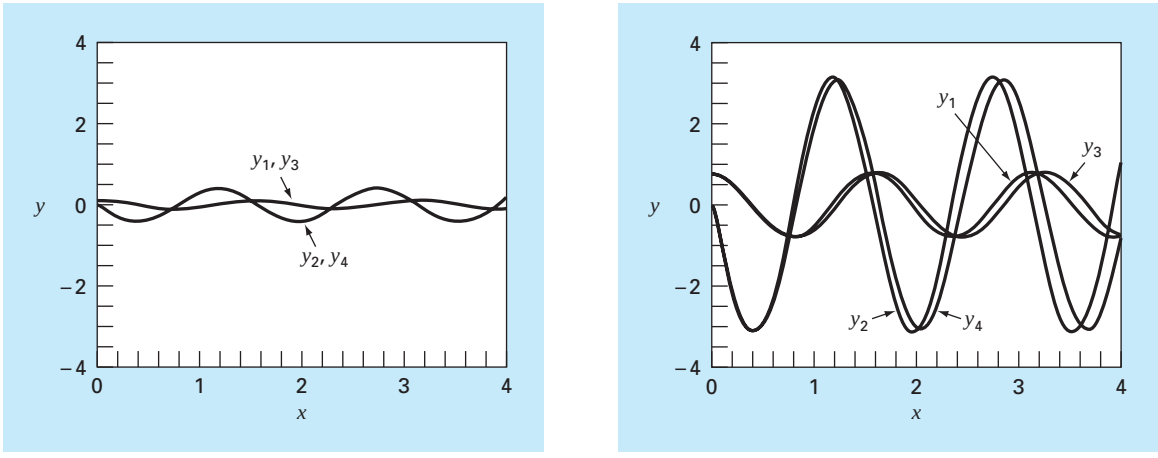
**(c) Fourth-Order RK Method for a System of ODEs**

```
SUB RK4 (x, y, n, h)
  CALL Derivs (x, y, k1)
  DOFOR i = 1, n
    ym_i = y_i + k1_i * h / 2
  END DO
  CALL Derivs (x + h / 2, ym, k2)
  DOFOR i = 1, n
    ym_i = y_i + k2_i * h / 2
  END DO
  CALL Derivs (x + h / 2, ym, k3)
  DOFOR i = 1, n
    ye_i = y_i + k3_i * h
  END DO
  CALL Derivs (x + h, ye, k4)
  DOFOR i = 1, n
    slope_i = (k1_i + 2*(k2_i+k3_i)+k4_i)/6
    y_i = y_i + slope_i * h
  END DO
  x = x + h
END SUB
```

**(d) Routine to Determine Derivatives**

```
SUB Derivs (x, y, dy)
  dy_1 = . . .
  dy_2 = . . .
END SUB
```

**FIGURE 25.18**
Pseudocode for the fourth-order RK method for systems.




(*a*) (*b*)

**FIGURE 25.19**
Solutions obtained with a computer program for the fourth-order RK method. The plots represent solutions for both linear and nonlinear pendulums with (*a*) small and (*b*) large initial displacements.

Solution.

**(a)** The calculated results for the linear and nonlinear models are almost identical (Fig. 25.19*a*). This is as expected because when the initial displacement is small, $\sin(\theta) \cong \theta$.

**(b)** When the initial displacement is $\pi/4 = 0.785398$, the solutions are much different and the difference is magnified as time becomes larger and larger (Fig. 25.19*b*). This is expected because the assumption that $\sin(\theta) = \theta$ is poor when theta is large.

## 25.5 ADAPTIVE RUNGE-KUTTA METHODS

To this point, we have presented methods for solving ODEs that employ a constant step size. For a significant number of problems, this can represent a serious limitation. For example, suppose that we are integrating an ODE with a solution of the type depicted in Fig. 25.20. For most of the range, the solution changes gradually. Such behavior suggests that a fairly large step size could be employed to obtain adequate results. However, for a localized region from $x = 1.75$ to $x = 2.25$, the solution undergoes an abrupt change. The practical consequence of dealing with such functions is that a very small step size would be required to accurately capture the impulsive behavior. If a constant step-size algorithm were employed, the smaller step size required for the region of abrupt change would have to be applied to the entire computation. As a consequence, a much smaller step size than necessary—and, therefore, many more calculations—would be wasted on the regions of gradual change.

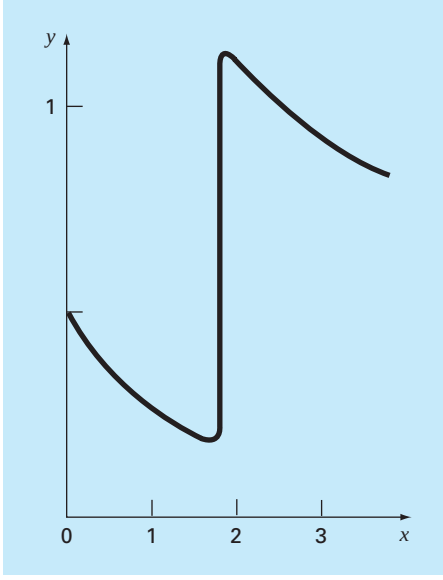



**FIGURE 25.20**
An example of a solution of an ODE that exhibits an abrupt change. Automatic step-size adjustment has great advantages for such cases.

Algorithms that automatically adjust the step size can avoid such overkill and hence be of great advantage. Because they "adapt" to the solution's trajectory, they are said to have *adaptive step-size control*. Implementation of such approaches requires that an estimate of the local truncation error be obtained at each step. This error estimate can then serve as a basis for either lengthening or decreasing the step size.

Before proceeding, we should mention that aside from solving ODEs, the methods described in this chapter can also be used to evaluate definite integrals. As mentioned previously in the introduction to Part Six, the evaluation of the integral

$$I = \int_a^b f(x)\,dx$$

is equivalent to solving the differential equation

$$\frac{dy}{dx} = f(x)$$

for $y(b)$ given the initial condition $y(a) = 0$. Thus, the following techniques can be employed to efficiently evaluate definite integrals involving functions that are generally smooth but exhibit regions of abrupt change.

There are two primary approaches to incorporate adaptive step-size control into one-step methods. In the first, the error is estimated as the difference between two predictions using the same-order RK method but with different step sizes. In the second, the local

truncation error is estimated as the difference between two predictions using different-order RK methods.

### 25.5.1 Adaptive RK or Step-Halving Method

*Step halving* (also called *adaptive RK*) involves taking each step twice, once as a full step and independently as two half steps. The difference in the two results represents an estimate of the local truncation error. If $y_1$ designates the single-step prediction and $y_2$ designates the prediction using the two half steps, the error $\Delta$ can be represented as

$$\Delta = y_2 - y_1 \tag{25.43}$$

In addition to providing a criterion for step-size control, Eq. (25.43) can also be used to correct the $y_2$ prediction. For the fourth-order RK version, the correction is

$$y_2 \leftarrow y_2 + \frac{\Delta}{15} \tag{25.44}$$

This estimate is fifth-order accurate.

EXAMPLE 25.12 Adaptive Fourth-Order RK Method

Problem Statement. Use the adaptive fourth-order RK method to integrate $y' = 4e^{0.8x} - 0.5y$ from $x = 0$ to 2 using $h = 2$ and an initial condition of $y(0) = 2$. This is the same differential equation that was solved previously in Example 25.5. Recall that the true solutions is $y(2) = 14.84392$.

Solution. The single prediction with a step of $h$ is computed as

$$y(2) = 2 + \frac{1}{6}[3 + 2(6.40216 + 4.70108) + 14.11105]2 = 15.10584$$

The two half-step predictions are

$$y(1) = 2 + \frac{1}{6}[3 + 2(4.21730 + 3.91297) + 5.945681]1 = 6.20104$$

and

$$y(2) = 6.20104 + \frac{1}{6}[5.80164 + 2(8.72954 + 7.99756) + 12.71283]1 = 14.86249$$

Therefore, the approximate error is

$$E_a = \frac{14.86249 - 15.10584}{15} = -0.01622$$

which compares favorably with the true error of

$$E_t = 14.84392 - 14.86249 = -0.01857$$

The error estimate can also be used to correct the prediction

$$y(2) = 14.86249 - 0.01622 = 14.84627$$

which has an $E_t = -0.00235$.

### 25.5.2 Runge-Kutta Fehlberg

Aside from step halving as a strategy to adjust step size, an alternative approach for obtaining an error estimate involves computing two RK predictions of different order. The results can then be subtracted to obtain an estimate of the local truncation error. One shortcoming of this approach is that it greatly increases the computational overhead. For example, a fourth- and fifth-order prediction amount to a total of 10 function evaluations per step. The *Runge-Kutta Fehlberg* or *embedded RK* method cleverly circumvents this problem by using a fifth-order RK method that employs the function evaluations from the accompanying fourth-order RK method. Thus, the approach yields the error estimate on the basis of only six function evaluations!

For the present case, we use the following fourth-order estimate

$$y_{i+1} = y_i + \left(\frac{37}{378}k_1 + \frac{250}{621}k_3 + \frac{125}{594}k_4 + \frac{512}{1771}k_6\right)h \tag{25.45}$$

along with the fifth-order formula:

$$y_{i+1} = y_i + \left(\frac{2825}{27{,}648}k_1 + \frac{18{,}575}{48{,}384}k_3 + \frac{13{,}525}{55{,}296}k_4 + \frac{277}{14{,}336}k_5 + \frac{1}{4}k_6\right)h \tag{25.46}$$

where

$$k_1 = f(x_i, y_i)$$

$$k_2 = f\left(x_i + \frac{1}{5}h, y_i + \frac{1}{5}k_1h\right)$$

$$k_3 = f\left(x_i + \frac{3}{10}h, y_i + \frac{3}{40}k_1h + \frac{9}{40}k_2h\right)$$

$$k_4 = f\left(x_i + \frac{3}{5}h, y_i + \frac{3}{10}k_1h - \frac{9}{10}k_2h + \frac{6}{5}k_3h\right)$$

$$k_5 = f\left(x_i + h, y_i - \frac{11}{54}k_1h + \frac{5}{2}k_2h - \frac{70}{27}k_3h + \frac{35}{27}k_4h\right)$$

$$k_6 = f\left(x_i + \frac{7}{8}h, y_i + \frac{1631}{55{,}296}k_1h + \frac{175}{512}k_2h + \frac{575}{13{,}824}k_3h + \frac{44{,}275}{110{,}592}k_4h + \frac{253}{4096}k_5h\right)$$

Thus, the ODE can be solved with Eq. (25.46) and the error estimated as the difference of the fifth- and fourth-order estimates. It should be noted that the particular coefficients used above were developed by Cash and Karp (1990). Therefore, it is sometimes called the *Cash-Karp* RK method.

EXAMPLE 25.13 Runge-Kutta Fehlberg Method

Problem Statement. Use the Cash-Karp version of the Runge-Kutta Fehlberg approach to perform the same calculation as in Example 25.12 from $x = 0$ to 2 using $h = 2$.

Solution. The calculation of the $k$'s can be summarized in the following table:

| | $x$ | $y$ | $f(x, y)$ |
|---|---|---|---|
| $k_1$ | 0 | 2 | 3 |
| $k_2$ | 0.4 | 3.2 | 3.908511 |
| $k_3$ | 0.6 | 4.20883 | 4.359883 |
| $k_4$ | 1.2 | 7.228398 | 6.832587 |
| $k_5$ | 2 | 15.42765 | 12.09831 |
| $k_6$ | 1.75 | 12.17686 | 10.13237 |

These can then be used to compute the fourth-order prediction

$$y_1 = 2 + \left(\frac{37}{378}3 + \frac{250}{621}4.359883 + \frac{125}{594}6.832587 + \frac{512}{1771}10.13237\right)2 = 14.83192$$

along with a fifth-order formula:

$$y_1 = 2 + \left(\frac{2825}{27{,}648}3 + \frac{18{,}575}{48{,}384}4.359883 + \frac{13{,}525}{55{,}296}6.832587 + \frac{277}{14{,}336}12.09831 + \frac{1}{4}10.13237\right)2 = 14.83677$$

The error estimate is obtained by subtracting these two equations to give

$$E_a = 14.83677 - 14.83192 = 0.004842$$

### 25.5.3 Step-Size Control

Now that we have developed ways to estimate the local truncation error, it can be used to adjust the step size. In general, the strategy is to increase the step size if the error is too small and decrease it if the error is too large. Press et al. (1992) have suggested the following criterion to accomplish this:

$$h_{\text{new}} = h_{\text{present}} \left| \frac{\Delta_{\text{new}}}{\Delta_{\text{present}}} \right|^{\alpha} \tag{25.47}$$

where $h_{\text{present}}$ and $h_{\text{new}}$ = the present and the new step sizes, respectively, $\Delta_{\text{present}}$ = the computed present accuracy, $\Delta_{\text{new}}$ = the desired accuracy, and $\alpha$ = a constant power that is equal to 0.2 when the step size is increased (that is, when $\Delta_{\text{present}} \leq \Delta_{\text{new}}$) and 0.25 when the step size is decreased ($\Delta_{\text{present}} > \Delta_{\text{new}}$).

The key parameter in Eq. (25.47) is obviously $\Delta_{\text{new}}$ because it is your vehicle for specifying the desired accuracy. One way to do this would be to relate $\Delta_{\text{new}}$ to a relative error level. Although this works well when only positive values occur, it can cause problems for solutions that pass through zero. For example, you might be simulating an oscillating function that repeatedly passes through zero but is bounded by maximum absolute values. For such a case, you might want these maximum values to figure in the desired accuracy.

A more general way to handle such cases is to determine $\Delta_{new}$ as

$$\Delta_{new} = \varepsilon y_{scale}$$

where $\varepsilon$ = an overall tolerance level. Your choice of $y_{scale}$ will then determine how the error is scaled. For example, if $y_{scale} = y$, the accuracy will be couched in terms of fractional relative errors. If you are dealing with a case where you desire constant errors relative to a prescribed maximum bound, set $y_{scale}$ equal to that bound. A trick suggested by Press et al. (1992) to obtain the constant relative errors except very near zero crossings is

$$y_{scale} = |y| + \left| h\frac{dy}{dx} \right|$$

This is the version we will use in our algorithm.

### 25.5.4 Computer Algorithm

Figures 25.21 and 25.22 outline pseudocode to implement the Cash-Karp version of the Runge-Kutta Fehlberg algorithm. This algorithm is patterned after a more detailed implementation by Press et al. (1992) for systems of ODEs.

Figure 25.21 implements a single step of the Cash-Karp routine (that is Eqs. 25.45 and 25.46). Figure 25.22 outlines a general driver program along with a subroutine that actually adapts the step size.

**FIGURE 25.21**
Pseudocode for a single step of the Cash-Karp RK method.

```
SUBROUTINE RKkc (y,dy,x,h,yout,yerr)
PARAMETER (a2=0.2,a3=0.3,a4=0.6,a5=1.,a6=0.875,
  b21=0.2,b31=3./40.,b32=9./40.,b41=0.3,b42=-0.9,
  b43=1.2,b51=-11./54.,b52=2.5,b53=-70./27.,
  b54=35./27.,b61=1631./55296.,b62=175./512.,
  b63=575./13824.,b64=44275./110592.,b65=253./4096.,
  c1=37./378.,c3=250./621.,c4=125./594.,
  c6=512./1771.,dc1=c1-2825./27648.,
  dc3=c3-18575./48384.,dc4=c4-13525./55296.,
  dc5=-277./14336.,dc6=c6-0.25)
ytemp=y+b21*h*dy
CALL Derivs (x+a2*h,ytemp,k2)
ytemp=y+h*(b31*dy+b32*k2)
CALL Derivs(x+a3*h,ytemp,k3)
ytemp=y+h*(b41*dy+b42*k2+b43*k3)
CALL Derivs(x+a4*h,ytemp,k4)
ytemp=y+h*(b51*dy+b52*k2+b53*k3+b54*k4)
CALL Derivs(x+a5*h,ytemp,k5)
ytemp=y+h*(b61*dy+b62*k2+b63*k3+b64*k4+b65*k5)
CALL Derivs(x+a6*h,ytemp,k6)
yout=y+h*(c1*dy+c3*k3+c4*k4+c6*k6)
yerr=h*(dc1*dy+dc3*k3+dc4*k4+dc5*k5+dc6*k6)
END RKkc
```

**(a) Driver Program**

```
INPUT xi, xf, yi
maxstep=100
hi=.5; tiny = 1. × 10^-30
eps=0.00005
print *, xi,yi
x=xi
y=yi
h=hi
istep=0
DO
  IF (istep > maxstep AND x ≤ xf) EXIT
  istep=istep+1
  CALL Derivs(x,y,dy)
  yscal=ABS(y)+ABS(h*dy)+tiny
  IF (x+h>xf) THEN h=xf-x
  CALL Adapt (x,y,dy,h,yscal,eps,hnxt)
  PRINT x,y
  h=hnxt
END DO
END
```

**(b) Adaptive Step Routine**

```
SUB Adapt (x,y,dy,htry,yscal,eps,hnxt)
PARAMETER (safety=0.9,econ=1.89e-4)
h=htry
DO
  CALL RKkc(y,dy,x,h,ytemp,yerr)
  emax=abs(yerr/yscal/eps)
  IF emax ≤ 1 EXIT
  htemp=safety*h*emax^-0.25
  h=max(abs(htemp),0.25*abs(h))
  xnew=x+h
  IF xnew = x THEN pause
END DO
IF emax > econ THEN
  hnxt=safety*emax^-.2*h
ELSE
  hnxt=4.*h
END IF
x=x+h
y=ytemp
END Adapt
```

**FIGURE 25.22**
Pseudocode for a (*a*) driver program and an (*b*) adaptive step routine to solve a single ODE.

EXAMPLE 25.14 Computer Application of an Adaptive Fourth-Order RK Scheme

Problem Statement. The adaptive RK method is well-suited for the following ordinary differential equation

$$\frac{dy}{dx} + 0.6y = 10e^{-(x-2)^2/[2(0.075)^2]} \tag{E25.14.1}$$

Notice for the initial condition, $y(0) = 0.5$, the general solution is

$$y = 0.5e^{-0.6x} \tag{E25.14.2}$$

which is a smooth curve that gradually approaches zero as $x$ increases. In contrast, the particular solution undergoes an abrupt transition in the vicinity of $x = 2$ due to the nature of the forcing function (Fig. 25.23*a*). Use a standard fourth-order RK scheme to solve Eq. (E25.14.1) from $x = 0$ to 4. Then employ the adaptive scheme described in this section to perform the same computation.

Solution. First, the classical fourth-order scheme is used to compute the solid curve in Fig. 25.23*b*. For this computation, a step size of 0.1 is used so that $4/(0.1) = 40$ applications of the technique are made. Then, the calculation is repeated with a step size of 0.05 for a total of 80 applications. The major discrepancy between the two results occurs in the region from 1.8 to 2.0. The magnitude of the discrepancy is about 0.1 to 0.2 percent.

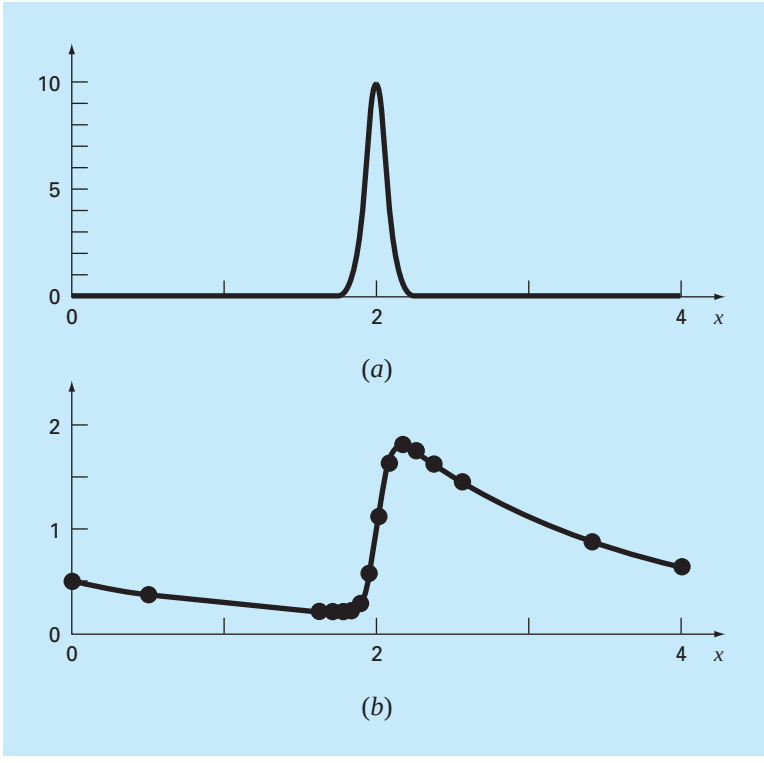



**FIGURE 25.23**
(*a*) A bell-shaped forcing function that induces an abrupt change in the solution of an ODE [Eq. (E25.14.1)]. (*b*) The solution. The points indicate the predictions of an adaptive step-size routine.

Next, the algorithm in Figs. 25.21 and 25.22 is developed into a computer program and used to solve the same problem. An initial step size of 0.5 and an $\varepsilon = 0.00005$ were chosen. The results were superimposed on Fig. 25.23*b*. Notice how large steps are taken in the regions of gradual change. Then, in the vicinity of $x = 2$, the steps are decreased to accommodate the abrupt nature of the forcing function.

The utility of an adaptive integration scheme obviously depends on the nature of the functions being modeled. It is particularly advantageous for those solutions with long smooth stretches and short regions of abrupt change. In addition, it has utility in those situations where the correct step size is not known a priori. For these cases, an adaptive routine will "feel" its way through the solution while keeping the results within the desired tolerance. Thus, it will tiptoe through the regions of abrupt change and step out briskly when the variations become more gradual.

## PROBLEMS

**25.1** Solve the following initial value problem over the interval from $t = 0$ to 2 where $y(0) = 1$. Display all your results on the same graph.

$$\frac{dy}{dt} = yt^3 - 1.5y$$

**(a)** Analytically.
**(b)** Euler's method with $h = 0.5$ and 0.25.
**(c)** Midpoint method with $h = 0.5$.
**(d)** Fourth-order RK method with $h = 0.5$.

**25.2** Solve the following problem over the interval from $x = 0$ to 1 using a step size of 0.25 where $y(0) = 1$. Display all your results on the same graph.

$$\frac{dy}{dx} = (1 + 2x)\sqrt{y}$$

**(a)** Analytically.
**(b)** Euler's method.
**(c)** Heun's method without the corrector.
**(d)** Ralston's method.
**(e)** Fourth-order RK method.

**25.3** Use the **(a)** Euler and **(b)** Heun (without iteration) methods to solve

$$\frac{d^2y}{dt^2} - t + y = 0$$

where $y(0) = 2$ and $y'(0) = 0$. Solve from $x = 0$ to 4 using $h = 0.1$. Compare the methods by plotting the solutions.

**25.4** Solve the following problem with the fourth-order RK method:

$$\frac{d^2y}{dx^2} + 0.5\frac{dy}{dx} + 7y = 0$$

where $y(0) = 4$ and $y'(0) = 0$. Solve from $x = 0$ to 5 with $h = 0.5$. Plot your results.

**25.5** Solve from $t = 0$ to 3 with $h = 0.1$ using **(a)** Heun (without corrector) and **(b)** Ralston's 2nd-order RK method:

$$\frac{dy}{dt} = y\sin^3(t) \qquad y(0) = 1$$

**25.6** Solve the following problem numerically from $t = 0$ to 3:

$$\frac{dy}{dt} = -y + t^2 \qquad y(0) = 1$$

Use the third-order RK method with a step size of 0.5.

**25.7** Use **(a)** Euler's and **(b)** the fourth-order RK method to solve

$$\frac{dy}{dx} = -2y + 4e^{-x}$$
$$\frac{dz}{dx} = -\frac{yz^2}{3}$$

over the range $x = 0$ to 1 using a step size of 0.2 with $y(0) = 2$ and $z(0) = 4$.

**25.8** Compute the first step of Example 25.14 using the adaptive fourth-order RK method with $h = 0.5$. Verify whether step-size adjustment is in order.

**25.9** If $\varepsilon = 0.001$, determine whether step size adjustment is required for Example 25.12.

**25.10** Use the RK-Fehlberg approach to perform the same calculation as in Example 25.12 from $x = 0$ to 1 with $h = 1$.

**25.11** Write a computer program based on Fig. 25.7. Among other things, place documentation statements throughout the program to identify what each section is intended to accomplish.

**25.12** Test the program you developed in Prob. 25.11 by duplicating the computations from Examples 25.1 and 25.4.

**25.13** Develop a user-friendly program for the Heun method with an iterative corrector. Test the program by duplicating the results in Table 25.2.

**25.14** Develop a user-friendly computer program for the classical fourth-order RK method. Test the program by duplicating Example 25.7.

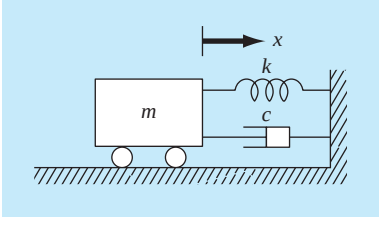
**25.15** Develop a user-friendly computer program for systems of equations using the fourth-order RK method. Use this program to duplicate the computation in Example 25.10.

**25.16** The motion of a damped spring-mass system (Fig. P25.16) is described by the following ordinary differential equation:

$$m\frac{d^2x}{dt^2} + c\frac{dx}{dt} + kx = 0$$

where $x$ = displacement from equilibrium position (m), $t$ = time (s), $m$ = 20-kg mass, and $c$ = the damping coefficient (N · s/m). The damping coefficient $c$ takes on three values of 5 (underdamped), 40 (critically damped), and 200 (overdamped). The spring constant $k = 20$ N/m. The initial velocity is zero, and the initial displacement $x = 1$ m. Solve this equation using a numerical method over the time period $0 \le t \le 15$ s. Plot the displacement versus time for each of the three values of the damping coefficient on the same curve.

**Figure P25.16**



**25.17** If water is drained from a vertical cylindrical tank by opening a valve at the base, the water will flow fast when the tank is full and slow down as it continues to drain. As it turns out, the rate at which the water level drops is:

$$\frac{dy}{dt} = -k\sqrt{y}$$

where $k$ is a constant depending on the shape of the hole and the cross-sectional area of the tank and drain hole. The depth of the water $y$ is measured in meters and the time $t$ in minutes. If $k = 0.06$, determine how long it takes the tank to drain if the fluid level is initially 3 m. Solve by applying Euler's equation and writing a computer program or using Excel. Use a step of 0.5 minutes.

**25.18** The following is an initial value, second-order differential equation:

$$\frac{d^2x}{dt^2} + (5x)\frac{dx}{dt} + (x + 7)\sin(\omega t) = 0$$

where

$$\frac{dx}{dt}(0) = 1.5 \quad \text{and} \quad x(0) = 6$$

Note that $\omega = 1$. Decompose the equation into two first-order differential equations. After the decomposition, solve the system from $t = 0$ to 15 and plot the results.

**25.19** Assuming that drag is proportional to the square of velocity, we can model the velocity of a falling object like a parachutist with the following differential equation:

$$\frac{dv}{dt} = g - \frac{c_d}{m}v^2$$

where $v$ is velocity (m/s), $t$ = time (s), $g$ is the acceleration due to gravity (9.81 m/s$^2$), $c_d$ = a second-order drag coefficient (kg/m), and $m$ = mass (kg). Solve for the velocity and distance fallen by a 90-kg object with a drag coefficient of 0.225 kg/m. If the initial height is 1 km, determine when it hits the ground. Obtain your solution with **(a)** Euler's method and **(b)** the fourth-order RK method.

**25.20** A spherical tank has a circular orifice in its bottom through which the liquid flows out (Fig. P25.20). The flow rate through the hole can be estimated as

$$Q_{\text{out}} = CA\sqrt{2gH}$$

where $Q_{\text{out}}$ = outflow (m$^3$/s), $C$ = an empirically-derived coefficient, $A$ = the area of the orifice (m$^2$), $g$ = the gravitational constant (= 9.81 m/s$^2$), and $H$ = the depth of liquid in the tank. Use one of the numerical methods described in this chapter to determine how long it will take for the water to flow out of a 3-m diameter tank with an initial height of 2.75 m. Note that the orifice has a diameter of 3 cm and $C = 0.55$.

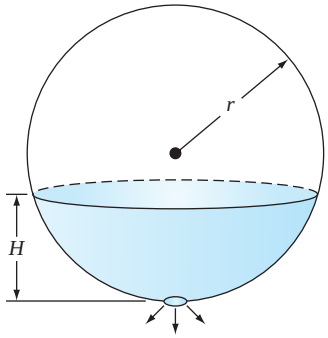



**Figure P25.20**
A spherical tank.

**25.21** The logistic model is used to simulate population as in

$$\frac{dp}{dt} = k_{gm}(1 - p/p_{\max})p$$

where $p$ = population, $k_{gm}$ = the maximum growth rate under unlimited conditions, and $p_{\max}$ = the carrying capacity. Simulate the world's population from 1950 to 2000 using one of the numerical methods described in this chapter. Employ the following initial conditions and parameter values for your simulation: $p_0$ (in 1950) = 2555 million people, $k_{gm}$ = 0.026/yr, and $p_{\max}$ = 12,000 million people. Have the function generate output corresponding to the dates for the following measured population data. Develop a plot of your simulation along with the data.

| $t$ | 1950 | 1960 | 1970 | 1980 | 1990 | 2000 |
|---|---|---|---|---|---|---|
| $p$ | 2555 | 3040 | 3708 | 4454 | 5276 | 6079 |

**25.22** Suppose that a projectile is launched upward from the earth's surface. Assume that the only force acting on the object is the downward force of gravity. Under these conditions, a force balance can be used to derive,

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

where $v$ = upward velocity (m/s), $t$ = time (s), $x$ = altitude (m) measured upwards from the earth's surface, $g(0)$ = the gravitational acceleration at the earth's surface ($\cong$ 9.81 m/s$^2$), and $R$ = the earth's radius ($\cong$ 6.37 $\times$ 10$^6$ m). Recognizing that $dx/dt = v$, use Euler's method to determine the maximum height that would be obtained if $v(t = 0) = 1400$ m/s.

**25.23** The following function exhibits both flat and steep regions over a relatively short $x$ region:

$$f(x) = \frac{1}{(x - 0.3)^2 + 0.01} + \frac{1}{(x - 0.9)^2 + 0.04} - 6$$

Determine the value of the definite integral of this function between $x = 0$ and 1 using an adaptive RK method.

CHAPTER 26

# Stiffness and Multistep Methods

This chapter covers two areas. First, we describe *stiff ODEs*. These are both individual and systems of ODEs that have both fast and slow components to their solution. We introduce the idea of an *implicit solution* technique as one commonly used remedy for this problem. Then we discuss *multistep methods*. These algorithms retain information of previous steps to more effectively capture the trajectory of the solution. They also yield the truncation error estimates that can be used to implement adaptive step-size control.

## 26.1 STIFFNESS

Stiffness is a special problem that can arise in the solution of ordinary differential equations. A *stiff system* is one involving rapidly changing components together with slowly changing ones. In many cases, the rapidly varying components are ephemeral transients that die away quickly, after which the solution becomes dominated by the slowly varying components. Although the transient phenomena exist for only a short part of the integration interval, they can dictate the time step for the entire solution.

Both individual and systems of ODEs can be stiff. An example of a single stiff ODE is

$$\frac{dy}{dt} = -1000y + 3000 - 2000e^{-t} \tag{26.1}$$

If $y(0) = 0$, the analytical solution can be developed as

$$y = 3 - 0.998e^{-1000t} - 2.002e^{-t} \tag{26.2}$$

As in Fig. 26.1, the solution is initially dominated by the fast exponential term ($e^{-1000t}$). After a short period ($t < 0.005$), this transient dies out and the solution becomes dictated by the slow exponential ($e^{-t}$).

Insight into the step size required for stability of such a solution can be gained by examining the homogeneous part of Eq. (26.1),

$$\frac{dy}{dt} = -ay \tag{26.3}$$

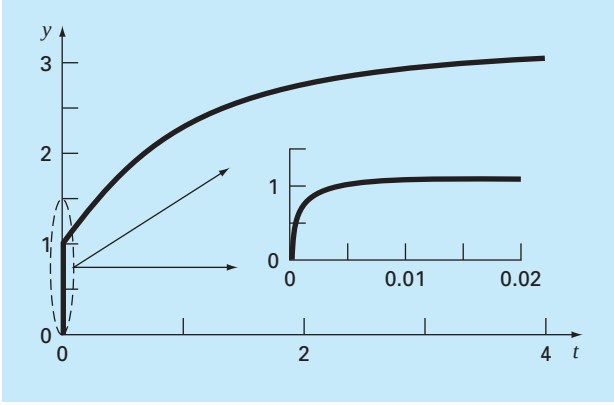



**FIGURE 26.1**
Plot of a stiff solution of a single ODE. Although the solution appears to start at 1, there is actually a fast transient from $y = 0$ to 1 that occurs in less than 0.005 time unit. This transient is perceptible only when the response is viewed on the finer timescale in the inset.

If $y(0) = y_0$, calculus can be used to determine the solution as

$$y = y_0 e^{-at}$$

Thus, the solution starts at $y_0$ and asymptotically approaches zero.

Euler's method can be used to solve the same problem numerically:

$$y_{i+1} = y_i + \frac{dy_i}{dt} h$$

Substituting Eq. (26.3) gives

$$y_{i+1} = y_i - a y_i h$$

or

$$y_{i+1} = y_i (1 - ah) \tag{26.4}$$

The stability of this formula clearly depends on the step size $h$. That is, $|1 - ah|$ must be less than 1. Thus, if $h > 2/a$, $|y_i| \to \infty$ as $i \to \infty$.

For the fast transient part of Eq. (26.2), this criterion can be used to show that the step size to maintain stability must be $< 2/1000 = 0.002$. In addition, it should be noted that, whereas this criterion maintains stability (that is, a bounded solution), an even smaller step size would be required to obtain an accurate solution. Thus, although the transient occurs for only a small fraction of the integration interval, it controls the maximum allowable step size.

Superficially, you might suppose that the adaptive step-size routines described at the end of the last chapter might offer a solution for this dilemma. You might think that they would use small steps during the rapid transients and large steps otherwise. However, this is not the case, because the stability requirement will still necessitate using very small steps throughout the entire solution.

Rather than using explicit approaches, implicit methods offer an alternative remedy. Such representations are called *implicit* because the unknown appears on both sides of the equation. An implicit form of Euler's method can be developed by evaluating the derivative at the future time,

$$y_{i+1} = y_i + \frac{dy_{i+1}}{dt}h$$

This is called the *backward*, or *implicit, Euler's method*. Substituting Eq. (26.3) yields

$$y_{i+1} = y_i - ay_{i+1}h$$

which can be solved for

$$y_{i+1} = \frac{y_i}{1 + ah} \tag{26.5}$$

For this case, regardless of the size of the step, $|y_i| \to 0$ as $i \to \infty$. Hence, the approach is called *unconditionally stable.*

EXAMPLE 26.1 Explicit and Implicit Euler

Problem Statement. Use both the explicit and implicit Euler methods to solve

$$\frac{dy}{dt} = -1000y + 3000 - 2000e^{-t}$$

where $y(0) = 0$. **(a)** Use the explicit Euler with step sizes of 0.0005 and 0.0015 to solve for $y$ between $t = 0$ and 0.006. **(b)** Use the implicit Euler with a step size of 0.05 to solve for $y$ between 0 and 0.4.

Solution.

**(a)** For this problem, the explicit Euler's method is

$$y_{i+1} = y_i + (-1000y_i + 3000 - 2000e^{-t_i})h$$

The result for $h = 0.0005$ is displayed in Fig. 26.2*a* along with the analytical solution. Although it exhibits some truncation error, the result captures the general shape of the analytical solution. In contrast, when the step size is increased to a value just below the stability limit ($h = 0.0015$), the solution manifests oscillations. Using $h > 0.002$ would result in a totally unstable solution, that is, it would go infinite as the solution progressed.

**(b)** The implicit Euler's method is

$$y_{i+1} = y_i + (-1000y_{i+1} + 3000 - 2000e^{-t_{i+1}})h$$

Now because the ODE is linear, we can rearrange this equation so that $y_{i+1}$ is isolated on the left-hand side,

$$y_{i+1} = \frac{y_i + 3000h - 2000he^{-t_{i+1}}}{1 + 1000h}$$

The result for $h = 0.05$ is displayed in Fig. 26.2*b* along with the analytical solution. Notice that even though we have used a much bigger step size than the one that

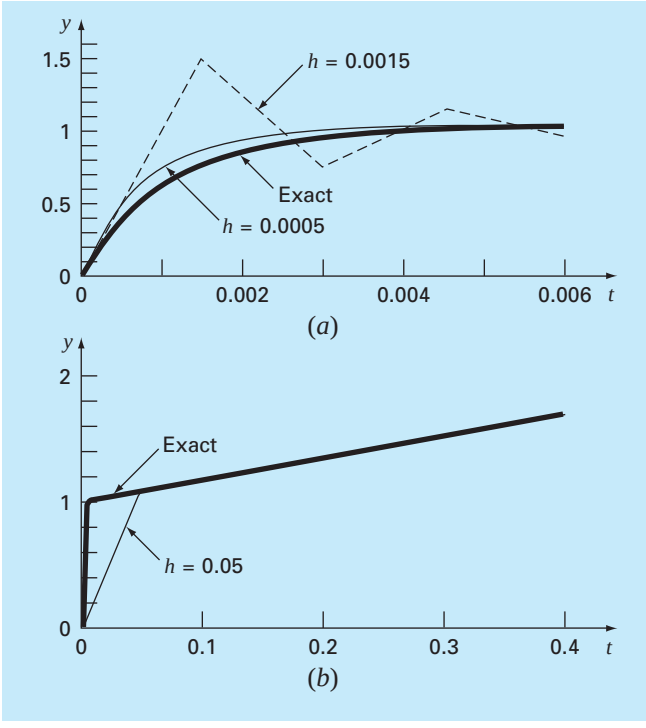



**FIGURE 26.2**
Solution of a "stiff" ODE with (*a*) the explicit and (*b*) implicit Euler methods.

induced instability for the explicit Euler, the numerical solution tracks nicely on the analytical result.

Systems of ODEs can also be stiff. An example is

$$\frac{dy_1}{dt} = -5y_1 + 3y_2 \tag{26.6a}$$

$$\frac{dy_2}{dt} = 100y_1 - 301y_2 \tag{26.6b}$$

For the initial conditions $y_1(0) = 52.29$ and $y_2(0) = 83.82$, the exact solution is

$$y_1 = 52.96e^{-3.9899t} - 0.67e^{-302.0101t} \tag{26.7a}$$

$$y_2 = 17.83e^{-3.9899t} + 65.99e^{-302.0101t} \tag{26.7b}$$

Note that the exponents are negative and differ by about 2 orders of magnitude. As with the single equation, it is the large exponents that respond rapidly and are at the heart of the system's stiffness.

An implicit Euler's method for systems can be formulated for the present example as

$$y_{1,i+1} = y_{1,i} + (-5y_{1,i+1} + 3y_{2,i+1})h \tag{26.8a}$$

$$y_{2,i+1} = y_{2,i} + (100y_{1,i+1} - 301y_{2,i+1})h \tag{26.8b}$$

Collecting terms gives

$$(1 + 5h)y_{1,i+1} - 3hy_{2,i+1} = y_{1,i} \tag{26.9a}$$

$$-100hy_{1,i+1} + (1 + 301h)y_{2,i+1} = y_{2,i} \tag{26.9b}$$

Thus, we can see that the problem consists of solving a set of simultaneous equations for each time step.

For nonlinear ODEs, the solution becomes even more difficult since it involves solving a system of nonlinear simultaneous equations (recall Sec. 6.5). Thus, although stability is gained through implicit approaches, a price is paid in the form of added solution complexity.

The implicit Euler method is unconditionally stable and only first-order accurate. It is also possible to develop in a similar manner a second-order accurate implicit trapezoidal rule integration scheme for stiff systems. It is usually desirable to have higher-order methods. The Adams-Moulton formulas described later in this chapter can also be used to devise higher-order implicit methods. However, the stability limits of such approaches are very stringent when applied to stiff systems. Gear (1971) developed a special series of implicit schemes that have much larger stability limits based on backward difference formulas. Extensive efforts have been made to develop software to efficiently implement Gear's methods. As a result, this is probably the most widely used method to solve stiff systems. In addition, Rosenbrock and others (see Press et al., 1992) have proposed implicit Runge-Kutta algorithms where the $k$ terms appear implicitly. These methods have good stability characteristics and are quite suitable for solving systems of stiff ordinary differential equations.

## 26.2 MULTISTEP METHODS

The one-step methods described in the previous sections utilize information at a single point $x_i$ to predict a value of the dependent variable $y_{i+1}$ at a future point $x_{i+1}$ (Fig. 26.3*a*). Alternative approaches, called *multistep methods* (Fig. 26.3*b*), are based on the insight that, once the computation has begun, valuable information from previous points is at our command. The curvature of the lines connecting these previous values provides information regarding the trajectory of the solution. The multistep methods explored in this chapter exploit this information to solve ODEs. Before describing the higher-order versions, we will present a simple second-order method that serves to demonstrate the general characteristics of multistep approaches.

### 26.2.1 The Non-Self-Starting Heun Method

Recall that the Heun approach uses *Euler's method* as a *predictor* [Eq. (25.15)]:

$$y_{i+1}^0 = y_i + f(x_i, y_i)h \tag{26.10}$$

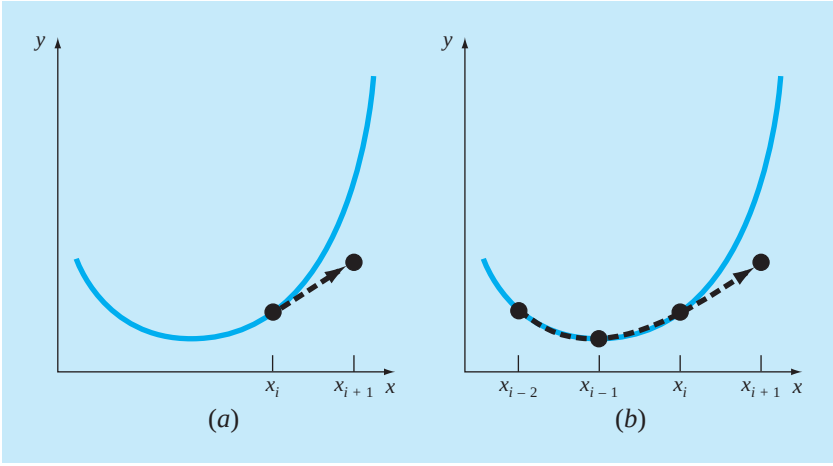



**FIGURE 26.3**
Graphical depiction of the fundamental difference between (*a*) one-step and (*b*) multistep methods for solving ODEs.

and the *trapezoidal rule* as a *corrector* [Eq. (25.16)]:

$$y_{i+1} = y_i + \frac{f(x_i, y_i) + f\left(x_{i+1}, y_{i+1}^0\right)}{2}h \tag{26.11}$$

Thus, the predictor and the corrector have local truncation errors of $O(h^2)$ and $O(h^3)$, respectively. This suggests that the predictor is the weak link in the method because it has the greatest error. This weakness is significant because the efficiency of the iterative corrector step depends on the accuracy of the initial prediction. Consequently, one way to improve Heun's method is to develop a predictor that has a local error of $O(h^3)$. This can be accomplished by using Euler's method and the slope at $y_i$, and extra information from a previous point $y_{i-1}$, as in

$$y_{i+1}^0 = y_{i-1} + f(x_i, y_i)2h \tag{26.12}$$

Notice that Eq. (26.12) attains $O(h^3)$ at the expense of employing a larger step size, $2h$. In addition, note that Eq. (26.12) is not self-starting because it involves a previous value of the dependent variable $y_{i-1}$. Such a value would not be available in a typical initial-value problem. Because of this fact, Eqs. (26.11) and (26.12) are called the *non-self-starting Heun method.*

As depicted in Fig. 26.4, the derivative estimate in Eq. (26.12) is now located at the midpoint rather than at the beginning of the interval over which the prediction is made. As demonstrated subsequently, this centering improves the error of the predictor to $O(h^3)$. However, before proceeding to a formal derivation of the non-self-starting Heun, we will summarize the method and express it using a slightly modified nomenclature:

$$\text{Predictor:} \quad y_{i+1}^0 = y_{i-1}^m + f\left(x_i, y_i^m\right)2h \tag{26.13}$$

$$\text{Corrector:} \quad y_{i+1}^j = y_i^m + \frac{f\left(x_i, y_i^m\right) + f\left(x_{i+1}, y_{i+1}^{j-1}\right)}{2}h \quad (\text{for } j = 1, 2, \ldots, m) \tag{26.14}$$

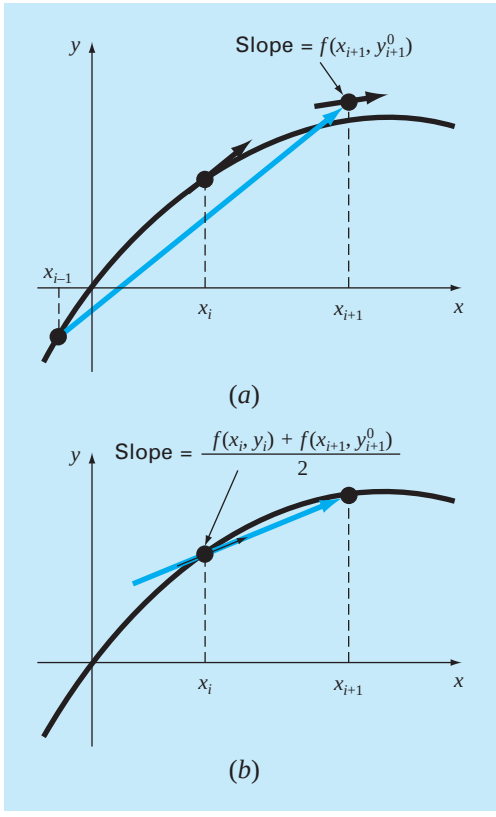



**FIGURE 26.4**
A graphical depiction of the non-self-starting Heun method. (*a*) The midpoint method that is used as a predictor. (*b*) The trapezoidal rule that is employed as a corrector.

where the superscripts have been added to denote that the corrector is applied iteratively from $j = 1$ to $m$ to obtain refined solutions. Note that $y_i^m$ and $y_{i-1}^m$ are the final results of the corrector iterations at the previous time steps. The iterations are terminated at any time step on the basis of the stopping criterion

$$|\varepsilon_a| = \left| \frac{y_{i+1}^j - y_{i+1}^{j-1}}{y_{i+1}^j} \right| 100\% \tag{26.15}$$

When $\varepsilon_a$ is less than a prespecified error tolerance $\varepsilon_s$, the iterations are terminated. At this point, $j = m$. The use of Eqs. (26.13) through (26.15) to solve an ODE is demonstrated in the following example.

EXAMPLE 26.2 Non-Self-Starting Heun Method

Problem Statement. Use the non-self-starting Heun method to perform the same computations as were performed previously in Example 25.5 using Heun's method. That is,

integrate $y' = 4e^{0.8x} - 0.5y$ from $x = 0$ to $x = 4$ using a step size of 1.0. As with Example 25.5, the initial condition at $x = 0$ is $y = 2$. However, because we are now dealing with a multistep method, we require the additional information that $y$ is equal to $-0.3929953$ at $x = -1$.

Solution. The predictor [Eq. (26.13)] is used to extrapolate linearly from $x = -1$ to $x = 1$.

$$y_1^0 = -0.3929953 + \left[4e^{0.8(0)} - 0.5(2)\right]2 = 5.607005$$

The corrector [Eq. (26.14)] is then used to compute the value:

$$y_1^1 = 2 + \frac{4e^{0.8(0)} - 0.5(2) + 4e^{0.8(1)} - 0.5(5.607005)}{2}1 = 6.549331$$

which represents a percent relative error of $-5.73$ percent (true value $= 6.194631$). This error is somewhat smaller than the value of $-8.18$ percent incurred in the self-starting Heun.

Now, Eq. (26.14) can be applied iteratively to improve the solution:

$$y_1^2 = 2 + \frac{3 + 4e^{0.8(1)} - 0.5(6.549331)}{2}1 = 6.313749$$

which represents an $\varepsilon_t$ of $-1.92\%$. An approximate estimate of the error can also be determined using Eq. (26.15):

$$|\varepsilon_a| = \left|\frac{6.313749 - 6.549331}{6.313749}\right| 100\% = 3.7\%$$

Equation (26.14) can be applied iteratively until $\varepsilon_a$ falls below a prespecified value of $\varepsilon_s$. As was the case with the Heun method (recall Example 25.5), the iterations converge on a value of 6.360865 ($\varepsilon_t = -2.68\%$). However, because the initial predictor value is more accurate, the multistep method converges at a somewhat faster rate.

For the second step, the predictor is

$$y_2^0 = 2 + \left[4e^{0.8(1)} - 0.5(6.360865)\right]2 = 13.44346 \qquad \varepsilon_t = 9.43\%$$

which is superior to the prediction of 12.08260 ($\varepsilon_t = 18\%$) that was computed with the original Heun method. The first corrector yields 15.76693 ($\varepsilon_t = 6.8\%$), and subsequent iterations converge on the same result as was obtained with the self-starting Heun method: 15.30224 ($\varepsilon_t = -3.1\%$). As with the previous step, the rate of convergence of the corrector is somewhat improved because of the better initial prediction.

Derivation and Error Analysis of Predictor-Corrector Formulas. We have just employed graphical concepts to derive the non-self-starting Heun. We will now show how the same equations can be derived mathematically. This derivation is particularly interesting because it ties together ideas from curve fitting, numerical integration, and ODEs. The exercise is also useful because it provides a simple procedure for developing higher-order multistep methods and estimating their errors.

The derivation is based on solving the general ODE

$$\frac{dy}{dx} = f(x, y)$$

This equation can be solved by multiplying both sides by $dx$ and integrating between limits at $i$ and $i + 1$:

$$\int_{y_i}^{y_{i+1}} dy = \int_{x_i}^{x_{i+1}} f(x, y)\, dx$$

The left side can be integrated and evaluated using the fundamental theorem [recall Eq. (25.21)]:

$$y_{i+1} = y_i + \int_{x_i}^{x_{i+1}} f(x, y)\, dx \tag{26.16}$$

Equation (26.16) represents a solution to the ODE if the integral can be evaluated. That is, it provides a means to compute a new value of the dependent variable $y_{i+1}$ on the basis of a prior value $y_i$ and the differential equation.

Numerical integration formulas such as those developed in Chap. 21 provide one way to make this evaluation. For example, the trapezoidal rule [Eq. (21.3)] can be used to evaluate the integral, as in

$$\int_{x_i}^{x_{i+1}} f(x, y)\, dx = \frac{f(x_i, y_i) + f(x_{i+1}, y_{i+1})}{2} h \tag{26.17}$$

where $h = x_{i+1} - x_i$ is the step size. Substituting Eq. (26.17) into Eq. (26.16) yields

$$y_{i+1} = y_i + \frac{f(x_i, y_i) + f(x_{i+1}, y_{i+1})}{2} h$$

which is the corrector equation for the Heun method. Because this equation is based on the trapezoidal rule, the truncation error can be taken directly from Table 21.2,

$$E_c = -\frac{1}{12} h^3 y^{(3)}(\xi_c) = -\frac{1}{12} h^3 f''(\xi_c) \tag{26.18}$$

where the subscript $c$ designates that this is the error of the corrector.

A similar approach can be used to derive the predictor. For this case, the integration limits are from $i - 1$ to $i + 1$:

$$\int_{y_{i-1}}^{y_{i+1}} dy = \int_{x_{i-1}}^{x_{i+1}} f(x, y)\, dx$$

which can be integrated and rearranged to yield

$$y_{i+1} = y_{i-1} + \int_{x_{i-1}}^{x_{i+1}} f(x, y)\, dx \tag{26.19}$$

Now, rather than using a closed formula from Table 21.2, the first Newton-Cotes open integration formula (see Table 21.4) can be used to evaluate the integral, as in

$$\int_{x_{i-1}}^{x_{i+1}} f(x, y)\, dx = 2h f(x_i, y_i) \tag{26.20}$$

which is called the *midpoint method.* Substituting Eq. (26.20) into Eq. (26.19) yields

$$y_{i+1} = y_{i-1} + 2h f(x_i, y_i)$$

which is the predictor for the non-self-starting Heun. As with the corrector, the local truncation error can be taken directly from Table 21.4:

$$E_p = \frac{1}{3}h^3 y^{(3)}(\xi_p) = \frac{1}{3}h^3 f''(\xi_p) \tag{26.21}$$

where the subscript $p$ designates that this is the error of the predictor.

Thus, the predictor and the corrector for the non-self-starting Heun method have truncation errors of the same order. Aside from upgrading the accuracy of the predictor, this fact has additional benefits related to error analysis, as elaborated in the next section.

Error Estimates. If the predictor and the corrector of a multistep method are of the same order, the local truncation error may be estimated during the course of a computation. This is a tremendous advantage because it establishes a criterion for adjustment of the step size.

The local truncation error for the predictor is estimated by Eq. (26.21). This error estimate can be combined with the estimate of $y_{i+1}$ from the predictor step to yield [recall our basic definition of Eq. (3.1)]

$$\text{True value} = y^0_{i+1} + \frac{1}{3}h^3 y^{(3)}(\xi_p) \tag{26.22}$$

Using a similar approach, the error estimate for the corrector [Eq. (26.18)] can be combined with the corrector result $y_{i+1}$ to give

$$\text{True value} = y^m_{i+1} - \frac{1}{12}h^3 y^{(3)}(\xi_c) \tag{26.23}$$

Equation (26.22) can be subtracted from Eq. (26.23) to yield

$$0 = y^m_{i+1} - y^0_{i+1} - \frac{5}{12}h^3 y^{(3)}(\xi) \tag{26.24}$$

where $\xi$ is now between $x_{i-1}$ and $x_{i+1}$. Now, dividing Eq. (26.24) by 5 and rearranging the result gives

$$\frac{y^0_{i+1} - y^m_{i+1}}{5} = -\frac{1}{12}h^3 y^{(3)}(\xi) \tag{26.25}$$

Notice that the right-hand sides of Eqs. (26.18) and (26.25) are identical, with the exception of the argument of the third derivative. If the third derivative does not vary appreciably over the interval in question, we can assume that the right-hand sides are equal, and therefore, the left-hand sides should also be equivalent, as in

$$E_c = -\frac{y^0_{i+1} - y^m_{i+1}}{5} \tag{26.26}$$

Thus, we have arrived at a relationship that can be used to estimate the per-step truncation error on the basis of two quantities—the predictor ($y^0_{i+1}$) and the corrector ($y^m_{i+1}$)—that are routine by-products of the computation.

EXAMPLE 26.3 Estimate of Per-Step Truncation Error

Problem Statement. Use Eq. (26.26) to estimate the per-step truncation error of Example 26.2. Note that the true values at $x = 1$ and 2 are 6.194631 and 14.84392, respectively.

Solution. At $x_{i+1} = 1$, the predictor gives 5.607005 and the corrector yields 6.360865. These values can be substituted into Eq. (26.26) to give

$$E_c = -\frac{6.360865 - 5.607005}{5} = -0.1507722$$

which compares well with the exact error,

$$E_t = 6.194631 - 6.360865 = -0.1662341$$

At $x_{i+1} = 2$, the predictor gives 13.44346 and the corrector yields 15.30224, which can be used to compute

$$E_c = -\frac{15.30224 - 13.44346}{5} = -0.3717550$$

which also compares favorably with the exact error, $E_t = 14.84392 - 15.30224 = -0.4583148$.

The ease with which the error can be estimated using Eq. (26.26) provides a rational basis for step-size adjustment during the course of a computation. For example, if Eq. (26.26) indicates that the error is greater than an acceptable level, the step size could be decreased.

Modifiers. Before discussing computer algorithms, we must note two other ways in which the non-self-starting Heun method can be made more accurate and efficient. First, you should realize that besides providing a criterion for step-size adjustment, Eq. (26.26) represents a numerical estimate of the discrepancy between the final corrected value at each step $y_{i+1}$ and the true value. Thus, it can be added directly to $y_{i+1}$ to refine the estimate further:

$$y_{i+1}^m \leftarrow y_{i+1}^m - \frac{y_{i+1}^m - y_{i+1}^0}{5} \tag{26.27}$$

Equation (26.27) is called a *corrector modifier.* (The symbol $\leftarrow$ is read "is replaced by.") The left-hand side is the modified value of $y_{i+1}^m$.

A second improvement, one that relates more to program efficiency, is a *predictor modifier,* which is designed to adjust the predictor result so that it is closer to the final convergent value of the corrector. This is advantageous because, as noted previously at the beginning of this section, the number of iterations of the corrector is highly dependent on the accuracy of the initial prediction. Consequently, if the prediction is modified properly, we might reduce the number of iterations required to converge on the ultimate value of the corrector.

Such a modifier can be derived simply by assuming that the third derivative is relatively constant from step to step. Therefore, using the result of the previous step at $i$, Eq. (26.25) can be solved for

$$h^3 y^{(3)}(\xi) = -\frac{12}{5}\left(y_i^0 - y_i^m\right) \tag{26.28}$$

which, assuming that $y^{(3)}(\xi) \cong y^{(3)}(\xi_p)$, can be substituted into Eq. (26.21) to give

$$E_p = \frac{4}{5}\left(y_i^m - y_i^0\right) \tag{26.29}$$

which can then be used to modify the predictor result:

$$y_{i+1}^0 \leftarrow y_{i+1}^0 + \frac{4}{5}\left(y_i^m - y_i^0\right) \tag{26.30}$$

EXAMPLE 26.4 Effect of Modifiers on Predictor-Corrector Results

Problem Statement. Recompute Example 26.3 using both modifiers.

Solution. As in Example 26.3, the initial predictor result is 5.607005. Because the predictor modifier [Eq. (26.30)] requires values from a previous iteration, it cannot be employed to improve this initial result. However, Eq. (26.27) can be used to modify the corrected value of 6.360865 ($\varepsilon_t = -2.684\%$), as in

$$y_1^m = 6.360865 - \frac{6.360865 - 5.607005}{5} = 6.210093$$

which represents an $\varepsilon_t = -0.25\%$. Thus, the error is reduced over an order of magnitude.

For the next iteration, the predictor [Eq. (26.13)] is used to compute

$$y_2^0 = 2 + \left[4e^{0.8(0)} - 0.5(6.210093)\right]2 = 13.59423 \qquad \varepsilon_t = 8.42\%$$

which is about half the error of the predictor for the second iteration of Example 26.3, which was $\varepsilon_t = 18.6\%$. This improvement occurs because we are using a superior estimate of $y$ (6.210093 as opposed to 6.360865) in the predictor. In other words, the propagated and global errors are reduced by the inclusion of the corrector modifier.

Now because we have information from the prior iteration, Eq. (26.30) can be employed to modify the predictor, as in

$$y_2^0 = 13.59423 + \frac{4}{5}(6.360865 - 5.607005) = 14.19732 \qquad \varepsilon_t = -4.36\%$$

which, again, halves the error.

This modification has no effect on the final outcome of the subsequent corrector step. Regardless of whether the unmodified or modified predictors are used, the corrector will ultimately converge on the same answer. However, because the rate or efficiency of convergence depends on the accuracy of the initial prediction, the modification can reduce the number of iterations required for convergence.

Implementing the corrector yields a result of 15.21178 ($\varepsilon_t = -2.48\%$), which represents an improvement over Example 26.3 because of the reduction of global error. Finally, this result can be modified using Eq. (26.27):

$$y_2^m = 15.21178 - \frac{15.21178 - 13.59423}{5} = 14.88827 \qquad \varepsilon_t = -0.30\%$$

Again, the error has been reduced an order of magnitude.

As in the previous example, the addition of the modifiers increases both the efficiency and accuracy of multistep methods. In particular, the corrector modifier effectively increases the order of the technique. Thus, the non-self-starting Heun with modifiers is third order rather than second order as is the case for the unmodified version. However, it should be noted that there are situations where the corrector modifier will affect the stability of the corrector iteration process. As a consequence, the modifier is not included in the algorithm for the non-self-starting Heun delineated in Fig. 26.5. Nevertheless, the corrector modifier can still have utility for step-size control, as discussed next.

**FIGURE 26.5**

The sequence of formulas used to implement the non-self-starting Heun method. Note that the corrector error estimates can be used to modify the corrector. However, because this can affect the corrector's stability, the modifier is not included in this algorithm. The corrector error estimate is included because of its utility for step-size adjustment.

**Predictor:**

$$y^0_{i+1} = y^m_{i-1} + f(x_i, y^m_i)2h$$

(Save result as $y^0_{i+1,u} = y^0_{i+1}$ where the subscript $u$ designates that the variable is unmodified.)

**Predictor Modifier:**

$$y^0_{i+1} \leftarrow y^0_{i+1,u} + \frac{4}{5}(y^m_{i,u} - y^0_{i,u})$$

**Corrector:**

$$y^j_{i+1} = y^m_i + \frac{f(x_i, y^m_i) + f(x_{i+1}, y^{j-1}_{i+1})}{2}h \qquad \text{(for } j = 1 \text{ to maximum iterations } m\text{)}$$

**Error Check:**

$$|\varepsilon_a| = \left|\frac{y^j_{i+1} - y^{j-1}_{i+1}}{y^j_{i+1}}\right| 100\%$$

(If $|\varepsilon_a| >$ error criterion, set $j = j + 1$ and repeat corrector; if $\varepsilon_a \leq$ error criterion, save result as $y^m_{i+1,u} = y^m_{i+1}$.)

**Corrector Error Estimate:**

$$E_c = -\frac{1}{5}(y^m_{i+1,u} - y^0_{i+1,u})$$

(If computation is to continue, set $i = i + 1$ and return to predictor.)

### 26.2.2 Step-Size Control and Computer Programs

Constant Step Size. It is relatively simple to develop a constant step-size version of the non-self-starting Heun method. About the only complication is that a one-step method is required to generate the extra point to start the computation.

Additionally, because a constant step size is employed, a value for $h$ must be chosen prior to the computation. In general, experience indicates that an optimal step size should be small enough to ensure convergence within two iterations of the corrector (Hull and Creemer, 1963). In addition, it must be small enough to yield a sufficiently small truncation error. At the same time, the step size should be as large as possible to minimize run-time cost and round-off error. As with other methods for ODEs, the only practical way to assess the magnitude of the global error is to compare the results for the same problem but with a halved step size.

Variable Step Size. Two criteria are typically used to decide whether a change in step size is warranted. First, if Eq. (26.26) is greater than some prespecified error criterion, the step size is decreased. Second, the step size is chosen so that the convergence criterion of the corrector is satisfied in two iterations. This criterion is intended to account for the trade-off between the rate of convergence and the total number of steps in the calculation. For smaller values of $h$, convergence will be more rapid but more steps are required. For larger $h$, convergence is slower but fewer steps result. Experience (Hull and Creemer, 1963) suggests that the total steps will be minimized if $h$ is chosen so that the corrector converges within two iterations. Therefore, if over two iterations are required, the step size is decreased, and if less than two iterations are required, the step size is increased.

Although the above strategy specifies when step size modifications are in order, it does not indicate how they should be changed. This is a critical question because multistep methods by definition require several points to compute a new point. Once the step size is changed, a new set of points must be determined. One approach is to restart the computation and use the one-step method to generate a new set of starting points.

A more efficient strategy that makes use of existing information is to increase and decrease by doubling and halving the step size. As depicted in Fig. 26.6*b*, if a sufficient number of previous values have been generated, increasing the step size by doubling is a relatively straightforward task (Fig. 26.6*c*). All that is necessary is to keep track of subscripts so that old values of $x$ and $y$ become the appropriate new values. Halving the step size is somewhat more complicated because some of the new values will be unavailable (Fig. 26.6*a*). However, interpolating polynomials of the type developed in Chap. 18 can be used to determine these intermediate values.

In any event, the decision to incorporate step-size control represents a trade-off between initial investment in program complexity versus the long-term return because of increased efficiency. Obviously, the magnitude and importance of the problem itself will have a strong bearing on this trade-off. Fortunately, several software packages and libraries have multistep routines that you can use to obtain solutions without having to program them from scratch. We will mention some of these when we review packages and libraries at the end of Chap. 27.

### 26.2.3 Integration Formulas

The non-self-starting Heun method is characteristic of most multistep methods. It employs an open integration formula (the midpoint method) to make an initial estimate. This

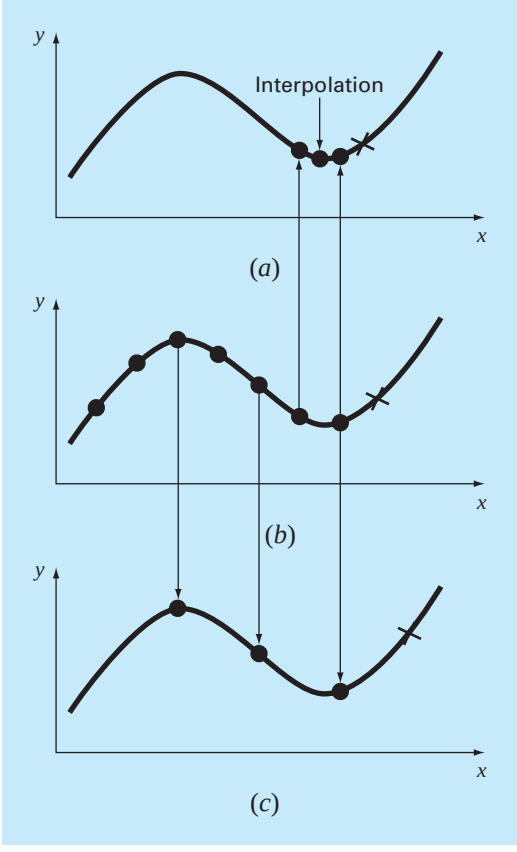



**FIGURE 26.6**
A plot indicating how a halving-doubling strategy allows the use of (*b*) previously calculated values for a third-order multistep method. (*a*) Halving; (*c*) doubling.

predictor step requires a previous data point. Then, a closed integration formula (the trapezoidal rule) is applied iteratively to improve the solution.

It should be obvious that a strategy for improving multistep methods would be to use higher-order integration formulas as predictors and correctors. For example, the higher-order Newton-Cotes formulas developed in Chap. 21 could be used for this purpose.

Before describing these higher-order methods, we will review the most common integration formulas upon which they are based. As mentioned above, the first of these are the Newton-Cotes formulas. However, there is a second class called the Adams formulas that we will also review and that are often preferred. As depicted in Fig. 26.7, the fundamental difference between the Newton-Cotes and Adams formulas relates to the manner in which the integral is applied to obtain the solution. As depicted in Fig. 26.7*a*, the Newton-Cotes formulas estimate the integral over an interval spanning several points. This integral is then used to project from the beginning of the interval to the end. In contrast, the Adams formulas (Fig. 26.7*b*) use a set of points from an interval to estimate the integral solely for the last segment in the interval. This integral is then used to project across this last segment.

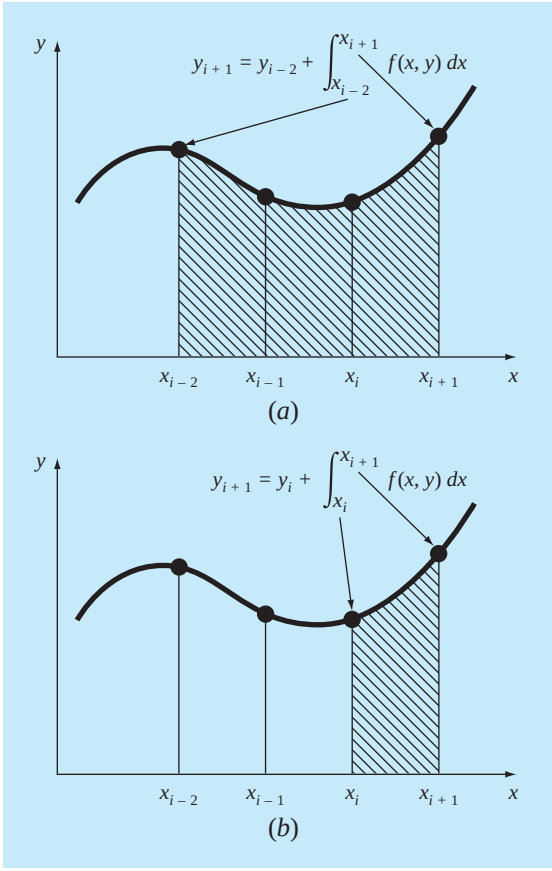



**FIGURE 26.7**
Illustration of the fundamental difference between the Newton-Cotes and Adams integration formulas. (*a*) The Newton-Cotes formulas use a series of points to obtain an integral estimate over a number of segments. The estimate is then used to project across the entire range. (*b*) The Adams formulas use a series of points to obtain an integral estimate for a single segment. The estimate is then used to project across the segment.

Newton-Cotes Formulas. Some of the most common formulas for solving ordinary differential equations are based on fitting an *n*th-degree interpolating polynomial to $n + 1$ known values of $y$ and then using this equation to compute the integral. As discussed previously in Chap. 21, the Newton-Cotes integration formulas are based on such an approach. These formulas are of two types: open and closed forms.

***Open Formulas.*** For $n$ equally spaced data points, the open formulas can be expressed in the form of a solution of an ODE, as was done previously for Eq. (26.19). The general equation for this purpose is

$$y_{i+1} = y_{i-n} + \int_{x_{i-n}}^{x_{i+1}} f_n(x)\,dx \tag{26.31}$$

where $f_n(x)$ is an $n$th-order interpolating polynomial. The evaluation of the integral employs the $n$th-order Newton-Cotes open integration formula (Table 21.4). For example, if $n = 1$,

$$y_{i+1} = y_{i-1} + 2hf_i \tag{26.32}$$

where $f_i$ is an abbreviation for $f(x_i, y_i)$—that is, the differential equation evaluated at $x_i$ and $y_i$. Equation (26.32) is referred to as the midpoint method and was used previously as the predictor in the non-self-starting Heun method. For $n = 2$,

$$y_{i+1} = y_{i-2} + \frac{3h}{2}(f_i + f_{i-1})$$

and for $n = 3$,

$$y_{i+1} = y_{i-3} + \frac{4h}{3}(2f_i - f_{i-1} + 2f_{i-2}) \tag{26.33}$$

Equation (26.33) is depicted graphically in Fig. 26.8*a*.

***Closed Formulas.*** The closed form can be expressed generally as

$$y_{i+1} = y_{i-n+1} + \int_{x_{i-n+1}}^{x_{i+1}} f_n(x)\,dx \tag{26.34}$$

where the integral is approximated by an $n$th-order Newton-Cotes closed integration formula (Table 21.2). For example, for $n = 1$,

$$y_{i+1} = y_i + \frac{h}{2}(f_i + f_{i+1})$$

which is equivalent to the trapezoidal rule. For $n = 2$,

$$y_{i+1} = y_{i-1} + \frac{h}{3}(f_{i-1} + 4f_i + f_{i+1}) \tag{26.35}$$

which is equivalent to Simpson's 1/3 rule. Equation (26.35) is depicted in Fig. 26.8*b*.

Adams Formulas. The other types of integration formulas that can be used to solve ODEs are the Adams formulas. Many popular computer algorithms for multistep solution of ODEs are based on these methods.

***Open Formulas (Adams-Bashforth).*** The Adams formulas can be derived in a variety of ways. One technique is to write a forward Taylor series expansion around $x_i$:

$$y_{i+1} = y_i + f_i h + \frac{f_i'}{2}h^2 + \frac{f_i''}{6}h^3 + \cdots$$

which can also be written as

$$y_{i+1} = y_i + h\left(f_i + \frac{h}{2}f_i' + \frac{h^2}{3!}f_i'' + \cdots\right) \tag{26.36}$$

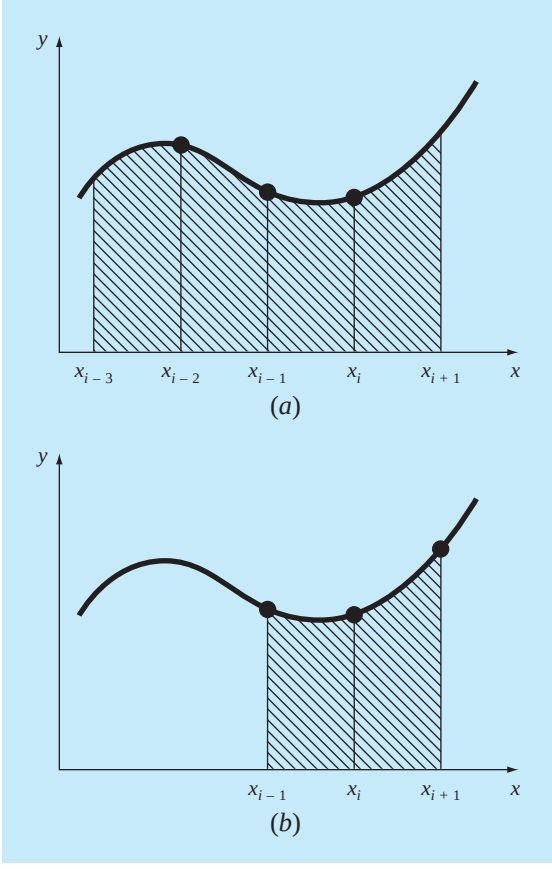



**FIGURE 26.8**
Graphical depiction of open and closed Newton-Cotes integration formulas. (*a*) The third open formula [Eq. (26.33)] and (*b*) Simpson's 1/3 rule [Eq. (26.35)].

Recall from Sec. 4.1.3 that a backward difference can be used to approximate the derivative:

$$f_i' = \frac{f_i - f_{i-1}}{h} + \frac{f_i''}{2}h + O(h^2)$$

which can be substituted into Eq. (26.36),

$$y_{i+1} = y_i + h\left\{ f_i + \frac{h}{2}\left[ \frac{f_i - f_{i-1}}{h} + \frac{f_i''}{2}h + O(h^2) \right] + \frac{h^2}{6} f_i'' + \cdots \right\}$$

or, collecting terms,

$$y_{i+1} = y_i + h\left( \frac{3}{2} f_i - \frac{1}{2} f_{i-1} \right) + \frac{5}{12} h^3 f_i'' + O(h^4) \tag{26.37}$$

**TABLE 26.1** Coefficients and truncation error for Adams-Bashforth predictors.

| Order | $\beta_0$ | $\beta_1$ | $\beta_2$ | $\beta_3$ | $\beta_4$ | $\beta_5$ | Local Truncation Error |
|---|---|---|---|---|---|---|---|
| 1 | 1 | | | | | | $\frac{1}{2}h^2f'(\xi)$ |
| 2 | 3/2 | −1/2 | | | | | $\frac{5}{12}h^3f''(\xi)$ |
| 3 | 23/12 | −16/12 | 5/12 | | | | $\frac{9}{24}h^4f^{(3)}(\xi)$ |
| 4 | 55/24 | −59/24 | 37/24 | −9/24 | | | $\frac{251}{720}h^5f^{(4)}(\xi)$ |
| 5 | 1901/720 | −2774/720 | 2616/720 | −1274/720 | 251/720 | | $\frac{475}{1440}h^6f^{(5)}(\xi)$ |
| 6 | 4277/720 | −7923/720 | 9982/720 | −7298/720 | 2877/720 | −475/720 | $\frac{19,087}{60,480}h^7f^{(6)}(\xi)$ |

This formula is called the *second-order open Adams formula*. Open Adams formulas are also referred to as *Adams-Bashforth formulas*. Consequently, Eq. (26.37) is sometimes called the *second Adams-Bashforth formula*.

Higher-order Adams-Bashforth formulas can be developed by substituting higher-difference approximations into Eq. (26.36). The *n*th-order open Adams formula can be represented generally as

$$y_{i+1} = y_i + h\sum_{k=0}^{n-1}\beta_k f_{i-k} + O(h^{n+1}) \tag{26.38}$$

The coefficients $\beta_k$ are compiled in Table 26.1. The fourth-order version is depicted in Fig. 26.9*a*. Notice that the first-order version is Euler's method.

***Closed Formulas (Adams-Moulton).*** A backward Taylor series around $x_{i+1}$ can be written as

$$y_i = y_{i+1} - f_{i+1}h + \frac{f'_{i+1}}{2}h^2 - \frac{f''_{i+1}}{3!}h^3 + \cdots$$

Solving for $y_{i+1}$ yields

$$y_{i+1} = y_i + h\left(f_{i+1} - \frac{h}{2}f'_{i+1} + \frac{h^2}{6}f''_{i+1} + \cdots\right) \tag{26.39}$$

A difference can be used to approximate the first derivative:

$$f'_{i+1} = \frac{f_{i+1} - f_i}{h} + \frac{f''_{i+1}}{2}h + O(h^2)$$

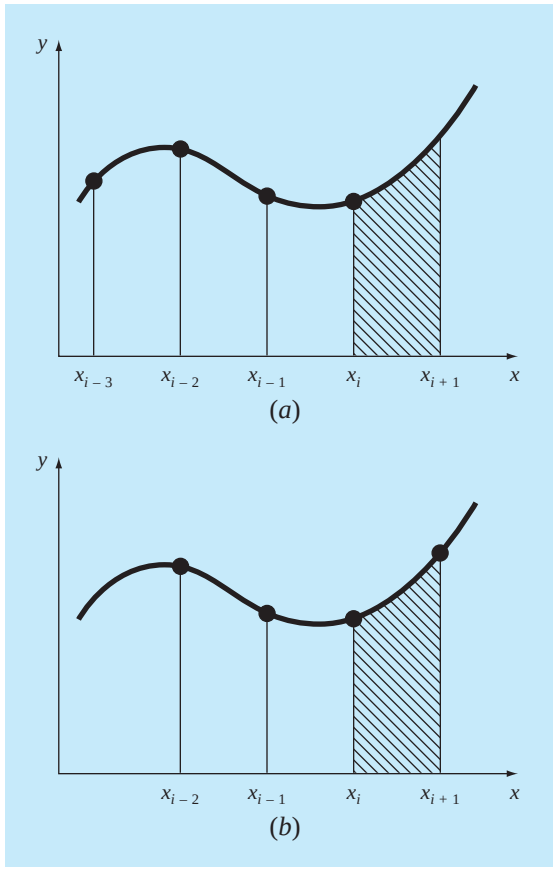



**FIGURE 26.9**
Graphical depiction of open and closed Adams integration formulas. (*a*) The fourth Adams-Bashforth open formula and (*b*) the fourth Adams-Moulton closed formula.

which can be substituted into Eq. (26.39), and collecting terms gives

$$y_{i+1} = y_i + h\left(\frac{1}{2}f_{i+1} + \frac{1}{2}f_i\right) - \frac{1}{12}h^3 f''_{i+1} - O(h^4)$$

This formula is called the *second-order closed Adams formula* or the *second Adams-Moulton formula*. Also, notice that it is the trapezoidal rule.

The *n*th-order closed Adams formula can be written generally as

$$y_{i+1} = y_i + h\sum_{k=0}^{n-1} \beta_k f_{i+1-k} + O(h^{n+1})$$

The coefficients $\beta_k$ are listed in Table 26.2. The fourth-order method is depicted in Fig. 26.9*b*.

## Box 26.1 Derivation of General Relationships for Modifiers

The relationship between the true value, the approximation, and the error of a predictor can be represented generally as

$$\text{True value} = y_{i+1}^0 + \frac{\eta_p}{\delta_p} h^{n+1} y^{(n+1)}(\xi_p) \tag{B26.1.1}$$

where $\eta_p$ and $\delta_p$ = the numerator and denominator, respectively, of the constant of the truncation error for either an open Newton-Cotes (Table 21.4) or an Adams-Bashforth (Table 26.1) predictor, and $n$ is the order.

A similar relationship can be developed for the corrector:

$$\text{True value} = y_{i+1}^m - \frac{\eta_c}{\delta_c} h^{n+1} y^{(n+1)}(\xi_c) \tag{B26.1.2}$$

where $\eta_c$ and $\delta_c$ = the numerator and denominator, respectively, of the constant of the truncation error for either a closed Newton-Cotes (Table 21.2) or an Adams-Moulton (Table 26.2) corrector. As was done in the derivation of Eq. (26.24), Eq. (B26.1.1) can be subtracted from Eq. (B26.1.2) to yield

$$0 = y_{i+1}^m - y_{i+1}^0 - \frac{\eta_c + \eta_p \delta_c/\delta_p}{\delta_c} h^{n+1} y^{(n+1)}(\xi) \tag{B26.1.3}$$

Now, dividing the equation by $\eta_c + \eta_p \delta_c/\delta_p$, multiplying the last term by $\delta_p/\delta_p$, and rearranging provides an estimate of the local truncation error of the corrector:

$$E_c \cong -\frac{\eta_c \delta_p}{\eta_c \delta_p + \eta_p \delta_c}\left(y_{i+1}^m - y_{i+1}^0\right) \tag{B26.1.4}$$

For the predictor modifier, Eq. (B26.1.3) can be solved at the previous step for

$$h^n y^{(n+1)}(\xi) = -\frac{\delta_c \delta_p}{\eta_c \delta_p + \eta_p \delta_c}\left(y_i^0 - y_i^m\right)$$

which can be substituted into the error term of Eq. (B26.1.1) to yield

$$E_p = \frac{\eta_p \delta_c}{\eta_c \delta_p + \eta_p \delta_c}\left(y_i^m - y_i^0\right) \tag{B26.1.5}$$

Equations (B26.1.4) and (B26.1.5) are general versions of modifiers that can be used to improve multistep algorithms. For example, Milne's method has $\eta_p = 14$, $\delta_p = 45$, $\eta_c = 1$, $\delta_c = 90$. Substituting these values into Eqs. (B26.1.4) and (B26.1.5) yields Eqs. (26.43) and (26.42), respectively. Similar modifiers can be developed for other pairs of open and closed formulas that have local truncation errors of the same order.

**TABLE 26.2** Coefficients and truncation error for Adams-Moulton correctors.

| Order | $\beta_0$ | $\beta_1$ | $\beta_2$ | $\beta_3$ | $\beta_4$ | $\beta_5$ | Local Truncation Error |
|---|---|---|---|---|---|---|---|
| 2 | 1/2 | 1/2 | | | | | $-\frac{1}{12}h^3 f''(\xi)$ |
| 3 | 5/12 | 8/12 | −1/12 | | | | $-\frac{1}{24}h^4 f^{(3)}(\xi)$ |
| 4 | 9/24 | 19/24 | −5/24 | 1/24 | | | $-\frac{19}{720}h^5 f^{(4)}(\xi)$ |
| 5 | 251/720 | 646/720 | −264/720 | 106/720 | −19/720 | | $-\frac{27}{1440}h^6 f^{(5)}(\xi)$ |
| 6 | 475/1440 | 1427/1440 | −798/1440 | 482/1440 | −173/1440 | 27/1440 | $-\frac{863}{60{,}480}h^7 f^{(6)}(\xi)$ |

### 26.2.4 Higher-Order Multistep Methods

Now that we have formally developed the Newton-Cotes and Adams integration formulas, we can use them to derive higher-order multistep methods. As was the case with the non-self-starting Heun method, the integration formulas are applied in tandem as predictor-corrector methods. In addition, if the open and closed formulas have local truncation errors

of the same order, modifiers of the type listed in Fig. 26.5 can be incorporated to improve accuracy and allow step-size control. Box 26.1 provides general equations for these modifiers. In the following section, we present two of the most common higher-order multistep approaches: Milne's method and the fourth-order Adams method.

Milne's Method. Milne's method is the most common multistep method based on Newton-Cotes integration formulas. It uses the three-point Newton-Cotes open formula as a predictor:

$$y_{i+1}^0 = y_{i-3}^m + \frac{4h}{3}(2f_i^m - f_{i-1}^m + 2f_{i-2}^m) \tag{26.40}$$

and the three-point Newton-Cotes closed formula (Simpson's 1/3 rule) as a corrector:

$$y_{i+1}^j = y_{i-1}^m + \frac{h}{3}(f_{i-1}^m + 4f_i^m + f_{i+1}^{j-1}) \tag{26.41}$$

where $j$ is an index representing the number of iterations of the modifier. The predictor and corrector modifiers for Milne's method can be developed from the formulas in Box 26.1 and the error coefficients in Tables 21.2 and 21.4:

$$E_p = \frac{28}{29}(y_i^m - y_i^0) \tag{26.42}$$

$$E_c \cong -\frac{1}{29}(y_{i+1}^m - y_{i+1}^0) \tag{26.43}$$

EXAMPLE 26.5 Milne's Method

Problem Statement. Use Milne's method to integrate $y' = 4e^{0.8x} - 0.5y$ from $x = 0$ to $x = 4$ using a step size of 1. The initial condition at $x = 0$ is $y = 2$. Because we are dealing with a multistep method, previous points are required. In an actual application, a one-step method such as a fourth-order RK would be used to compute the required points. For the present example, we will use the analytical solution [recall Eq. (E25.5.1) from Example 25.5] to compute exact values at $x_{i-3} = -3$, $x_{i-2} = -2$, and $x_{i-1} = -1$ of $y_{i-3} = -4.547302$, $y_{i-2} = -2.306160$, and $y_{i-1} = -0.3929953$, respectively.

Solution. The predictor [Eq. (26.40)] is used to calculate a value at $x = 1$:

$$y_1^0 = -4.54730 + \frac{4(1)}{3}[2(3) - 1.99381 + 2(1.96067)] = 6.02272 \qquad \varepsilon_t = 2.8\%$$

The corrector [Eq. (26.41)] is then employed to compute

$$y_1^1 = -0.3929953 + \frac{1}{3}[1.99381 + 4(3) + 5.890802] = 6.235210 \qquad \varepsilon_t = -0.66\%$$

This result can be substituted back into Eq. (26.41) to iteratively correct the estimate. This process converges on a final corrected value of 6.204855 ($\varepsilon_t = -0.17\%$).

This value is more accurate than the comparable estimate of 6.360865 ($\varepsilon_t = -2.68\%$) obtained previously with the non-self-starting Heun method (Examples 26.2 through 26.4). The results for the remaining steps are $y(2) = 14.86031$ ($\varepsilon_t = -0.11\%$), $y(3) = 33.72426$ ($\varepsilon_t = -0.14\%$), and $y(4) = 75.43295$ ($\varepsilon_t = -0.12\%$).

As in the previous example, Milne's method usually yields results of high accuracy. However, there are certain cases where it performs poorly (see Ralston and Rabinowitz, 1978). Before elaborating on these cases, we will describe another higher-order multistep approach—the fourth-order Adams method.

Fourth-Order Adams Method. A popular multistep method based on the Adams integration formulas uses the fourth-order Adams-Bashforth formula (Table 26.1) as the predictor:

$$y_{i+1}^0 = y_i^m + h\left(\frac{55}{24} f_i^m - \frac{59}{24} f_{i-1}^m + \frac{37}{24} f_{i-2}^m - \frac{9}{24} f_{i-3}^m\right) \tag{26.44}$$

and the fourth-order Adams-Moulton formula (Table 26.2) as the corrector:

$$y_{i+1}^j = y_i^m + h\left(\frac{9}{24} f_{i+1}^{j-1} + \frac{19}{24} f_i^m - \frac{5}{24} f_{i-1}^m + \frac{1}{24} f_{i-2}^m\right) \tag{26.45}$$

The predictor and the corrector modifiers for the fourth-order Adams method can be developed from the formulas in Box 26.1 and the error coefficients in Tables 26.1 and 26.2 as

$$E_p = \frac{251}{270}(y_i^m - y_i^0) \tag{26.46}$$

$$E_c = -\frac{19}{270}(y_{i+1}^m - y_{i+1}^0) \tag{26.47}$$

EXAMPLE 26.6 Fourth-Order Adams Method

Problem Statement. Use the fourth-order Adams method to solve the same problem as in Example 26.5.

Solution. The predictor [Eq. (26.44)] is used to compute a value at $x = 1$.

$$y_1^0 = 2 + 1\left(\frac{55}{24}3 - \frac{59}{24}1.993814 + \frac{37}{24}1.960667 - \frac{9}{24}2.6365228\right) = 6.007539$$

$$\varepsilon_t = 3.1\%$$

which is comparable to but somewhat less accurate than the result using the Milne method. The corrector [Eq. (26.45)] is then employed to calculate

$$y_1^1 = 2 + 1\left(\frac{9}{24}5.898394 + \frac{19}{24}3 - \frac{5}{24}1.993814 + \frac{1}{24}1.960666\right) = 6.253214$$

$$\varepsilon_t = -0.96\%$$

which again is comparable to but less accurate than the result using Milne's method. This result can be substituted back into Eq. (26.45) to iteratively correct the estimate. The process converges on a final corrected value of 6.214424 ($\varepsilon_t = 0.32\%$), which is an accurate result but again somewhat inferior to that obtained with the Milne method.

**Stability of Multistep Methods.** The superior accuracy of the Milne method exhibited in Examples 26.5 and 26.6 would be anticipated on the basis of the error terms for the predictors [Eqs. (26.42) and (26.46)] and the correctors [Eqs. (26.43) and (26.47)]. The coefficients for the Milne method, 14/45 and 1/90, are smaller than for the fourth-order Adams, 251/720 and 19/720. Additionally, the Milne method employs fewer function evaluations to attain these higher accuracies. At face value, these results might lead to the conclusion that the Milne method is superior and, therefore, preferable to the fourth-order Adams. Although this conclusion holds for many cases, there are instances where the Milne method performs unacceptably. Such behavior is exhibited in the following example.

EXAMPLE 26.7 Stability of Milne's and Fourth-Order Adams Methods

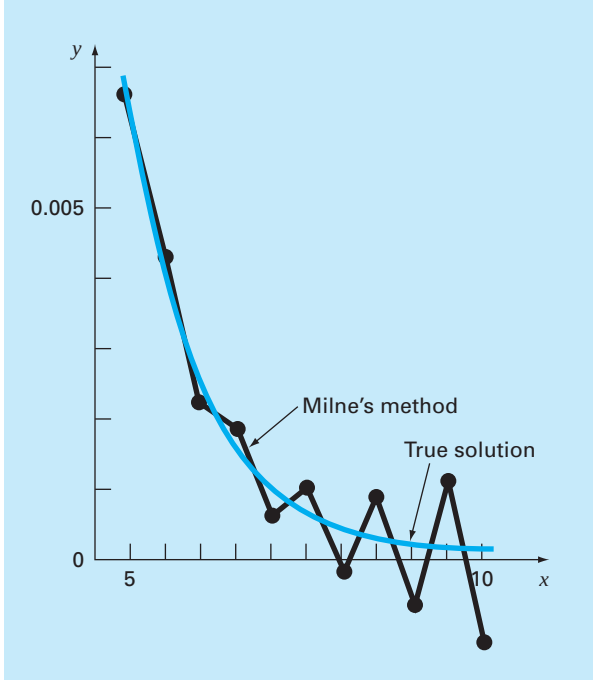
**Problem Statement.** Employ Milne's and the fourth-order Adams methods to solve

$$\frac{dy}{dx} = -y$$

with the initial condition that $y = 1$ at $x = 0$. Solve this equation from $x = 0$ to $x = 10$ using a step size of $h = 0.5$. Note that the analytical solution is $y = e^{-x}$.

**Solution.** The results, as summarized in Fig. 26.10, indicate problems with Milne's method. Shortly after the onset of the computation, the errors begin to grow and oscillate

**FIGURE 26.10**
Graphical depiction of the instability of Milne's method.



in sign. By $x = 10$, the relative error has inflated to 2831 percent and the predicted value itself has started to oscillate in sign.

In contrast, the results for the Adams method would be much more acceptable. Although the error also grows, it would do so at a slow rate. Additionally, the discrepancies would not exhibit the wild swings in sign exhibited by the Milne method.

The unacceptable behavior manifested in the previous example by the Milne method is referred to as instability. Although it does not always occur, its possibility leads to the conclusion that Milne's approach should be avoided. Thus, the fourth-order Adams method is normally preferred.

The instability of Milne's method is due to the corrector. Consequently, attempts have been made to rectify the shortcoming by developing stable correctors. One commonly used alternative that employs this approach is *Hamming's method*, which uses the Milne predictor and a stable corrector:

$$y_{i+1}^{j} = \frac{9y_i^m - y_{i-2}^m + 3h\left(y_{i+1}^{j-1} + 2f_i^m - f_{i-1}^m\right)}{8}$$

which has a local truncation error:

$$E_c = \frac{1}{40}h^5 y^{(4)}(\xi_c)$$

Hamming's method also includes modifiers of the form

$$E_p = \frac{9}{121}\left(y_i^m - y_i^0\right)$$

$$E_c = -\frac{112}{121}\left(y_{i+1}^m - y_{i+1}^0\right)$$

The reader can obtain additional information on this and other multistep methods elsewhere (Hamming, 1973; Lapidus and Seinfield, 1971).

## PROBLEMS

**26.1** Given

$$\frac{dy}{dt} = -100{,}000y + 99{,}999e^{-t}$$

**(a)** Estimate the step-size required to maintain stability using the explicit Euler method.

**(b)** If $y(0) = 0$, use the implicit Euler to obtain a solution from $t = 0$ to 2 using a step size of 0.1.

**26.2** Given

$$\frac{dy}{dt} = 30(\sin t - y) + 3\cos t$$

If $y(0) = 1$, use the implicit Euler to obtain a solution from $t = 0$ to 4 using a step size of 0.4.

**26.3** Given

$$\frac{dx_1}{dt} = 999x_1 + 1999x_2$$

$$\frac{dx_2}{dt} = -1000x_1 - 2000x_2$$

If $x_1(0) = x_2(0) = 1$, obtain a solution from $t = 0$ to 0.2 using a step size of 0.05 with the **(a)** explicit and **(b)** implicit Euler methods.

**26.4** Solve the following initial-value problem over the interval from $t = 2$ to 3:

$$\frac{dy}{dt} = -0.5y + e^{-t}$$

Use the non-self-starting Heun method with a step size of 0.5 and initial conditions of $y(1.5) = 5.222138$ and $y(2.0) = 4.143883$. Iterate the corrector to $\varepsilon_s = 0.1\%$. Compute the true percent relative errors $\varepsilon_t$ for your results based on the analytical solution.

**26.5** Repeat Prob. 26.4, but use the fourth-order Adams method. [Note: $y(0.5) = 8.132548$ and $y(1.0) = 6.542609$.] Iterate the corrector to $\varepsilon_s = 0.01\%$.

**26.6** Solve the following initial-value problem from $t = 4$ to 5:

$$\frac{dy}{dt} = -\frac{2y}{t}$$

Use a step size of 0.5 and initial values of $y(2.5) = 0.48$, $y(3) = 0.333333$, $y(3.5) = 0.244898$, and $y(4) = 0.1875$. Obtain your solutions using the following techniques: **(a)** the non-self-starting Heun method ($\varepsilon_s = 1\%$), and **(b)** the fourth-order Adams method ($\varepsilon_s = 0.01\%$). [Note: The exact answers obtained analytically are $y(4.5) = 0.148148$ and $y(5) = 0.12$.] Compute the true percent relative errors $\varepsilon_t$ for your results.

**26.7** Solve the following initial-value problem from $x = 0$ to $x = 0.75$:

$$\frac{dy}{dx} = yx^2 - y$$

Use the non-self-starting Heun method with a step size of 0.25. If $y(0) = 1$, employ the fourth-order RK method with a step size of 0.25 to predict the starting value at $y(0.25)$.

**26.8** Solve the following initial-value problem from $t = 1.5$ to $t = 2.5$

$$\frac{dy}{dt} = \frac{-2y}{1+t}$$

Use the fourth-order Adams method. Employ a step size of 0.5 and the fourth-order RK method to predict the start-up values if $y(0) = 2$.

**26.9** Develop a program for the implicit Euler method for a single linear ODE. Test it by duplicating Prob. 26.1*b*.

**26.10** Develop a program for the implicit Euler method for a pair of linear ODEs. Test it by solving Eq. (26.6).

**26.11** Develop a user-friendly program for the non-self-starting Heun method with a predictor modifier. Employ a fourth-order RK method to compute starter values. Test the program by duplicating Example 26.4.

**26.12** Use the program developed in Prob. 26.11 to solve Prob. 26.7.

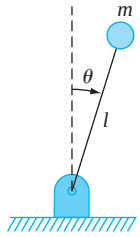



**Figure P26.13**

**26.13** Consider the thin rod of length $l$ moving in the $x$-$y$ plane as shown in Fig. P26.13. The rod is fixed with a pin on one end and a mass at the other. Note that $g = 9.81$ m/s$^2$ and $l = 0.5$ m. This system can be solved using

$$\ddot{\theta} - \frac{g}{l}\theta = 0$$

Let $\theta(0) = 0$ and $\dot{\theta}(0) = 0.25$ rad/s. Solve using any method studied in this chapter. Plot the angle versus time and the angular velocity versus time. (Hint: Decompose the second-order ODE.)

**26.14** Given the first-order ODE

$$\frac{dx}{dt} = -700x - 1000e^{-t}$$
$$x(t = 0) = 4$$

Solve this stiff differential equation using a numerical method over the time period $0 \le t \le 5$. Also solve analytically and plot the analytic and numerical solution for both the fast transient and slow transition phase of the timescale.

**26.15** The following second-order ODE is considered to be stiff

$$\frac{d^2y}{dx^2} = -1001\frac{dy}{dx} - 1000y$$

Solve this differential equation **(a)** analytically and **(b)** numerically for $x = 0$ to 5. For **(b)** use an implicit approach with $h = 0.5$. Note that the initial conditions are $y(0) = 1$ and $y'(0) = 0$. Display both results graphically.

**26.16** Solve the following differential equation from $t = 0$ to 1

$$\frac{dy}{dt} = -10y$$

with the initial condition $y(0) = 1$. Use the following techniques to obtain your solutions: **(a)** analytically, **(b)** the explicit Euler method, and **(c)** the implicit Euler method. For **(b)** and **(c)** use $h = 0.1$ and 0.2. Plot your results.

CHAPTER 27

# Boundary-Value and Eigenvalue Problems

Recall from our discussion at the beginning of Part Seven that an ordinary differential equation is accompanied by auxiliary conditions. These conditions are used to evaluate the constants of integration that result during the solution of the equation. For an *n*th-order equation, *n* conditions are required. If all the conditions are specified at the same value of the independent variable, then we are dealing with an *initial-value problem* (Fig. 27.1*a*). To this point, the material in Part Seven has been devoted to this type of problem.

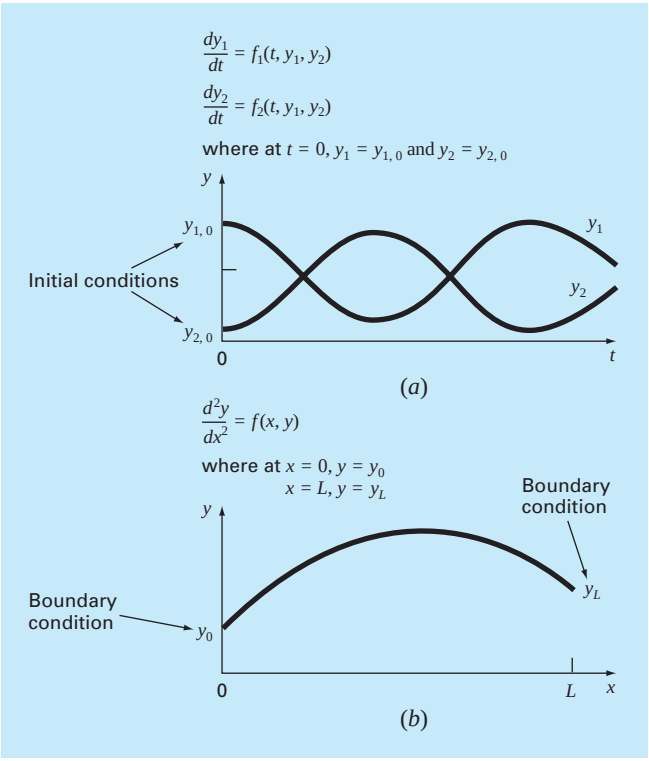



**FIGURE 27.1**
Initial-value versus boundary-value problems. (*a*) An initial-value problem where all the conditions are specified at the same value of the independent variable. (*b*) A boundary-value problem where the conditions are specified at different values of the independent variable.

In contrast, there is another application for which the conditions are not known at a single point, but rather, are known at different values of the independent variable. Because these values are often specified at the extreme points or boundaries of a system, they are customarily referred to as *boundary-value problems* (Fig. 27.1*b*). A variety of significant engineering applications fall within this class. In this chapter, we discuss two general approaches for obtaining their solution: the shooting method and the finite-difference approach. Additionally, we present techniques to approach a special type of boundary-value problem: the determination of eigenvalues. Of course, eigenvalues also have many applications beyond those involving boundary-value problems.

## 27.1 GENERAL METHODS FOR BOUNDARY-VALUE PROBLEMS

The conservation of heat can be used to develop a heat balance for a long, thin rod (Fig. 27.2). If the rod is not insulated along its length and the system is at a steady state, the equation that results is

$$\frac{d^2T}{dx^2} + h'(T_a - T) = 0 \tag{27.1}$$

where $h'$ is a heat transfer coefficient ($m^{-2}$) that parameterizes the rate of heat dissipation to the surrounding air and $T_a$ is the temperature of the surrounding air (°C).

To obtain a solution for Eq. (27.1), there must be appropriate boundary conditions. A simple case is where the temperatures at the ends of the bar are held at fixed values. These can be expressed mathematically as

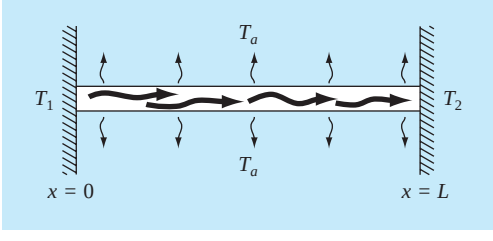
$$T(0) = T_1$$
$$T(L) = T_2$$

With these conditions. Eq. (27.1) can be solved analytically using calculus. For a 10-m rod with $T_a = 20$, $T_1 = 40$, $T_2 = 200$, and $h' = 0.01$, the solution is

$$T = 73.4523e^{0.1x} - 53.4523e^{-0.1x} + 20 \tag{27.2}$$

In the following sections, the same problem will be solved using numerical approaches.

**FIGURE 27.2**
A noninsulated uniform rod positioned between two bodies of constant but different temperature. For this case $T_1 > T_2$ and $T_2 > T_a$.



### 27.1.1 The Shooting Method

The *shooting method* is based on converting the boundary-value problem into an equivalent initial-value problem. A trial-and-error approach is then implemented to solve the initial-value version. The approach can be illustrated by an example.

EXAMPLE 27.1 The Shooting Method

Problem Statement. Use the shooting method to solve Eq. (27.1) for a 10-m rod with $h' = 0.01\ \text{m}^{-2}$, $T_a = 20$, and the boundary conditions

$$T(0) = 40 \qquad T(10) = 200$$

Solution. Using the same approach as was employed to transform Eq. (PT7.2) into Eqs. (PT7.3) through (PT7.6), the second-order equation can be expressed as two first-order ODEs:

$$\frac{dT}{dx} = z \tag{E27.1.1}$$

$$\frac{dz}{dx} = h'(T - T_a) \tag{E27.1.2}$$

To solve these equations, we require an initial value for $z$. For the shooting method, we guess a value—say, $z(0) = 10$. The solution is then obtained by integrating Eq. (E27.1.1) and (E27.1.2) simultaneously. For example, using a fourth-order RK method with a step size of 2, we obtain a value at the end of the interval of $T(10) = 168.3797$ (Fig. 27.3*a*), which differs from the boundary condition of $T(10) = 200$. Therefore, we make another guess, $z(0) = 20$, and perform the computation again. This time, the result of $T(10) = 285.8980$ is obtained (Fig. 27.3*b*).

Now, because the original ODE is linear, the values

$$z(0) = 10 \qquad T(10) = 168.3797$$

and

$$z(0) = 20 \qquad T(10) = 285.8980$$

are linearly related. As such, they can be used to compute the value of $z(0)$ that yields $T(10) = 200$. A linear interpolation formula [recall Eq. (18.2)] can be employed for this purpose:

$$z(0) = 10 + \frac{20 - 10}{285.8980 - 168.3797}(200 - 168.3797) = 12.6907$$

This value can then be used to determine the correct solution, as depicted in Fig. 27.3*c*.

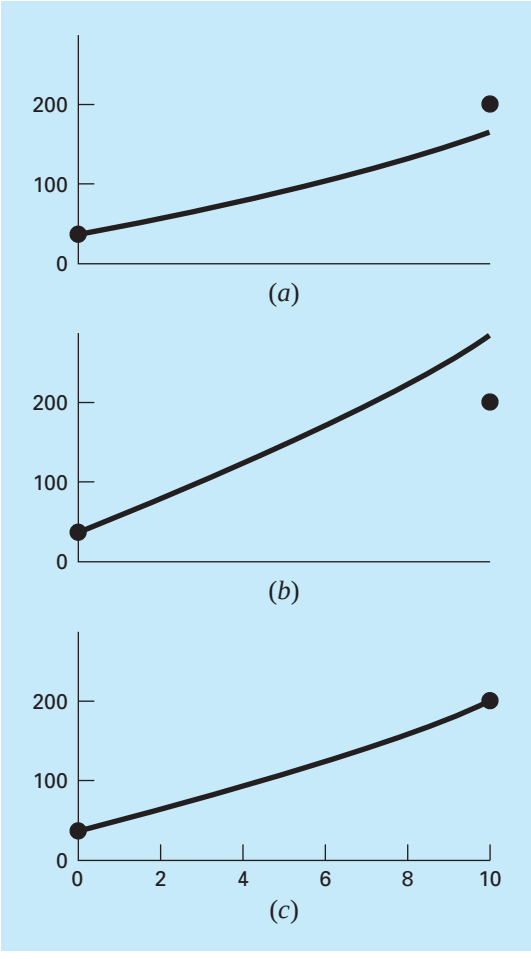



**FIGURE 27.3**
The shooting method: (*a*) the first "shot," (*b*) the second "shot," and (*c*) the final exact "hit."

Nonlinear Two-Point Problems. For nonlinear boundary-value problems, linear interpolation or extrapolation through two solution points will not necessarily result in an accurate estimate of the required boundary condition to attain an exact solution. An alternative is to perform three applications of the shooting method and use a quadratic interpolating polynomial to estimate the proper boundary condition. However, it is unlikely that such an approach would yield the exact answer, and additional iterations would be necessary to obtain the solution.

Another approach for a nonlinear problem involves recasting it as a roots problem. Recall that the general form of a root problem is to find the value of $x$ that makes the

function $f(x) = 0$. Now, let us use Example 27.1 to understand how the shooting method can be recast in this form.

First, recognize that the solution of the pair of differential equations is also a "function" in the sense that we guess a condition at the left-hand end of the bar, $z_0$, and the integration yields a prediction of the temperature at the right-hand end, $T_{10}$. Thus, we can think of the integration as

$$T_{10} = f(z_0)$$

That is, it represents a process whereby a guess of $z_0$ yields a prediction of $T_{10}$. Viewed in this way, we can see that what we desire is the value of $z_0$ that yields a specific value of $T_{10}$. If, as in the example, we desire $T_{10} = 200$, the problem can be posed as

$$200 = f(z_0)$$

By bringing the goal of 200 over to the right-hand side of the equation, we generate a new function, $g(z_0)$, that represents the difference between what we have, $f(z_0)$, and what we want, 200.

$$g(z_0) = f(z_0) - 200$$

If we drive this new function to zero, we will obtain the solution. The next example illustrates the approach.

EXAMPLE 27.2 The Shooting Method for Nonlinear Problems

Problem Statement. Although it served our purposes for proving a simple boundary-value problem, our model for the bar in Eq. (27.1) was not very realistic. For one thing, such a bar would lose heat by mechanisms such as radiation that are nonlinear.

Suppose that the following nonlinear ODE is used to simulate the temperature of the heated bar:

$$\frac{d^2T}{dx^2} + h''(T_a - T)^4 = 0$$

where $h'' = 5 \times 10^{-8}$. Now, although it is still not a very good representation of heat transfer, this equation is straightforward enough to allow us to illustrate how the shooting method can be used to solve a two-point nonlinear boundary-value problem. The remaining problem conditions are as specified in Example 27.1.

Solution. The second-order equation can be expressed as two first-order ODEs:

$$\frac{dT}{dx} = z$$

$$\frac{dz}{dx} = h''(T - T_a)^4$$

Now, these equations can be integrated using any of the methods described in Chaps. 25 and 26. We used the constant step-size version of the fourth-order RK approach described in Chap. 25. We implemented this approach as an Excel macro function written in Visual

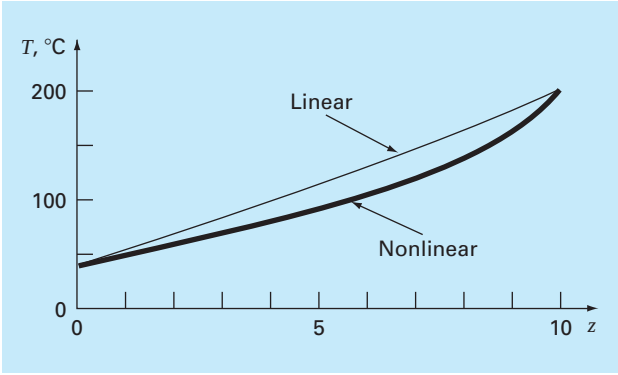



**FIGURE 27.4**
The result of using the shooting method to solve a nonlinear problem.

BASIC. The function integrated the equations based on an initial guess for $z(0)$ and returned the temperature at $x = 10$. The difference between this value and the goal of 200 was then placed in a spreadsheet cell. The Excel Solver was then invoked to adjust the value of $z(0)$ until the difference was driven to zero.

The result is shown in Fig. 27.4 along with the original linear case. As might be expected, the nonlinear case is curved more than the linear model. This is due to the power of four term in the heat transfer relationship.

The shooting method can become arduous for higher-order equations where the necessity to assume two or more conditions makes the approach somewhat more difficult. For these reasons, alternative methods are available, as described next.

### 27.1.2 Finite-Difference Methods

The most common alternatives to the shooting method are *finite-difference approaches*. In these techniques, finite divided differences are substituted for the derivatives in the original equation. Thus, a linear differential equation is transformed into a set of simultaneous algebraic equations that can be solved using the methods from Part Three.

For the case of Fig. 27.2, the finite-divided-difference approximation for the second derivative is (recall Fig. 23.3)

$$\frac{d^2T}{dx^2} = \frac{T_{i+1} - 2T_i + T_{i-1}}{\Delta x^2}$$

This approximation can be substituted into Eq. (27.1) to give

$$\frac{T_{i+1} - 2T_i + T_{i-1}}{\Delta x^2} - h'(T_i - T_a) = 0$$

Collecting terms gives

$$-T_{i-1} + (2 + h'\Delta x^2)T_i - T_{i+1} = h'\Delta x^2 T_a \tag{27.3}$$

This equation applies for each of the interior nodes of the rod. The first and last interior nodes, $T_{i-1}$ and $T_{i+1}$, respectively, are specified by the boundary conditions. Therefore, the resulting set of linear algebraic equations will be tridiagonal. As such, it can be solved with the efficient algorithms that are available for such systems (Sec. 11.1).

EXAMPLE 27.3 Finite-Difference Approximation of Boundary-Value Problems

Problem Statement. Use the finite-difference approach to solve the same problem as in Example 27.1.

Solution. Employing the parameters in Example 27.1, we can write Eq. (27.3) for the rod from Fig. 27.2. Using four interior nodes with a segment length of $\Delta x = 2$ m results in the following equations:

$$\begin{bmatrix} 2.04 & -1 & 0 & 0 \\ -1 & 2.04 & -1 & 0 \\ 0 & -1 & 2.04 & -1 \\ 0 & 0 & -1 & 2.04 \end{bmatrix} \begin{Bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \end{Bmatrix} = \begin{Bmatrix} 40.8 \\ 0.8 \\ 0.8 \\ 200.8 \end{Bmatrix}$$

which can be solved for

$$\{T\}^T = \lfloor 65.9698 \quad 93.7785 \quad 124.5382 \quad 159.4795 \rfloor$$

Table 27.1 provides a comparison between the analytical solution [Eq. (27.2)] and the numerical solutions obtained in Examples 27.1 and 27.3. Note that there are some discrepancies among the approaches. For both numerical methods, these errors can be mitigated by decreasing their respective step sizes. Although both techniques perform well for the present case, the finite-difference approach is preferred because of the ease with which it can be extended to more complex cases.

The fixed (or *Dirichlet*) boundary condition used in the previous example is but one of several types that are commonly employed in engineering and science. A common alternative, called the *Neumann boundary condition*, is the case where the derivative is given.

**TABLE 27.1** Comparison of the exact analytical solution with the shooting and finite-difference methods.

| $x$ | True | Shooting Method | Finite Difference |
|---|---|---|---|
| 0 | 40 | 40 | 40 |
| 2 | 65.9518 | 65.9520 | 65.9698 |
| 4 | 93.7478 | 93.7481 | 93.7785 |
| 6 | 124.5036 | 124.5039 | 124.5382 |
| 8 | 159.4534 | 159.4538 | 159.4795 |
| 10 | 200 | 200 | 200 |

We can use the heated rod model to demonstrate how derivative boundary condition can be incorporated into the finite-difference approach,

$$0 = \frac{d^2T}{dx^2} + h'(T_\infty - T)$$

However, in contrast to our previous discussions, we will prescribe a derivative boundary condition at one end of the rod,

$$\frac{dT}{dx}(0) = T'_a$$

$$T(L) = T_b$$

Thus, we have a derivative boundary condition at one end of the solution domain and a fixed boundary condition at the other.

As was done in Example 27.3, the rod is divided into a series of nodes and a finite-difference version of the differential equation (Eq. 27.3) is applied to each interior node. However, because its temperature is not specified, the node at the left end must also be included. Writing Eq. (27.3) for this node gives

$$-T_{-1} + (2 + h'\Delta x^2)T_0 - T_1 = h'\Delta x^2 T_\infty \tag{27.3a}$$

Notice that an imaginary node $(-1)$ lying to the left of the rod's end is required for this equation. Although this exterior point might seem to represent a difficulty, it actually serves as the vehicle for incorporating the derivative boundary condition into the problem. This is done by representing the first derivative in the $x$ dimension at (0) by the centered difference

$$\frac{dT}{dx} = \frac{T_1 - T_{-1}}{2\Delta x}$$

which can be solved for

$$T_{-1} = T_1 - 2\Delta x \frac{dT}{dx}$$

Now we have a formula for $T_{-1}$ that actually reflects the impact of the derivative. It can be substituted into Eq. (27.3*a*) to give

$$(2 + h'\Delta x^2)T_0 - 2T_1 = h'\Delta x^2 T_\infty - 2\Delta x \frac{dT}{dx} \tag{27.3b}$$

Consequently, we have incorporated the derivative into the balance.

A common example of a derivative boundary condition is the situation where the end of the rod is insulated. In this case, the derivative is set to zero. This conclusion follows directly from Fourier's law, which states that the heat flux is directly proportional to the temperature gradient. Thus, insulating a boundary means that the heat flux (and consequently the gradient) must be zero.

Aside from the shooting and finite-difference methods, there are other techniques available for solving boundary-value problems. Some of these will be described in Part Eight. These include steady-state (Chap. 29) and transient (Chap. 30) solution of two-dimensional

boundary-value problems using finite differences and steady-state solutions of the one-dimensional problem with the finite-element approach (Chap. 31).

## 27.2 EIGENVALUE PROBLEMS

*Eigenvalue*, or characteristic-value, problems are a special class of boundary-value problems that are common in engineering problem contexts involving vibrations, elasticity, and other oscillating systems. In addition, they are used in a wide variety of engineering contexts beyond boundary-value problems. Before describing numerical methods for solving these problems, we will present some general background information. This includes discussion of both the mathematics and the engineering significance of eigenvalues.

### 27.2.1 Mathematical Background

Part Three dealt with methods for solving sets of linear algebraic equations of the general form

$$[A]\{X\} = \{B\}$$

Such systems are called *nonhomogeneous* because of the presence of the vector $\{B\}$ on the right-hand side of the equality. If the equations comprising such a system are linearly independent (that is, have a nonzero determinant), they will have a unique solution. In other words, there is one set of $x$ values that will make the equations balance.

In contrast, a homogeneous linear algebraic system has the general form

$$[A]\{X\} = 0$$

Although nontrivial solutions (that is, solutions other than all $x$'s $= 0$) of such systems are possible, they are generally not unique. Rather, the simultaneous equations establish relationships among the $x$'s that can be satisfied by various combinations of values.

Eigenvalue problems associated with engineering are typically of the general form

$$\begin{aligned}
(a_{11} - \lambda)x_1 + \quad & a_{12}x_2 + \cdots + \quad & a_{1n}x_n = 0 \\
a_{21}x_1 + \ & (a_{22} - \lambda)x_2 + \cdots + \quad & a_{2n}x_n = 0 \\
\cdot \quad & \cdot \quad & \cdot \quad \cdot \\
\cdot \quad & \cdot \quad & \cdot \quad \cdot \\
\cdot \quad & \cdot \quad & \cdot \quad \cdot \\
a_{n1}x_1 + \quad & a_{n2}x_2 + \cdots + (a_{nn} - \lambda)x_n = 0
\end{aligned}$$

where $\lambda$ is an unknown parameter called the *eigenvalue*, or *characteristic value*. A solution $\{X\}$ for such a system is referred to as an *eigenvector*. The above set of equations may also be expressed concisely as

$$\big[[A] - \lambda[I]\big]\{X\} = 0 \tag{27.4}$$

The solution of Eq. (27.4) hinges on determining $\lambda$. One way to accomplish this is based on the fact that the determinant of the matrix $[[A] - \lambda[I]]$ must equal zero for nontrivial solutions to be possible. Expanding the determinant yields a polynomial in $\lambda$. The roots of this polynomial are the solutions for the eigenvalues. An example of this approach will be provided in the next section.

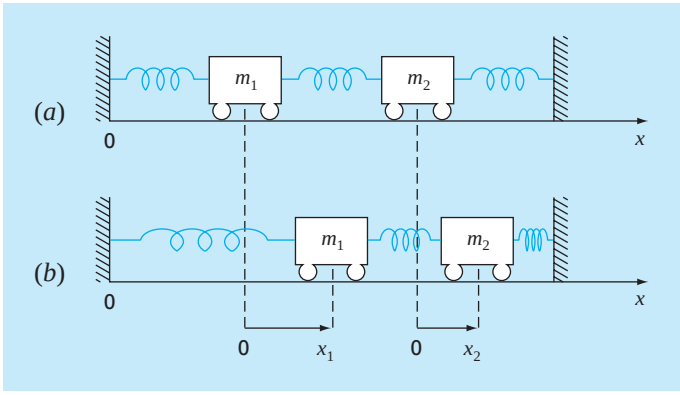



**FIGURE 27.5**
Positioning the masses away from equilibrium creates forces in the springs that upon release lead to oscillations of the masses. The positions of the masses can be referenced to local coordinates with origins at their respective equilibrium positions.

### 27.2.2 Physical Background

The mass-spring system in Fig. 27.5*a* is a simple context to illustrate how eigenvalues occur in physical problem settings. It also will help to illustrate some of the mathematical concepts introduced in the previous section.

To simplify the analysis, assume that each mass has no external or damping forces acting on it. In addition, assume that each spring has the same natural length $l$ and the same spring constant $k$. Finally, assume that the displacement of each spring is measured relative to its own local coordinate system with an origin at the spring's equilibrium position (Fig. 27.5*a*). Under these assumptions, Newton's second law can be employed to develop a force balance for each mass (recall Sec. 12.4),

$$m_1 \frac{d^2x_1}{dt^2} = -kx_1 + k(x_2 - x_1)$$

and

$$m_2 \frac{d^2x_2}{dt^2} = -k(x_2 - x_1) - kx_2$$

where $x_i$ is the displacement of mass $i$ away from its equilibrium position (Fig. 27.5*b*). These equations can be expressed as

$$m_1 \frac{d^2x_1}{dt^2} - k(-2x_1 + x_2) = 0 \tag{27.5a}$$

$$m_2 \frac{d^2x_2}{dt^2} - k(x_1 - 2x_2) = 0 \tag{27.5b}$$

From vibration theory, it is known that solutions to Eq. (27.5) can take the form

$$x_i = A_i \sin(\omega t) \tag{27.6}$$

where $A_i$ = the amplitude of the vibration of mass $i$ and $\omega$ = the frequency of the vibration, which is equal to

$$\omega = \frac{2\pi}{T_p} \tag{27.7}$$

where $T_p$ is the period. From Eq. (27.6) it follows that

$$x_i'' = -A_i \omega^2 \sin(\omega t) \tag{27.8}$$

Equations (27.6) and (27.8) can be substituted into Eq. (27.5), which, after collection of terms, can be expressed as

$$\left(\frac{2k}{m_1} - \omega^2\right) A_1 - \frac{k}{m_1} A_2 = 0 \tag{27.9a}$$

$$-\frac{k}{m_2} A_1 + \left(\frac{2k}{m_2} - \omega^2\right) A_2 = 0 \tag{27.9b}$$

Comparison of Eq. (27.9) with Eq. (27.4) indicates that at this point, the solution has been reduced to an eigenvalue problem.

## EXAMPLE 27.4 Eigenvalues and Eigenvectors for a Mass-Spring System

**Problem Statement.** Evaluate the eigenvalues and the eigenvectors of Eq. (27.9) for the case where $m_1 = m_2 = 40$ kg and $k = 200$ N/m.

**Solution.** Substituting the parameter values into Eq. (27.9) yields

$$(10 - \omega^2) A_1 - 5A_2 = 0$$
$$-5A_1 + (10 - \omega^2) A_2 = 0$$

The determinant of this system is [recall Eq. (9.3)]

$$(\omega^2)^2 - 20\omega^2 + 75 = 0$$

which can be solved by the quadratic formula for $\omega^2 = 15$ and 5 $s^{-2}$. Therefore, the frequencies for the vibrations of the masses are $\omega = 3.873$ $s^{-1}$ and 2.236 $s^{-1}$, respectively. These values can be used to determine the periods for the vibrations with Eq. (27.7). For the first mode, $T_p = 1.62$ s, and for the second, $T_p = 2.81$ s.

As stated in Sec. 27.2.1, a unique set of values cannot be obtained for the unknowns. However, their ratios can be specified by substituting the eigenvalues back into the equations. For example, for the first mode ($\omega^2 = 15$ $s^{-2}$), $A_1 = -A_2$. For the second mode ($\omega^2 = 5$ $s^{-2}$), $A_1 = A_2$.

This example provides valuable information regarding the behavior of the system in Fig. 27.5. Aside from its period, we know that if the system is vibrating in the first mode, the amplitude of the second mass will be equal but of opposite sign to the amplitude of the first. As in Fig. 27.6*a*, the masses vibrate apart and then together indefinitely.

In the second mode, the two masses have equal amplitudes at all times. Thus, as in Fig. 27.6*b*, they vibrate back and forth in unison. It should be noted that the configuration of the amplitudes provides guidance on how to set their initial values to attain pure motion

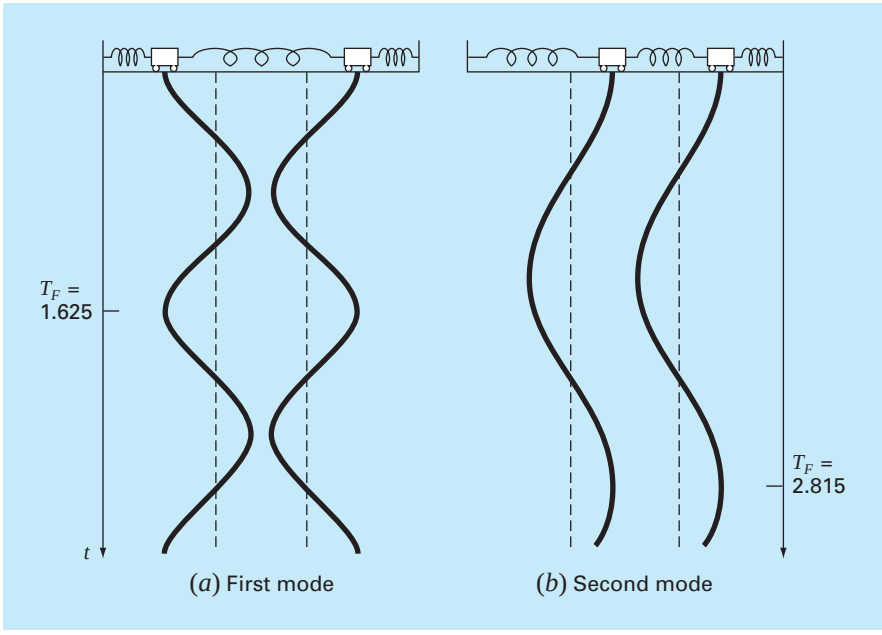



**FIGURE 27.6**
The principal modes of vibration of two equal masses connected by three identical springs between fixed walls.

in either of the two modes. Any other configuration will lead to superposition of the modes (recall Chap. 19).

### 27.2.3 A Boundary-Value Problem

Now that you have been introduced to eigenvalues, we turn to the type of problem that is the subject of the present chapter: boundary-value problems for ordinary differential equations. Figure 27.7 shows a physical system that can serve as a context for examining this type of problem.

The curvature of a slender column subject to an axial load $P$ can be modeled by

$$\frac{d^2y}{dx^2} = \frac{M}{EI} \tag{27.10}$$

where $d^2y/dx^2$ specifies the curvature, $M =$ the bending moment, $E =$ the modulus of elasticity, and $I =$ the moment of inertia of the cross section about its neutral axis. Considering the free body in Fig. 27.7*b*, it is clear that the bending moment at $x$ is $M = -Py$. Substituting this value into Eq. (27.10) gives

$$\frac{d^2y}{dx^2} + p^2y = 0 \tag{27.11}$$

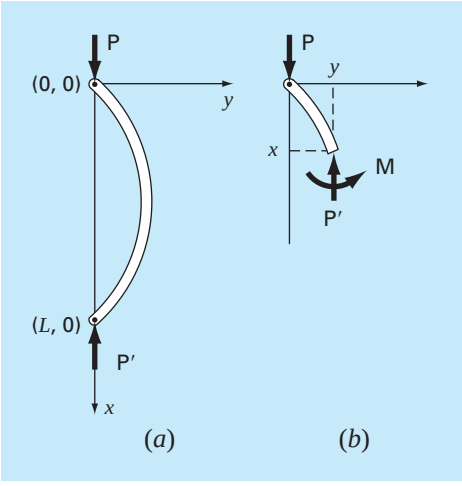



**FIGURE 27.7**
(*a*) A slender rod. (*b*) A free-body diagram of a rod.

where

$$p^2 = \frac{P}{EI} \tag{27.12}$$

For the system in Fig. 27.7, subject to the boundary conditions

$$y(0) = 0 \tag{27.13a}$$

$$y(L) = 0 \tag{27.13b}$$

the general solution for Eq. (27.11) is

$$y = A\sin(px) + B\cos(px) \tag{27.14}$$

where $A$ and $B$ are arbitrary constants that are to be evaluated via the boundary conditions. According to the first condition [Eq. (27.13*a*)],

$$0 = A\sin(0) + B\cos(0)$$

Therefore, we conclude that $B = 0$.

According to the second condition [Eq. (27.13*b*)],

$$0 = A\sin(pL) + B\cos(pL)$$

But, since $B = 0$, $A\sin(pL) = 0$. Because $A = 0$ represents a trivial solution, we conclude that $\sin(pL) = 0$. For this equality to hold,

$$pL = n\pi \qquad \text{for } n = 1, 2, 3, \ldots \pi \tag{27.15}$$

Thus, there are an infinite number of values that meet the boundary condition. Equation (27.15) can be solved for

$$p = \frac{n\pi}{L} \qquad \text{for } n = 1, 2, 3, \ldots \tag{27.16}$$

which are the eigenvalues for the column.

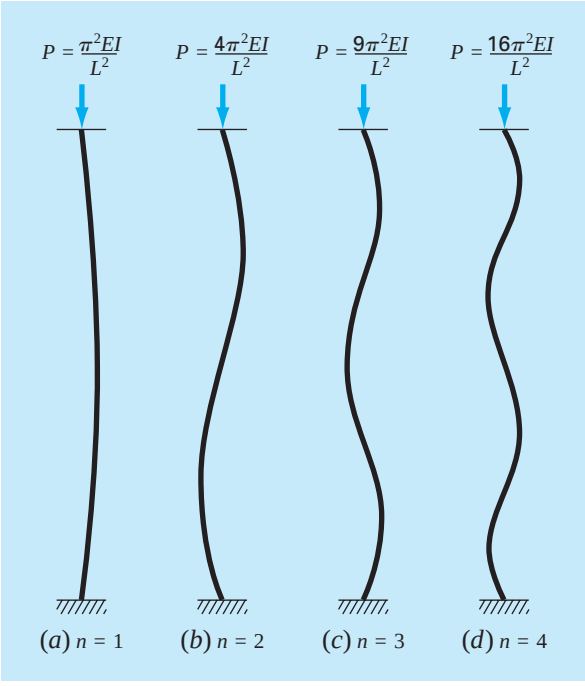



**FIGURE 27.8**
The first four eigenvalues for the slender rod from Fig. 27.7.

Figure 27.8, which shows the solution for the first four eigenvalues, can provide insight into the physical significance of the results. Each eigenvalue corresponds to a way in which the column buckles. Combining Eqs. (27.12) and (27.16) gives

$$P = \frac{n^2 \pi^2 E I}{L^2} \qquad \text{for } n = 1, 2, 3, \ldots \tag{27.17}$$

These can be thought of as *buckling loads* because they represent the levels at which the column moves into each succeeding buckling configuration. In a practical sense, it is usually the first value that is of interest because failure will usually occur when the column first buckles. Thus, a critical load can be defined as

$$P = \frac{\pi^2 E I}{L^2}$$

which is formally known as *Euler's formula.*

EXAMPLE 27.5 Eigenvalue Analysis of an Axially Loaded Column

**Problem Statement.** An axially loaded wooden column has the following characteristics: $E = 10 \times 10^9$ Pa, $I = 1.25 \times 10^{-5}$ m$^4$, and $L = 3$ m. Determine the first eight eigenvalues and the corresponding buckling loads.

Solution. Equations (27.16) and (27.17) can be used to compute

| $n$ | $p$, m$^{-2}$ | $P$, kN |
|---|---|---|
| 1 | 1.0472 | 137.078 |
| 2 | 2.0944 | 548.311 |
| 3 | 3.1416 | 1233.701 |
| 4 | 4.1888 | 2193.245 |
| 5 | 5.2360 | 3426.946 |
| 6 | 6.2832 | 4934.802 |
| 7 | 7.3304 | 6716.814 |
| 8 | 8.3776 | 8772.982 |

The critical buckling load is, therefore, 137.078 kN.

Although analytical solutions of the sort obtained above are useful, they are often difficult or impossible to obtain. This is usually true when dealing with complicated systems or those with heterogeneous properties. In such cases, numerical methods of the sort described next are the only practical alternative.

### 27.2.4 The Polynomial Method

Equation (27.11) can be solved numerically by substituting a central finite-divided-difference approximation (Fig. 23.3) for the second derivative to give

$$\frac{y_{i+1} - 2y_i + y_{i-1}}{h^2} + p^2 y_i = 0$$

which can be expressed as

$$y_{i-1} - (2 - h^2 p^2) y_i + y_{i+1} = 0 \tag{27.18}$$

Writing this equation for a series of nodes along the axis of the column yields a homogeneous system of equations. For example, if the column is divided into five segments (that is, four interior nodes), the result is

$$\begin{bmatrix} (2-h^2p^2) & -1 & 0 & 0 \\ -1 & (2-h^2p^2) & -1 & 0 \\ 0 & -1 & (2-h^2p^2) & -1 \\ 0 & 0 & -1 & (2-h^2p^2) \end{bmatrix} \begin{Bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{Bmatrix} = 0 \tag{27.19}$$

Expansion of the determinant of the system yields a polynomial, the roots of which are the eigenvalues. This approach, called the *polynomial method*, is performed in the following example.

EXAMPLE 27.6 The Polynomial Method

Problem Statement. Employ the polynomial method to determine the eigenvalues for the axially loaded column from Example 27.5 using (*a*) one, (*b*) two, (*c*) three, and (*d*) four interior nodes.

Solution.

(**a**) Writing Eq. (27.18) for one interior node yields ($h = 3/2$)

$$-(2 - 2.25p^2)y_1 = 0$$

Thus, for this simple case, the eigenvalue is analyzed by setting the determinant equal to zero

$$2 - 2.25p^2 = 0$$

and solving for $p = \pm 0.9428$, which is about 10 percent less than the exact value of 1.0472 obtained in Example 27.4.

(**b**) For two interior nodes ($h = 3/3$), Eq. (27.18) is written as

$$\begin{bmatrix} (2-p^2) & -1 \\ -1 & (2-p^2) \end{bmatrix} \begin{Bmatrix} y_1 \\ y_2 \end{Bmatrix} = 0$$

Expansion of the determinant gives

$$(2 - p^2)^2 - 1 = 0$$

which can be solved for $p = \pm 1$ and $\pm 1.73205$. Thus, the first eigenvalue is now about 4.5 percent low and a second eigenvalue is obtained that is about 17 percent low.

(**c**) For three interior points ($h = 3/4$), Eq. (27.18) yields

$$\begin{bmatrix} 2-0.5625p^2 & -1 & 0 \\ -1 & 2-0.5625p^2 & -1 \\ 0 & -1 & 2-0.5625p^2 \end{bmatrix} \begin{Bmatrix} y_1 \\ y_2 \\ y_3 \end{Bmatrix} = 0 \qquad \text{(E27.6.1)}$$

The determinant can be set equal to zero and expanded to give

$$(2 - 0.5625p^2)^3 - 2(2 - 0.5625p^2) = 0$$

For this equation to hold, $2 - 0.5625p^2 = 0$ and $2 - 0.5625p^2 = \sqrt{2}$. Therefore, the first three eigenvalues can be determined as

$$p = \pm 1.0205 \qquad |\varepsilon_t| = 2.5\%$$
$$p = \pm 1.8856 \qquad |\varepsilon_t| = 10\%$$
$$p = \pm 2.4637 \qquad |\varepsilon_t| = 22\%$$

(**d**) For four interior points ($h = 3/5$), the result is Eq. (27.19) with $2 - 0.36p^2$ on the diagonal. Setting the determinant equal to zero and expanding it gives

$$(2 - 0.36p^2)^4 - 3(2 - 0.36p^2)^2 + 1 = 0$$

which can be solved for the first four eigenvalues

$$p = \pm 1.0301 \qquad |\varepsilon_t| = 1.6\%$$
$$p = \pm 1.9593 \qquad |\varepsilon_t| = 6.5\%$$
$$p = \pm 2.6967 \qquad |\varepsilon_t| = 14\%$$
$$p = \pm 3.1702 \qquad |\varepsilon_t| = 24\%$$

**TABLE 27.2** The results of applying the polynomial method to an axially loaded column. The numbers in parentheses represent the absolute value of the true percent relative error.

| | | Polynomial Method | | | |
|---|---|---|---|---|---|
| **Eigenvalue** | **True** | ***h* = 3/2** | ***h* = 3/3** | ***h* = 3/4** | ***h* = 3/5** |
| 1 | 1.0472 | 0.9428 (10%) | 1.0000 (4.5%) | 1.0205 (2.5%) | 1.0301 (1.6%) |
| 2 | 2.0944 | | 1.7321 (21%) | 1.8856 (10%) | 1.9593 (65%) |
| 3 | 3.1416 | | | 2.4637 (22%) | 2.6967 (14%) |
| 4 | 4.1888 | | | | 3.1702 (24%) |

Table 27.2, which summarizes the results of this example, illustrates some fundamental aspects of the polynomial method. As the segmentation is made more refined, additional eigenvalues are determined and the previously determined values become progressively more accurate. Thus, the approach is best suited for cases where the lower eigenvalues are required.

### 27.2.5 The Power Method

The *power method* is an iterative approach that can be employed to determine the largest eigenvalue. With slight modification, it can also be employed to determine the smallest and the intermediate values. It has the additional benefit that the corresponding eigenvector is obtained as a by-product of the method.

Determination of the Largest Eigenvalue. To implement the power method, the system being analyzed must be expressed in the form

$$[A]\{X\} = \lambda\{X\} \tag{27.20}$$

As illustrated by the following example, Eq. (27.20) forms the basis for an iterative solution technique that eventually yields the highest eigenvalue and its associated eigenvector.

EXAMPLE 27.7 Power Method for Highest Eigenvalue

Problem Statement. Employ the power method to determine the highest eigenvalue for part **(c)** of Example 27.6.

Solution. The system is first written in the form of Eq. (27.20),

$$\begin{aligned} 3.556x_1 - 1.778x_2 &= \lambda x_1 \\ -1.778x_1 + 3.556x_2 - 1.778x_3 &= \lambda x_2 \\ -1.778x_2 + 3.556x_3 &= \lambda x_3 \end{aligned}$$

Then, assuming the $x$'s on the left-hand side of the equation are equal to 1,

$$\begin{aligned} 3.556(1) - 1.778(1) &= 1.778 \\ -1.778(1) + 3.556(1) - 1.778(1) &= 0 \\ -1.778(1) + 3.556(1) &= 1.778 \end{aligned}$$

Next, the right-hand side is normalized by 1.778 to make the largest element equal to

$$\begin{Bmatrix} 1.778 \\ 0 \\ 1.778 \end{Bmatrix} = 1.778 \begin{Bmatrix} 1 \\ 0 \\ 1 \end{Bmatrix}$$

Thus, the first estimate of the eigenvalue is 1.778. This iteration can be expressed concisely in matrix form as

$$\begin{bmatrix} 3.556 & -1.778 & 0 \\ -1.778 & 3.556 & -1.778 \\ 0 & -1.778 & 3.556 \end{bmatrix} \begin{Bmatrix} 1 \\ 1 \\ 1 \end{Bmatrix} = \begin{Bmatrix} 1.778 \\ 0 \\ 1.778 \end{Bmatrix} = 1.778 \begin{Bmatrix} 1 \\ 0 \\ 1 \end{Bmatrix}$$

The next iteration consists of multiplying $[A]$ by $\lfloor 1 \quad 0 \quad 1 \rfloor^T$ to give

$$\begin{bmatrix} 3.556 & -1.778 & 0 \\ -1.778 & 3.556 & -1.778 \\ 0 & -1.778 & 3.556 \end{bmatrix} \begin{Bmatrix} 1 \\ 0 \\ 1 \end{Bmatrix} = \begin{Bmatrix} 3.556 \\ -3.556 \\ 3.556 \end{Bmatrix} = 3.556 \begin{Bmatrix} 1 \\ -1 \\ 1 \end{Bmatrix}$$

Therefore, the eigenvalue estimate for the second iteration is 3.556, which can be employed to determine the error estimate

$$|\varepsilon_a| = \left| \frac{3.556 - 1.778}{3.556} \right| 100\% = 50\%$$

The process can then be repeated.

*Third iteration:*

$$\begin{bmatrix} 3.556 & -1.778 & 0 \\ -1.778 & 3.556 & -1.778 \\ 0 & -1.778 & 3.556 \end{bmatrix} \begin{Bmatrix} 1 \\ -1 \\ 1 \end{Bmatrix} = \begin{Bmatrix} 5.334 \\ -7.112 \\ 5.334 \end{Bmatrix} = -7.112 \begin{Bmatrix} -0.75 \\ 1 \\ -0.75 \end{Bmatrix}$$

where $|\varepsilon_a| = 150\%$ (which is high because of the sign change).

*Fourth iteration:*

$$\begin{bmatrix} 3.556 & -1.778 & 0 \\ -1.778 & 3.556 & -1.778 \\ 0 & -1.778 & 3.556 \end{bmatrix} \begin{Bmatrix} -0.75 \\ 1 \\ -0.75 \end{Bmatrix} = \begin{Bmatrix} -4.445 \\ 6.223 \\ -4.445 \end{Bmatrix} = 6.223 \begin{Bmatrix} -0.714 \\ 1 \\ -0.714 \end{Bmatrix}$$

where $|\varepsilon_a| = 214\%$ (again inflated because of sign change).

*Fifth iteration:*

$$\begin{bmatrix} 3.556 & -1.778 & 0 \\ -1.778 & 3.556 & -1.778 \\ 0 & -1.778 & 3.556 \end{bmatrix} \begin{Bmatrix} -0.714 \\ 1 \\ -0.714 \end{Bmatrix} = \begin{Bmatrix} -4.317 \\ 6.095 \\ -4.317 \end{Bmatrix} = 6.095 \begin{Bmatrix} -0.708 \\ 1 \\ -0.708 \end{Bmatrix}$$

Thus, the normalizing factor is converging on the value of 6.070 ($= 2.4637^2$) obtained in part **(c)** of Example 27.6.

Note that there are some instances where the power method will converge to the second-largest eigenvalue instead of to the largest. James, Smith, and Wolford (1985) provide an illustration of such a case. Other special cases are discussed in Fadeev and Fadeeva (1963).

Determination of the Smallest Eigenvalue. There are often cases in engineering where we are interested in determining the smallest eigenvalue. Such was the case for the rod in Fig. 27.7, where the smallest eigenvalue could be used to identify a critical buckling load. This can be done by applying the power method to the matrix inverse of $[A]$. For this case, the power method will converge on the largest value of $1/\lambda$—in other words, the smallest value of $\lambda$.

## EXAMPLE 27.8 Power Method for Lowest Eigenvalue

Problem Statement. Employ the power method to determine the lowest eigenvalue for part **(c)** of Example 27.6.

Solution. After dividing Eq. E27.6.1 by $h^2$ ($= 0.5625$), its matrix inverse can be evaluated as

$$[A]^{-1} = \begin{bmatrix} 0.422 & 0.281 & 0.141 \\ 0.281 & 0.562 & 0.281 \\ 0.141 & 0.281 & 0.422 \end{bmatrix}$$

Using the same format as in Example 27.9, the power method can be applied to this matrix.

*First iteration:*

$$\begin{bmatrix} 0.422 & 0.281 & 0.141 \\ 0.281 & 0.562 & 0.281 \\ 0.141 & 0.281 & 0.422 \end{bmatrix} \begin{Bmatrix} 1 \\ 1 \\ 1 \end{Bmatrix} = \begin{Bmatrix} 0.884 \\ 1.124 \\ 0.884 \end{Bmatrix} = 1.124 \begin{Bmatrix} 0.751 \\ 1 \\ 0.751 \end{Bmatrix}$$

*Second iteration:*

$$\begin{bmatrix} 0.422 & 0.281 & 0.141 \\ 0.281 & 0.562 & 0.281 \\ 0.141 & 0.281 & 0.422 \end{bmatrix} \begin{Bmatrix} 0.751 \\ 1 \\ 0.751 \end{Bmatrix} = \begin{Bmatrix} 0.704 \\ 0.984 \\ 0.704 \end{Bmatrix} = 0.984 \begin{Bmatrix} 0.715 \\ 1 \\ 0.715 \end{Bmatrix}$$

where $|\varepsilon_a| = 14.6\%$.

*Third iteration:*

$$\begin{bmatrix} 0.422 & 0.281 & 0.141 \\ 0.281 & 0.562 & 0.281 \\ 0.141 & 0.281 & 0.422 \end{bmatrix} \begin{Bmatrix} 0.715 \\ 1 \\ 0.715 \end{Bmatrix} = \begin{Bmatrix} 0.684 \\ 0.964 \\ 0.684 \end{Bmatrix} = 0.964 \begin{Bmatrix} 0.709 \\ 1 \\ 0.709 \end{Bmatrix}$$

where $|\varepsilon_a| = 4\%$.

Thus, after only three iterations, the result is converging on the value of 0.9602, which is the reciprocal of the smallest eigenvalue, 1.0205($= \sqrt{1/0.9602}$), obtained in Example 27.6c.

Determination of Intermediate Eigenvalues. After finding the largest eigenvalue, it is possible to determine the next highest by replacing the original matrix by one that includes only the remaining eigenvalues. The process of removing the largest known eigenvalue is called *deflation.* The technique outlined here, *Hotelling's method,* is designed for symmetric matrices. This is because it exploits the orthogonality of the eigenvectors of such matrices, which can be expressed as

$$\{X\}_i^T\{X\}_j = \begin{cases} 0 & \text{for } i \neq j \\ 1 & \text{for } i = j \end{cases} \tag{27.21}$$

where the components of the eigenvector $\{X\}$ have been normalized so that $\{X\}^T\{X\} = 1$, that is, so that the sum of the squares of the components equals 1. This can be accomplished by dividing each of the elements by the normalizing factor

$$\sqrt{\sum_{k=1}^{n} x_k^2}$$

Now, a new matrix $[A]_2$ can be computed as

$$[A]_2 = [A]_1 - \lambda_1\{X\}_1\{X\}_1^T \tag{27.22}$$

where $[A]_1$ = the original matrix and $\lambda_1$ = the largest eigenvalue. If the power method is applied to this matrix, the iteration process will converge to the second largest eigenvalue, $\lambda_2$. To show this, first postmultiply Eq. (27.22) by $\{X\}_1$,

$$[A]_2\{X\}_1 = [A]_1\{X\}_1 - \lambda_1\{X\}_1\{X\}_1^T\{X\}_1$$

Invoking the orthogonality principle converts this equation to

$$[A]_2\{X\}_1 = [A]_1\{X\}_1 - \lambda_1\{X\}_1$$

where the right-hand side is equal to zero according to Eq. (27.20). Thus, $[A]_2\{X\}_1 = 0$. Consequently, $\lambda = 0$ and $\{X\} = \{X\}_1$ is a solution to $[A]_2\{X\} = \lambda\{X\}$. In other words, the $[A]_2$ has eigenvalues of $0, \lambda_2, \lambda_3, \ldots, \lambda_n$. The largest eigenvalue, $\lambda_1$, has been replaced by a 0 and, therefore, the power method will converge on the next biggest $\lambda_2$.

The above process can be repeated by generating a new matrix $[A]_3$, etc. Although in theory this process could be continued to determine the remaining eigenvalues, it is limited by the fact that errors in the eigenvectors are passed along at each step. Thus, it is only of value in determining several of the highest eigenvalues. Although this is somewhat of a shortcoming, such information is precisely what is required in many engineering problems.

### 27.2.6 Other Methods

A wide variety of additional methods are available for solving eigenvalue problems. Most are based on a two-step process. The first step involves transforming the original matrix to a simpler form (for example, tridiagonal) that retains all the original eigenvalues. Then, iterative methods are used to determine these eigenvalues.

Many of these approaches are designed for special types of matrices. In particular, a variety of techniques are devoted to symmetric systems. For example, *Jacobi's method*

transforms a symmetric matrix to a diagonal matrix by eliminating off-diagonal terms in a systematic fashion. Unfortunately, the method requires an infinite number of operations because the removal of each nonzero element often creates a new nonzero value at a previous zero element. Although an infinite time is required to create all nonzero off-diagonal elements, the matrix will eventually tend toward a diagonal form. Thus, the approach is iterative in that it is repeated until the off-diagonal terms are "sufficiently" small.

*Given's method* also involves transforming a symmetric matrix into a simpler form. However, in contrast to the Jacobi method, the simpler form is tridiagonal. In addition, it differs in that the zeros that are created in off-diagonal positions are retained. Consequently, it is finite and, thus, more efficient than Jacobi's method.

*Householder's method* also transforms a symmetric matrix into a tridiagonal form. It is a finite method and is more efficient than Given's approach in that it reduces whole rows and columns of off-diagonal elements to zero.

Once a tridiagonal system is obtained from Given's or Householder's method, the remaining step involves finding the eigenvalues. A direct way to do this is to expand the determinant. The result is a sequence of polynomials that can be evaluated iteratively for the eigenvalues.

Aside from symmetric matrices, there are also techniques that are available when all eigenvalues of a general matrix are required. These include the *LR method* of Rutishauser and the *QR method* of Francis. Although the QR method is less efficient, it is usually the preferred approach because it is more stable. As such, it is considered to be the best general-purpose solution method.

Finally, it should be mentioned that the aforementioned techniques are often used in tandem to capitalize on their respective strengths. For example, Given's and Householder's methods can also be applied to nonsymmetric systems. The result will not be tridiagonal but rather a special type called the *Hessenberg form.* One approach is to exploit the speed of Householder's approach by employing it to transform the matrix to this form and then use the stable QR algorithm to find the eigenvalues. Additional information on these and other issues related to eigenvalues can be found in Ralston and Rabinowitz (1978), Wilkinson (1965), Fadeev and Fadeeva (1963), and Householder (1953, 1964). Computer codes can be found in a number of sources including Press et al. (1992). Rice (1983) discusses available software packages.

SOFTWARE

## 27.3 ODES AND EIGENVALUES WITH SOFTWARE PACKAGES

Software packages have great capabilities for solving ODEs and determining eigenvalues. This section outlines some of the ways in which they can be applied for this purpose.

### 27.3.1 Excel

Excel's direct capabilities for solving eigenvalue problems and ODEs are limited. However, if some programming is done (for example, macros), they can be combined with Excel's visualization and optimization tools to implement some interesting applications. Section 28.1 provides an example of how the Excel Solver can be used for parameter estimation of an ODE.

### 27.3.2 MATLAB

As might be expected, the standard MATLAB software package has excellent capabilities for determining eigenvalues and eigenvectors. However, it also has built-in functions for solving ODEs. The standard ODE solvers include two functions to implement the adaptive step-size Runge-Kutta Fehlberg method (recall Sec. 25.5.2). These are **ODE23,** which uses second- and third-order formulas to attain medium accuracy, and **ODE45,** which uses fourth- and fifth-order formulas to attain higher accuracy. The following example illustrates how they can be used to solve a system of ODEs.

EXAMPLE 27.9 Using MATLAB for Eigenvalues and ODEs

Problem Statement. Explore how MATLAB can be used to solve the following set of nonlinear ODEs from $t = 0$ to 20:

$$\frac{dx}{dt} = 1.2x - 0.6xy \qquad \frac{dy}{dt} = -0.8y + 0.3xy$$

where $x = 2$ and $y = 1$ at $t = 0$. As we will see in the next chapter (Sec. 28.2), such equations are referred to as *predator-prey equations.*

Solution. Before obtaining a solution with MATLAB, you must use a text processor to create an M-file containing the right-hand side of the ODEs. This M-file will then be accessed by the ODE solver [where $x = y(1)$ and $y = y(2)$]:

```
function yp = predprey(t,y)
yp = [1.2*y(1)-0.6*y(1)*y(2);-0.8*y(2)+0.3*y(1)*y(2)];
```

We stored this M-file under the name: predprey.m.

Next, start up MATLAB, and enter the following commands to specify the integration range and the initial conditions:

```
>> tspan = [0,20];
>> y0=[2,1];
```

The solver can then be invoked by

```
>> [t,y]=ode23('predprey',tspan,y0);
```

This command will then solve the differential equations in predprey.m over the range defined by tspan using the initial conditions found in y0. The results can be displayed by simply typing

```
>> plot(t,y)
```

which yields Fig. 27.9.

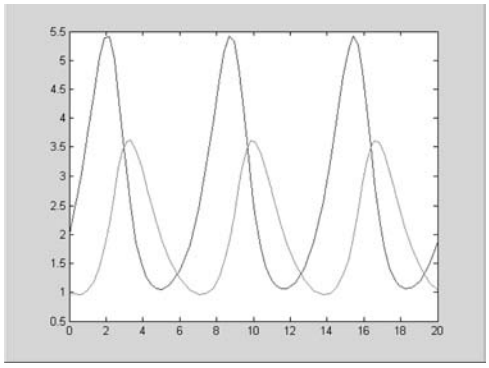



**FIGURE 27.9**
Solution of predator-prey model with MATLAB.

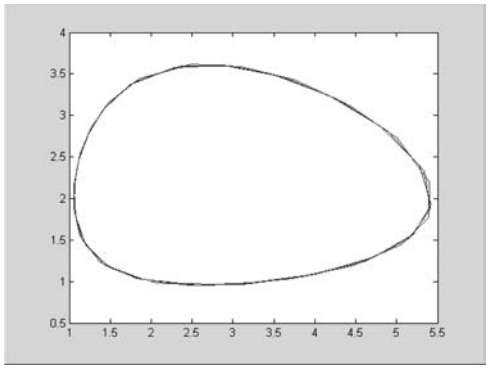



**FIGURE 27.10**
State-space plot of predator-prey model with MATLAB.

In addition, it is also instructive to generate a state-space plot, that is, a plot of the dependent variables versus each other by

```
>> plot(y(:,1),y(:,2))
```

which yields Fig. 27.10.

MATLAB also has a range of functions designed for stiff systems. These include ODE15S and ODE23S. As in the following example, they succeed where the standard functions fail.

EXAMPLE 27.10 MATLAB for Stiff ODEs

Problem Statement. *Van der Pol's equation* can be written as

$$\frac{dy_1}{dt} = y_2$$

$$\frac{dy_2}{dt} = \mu(1 - y_1^2)y_2 - y_1$$

As the parameter $\mu$ gets large, the system becomes progressively stiffer. Given the initial conditions, $y_1(0) = y_2(0) = 1$, use MATLAB to solve the following two cases

**(a)** For $\mu = 1$, use ODE45 to solve from $t = 0$ to 20.
**(b)** For $\mu = 1000$, use ODE23S to solve from $t = 0$ to 3000.

Solution.

**(a)** An M-file can be created to hold the differential equations,

```
function yp = vanderpol(t,y)
yp=[y(2);1*(1-y(1)^2)*y(2)-y(1)];
```

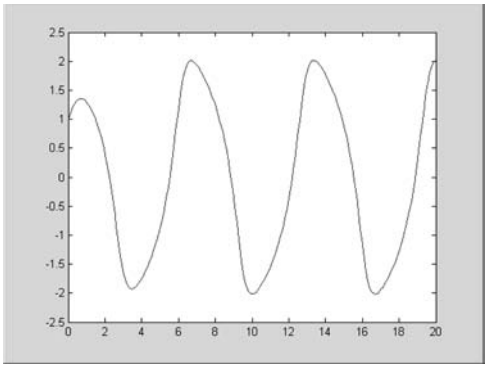
Then, as in Example 27.9, ODE45 can be invoked and the results plotted (Fig. 27.11),

```
>> tspan=[0,20];
>> y0=[1,1];
>> [t,y]=ode45('vanderpol',tspan,y0);
>> plot(t,y(:,1))
```

**(b)** If a standard solver like ODE45 is used for the stiff case ($\mu = 1000$), it will fail miserably (try it, if you like). However, ODE23S does an efficient job. After revising the M-file to reflect the new value of $\mu$, the solution can be obtained and graphed (Fig. 27.12),

```
>> tspan=[0,3000];
>> y0=[1,1];
```

**FIGURE 27.11**
Nonstiff form of Van der Pol's equation solved with MATLAB's ODE45 function.



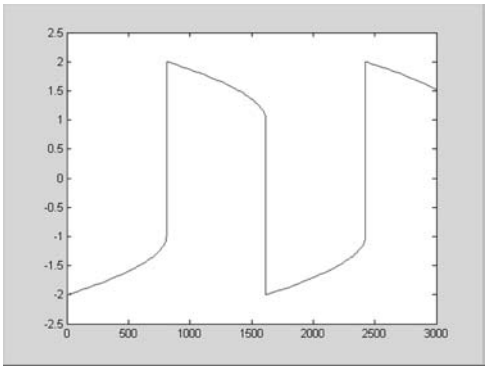



**FIGURE 27.12**
Stiff form of Van der Pol's equation solved with MATLAB's ODE23S function.

```
>> [t,y]=ode23S('vanderpol',tspan,y0);
>> plot(t,y(:,1))
```

Notice how this solution has much sharper edges than for case **(a)**. This is a visual manifestation of the "stiffness" of the solution.

For eigenvalues, the capabilities are also very easy to apply. Recall that, in our discussion of stiff systems in Chap. 26, we presented the stiff system defined by Eq. (26.6). Such linear ODEs can be written as an eigenvalue problem of the form

$$\begin{bmatrix} 5-\lambda & -3 \\ -100 & 301-\lambda \end{bmatrix} \begin{Bmatrix} e_1 \\ e_2 \end{Bmatrix} = \{0\}$$

where $\lambda$ and $\{e\}$ = the eigenvalue and eigenvector, respectively.

MATLAB can then be employed to evaluate both the eigenvalues (d) and eigenvectors (v) with the following simple commands:

```
>> a=[5 -3;-100 301];
>> [v,d]=eig(a)

v =
   -0.9477    0.0101
   -0.3191   -0.9999

d =
    3.9899         0
         0  302.0101
```

Thus, we see that the eigenvalues are of quite different magnitudes, which is typical of a stiff system.

The eigenvalues can be interpreted by recognizing that the general solution for a system of ODEs can be represented as the sum of exponentials. For example, the solution for the present case would be of the form

$$y_1 = c_{11}e^{-3.9899t} + c_{12}e^{-302.0101t}$$
$$y_2 = c_{21}e^{-3.9899t} + c_{22}e^{-302.0101t}$$

where $c_{ij}$ = the part of the initial condition for $y_i$ that is associated with the $j$th eigenvalue. It should be noted that the $c$'s can be evaluated from the initial conditions and the eigenvectors. Any good book on differential equations, for example, Boyce and DiPrima (1992), will provide an explanation of how this can be done.

Because, for the present case, all the eigenvalues are positive (and hence negative in the exponential function), the solution consists of a series of decaying exponentials. The one with the largest eigenvalue (in this case, 302.0101) would dictate the step size if an explicit solution technique were used.

### 27.3.3 Mathcad

Mathcad has a number of different functions that solve differential equations and determine eigenvalues and eigenvectors. The most basic technique employed by Mathcad to solve systems of first-order differential equations is a fixed step-size fourth-order Runge Kutta algorithm. This is provided by the **rkfixed** function. Although this is a good all-purpose integrator, it is not always efficient. Therefore, Mathcad supplies **Rkadapt**, which is a variable step sized version of **rkfixed**. It is well suited for functions that change rapidly in some regions and slowly in others. Similarly, if you know your solution is a smooth function, then you may find that the Mathcad **Bulstoer** function works well. This function employs the Bulirsch-Stoer method and is often both efficient and highly accurate for smooth functions.

Stiff differential equations are at the opposite end of the spectrum. Under these conditions the **rkfixed** function may be very inefficient or unstable. Therefore, Mathcad provides two special methods specifically designed to handle stiff systems. These functions are called **Stiffb** and **Stiffr** and are based on a modified Bulirsch-Stoer method for stiff systems and the Rosenbrock method.

As an example, let's use Mathcad to solve the following nonlinear ODEs,

$$\frac{dy_1}{dt} = 1.2y_1 - 0.6y_1y_2$$
$$\frac{dy_2}{dt} = -0.8y_2 + 0.3y_1y_2$$

with the initial conditions, $y_1 = 2$ and $y_2 = 1$. This system, called *Lotka-Volterra equations*, are used by environmental engineers and ecologists to evaluate the interactions of predators ($y_2$) and prey ($y_1$).

As in Fig. 27.13, the definition symbol is first used to define the vector D(u, y) holding the right-hand sides of the ODEs for input to **rkfixed**. Note that $y_1$ and $y_2$ in the ODEs are changed to $y_0$ and $y_1$ to comply with Mathcad requirements. In addition, we define the

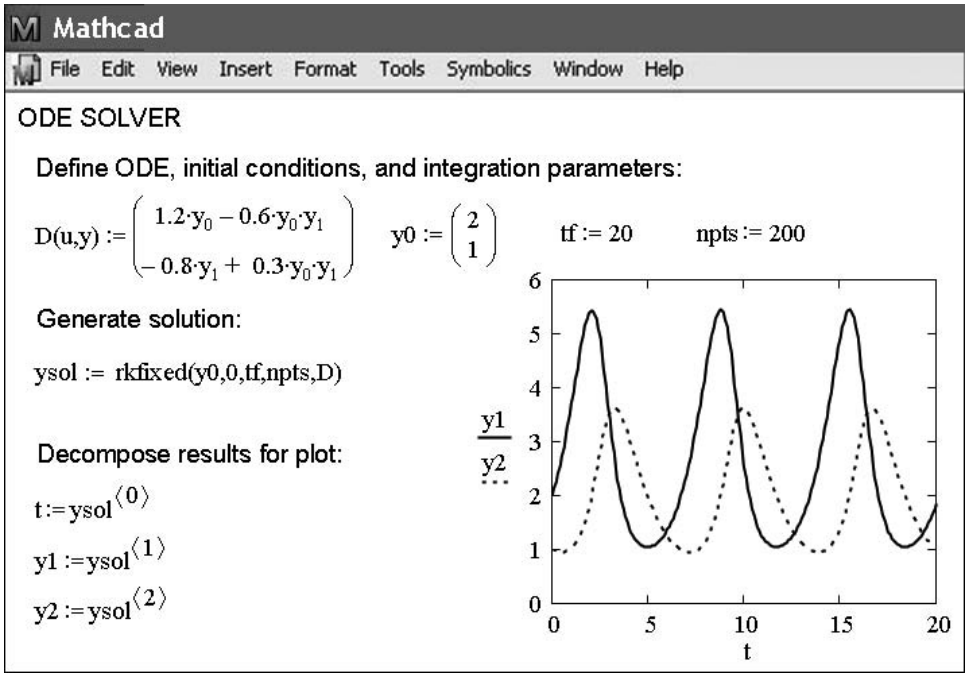



**FIGURE 27.13**
Mathcad screen to solve a system of ODEs.

initial conditions ($y_0$), the integration limit (tf) and the number of values we want to generate (npts). The solutions for **rkfixed** with 200 steps between $t = 0$ and tf are stored in the ysol matrix. The solution is displayed graphically in the plot in Fig. 27.13.

Next, we can illustrate how Mathcad evaluates eigenvalues and eigenvectors. The function **eigenvals**(**M**) returns the eigenvalues of the square matrix **M**. The function **eigenvecs**(**M**) returns a matrix containing normalized eigenvectors corresponding to the eigenvectors of **M** whereas **eigenvec**(**M,e**) returns the eigenvector corresponding to the eigenvalue **e**. We can illustrate these functions for the system given by [recall Eq. (26.6)]

$$\frac{dy_1}{dt} = -5y_1 + 3y_2$$

$$\frac{dy_2}{dt} = 100y_1 - 301y_2$$

The results are shown in Fig. 27.14. Because the eigenvalues (aa) are of different magnitudes, the system is stiff. Note that bb holds the specific eigenvector associated with the smaller eigenvalue. The result cc is a matrix containing both eigenvectors as its columns.

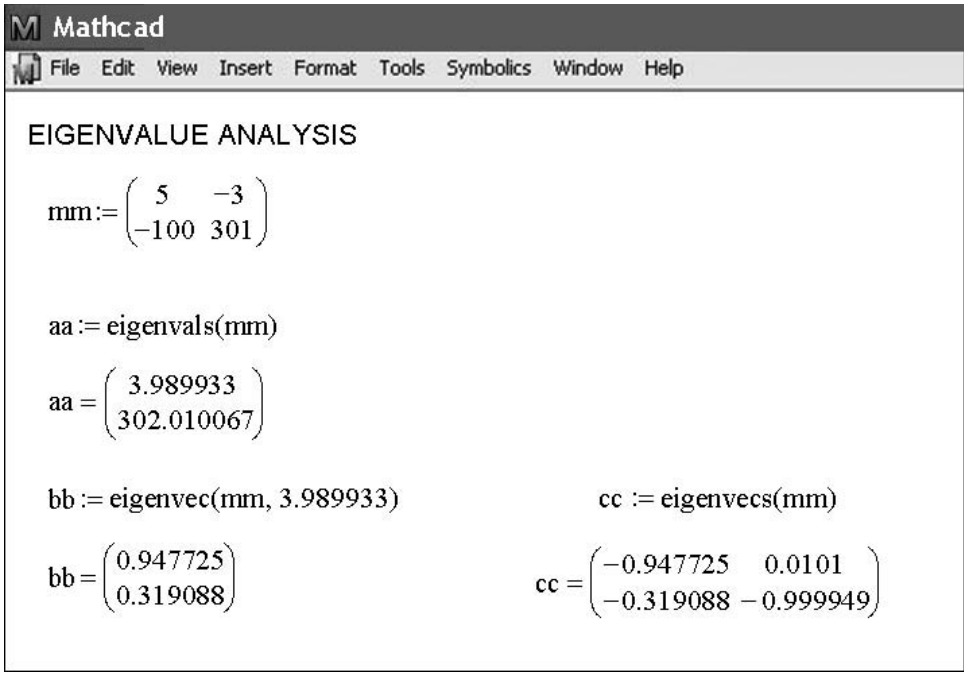



**FIGURE 27.14**
Mathcad screen to solve for the eigenvalues of a system of ODEs.

## PROBLEMS

**27.1** A steady-state heat balance for a rod can be represented as

$$\frac{d^2T}{dx^2} - 0.15T = 0$$

Obtain an analytical solution for a 10-m rod with $T(0) = 240$ and $T(10) = 150$.

**27.2** Use the shooting method to solve Prob. 27.1.

**27.3** Use the finite-difference approach with $\Delta x = 1$ to solve Prob. 27.1.

**27.4** Use the shooting method to solve

$$7\frac{d^2y}{dx^2} - 2\frac{dy}{dx} - y + x = 0$$

with the boundary conditions $y(0) = 5$ and $y(20) = 8$.

**27.5** Solve Prob. 27.4 with the finite-difference approach using $\Delta x = 2$.

**27.6** Use the shooting method to solve

$$\frac{d^2T}{dx^2} - 1 \times 10^{-7}(T + 273)^4 + 4(150 - T) = 0 \qquad \text{(P27.6)}$$

Obtain a solution for boundary conditions: $T(0) = 200$ and $T(0.5) = 100$.

**27.7** Differential equations like the one solved in Prob. 27.6 can often be simplified by linearizing their nonlinear terms. For example, a first-order Taylor series expansion can be used to linearize the quartic term in Eq. (P27.6) as

$$1 \times 10^{-7}(T + 273)^4 = 1 \times 10^{-7}(T_b + 273)^4 + 4 \times 10^{-7}(T_b + 273)^3(T - T_b)$$

where $T_b$ is a base temperature about which the term is linearized. Substitute this relationship into Eq. (P27.6), and then solve the resulting linear equation with the finite-difference approach. Employ $T_b = 150$ and $\Delta x = 0.01$ to obtain your solution.

**27.8** Repeat Example 27.4 but for three masses. Produce a plot like Fig. 27.6 to identify the principle modes of vibration. Change all the $k$'s to 240.

**27.9** Repeat Example 27.6, but for five interior points ($h = 3/6$).

**27.10** Use minors to expand the determinant of

$$\begin{bmatrix} 2-\lambda & 8 & 10 \\ 8 & 4-\lambda & 5 \\ 10 & 5 & 7-\lambda \end{bmatrix}$$

**27.11** Use the power method to determine the highest eigenvalue and corresponding eigenvector for Prob. 27.10.

**27.12** Use the power method to determine the lowest eigenvalue and corresponding eigenvector for Prob. 27.10.

**27.13** Develop a user-friendly computer program to implement the shooting method for a linear second-order ODE. Test the program by duplicating Example 27.1.

**27.14** Use the program developed in Prob. 27.13 to solve Probs. 27.2 and 27.4.

**27.15** Develop a user-friendly computer program to implement the finite-difference approach for solving a linear second-order ODE. Test it by duplicating Example 27.3.

**27.16** Use the program developed in Prob. 27.15 to solve Probs. 27.3 and 27.5.

**27.17** Develop a user-friendly program to solve for the largest eigenvalue with the power method. Test it by duplicating Example 27.7.

**27.18** Develop a user-friendly program to solve for the smallest eigenvalue with the power method. Test it by duplicating Example 27.8.

**27.19** Use the Excel Solver to directly solve (that is, without linearization) Prob. 27.6 using the finite-difference approach. Employ $\Delta x = 0.1$ to obtain your solution.

**27.20** Use MATLAB to integrate the following pair of ODEs from $t = 0$ to 100:

$$\frac{dy_1}{dt} = 0.35y_1 - 1.6y_1y_2 \qquad \frac{dy_2}{dt} = 0.04y_1y_2 - 0.15y_2$$

where $y_1 = 1$ and $y_2 = 0.05$ at $t = 0$. Develop a state-space plot ($y_1$ versus $y_2$) of your results.

**27.21** The following differential equation was used in Sec. 8.4 to analyze the vibrations of an automobile shock absorber:

$$1.25 \times 10^6 \frac{d^2x}{dt^2} + 1 \times 10^7 \frac{dx}{dt} + 1.5 \times 10^9 x = 0$$

Transform this equation into a pair of ODEs. **(a)** Use MATLAB to solve these equations from $t = 0$ to 0.4 for the case where $x = 0.5$, and $dx/dt = 0$ at $t = 0$. **(b)** Use MATLAB to determine the eigenvalues and eigenvectors for the system.

**27.22** Use MATLAB or Mathcad to integrate

$$\frac{dx}{dt} = -\sigma x + \sigma y$$

$$\frac{dy}{dt} = rx - y - xz$$

$$\frac{dz}{dt} = -bz + xy$$

where $\sigma = 10$, $b = 2.666667$, and $r = 28$. Employ initial conditions of $x = y = z = 5$ and integrate from $t = 0$ to 20.

**27.23** Use finite differences to solve the boundary-value ordinary differential equation

$$\frac{d^2u}{dx^2} + 6\frac{du}{dx} - u = 2$$

with boundary conditions $u(0) = 10$ and $u(2) = 1$. Plot the results of $u$ versus $x$. Use $\Delta x = 0.1$.

**27.24** Solve the nondimensionalized ODE using finite difference methods that describe the temperature distribution in a circular rod with internal heat source $S$

$$\frac{d^2T}{dr^2} + \frac{1}{r}\frac{dT}{dr} + S = 0$$

over the range $0 \le r \le 1$, with the boundary conditions

$$T(r = 1) = 1 \qquad \left.\frac{dT}{dr}\right|_{r=0} = 0$$

for $S = 1$, 10, and 20 K/m$^2$. Plot the temperature versus radius.

**27.25** Derive the set of differential equations for a three mass–four spring system (Fig. P27.25) that describes their time motion. Write the three differential equations in matrix form,

$$[\text{Acceleration vector}] + [k/m \text{ matrix}][\text{displacement vector } x] = 0$$

Note each equation has been divided by the mass. Solve for the eigenvalues and natural frequencies for the following values of mass and spring constants: $k_1 = k_4 = 15$ N/m, $k_2 = k_3 = 35$ N/m, and $m_1 = m_2 = m_3 = 1.5$ kg.

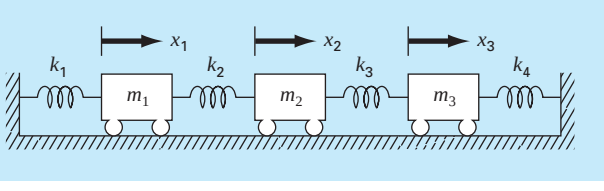



**Figure P27.25**

**27.26** Consider the mass-spring system in Fig. P27.26. The frequencies for the mass vibrations can be determined by solving for the eigenvalues and by applying $M\ddot{x} + kx = 0$, which yields

$$\begin{bmatrix} m_1 & 0 & 0 \\ 0 & m_2 & 0 \\ 0 & 0 & m_3 \end{bmatrix} \begin{Bmatrix} \ddot{x}_1 \\ \ddot{x}_2 \\ \ddot{x}_3 \end{Bmatrix} + \begin{bmatrix} 2k & -k & -k \\ -k & 2k & -k \\ -k & -k & 2k \end{bmatrix} \begin{Bmatrix} x_1 \\ x_2 \\ x_3 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix}$$

Applying the guess $x = x_0 e^{i\omega t}$ as a solution, we get the following matrix:

$$\begin{bmatrix} 2k - m_1\omega^2 & -k & -k \\ -k & 2k - m_2\omega^2 & -k \\ -k & -k & 2k - m_3\omega^2 \end{bmatrix} \begin{Bmatrix} x_{01} \\ x_{02} \\ x_{03} \end{Bmatrix} e^{i\omega t} = \begin{Bmatrix} 0 \\ 0 \\ 0 \end{Bmatrix}$$

Use MATLAB's `eig` command to solve for the eigenvalues of the $k - m\omega^2$ matrix above. Then use these eigenvalues to solve for the frequencies ($\omega$). Let $m_1 = m_2 = m_3 = 1$ kg, and $k = 2$ N/m.

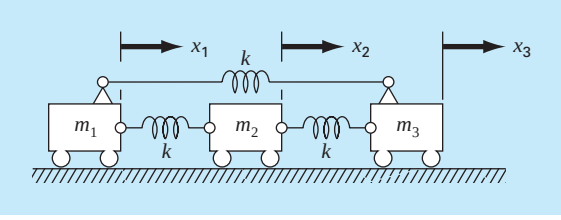



**Figure P27.26**

27.27 The following nonlinear, parasitic ODE was suggested by Hornbeck (1975):

$$\frac{dy_1}{dt} = 5(y_1 - t^2)$$

If the initial condition is $y_1(0) = 0.08$, obtain a solution from $t = 0$ to 5:

**(a)** Analytically.

**(b)** Using the fourth-order RK method with a constant step size of 0.03125.

**(c)** Using the MATLAB function ODE45.

**(d)** Using the MATLAB function ODE23s.

**(e)** Using the MATLAB function ODE23tb.

Present your results in graphical form.

27.28 A heated rod with a uniform heat source can be modeled with the *Poisson equation*,

$$\frac{d^2T}{dx^2} = -f(x)$$

Given a heat source $f(x) = 25$ and the boundary conditions, $T(x = 0) = 40$ and $T(x = 10) = 200$, solve for the temperature distribution with **(a)** the shooting method and **(b)** the finite-difference method ($\Delta x = 2$).

27.29 Repeat Prob. 27.28, but for the following heat source: $f(x) = 0.12x^3 - 2.4x^2 + 12x$.

27.30 Suppose that the position of a falling object is governed by the following differential equation,

$$\frac{d^2x}{dt^2} + \frac{c}{m}\frac{dx}{dt} - g = 0$$

where $c$ = a first-order drag coefficient = 12.5 kg/s, $m$ = mass = 70 kg, and $g$ = gravitational acceleration = 9.81 m/s$^2$. Use the shooting method to solve this equation for position and velocity given the boundary conditions, $x(0) = 0$ and $x(12) = 500$.

27.31 Repeat Example 27.3, but insulate the left end of the rod. That is, change the boundary condition at the left end of the rod to $T'(0) = 0$.

CHAPTER 28

# Case Studies: Ordinary Differential Equations

The purpose of this chapter is to solve some ordinary differential equations using the numerical methods presented in Part Seven. The equations originate from practical engineering applications. Many of these applications result in nonlinear differential equations that cannot be solved using analytic techniques. Therefore, numerical methods are usually required. Thus, the techniques for the numerical solution of ordinary differential equations are fundamental capabilities that characterize good engineering practice. The problems in this chapter illustrate some of the trade-offs associated with various methods developed in Part Seven.

Section 28.1 derives from a chemical engineering problem context. It demonstrates how the transient behavior of chemical reactors can be simulated. It also illustrates how optimization can be used to estimate parameters for ODEs.

Sections 28.2 and 28.3, which are taken from civil and electrical engineering, respectively, deal with the solution of systems of equations. In both cases, high accuracy is demanded, and as a consequence, a fourth-order RK scheme is used. In addition, the electrical engineering application also deals with determining eigenvalues.

Section 28.4 employs a variety of different approaches to investigate the behavior of a swinging pendulum. This problem also utilizes two simultaneous equations. An important aspect of this example is that it illustrates how numerical methods allow nonlinear effects to be incorporated easily into an engineering analysis.

## 28.1 USING ODES TO ANALYZE THE TRANSIENT RESPONSE OF A REACTOR (CHEMICAL/BIO ENGINEERING)

Background. In Sec. 12.1, we analyzed the steady state of a series of reactors. In addition to steady-state computations, we might also be interested in the transient response of a completely mixed reactor. To do this, we have to develop a mathematical expression for the accumulation term in Eq. (12.1).

Accumulation represents the change in mass in the reactor per change in time. For a constant-volume system, it can be simply formulated as

$$\text{Accumulation} = V\frac{dc}{dt} \tag{28.1}$$

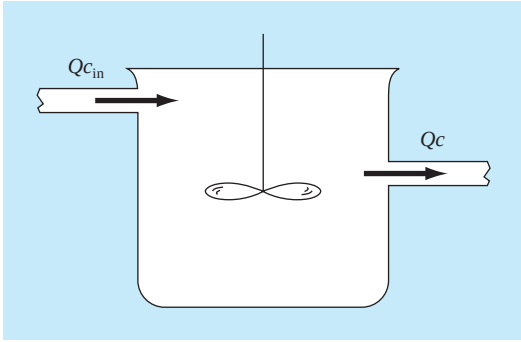



**FIGURE 28.1**
A single, completely mixed reactor with an inflow and an outflow.

where $V$ = volume and $c$ = concentration. Thus, a mathematical formulation for accumulation is volume times the derivative of $c$ with respect to $t$.

In this application we will incorporate the accumulation term into the general mass-balance framework we developed in Sec. 12.1. We will then use it to simulate the dynamics of a single reactor and a system of reactors. In the latter case, we will show how the system's eigenvalues can be determined and provide insight into its dynamics. Finally, we will illustrate how optimization can be used to estimate the parameters of mass-balance models.

Solution. Equations (28.1) and (12.1) can be used to represent the mass balance for a single reactor such as the one shown in Fig. 28.1:

$$V\frac{dc}{dt} = Qc_{in} - Qc \tag{28.2}$$

Accumulation = inputs − outputs

Equation (28.2) can be used to determine transient or time-variable solutions for the reactor. For example, if $c = c_0$ at $t = 0$, calculus can be employed to analytically solve Eq. (28.2) for

$$c = c_{in}\left(1 - e^{-(Q/V)t}\right) + c_0 e^{-(Q/V)t}$$

If $c_{in} = 50$ mg/m$^3$, $Q = 5$ m$^3$/min, $V = 100$ m$^3$, and $c_0 = 10$ mg/m$^3$, the equation is

$$c = 50(1 - e^{-0.05t}) + 10e^{-0.05t}$$

Figure 28.2 shows this exact, analytical solution.

Euler's method provides an alternative approach for solving Eq. (28.2). Figure 28.2 includes two solutions with different step sizes. As the step size is decreased, the numerical solution converges on the analytical solution. Thus, for this case, the numerical method can be used to check the analytical result.

Besides checking the results of an analytical solution, numerical solutions have added value in those situations where analytical solutions are impossible or so difficult that they are impractical. For example, aside from a single reactor, numerical methods have utility when simulating the dynamics of systems of reactors. For example, ODEs can be written

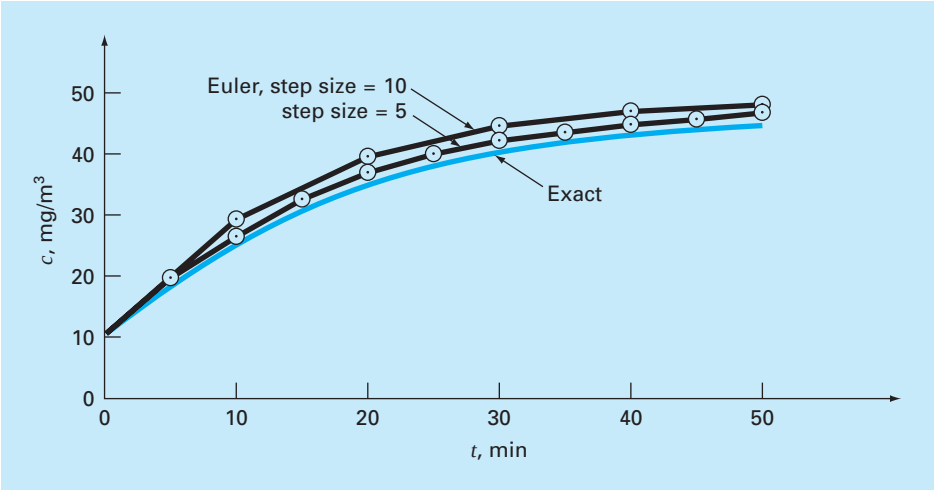



**FIGURE 28.2**
Plot of analytical and numerical solutions of Eq. (28.2). The numerical solutions are obtained with Euler's method using different step sizes.

for the five coupled reactors in Fig. 12.3. The mass balance for the first reactor can be written as

$$V_1\frac{dc_1}{dt} = Q_{01}c_{01} + Q_{31}c_3 - Q_{12}c_1 - Q_{15}c_1$$

or, substituting parameters (note that $Q_{01}c_{01} = 50$ mg/min, $Q_{03}c_{03} = 160$ mg/min, $V_1 = 50$ m$^3$, $V_2 = 20$ m$^3$, $V_3 = 40$ m$^3$, $V_4 = 80$ m$^3$, and $V_5 = 100$ m$^3$),

$$\frac{dc_1}{dt} = -0.12c_1 + 0.02c_3 + 1$$

Similarly, balances can be developed for the other reactors as

$$\frac{dc_2}{dt} = 0.15c_1 - 0.15c_2$$

$$\frac{dc_3}{dt} = 0.025c_2 - 0.225c_3 + 4$$

$$\frac{dc_4}{dt} = 0.1c_3 - 0.1375c_4 + 0.025c_5$$

$$\frac{dc_5}{dt} = 0.03c_1 + 0.01c_2 - 0.04c_5$$

Suppose that at $t = 0$ all the concentrations in the reactors are at zero. Compute how their concentrations will increase over the next hour.

The equations can be integrated with the fourth-order RK method for systems of equations and the results are depicted in Fig. 28.3. Notice that each of the reactors shows a different transient response to the introduction of chemical. These responses can be parameterized by a 90 percent response time $t_{90}$, which measures the time required for each reactor to reach 90 percent of its ultimate steady-state level. The times range from about

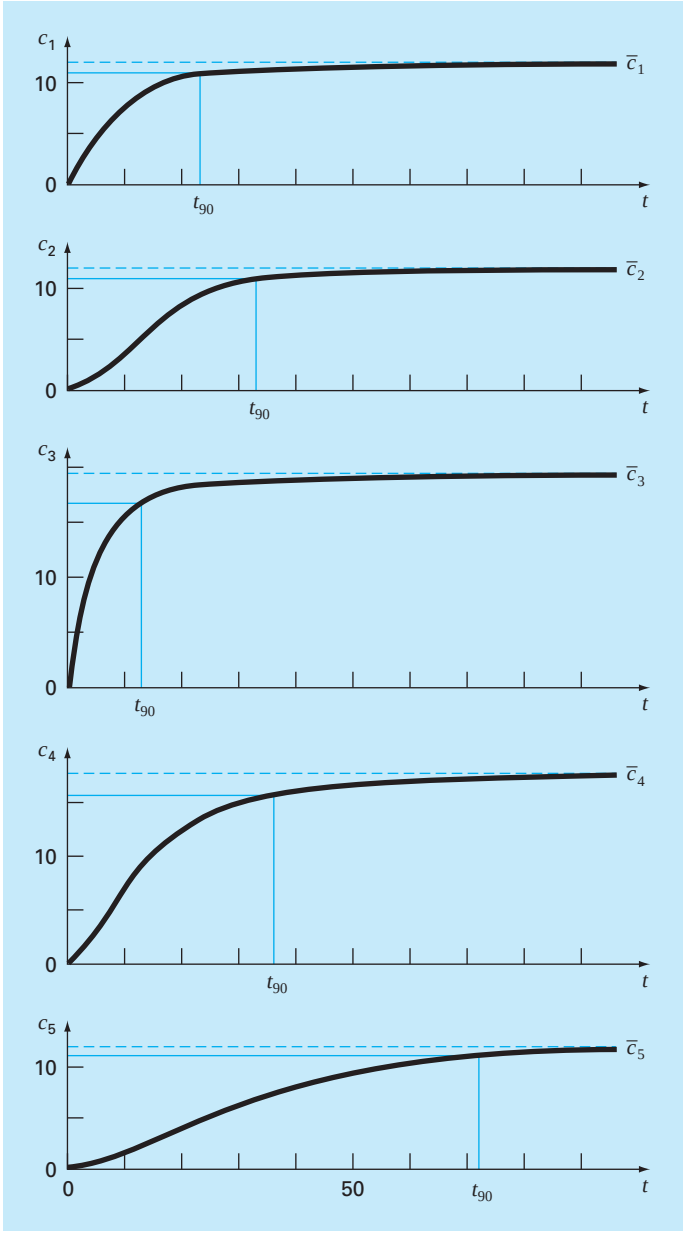



**FIGURE 28.3**
Plots of transient or dynamic response of the network of reactors from Fig. 12.3. Note that all the reactors eventually approach their steady-state concentrations previously computed in Sec. 12.1. In addition, the time to steady state is parameterized by the 90 percent response time $t_{90}$.

10 min for reactor 3 to about 70 min for reactor 5. The response times of reactors 4 and 5 are of particular concern because the two outflow streams for the system exit these tanks. Thus, a chemical engineer designing the system might change the flows or volumes of the reactors to speed up the response of these tanks while still maintaining the desired outputs. Numerical methods of the sort described in this part of the book can prove useful in these design calculations.

Further insight into the system's response characteristics can be developed by computing its eigenvalues. First, the system of ODEs can be written as an eigenvalue problem as

$$\begin{bmatrix} 0.12-\lambda & 0 & -0.02 & 0 & 0 \\ -0.15 & 0.15-\lambda & 0 & 0 & 0 \\ 0 & -0.025 & 0.225-\lambda & 0 & 0 \\ 0 & 0 & -0.1 & 0.1375-\lambda & -0.025 \\ -0.03 & -0.01 & 0 & 0 & 0.04-\lambda \end{bmatrix} \begin{Bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \end{Bmatrix} = \{0\}$$

where $\lambda$ and $\{e\}$ = the eigenvalue and the eigenvector, respectively.

A package like MATLAB software can be used to very conveniently generate the eigenvalues and eigenvectors,

```
>> a=[0.12 0.0 -0.02 0.0 0.0;-.15 0.15 0.0 0.0 0.0;0.0
-0.025 0.225 0.0 0.0; 0.0 0.0 -.1 0.1375 -0.025;-0.03 -0.01
0.0 0.0 0.04];

>> [e,l]=eig(a)
e =
         0         0   -0.1228   -0.1059    0.2490
         0         0    0.2983    0.5784    0.8444
         0         0    0.5637    0.3041    0.1771
    1.0000    0.2484   -0.7604   -0.7493    0.3675
         0    0.9687    0.0041   -0.0190   -0.2419

l =
    0.1375         0         0         0         0
         0    0.0400         0         0         0
         0         0    0.2118         0         0
         0         0         0    0.1775         0
         0         0         0         0    0.1058
```

The eigenvalues can be interpreted by recognizing that the general solution for a system of ODEs can be represented as the sum of exponentials. For example, for reactor 1, the general solution would be of the form

$$c_1 = c_{11}e^{-\lambda_1 t} + c_{12}e^{-\lambda_2 t} + c_{13}e^{-\lambda_3 t} + c_{14}e^{-\lambda_4 t} + c_{15}e^{-\lambda_5 t}$$

where $c_{ij}$ = the part of the initial condition for reactor $i$ that is associated with the $j$th eigenvalue. Thus, because, for the present case, all the eigenvalues are positive (and hence negative in the exponential function), the solution consists of a series of decaying exponentials. The one with the smallest eigenvalue (in our case, 0.04) will be the slowest. In some

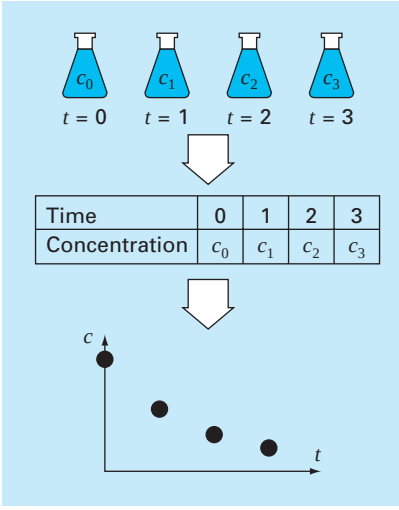



**FIGURE 28.4**
A simple experiment to collect rate data for a chemical compound that decays with time (reprinted from Chapra 1997).

cases, the engineer performing this analysis could be able to relate this eigenvalue back to the system parameters. For example, the ratio of the outflow from reactor 5 to its volume is $(Q_{55} + Q_{54})/V_5 = 4/100 = 0.04$. Such information can then be used to modify the system's dynamic performance.

The final topic we would like to review within the present context is *parameter estimation*. One area where this occurs frequently is in *reaction kinetics*, that is, the quantification of chemical reaction rates.

A simple example is depicted in Fig. 28.4. A series of beakers are set up containing a chemical compound that decays over time. At time intervals, the concentration in one of the beakers is measured and recorded. Thus, the result is a table of times and concentrations.

One model that is commonly used to describe such data is

$$\frac{dc}{dt} = -kc^n \tag{28.3}$$

where $k =$ a reaction rate and $n =$ the order of the reaction. Chemical engineers use concentration-time data of the sort depicted in Fig. 28.4 to estimate $k$ and $n$. One way to do this is to guess values of the parameters and then solve Eq. (28.3) numerically. The predicted values of concentration can be compared with the measured concentrations and an assessment of the fit made. If the fit is deemed inadequate (for example, by examining a plot or a statistical measure like the sum of the squares of the residuals), the guesses are adjusted and the procedure repeated until a decent fit is attained.

The following data can be fit in this fashion:

| $t$, d | 0 | 1 | 3 | 5 | 10 | 15 | 20 |
|---|---|---|---|---|---|---|---|
| $c$, mg/L | 12 | 10.7 | 9 | 7.1 | 4.6 | 2.5 | 1.8 |

| | A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|---|
| 1 | Fitting of reaction rate | | | | | | | |
| 2 | data with the integral/least-squares approach | | | | | | | |
| 3 | k | 0.091528 | | | | | | |
| 4 | n | 1.044425 | | | | | | |
| 5 | dt | 1 | | | | | | |
| 6 | t | k1 | k2 | k3 | k4 | cp | cm | (cp-cm)^2 |
| 7 | 0 | −1.22653 | −1.16114 | −1.16462 | −1.10248 | 12 | 12 | 0 |
| 8 | 1 | −1.10261 | −1.04409 | −1.04719 | −0.99157 | 10.83658 | 10.7 | 0.018653 |
| 9 | 2 | −0.99169 | −0.93929 | −0.94206 | −0.89225 | 9.790448 | | |
| 10 | 3 | −0.89235 | −0.84541 | −0.84788 | −0.80325 | 8.849344 | 9 | 0.022697 |
| 11 | 4 | −0.80334 | −0.76127 | −0.76347 | −0.72346 | 8.002317 | | |
| 12 | 5 | −0.72354 | −0.68582 | −0.68779 | −0.65191 | 7.239604 | 7.1 | 0.019489 |
| 13 | 6 | −0.65198 | −0.61814 | −0.61989 | −0.5877 | 6.552494 | | |
| 14 | 7 | −0.58776 | −0.55739 | −0.55895 | −0.53005 | 5.933207 | | |
| 15 | 8 | −0.53011 | −0.50283 | −0.50424 | −0.47828 | 5.374791 | | |
| 16 | 9 | −0.47833 | −0.45383 | −0.45508 | −0.43175 | 4.871037 | | |
| 17 | 10 | −0.4318 | −0.40978 | −0.4109 | −0.38993 | 4.416389 | 4.6 | 0.033713 |
| 18 | 11 | −0.38997 | −0.37016 | −0.37117 | −0.35231 | 4.005877 | | |
| 19 | 12 | −0.35234 | −0.33453 | −0.33543 | −0.31846 | 3.635053 | | |
| 20 | 13 | −0.31849 | −0.30246 | −0.30326 | −0.28798 | 3.299934 | | |
| 21 | 14 | −0.28801 | −0.27357 | −0.2743 | −0.26054 | 2.996949 | | |
| 22 | 15 | −0.26056 | −0.24756 | −0.24821 | −0.23581 | 2.7229 | 2.5 | 0.049684 |
| 23 | 16 | −0.23583 | −0.22411 | −0.22469 | −0.21352 | 2.474917 | | |
| 24 | 17 | −0.21354 | −0.20297 | −0.20349 | −0.19341 | 2.250426 | | |
| 25 | 18 | −0.19343 | −0.18389 | −0.18436 | −0.17527 | 2.047117 | | |
| 26 | 19 | −0.17529 | −0.16668 | −0.16711 | −0.1589 | 1.862914 | | |
| 27 | 20 | −0.15891 | −0.15115 | −0.15153 | −0.14412 | 1.695953 | 1.8 | 0.010826 |
| 28 | | | | | | | | |
| 29 | | | | | | | SSR = | 0.155062 |

**FIGURE 28.5**
The application of a spreadsheet and numerical methods to determine the order and rate coefficient of reaction data. This application was performed with the Excel spreadsheet.

The solution to this problem is shown in Fig. 28.5. The Excel spreadsheet was used to perform the computation.

Initial guesses for the reaction rate and order are entered into cells B3 and B4, respectively, and the time step for the numerical calculation is typed into cell B5. For this case, a column of calculation times is entered into column A starting at 0 (cell A7) and ending at 20 (cell A27). The $k_1$ through $k_4$ coefficients of the fourth-order RK method are then calculated in the block B7..E27. These are then used to determine the predicted concentrations (the $c_p$ values) in column F. The measured values ($c_m$) are entered in column G adjacent to the corresponding predicted values. These are then used in conjunction with the predicted values to compute the squared residual in column H. These values are then summed in cell H29.

At this point, the Excel Solver can be used to determine the best parameter values. Once you have accessed the Solver, you are prompted for a target or solution cell (H29), queried whether you want to maximize or minimize the target cell (minimize), and prompted for the cells that are to be varied (B3..B4). You then activate the algorithm

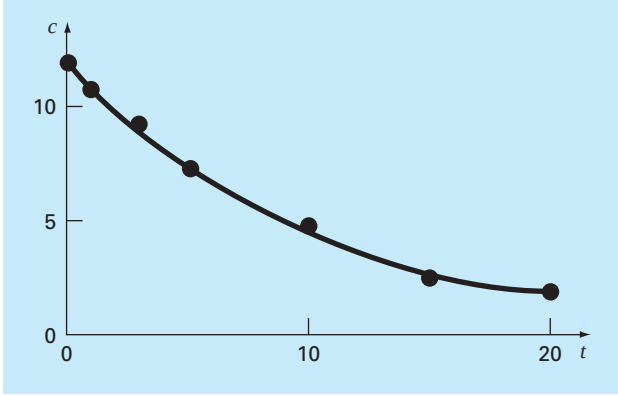



**FIGURE 28.6**
Plot of fit generated with the integral/least-squares approach.

[*s*(olve)], and the results are as in Fig. 28.5. As shown, the values in cells B3..B4 ($k = 0.0915$ and $n = 1.044$) minimize the sum of the squares of the residuals (SSR = 0.155) between the predicted and measured data. A plot of the fit along with the data is shown in Fig. 28.6.

## 28.2 PREDATOR-PREY MODELS AND CHAOS (CIVIL/ENVIRONMENTAL ENGINEERING)

Background. Environmental engineers deal with a variety of problems involving systems of nonlinear ordinary differential equations. In this section, we will focus on two of these applications. The first relates to the so-called predator-prey models that are used to study the cycling of nutrient and toxic pollutants in aquatic food chains and biological treatment systems. The second are equations derived from fluid dynamics that are used to simulate the atmosphere. Aside from their obvious application to weather prediction, such equations have also been used to study air pollution and global climate change.

*Predator-prey models* were developed independently in the early part of the twentieth century by the Italian mathematician Vito Volterra and the American biologist Alfred J. Lotka. These equations are commonly called *Lotka-Volterra equations*. The simplest example is the following pair of ODEs:

$$\frac{dx}{dt} = ax - bxy \tag{28.4}$$

$$\frac{dy}{dt} = -cy + dxy \tag{28.5}$$

where $x$ and $y$ = the number of prey and predators, respectively, $a$ = the prey growth rate, $c$ = the predator death rate, and $b$ and $d$ = the rate characterizing the effect of the predator-prey interaction on prey death and predator growth, respectively. The multiplicative terms (that is, those involving $xy$) are what make such equations nonlinear.

An example of a simple model based on atmospheric fluid dynamics is the *Lorenz equations* developed by the American meteorologist Edward Lorenz,

$$\frac{dx}{dt} = -\sigma x + \sigma y \tag{28.6}$$

$$\frac{dy}{dt} = rx - y - xz \tag{28.7}$$

$$\frac{dz}{dt} = -bz + xy \tag{28.8}$$

Lorenz developed these equations to relate the intensity of atmospheric fluid motion, $x$, to temperature variations $y$ and $z$ in the horizontal and vertical directions, respectively. As with the predator-prey model, we see that the nonlinearity is localized in simple multiplicative terms ($xz$ and $xy$).

Use numerical methods to obtain solutions for these equations. Plot the results to visualize how the dependent variables change temporally. In addition, plot the dependent variables versus each other to see whether any interesting patterns emerge.

Solution. Use the following parameter values for the predator-prey simulation: $a = 1.2$, $b = 0.6$, $c = 0.8$, and $d = 0.3$. Employ initial conditions of $x = 2$ and $y = 1$ and integrate from $t = 0$ to 30. We will use the fourth-order RK method with double precision to obtain solutions.

The results using a step size of 0.1 are shown in Fig. 28.7. Note that a cyclical pattern emerges. Thus, because predator population is initially small, the prey grows exponentially. At a certain point, the prey become so numerous, that the predator population begins to grow. Eventually, the increased predators cause the prey to decline. This decrease, in turn, leads to a decrease of the predators. Eventually, the process repeats. Notice that, as expected, the predator peak lags the prey. Also, observe that the process has a fixed period, that is, it repeats in a set time.

Now, if the parameters used to simulate Fig. 28.7 were changed, although the general pattern would remain the same, the magnitudes of the peaks, lags, and period would change. Thus, there are an infinite number of cycles that could occur.

A phase-plane representation is useful in discerning the underlying structure of the model. Rather than plotting $x$ and $y$ versus $t$, we can plot $x$ versus $y$. This plot illustrates

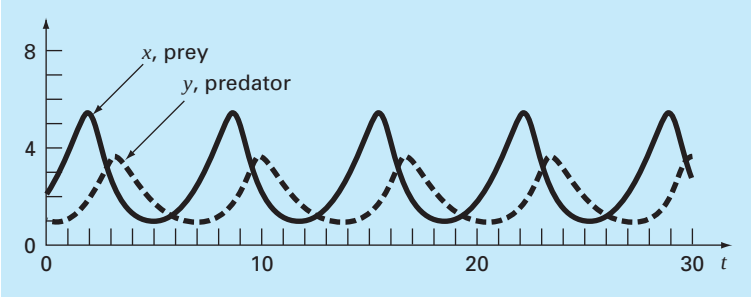



**FIGURE 28.7**
Time-domain representation of numbers of prey and predators for the Lotka-Volterra model.

the way that the state variables ($x$ and $y$) interact, and is referred to as a *phase-plane representation*.

Figure 28.8 shows the phase-plane representation for the case we are studying. Thus, the interaction between the predator and the prey defines a closed counterclockwise orbit. Notice that there is a critical or rest point at the center of the orbit. The exact location of this point can be determined by setting Eqs. (28.4) and (28.5) to steady state ($dy/dt = dx/dt = 0$) and solving for $(x, y) = (0, 0)$ and $(c/d, a/b)$. The former is the trivial result that if we start with neither predators nor prey, nothing will happen. The latter is the more interesting outcome that if the initial conditions are set at $x = c/d$ and $y = a/b$, the derivative will be zero and the populations will remain constant.

Now, let us use the same approach to investigate the trajectories of the Lorenz equations with the following parameter values: $\sigma = 10$, $b = 2.666667$, and $r = 28$. Employ initial conditions of $x = y = z = 5$ and integrate from $t = 0$ to 20. Again, we will use the fourth-order RK method with double precision to obtain solutions.

The results shown in Fig. 28.9 are quite different from the behavior of the Lotka-Volterra equations. The variable $x$ seems to be undergoing an almost random pattern of oscillations, bouncing around from negative values to positive values. However, even though

**FIGURE 28.8**
Phase-plane representation for the Lotka-Volterra model.

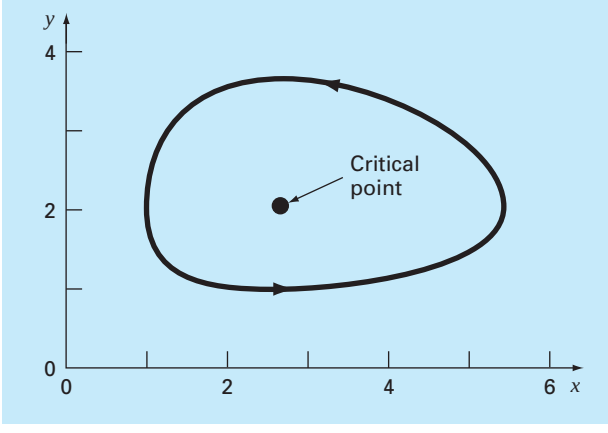
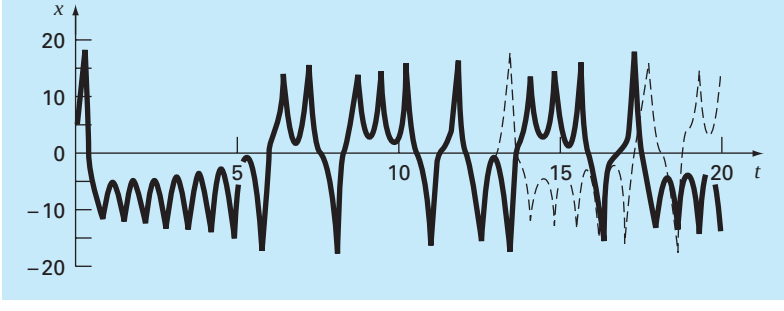



**FIGURE 28.9**
Time-domain representation of $x$ versus $t$ for the Lorenz equations. The solid time series is for the initial conditions (5, 5, 5). The dotted line is where the initial condition for $x$ is perturbed slightly (5.001, 5, 5).



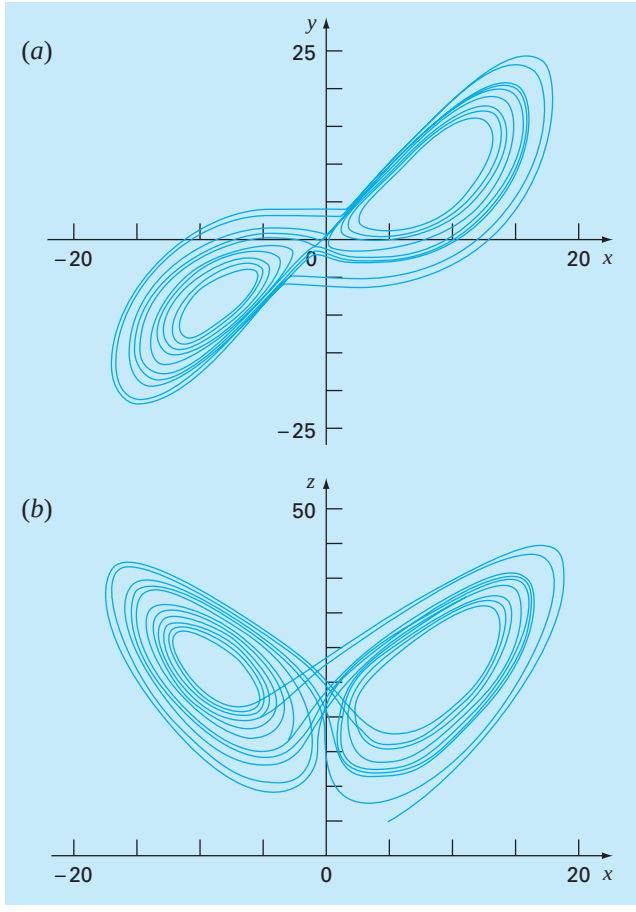



**FIGURE 28.10**
Phase-plane representation for the Lorenz equations. (*a*) *xy* projection and (*b*) *xz* projection.

the patterns seem random, the frequency of the oscillation and the amplitudes seem fairly consistent.

Another interesting feature can be illustrated by changing the initial condition for $x$ slightly (from 5 to 5.001). The results are superimposed as the dotted line in Fig. 28.9. Although the solutions track on each other for a time, after about $t = 12.5$ they diverge significantly. Thus, we can see that the Lorenz equations are quite sensitive to their initial conditions. In his original study, this led Lorenz to the conclusion that long-range weather forecasts might be impossible!

Finally, let us examine the phase-plane plots. Because we are dealing with three independent variables, we are limited to projections. Figure 28.10 shows projections in the *xy* and the *xz* planes. Notice how a structure is manifest when perceived from the phase-plane perspective. The solution forms orbits around what appear to be critical points. These points

are called *strange attractors* in the jargon of mathematicians who study such nonlinear systems.

Solutions such as the type we have explored for the Lorenz equations are referred to as *chaotic* solutions. The study of chaos and nonlinear systems presently represents an exciting area of analysis that has implications to mathematics as well as to science and engineering.

From a numerical perspective, the primary point is the sensitivity of such solutions to initial conditions. Thus, different numerical algorithms, computer precision, and integration time steps can all have an impact on the resulting numerical solution.

## 28.3 SIMULATING TRANSIENT CURRENT FOR AN ELECTRIC CIRCUIT (ELECTRICAL ENGINEERING)

Background. Electric circuits where the current is time-variable rather than constant are common. A transient current is established in the right-hand loop of the circuit shown in Fig. 28.11 when the switch is suddenly closed.

Equations that describe the transient behavior of the circuit in Fig. 28.11 are based on Kirchhoff's law, which states that the algebraic sum of the voltage drops around a closed loop is zero (recall Sec. 8.3). Thus,

$$L\frac{di}{dt} + Ri + \frac{q}{C} - E(t) = 0 \tag{28.9}$$

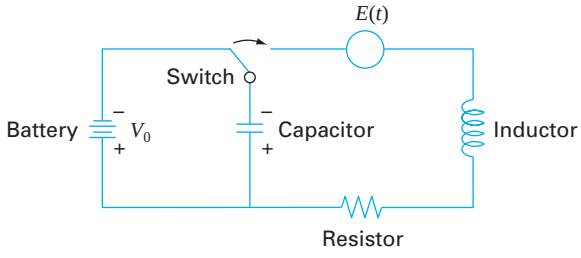
where $L(di/dt)$ = voltage drop across the inductor, $L$ = inductance (H), $R$ = resistance ($\Omega$), $q$ = charge on the capacitor (C), $C$ = capacitance (F), $E(t)$ = time-variable voltage source (V), and

$$i = \frac{dq}{dt} \tag{28.10}$$

Equations (28.9) and (28.10) are a pair of first-order linear differential equations that can be solved analytically. For example, if $E(t) = E_0 \sin \omega t$ and $R = 0$,

$$q(t) = \frac{-E_0}{L(p^2 - \omega^2)}\frac{\omega}{p}\sin pt + \frac{E_0}{L(p^2 - \omega^2)}\sin \omega t \tag{28.11}$$

**FIGURE 28.11**
An electric circuit where the current varies with time.



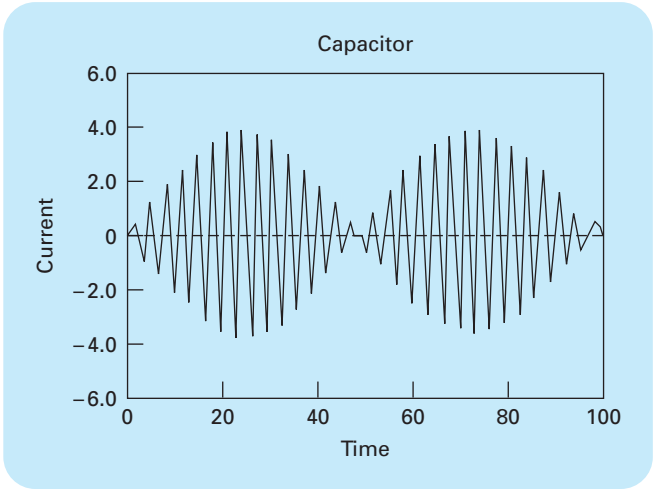



**FIGURE 28.12**
Computer screen showing the plot of the function represented by Eq. (28.11).

where $p = 1/\sqrt{LC}$. The values of $q$ and $dq/dt$ are zero for $t = 0$. Use a numerical approach to solve Eqs. (28.9) and (28.10) and compare the results with Eq. (28.11).

Solution. This problem involves a rather long integration range and demands the use of a highly accurate scheme to solve the differential equation if good results are expected. Let us assume that $L = 1$ H, $E_0 = 1$ V, $C = 0.25$ C, and $\omega^2 = 3.5$ s$^2$. This gives $p = 2$, and Eq. (28.11) becomes

$$q(t) = -1.8708 \sin(2t) + 2 \sin(1.8708t)$$

for the analytical solution. This function is plotted in Fig. 28.12. The rapidly changing nature of the function places a severe requirement on any numerical procedure to find $q(t)$. Furthermore, because the function exhibits a slowly varying periodic nature as well as a rapidly varying component, long integration ranges are necessary to portray the solution. Thus, we expect that a high-order method is preferred for this problem.

However, we can try both Euler and fourth-order RK methods and compare the results. Using a step size of 0.1 s gives a value for $q$ at $t = 10$ s of $-6.638$ with Euler's method and a value of $-1.9897$ with the fourth-order RK method. These results compare to an exact solution of $-1.996$ C.

Figure 28.13 shows the results of Euler integration every 1.0 s compared to the exact solution. Note that only every tenth output point is plotted. It is seen that the global error increases as $t$ increases. This divergent behavior intensifies as $t$ approaches infinity.

In addition to directly simulating a network's transient response, numerical methods can also be used to determine its eigenvalues. For example, Fig. 28.14 shows an $LC$ network for which Kirchhoff's voltage law can be employed to develop the following system

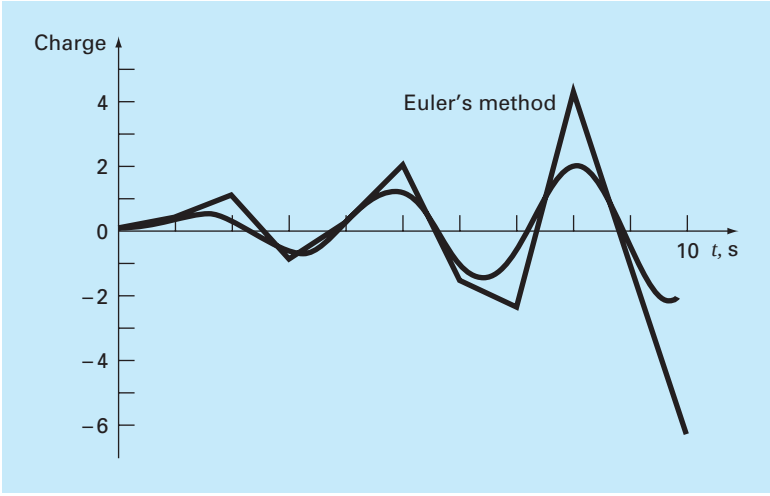



**FIGURE 28.13**
Results of Euler integration versus exact solution. Note that only every tenth output point is plotted.

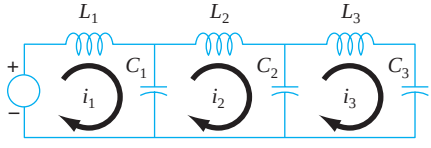



**FIGURE 28.14**
An *LC* network.

of ODEs:

$$-L_1\frac{di_1}{dt} - \frac{1}{C_1}\int_{-\infty}^{t}(i_1 - i_2)\,dt = 0$$

$$-L_2\frac{di_2}{dt} - \frac{1}{C_2}\int_{-\infty}^{t}(i_2 - i_3)\,dt + \frac{1}{C_1}\int_{-\infty}^{t}(i_1 - i_2)\,dt = 0$$

$$-L_3\frac{di_3}{dt} - \frac{1}{C_3}\int_{-\infty}^{t} i_3\,dt + \frac{1}{C_2}\int_{-\infty}^{t}(i_2 - i_3)\,dt = 0$$

Notice that we have represented the voltage drop across the capacitor as

$$V_C = \frac{1}{C}\int_{-\infty}^{t} i\,dt$$

This is an alternative and equivalent expression to the relationship used in Eq. (28.9) and introduced in Sec. 8.3.

The system of ODEs can be differentiated and rearranged to yield

$$L_1\frac{d^2i_1}{dt^2} + \frac{1}{C_1}(i_1 - i_2) = 0$$
$$L_2\frac{d^2i_2}{dt^2} + \frac{1}{C_2}(i_2 - i_3) - \frac{1}{C_1}(i_1 - i_2) = 0$$
$$L_3\frac{d^2i_3}{dt^2} + \frac{1}{C_3}i_3 - \frac{1}{C_2}(i_2 - i_3) = 0$$

Comparison of this system with the one in Eq. (27.5) indicates an analogy between a spring-mass system and an *LC* circuit. As was done with Eq. (27.5), the solution can be assumed to be of the form

$$i_j = A_j \sin(\omega t)$$

This solution along with its second derivative can be substituted into the simultaneous ODEs. After simplification, the result is

$$\begin{aligned}
\left(\frac{1}{C_1} - L_1\omega^2\right)A_1 \qquad & -\frac{1}{C_2}A_2 & & = 0\\
-\frac{1}{C_1}A_1 \qquad & +\left(\frac{1}{C_1} + \frac{1}{C_2} - L_2\omega^2\right)A_2 \qquad & -\frac{1}{C_2}A_3 \qquad & = 0\\
& -\frac{1}{C_2}A_2 \qquad & +\left(\frac{1}{C_2} + \frac{1}{C_3} - L_3\omega^2\right)A_3 & = 0
\end{aligned}$$

Thus, we have formulated an eigenvalue problem. Further simplification results for the special case where the *C*'s and *L*'s are constant. For this situation, the system can be expressed in matrix form as

$$\begin{bmatrix} 1-\lambda & -1 & 0 \\ -1 & 2-\lambda & -1 \\ 0 & -1 & 2-\lambda \end{bmatrix}\begin{Bmatrix} A_1 \\ A_2 \\ A_3 \end{Bmatrix} = \{0\} \tag{28.12}$$

where

$$\lambda = LC\omega^2 \tag{28.13}$$

Numerical methods can be employed to determine values for the eigenvalues and eigenvectors. MATLAB is particularly convenient in this regard. The following MATLAB session has been developed to do this:

```
>>a=[1 -1 0; -1 2 -1; 0 -1 2]

a =

     1    -1     0
    -1     2    -1
     0    -1     2
```

```
>>[v,d]=eig(a)

v =

    0.7370    0.5910    0.3280
    0.5910   -0.3280   -0.7370
    0.3280   -0.7370    0.5910

d =

    0.1981         0         0
         0    1.5550         0
         0         0    3.2470
```

The matrix v consists of the system's three eigenvectors (arranged as columns), and d is a matrix with the corresponding eigenvalues on the diagonal. Thus, the package computes that the eigenvalues are $\lambda = 0.1981$, 1.555, and 3.247. These values in turn can be substituted into Eq. (28.13) to solve for the natural circular frequencies of the system

$$\omega = \begin{Bmatrix} \dfrac{0.4450}{\sqrt{LC}} \\ \dfrac{1.2470}{\sqrt{LC}} \\ \dfrac{1.8019}{\sqrt{LC}} \end{Bmatrix}$$

Aside from providing the natural frequencies, the eigenvalues can be substituted into Eq. (28.12) to gain further insight into the circuit's physical behavior. For example, substituting $\lambda = 0.1981$ yields

$$\begin{bmatrix} 0.8019 & -1 & 0 \\ -1 & 1.8019 & -1 \\ 0 & -1 & 1.8019 \end{bmatrix} \begin{Bmatrix} i_1 \\ i_2 \\ i_3 \end{Bmatrix} = \{0\}$$

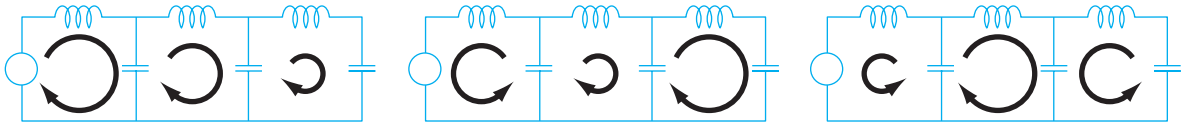
Although this system does not have a unique solution, it will be satisfied if the currents are in fixed ratios, as in

$$0.8019i_1 = i_2 = 1.8019i_3 \tag{28.14}$$

Thus, as depicted in Fig. 28.15*a*, they oscillate in the same direction with different magnitudes. Observe that if we assume that $i_1 = 0.737$, we can use Eq. (28.14) to compute the other currents with the result

$$\{i\} = \begin{Bmatrix} 0.737 \\ 0.591 \\ 0.328 \end{Bmatrix}$$

which is the first column of the v matrix calculated with MATLAB.



(a) $\omega = \frac{0.4451}{\sqrt{LC}}$ (b) $\omega = \frac{1.2470}{\sqrt{LC}}$ (c) $\omega = \frac{1.8019}{\sqrt{LC}}$

**FIGURE 28.15**
A visual representation of the natural modes of oscillation of the *LC* network of Fig. 28.14. Note that the diameters of the circular arrows are proportional to the magnitudes of the currents for each loop.

In a similar fashion, the second eigenvalue of $\lambda = 1.555$ can be substituted and the result evaluated to yield

$$-1.8018i_1 = i_2 = 2.247i_3$$

As depicted in Fig. 28.15*b*, the first loop oscillates in the opposite direction from the second and third. Finally, the third mode can be determined as

$$-0.445i_1 = i_2 = -0.8718i_3$$

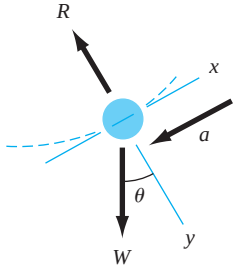
Consequently, as in Fig. 28.15*c*, the first and third loops oscillate in the opposite direction from the second.

## 28.4 THE SWINGING PENDULUM (MECHANICAL/AEROSPACE ENGINEERING)

Background. Mechanical engineers (as well as all other engineers) are frequently faced with problems concerning the periodic motion of free bodies. The engineering approach to such problems ultimately requires that the position and velocity of the body be known as a function of time. These functions of time invariably are the solution of ordinary differential equations. The differential equations are usually based on Newton's laws of motion.

As an example, consider the simple pendulum shown previously in Fig. PT7.1. The particle of weight $W$ is suspended on a weightless rod of length $l$. The only forces acting on the particle are its weight and the tension $R$ in the rod. The position of the particle at any time is completely specified in terms of the angle $\theta$ and $l$.

**FIGURE 28.16**
A free-body diagram of the swinging pendulum showing the forces on the particle and the acceleration.




The free-body diagram in Fig. 28.16 shows the forces on the particle and the acceleration. It is convenient to apply Newton's laws of motion in the $x$ direction tangent to the path of the particle:

$$\Sigma F = -W \sin\theta = \frac{W}{g}a$$

where $g$ = the gravitational constant (32.2 ft/s$^2$) and $a$ = the acceleration in the $x$ direction. The angular acceleration of the particle ($\alpha$) becomes

$$\alpha = \frac{a}{l}$$

Therefore, in polar coordinates ($\alpha = d^2\theta/dt^2$),

$$-W \sin\theta = \frac{Wl}{g}\alpha = \frac{Wl}{g}\frac{d^2\theta}{dt^2}$$

or

$$\frac{d^2\theta}{dt^2} + \frac{g}{l}\sin\theta = 0 \tag{28.15}$$

This apparently simple equation is a second-order nonlinear differential equation. In general, such equations are difficult or impossible to solve analytically. You have two choices regarding further progress. First, the differential equation might be reduced to a form that can be solved analytically (recall Sec. PT7.1.1), or second, a numerical approximation technique can be used to solve the differential equation directly. We will examine both of these alternatives in this example.

Solution. Proceeding with the first approach, we note that the series expansion for $\sin\theta$ is given by

$$\sin\theta = \theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \frac{\theta^7}{7!} + \cdots \tag{28.16}$$

For small angular displacements, $\sin\theta$ is approximately equal to $\theta$ when expressed in radians. Therefore, for small displacements, Eq. (28.15) becomes

$$\frac{d^2\theta}{dt^2} + \frac{g}{l}\theta = 0 \tag{28.17}$$

which is a second-order linear differential equation. This approximation is very important because Eq. (28.17) is easy to solve analytically. The solution, based on the theory of differential equations, is given by

$$\theta(t) = \theta_0 \cos\sqrt{\frac{g}{l}}t \tag{28.18}$$

where $\theta_0 =$ the displacement at $t = 0$ and where it is assumed that the velocity ($v = d\theta/dt$) is zero at $t = 0$. The time required for the pendulum to complete one cycle of oscillation is called the period and is given by

$$T = 2\pi\sqrt{\frac{l}{g}} \tag{28.19}$$

Figure 28.17 shows a plot of the displacement $\theta$ and velocity $d\theta/dt$ as a function of time, as calculated from Eq. (28.18) with $\theta_0 = \pi/4$ and $l = 2$ ft. The period, as calculated from Eq. (28.19), is 1.5659 s.

The above calculations essentially are a complete solution of the motion of the pendulum. However, you must also consider the accuracy of the results because of the assumptions

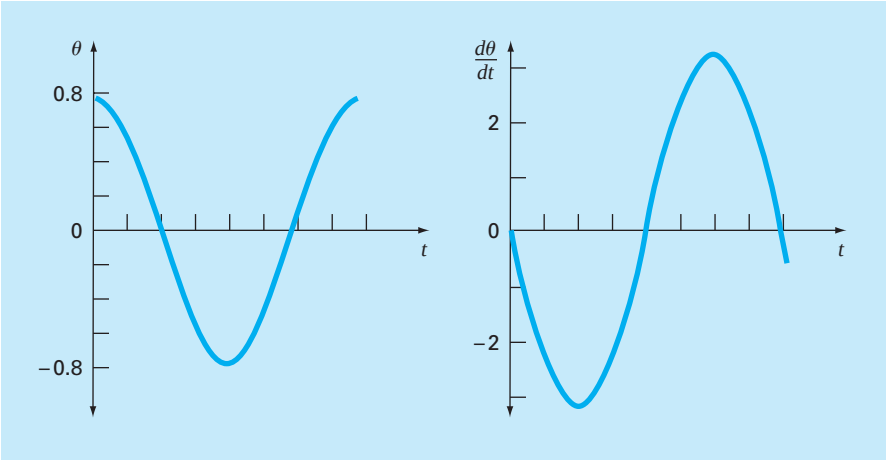



**FIGURE 28.17**
Plot of displacement $\theta$ and velocity $d\theta/dt$ as a function of time $t$, as calculated from Eq. (28.18). $\theta_0$ is $\pi/4$ and the length is 2 ft.

inherent in Eq. (28.17). To evaluate the accuracy, it is necessary to obtain a numerical solution for Eq. (28.15), which is a more complete physical representation of the motion. Any of the methods discussed in Chaps. 25 and 26 could be used for this purpose—for example, the Euler and fourth-order RK methods. Equation (28.15) must be transformed into two first-order equations to be compatible with the above methods. This is accomplished as follows. The velocity $v$ is defined by

$$\frac{d\theta}{dt} = v \tag{28.20}$$

and, therefore, Eq. (28.15) can be expressed as

$$\frac{dv}{dt} = -\frac{g}{l}\sin\theta \tag{28.21}$$

Equations (28.20) and (28.21) are a coupled system of two ordinary differential equations. The numerical solutions by the Euler method and the fourth-order RK method give the results shown in Table 28.1, which compares the analytic solution for the linear equation of motion [Eq. (28.18)] in column (*a*) with the numerical solutions in columns (*b*), (*c*), and (*d*).

The Euler and fourth-order RK methods yield different results and both disagree with the analytic solution, although the fourth-order RK method for the nonlinear case is closer to the analytic solution than is the Euler method. To properly evaluate the difference between the linear and nonlinear models, it is important to determine the accuracy of the numerical results. This is accomplished in three ways. First, the Euler numerical solution is easily recognized as inadequate because it overshoots the initial condition at $t = 0.8$ s. This clearly violates conservation of energy. Second, column (*c*) and (*d*) in Table 28.1 show the solution of the fourth-order RK method for step sizes of 0.05 and 0.01. Because these vary

**TABLE 28.1** Comparison of a linear analytical solution of the swinging pendulum problem with three nonlinear numerical solutions.

| | | Nonlinear Numerical Solutions | | |
|---|---|---|---|---|
| **Time, s** | **Linear Analytical Solution (*a*)** | **Euler ($h = 0.05$) (*b*)** | **4th-Order RK ($h = 0.05$) (*c*)** | **4th-Order RK ($h = 0.01$) (*d*)** |
| 0.0 | 0.785398 | 0.785398 | 0.785398 | 0.785398 |
| 0.2 | 0.545784 | 0.615453 | 0.566582 | 0.566579 |
| 0.4 | −0.026852 | 0.050228 | 0.021895 | 0.021882 |
| 0.6 | −0.583104 | −0.639652 | −0.535802 | −0.535820 |
| 0.8 | −0.783562 | −1.050679 | −0.784236 | −0.784242 |
| 1.0 | −0.505912 | −0.940622 | −0.595598 | −0.595583 |
| 1.2 | 0.080431 | −0.299819 | −0.065611 | −0.065575 |
| 1.4 | 0.617698 | 0.621700 | 0.503352 | 0.503392 |
| 1.6 | 0.778062 | 1.316795 | 0.780762 | 0.780777 |

**TABLE 28.2** Comparison of the period of an oscillating body calculated from linear and nonlinear models.

| | Period, *s* | |
|---|---|---|
| **Initial Displacement, $\theta_0$** | **Linear Model ($T = 2\pi\sqrt{l/g}$)** | **Nonlinear Model [Numerical Solution of Eq. (28.15)]** |
| $\pi/16$ | 1.5659 | 1.57 |
| $\pi/4$ | 1.5659 | 1.63 |
| $\pi/2$ | 1.5659 | 1.85 |

in the fourth decimal place, it is reasonable to assume that the solution with a step size of 0.01 is also accurate with this degree of certainty. Third, for the 0.01-s step-size case, $\theta$ obtains a local maximum value of 0.785385 at $t = 1.63$ s (not shown in Table 28.1). This indicates that the pendulum returns to its original position with four-place accuracy with a period of 1.63 s. These considerations allow you to safely assume that the difference between columns (*a*) and (*d*) in Table 28.1 truly represents the difference between the linear and nonlinear model.

Another way to characterize the difference between the linear and the nonlinear model is on the basis of period. Table 28.2 shows the period of oscillation as calculated by the linear model and nonlinear model for three different initial displacements. It is seen that the calculated periods agree closely when $\theta$ is small because $\theta$ is a good approximation for $\sin\theta$ in Eq. (28.16). This approximation deteriorates when $\theta$ becomes large.

These analyses are typical of cases you will routinely encounter as an engineer. The utility of the numerical techniques becomes particularly significant in nonlinear problems, and in many cases real-world problems are nonlinear.

## PROBLEMS

**Chemical/Bio Engineering**

**28.1** Perform the first computation in Sec. 28.1, but for the case where $h = 10$. Use the Heun (without iteration) and the fourth-order RK method to obtain solutions.

**28.2** Perform the second computation in Sec. 28.1, but for the system described in Prob. 12.4.

**28.3** A mass balance for a chemical in a completely mixed reactor can be written as

$$V\frac{dc}{dt} = F - Qc - kVc^2$$

where $V$ = volume (12 m$^3$), $c$ = concentration (g/m$^3$), $F$ = feed rate (175 g/min), $Q$ = flow rate (1 m$^3$/min), and $k$ = a second-order reaction rate (0.15 m$^3$/g/min). If $c(0) = 0$, solve the ODE until the concentration reaches a stable level. Use the midpoint method ($h = 0.5$) and plot your results.

Challenge question: If one ignores the fact that concentrations must be positive, find a range of initial conditions such that you obtain a very different trajectory than was obtained with $c(0) = 0$. Relate your results to the steady-state solutions.

**28.4** If $c_{in} = c_b(1 - e^{-0.12t})$, calculate the outflow concentration of a conservative substance (no reaction) for a single, completely mixed reactor as a function of time. Use Heun's method (without iteration) to perform the computation. Employ values of $c_b = 40$ mg/m$^3$, $Q = 6$ m$^3$/min, $V = 100$ m$^3$, and $c_0 = 20$ mg/m$^3$. Perform the computation from $t = 0$ to 100 min using $h = 2$. Plot your results along with the inflow concentration versus time.

**28.5** Seawater with a concentration of 8000 g/m$^3$ is pumped into a well-mixed tank at a rate of 0.6 m$^3$/hr. Because of faulty design work, water is evaporating from the tank at a rate of 0.025 m$^3$/hr. The salt solution leaves the tank at a rate of 0.6 m$^3$/hr.

**(a)** If the tank originally contains 1 m$^3$ of the inlet solution, how long after the outlet pump is turned on will the tank run dry?

**(b)** Use numerical methods to determine the salt concentration in the tank as a function of time.

**28.6** A spherical ice cube (an "ice sphere") that is 6 cm in diameter is removed from a 0°C freezer and placed on a mesh screen at room temperature $T_a = 20$°C. What will be the diameter of the ice cube as a function of time out of the freezer (assuming that all the water that has melted immediately drips through the screen)? The heat transfer coefficient $h$ for a sphere in a still room is about 3 W/(m$^2 \cdot$ K). The heat flux from the ice sphere to the air is given by

$$\text{Flux} = \frac{q}{A} = h(T_a - T)$$

where $q$ = heat and $A$ = surface area of the sphere. Use a numerical method to make your calculation. Note that the latent heat of fusion is 333 kJ/kg and the density of ice is approximately 0.917 kg/m$^3$.

**28.7** The following equations define the concentrations of three reactants:

$$\frac{dc_a}{dt} = -10c_ac_c + c_b$$
$$\frac{dc_b}{dt} = 10c_ac_c - c_b$$
$$\frac{dc_c}{dt} = -10c_ac_c + c_b - 2c_c$$

If the initial conditions are $c_a = 50$, $c_b = 0$, and $c_c = 40$, find the concentrations for the times from 0 to 3 s.

**28.8** Compound $A$ diffuses through a 4-cm-long tube and reacts as it diffuses. The equation governing diffusion with reaction is

$$D\frac{d^2A}{dx^2} - kA = 0$$

At one end of the tube, there is a large source of $A$ at a concentration of 0.1 M. At the other end of the tube there is an adsorbent material that quickly absorbs any $A$, making the concentration 0 M. If $D = 1.5 \times 10^{-6}$ cm$^2$/s and $k = 5 \times 10^{-6}$ s$^{-1}$, what is the concentration of $A$ as a function of distance in the tube?

**28.9** In the investigation of a homicide or accidental death, it is often important to estimate the time of death. From the experimental observations, it is known that the surface temperature of an object changes at a rate proportional to the difference between the temperature of the object and that of the surrounding environment or ambient temperature. This is known as Newton's law of cooling. Thus, if $T(t)$ is the temperature of the object at time $t$, and $T_a$ is the constant ambient temperature:

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

where $K > 0$ is a constant of proportionality. Suppose that at time $t = 0$ a corpse is discovered and its temperature is measured to be $T_o$. We assume that at the time of death, the body temperature, $T_d$, was at the normal value of 37°C. Suppose that the temperature of the corpse when it was discovered was 29.5°C, and that two hours later, it is 23.5°C. The ambient temperature is 20°C.

**(a)** Determine $K$ and the time of death.

**(b)** Solve the ODE numerically and plot the results.

**28.10** The reaction $A \rightarrow B$ takes place in two reactors in series. The reactors are well mixed but are not at steady state. The

unsteady-state mass balance for each stirred tank reactor is shown below:

$$\frac{dCA_1}{dt} = \frac{1}{\tau}(CA_0 - CA_1) - kCA_1$$

$$\frac{dCB_1}{dt} = -\frac{1}{\tau}CB_1 + kCA_1$$

$$\frac{dCA_2}{dt} = \frac{1}{\tau}(CA_1 - CA_2) - kCA_2$$

$$\frac{dCB_2}{dt} = \frac{1}{\tau}(CB_1 - CB_2) + kCA_2$$

where $CA_0$ = concentration of $A$ at the inlet of the first reactor, $CA_1$ = concentration of $A$ at the outlet of the first reactor (and inlet of the second), $CA_2$ = concentration of $A$ at the outlet of the second reactor, $CB_1$ = concentration of $B$ at the outlet of the first reactor (and inlet of the second), $CB_2$ = concentration of $B$ in the second reactor, $\tau$ = residence time for each reactor, and $k$ = the rate constant for reaction of $A$ to produce $B$. If $CA_0$ is equal to 20, find the concentrations of $A$ and $B$ in both reactors during their first 10 minutes of operation. Use $k = 0.12$/min and $\tau = 5$ min and assume that the initial conditions of all the dependent variables are zero.

**28.11** A nonisothermal batch reactor can be described by the following equations:

$$\frac{dC}{dt} = -e^{(-10/(T+273))}C$$

$$\frac{dT}{dt} = 1000e^{(-10/(T+273))}C - 10(T - 20)$$

where $C$ is the concentration of the reactant and $T$ is the temperature of the reactor. Initially the reactor is at 15°C and has a concentration of reactant $C$ of 1.0 gmol/L. Find the concentration and temperature of the reactor as a function of time.

**28.12** The following system is a classic example of stiff ODEs that can occur in the solution of chemical reaction kinetics:

$$\frac{dc_1}{dt} = -0.013c_1 - 1000c_1c_3$$

$$\frac{dc_2}{dt} = -2500c_2c_3$$

$$\frac{dc_3}{dt} = -0.013c_1 - 1000c_1c_3 - 2500c_2c_3$$

Solve these equations from $t = 0$ to 50 with initial conditions $c_1(0) = c_2(0) = 1$ and $c_3(0) = 0$. If you have access to MATLAB software, use both standard (for example, `ode45`) and stiff (for example, `ode23s`) functions to obtain your solutions.

**28.13** A *biofilm* with a thickness, $L_f$ [cm], grows on the surface of a solid (Fig. P28.13). After traversing a diffusion layer of thickness, $L$ [cm], a chemical compound, $A$, diffuses into the biofilm where it is subject to an irreversible first-order reaction that converts it to a product, $B$.

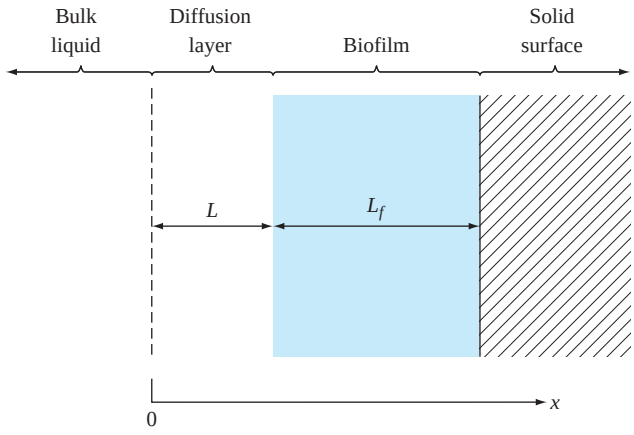



**Figure P28.13**
A biofilm growing on a solid surface.

Steady-state mass balances can be used to derive the following ordinary differential equations for compound $A$:

$$D\frac{d^2c_a}{dx^2} = 0 \qquad 0 \le x < L$$

$$D_f\frac{d^2c_a}{dx^2} - kc_a = 0 \qquad L \le x < L + L_f$$

where $D$ = the diffusion coefficient in the diffusion layer = 0.8 cm$^2$/d, $D_f$ = the diffusion coefficient in the biofilm = 0.64 cm$^2$/d, and $k$ = the first-order rate for the conversion of $A$ to $B$ = 0.1/d. The following boundary conditions hold:

$$c_a = c_{a0} \qquad \text{at } x = 0$$

$$\frac{dc_a}{dx} = 0 \qquad \text{at } x = L + L_f$$

where $c_{a\,0}$ = the concentration of $A$ in the bulk liquid = 100 mol/L. Use the finite-difference method to compute the steady-state distribution of $A$ from $x = 0$ to $L + L_f$, where $L = 0.008$ cm and $L_f = 0.004$ cm. Employ centered finite differences with $\Delta x = 0.001$ cm.

**28.14** The following differential equation describes the steady-state concentration of a substance that reacts with first-order kinetics in an axially-dispersed plug-flow reactor (Fig. P28.14),

$$D\frac{d^2c}{dx^2} - U\frac{dc}{dx} - kc = 0$$

where $D$ = the dispersion coefficient [m$^2$/hr], $c$ = concentration[mol/L], $x$ = distance [m], $U$ = the velocity [m/hr],

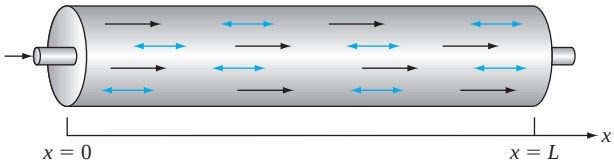



**Figure P28.14**
An axially-dispersed plug-flow reactor.

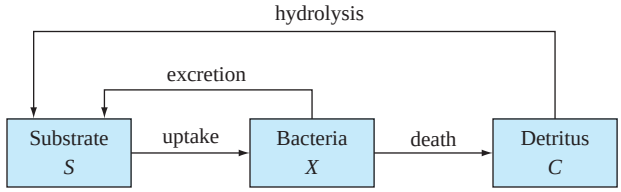



**Figure P28.16**

and $k =$ the reaction rate [/hr]. The boundary conditions can be formulated as

$$Uc_{\text{in}} = Uc(x=0) - D\frac{dc}{dx}(x=0)$$

$$\frac{dc}{dx}(x=L) = 0$$

where $c_{\text{in}} =$ the concentration in the inflow [mol/L], and $L =$ the length of the reactor [m]. These are called *Danckwerts boundary conditions*. Use the finite-difference approach to solve for concentration as a function of distance given the following parameters: $D = 5000$ m$^2$/hr, $U = 100$ m/hr, $k = 2$/hr, $L = 100$ m and $c_{\text{in}} = 100$ mol/L. Employ centered finite-difference approximations with $\Delta x = 10$ m to obtain your solutions. Compare your numerical results with the analytical solution,

$$c = \frac{Uc_{in}}{(U - D\lambda_1)\lambda_2 e^{\lambda_2 L} - (U - D\lambda_2)\lambda_1 e^{\lambda_1 L}} \times \left(\lambda_2 e^{\lambda_2 L} e^{\lambda_1 x} - \lambda_1 e^{\lambda_1 L} e^{\lambda_2 x}\right)$$

where

$$\frac{\lambda_1}{\lambda_2} = \frac{U}{2D}\left(1 \pm \sqrt{1 + \frac{4kD}{U^2}}\right)$$

**28.15** A series of first-order, liquid-phase reactions create a desirable product ($B$) and an undesirable byproduct ($C$)

$$A \xrightarrow{k_1} B \xrightarrow{k_2} C$$

If the reactions take place in an axially-dispersed plug-flow reactor (Fig. P28.14), steady-state mass balances can be used to develop the following second-order ODEs,

$$D\frac{d^2c_a}{dx^2} - U\frac{dc_a}{dx} - k_1 c_a = 0$$

$$D\frac{d^2c_b}{dx^2} - U\frac{dc_b}{dx} + k_1 c_a - k_2 c_b = 0$$

$$D\frac{d^2c_c}{dx^2} - U\frac{dc_c}{dx} + k_2 c_b = 0$$

Use the finite-difference approach to solve for the concentration of each reactant as a function of distance given: $D = 0.1$ m$^2$/min, $U =$ 1 m/min, $k_1 = 3$/min, $k_2 = 1$/min, $L = 0.5$ m, $c_{a,\text{in}} = 10$ mol/L. Employ centered finite-difference approximations with $\Delta x = 0.05$ m to obtain your solutions and assume Danckwerts boundary conditions as described in Prob. 28.14. Also, compute the sum of the reactants as a function of distance. Do your results make sense?

**28.16** Bacteria growing in a batch reactor utilize a soluble food source (substrate) as depicted in Fig. P28.16. The uptake of the substrate is represented by a logistic model with Michaelis-Menten limitation. Death of the bacteria produces detritus which is subsequently converted to the substrate by hydrolysis. In addition, the bacteria also excrete some substrate directly. Death, hydrolysis and excretion are all simulated as first-order reactions.

Mass balances can be written as

$$\frac{dX}{dt} = \mu_{\max}\left(1 - \frac{X}{K}\right)\left(\frac{S}{K_s + S}\right)X - k_d X - k_e X$$

$$\frac{dC}{dt} = k_d X - k_h C$$

$$\frac{dS}{dt} = k_e X + k_h C - \mu_{\max}\left(1 - \frac{X}{K}\right)\left(\frac{S}{K_s + S}\right)X$$

where $X$, $C$, and $S =$ the concentrations [mg/L] of bacteria, detritus, and substrate, respectively; $\mu_{\max} =$ maximum growth rate [/d], $K =$ the logistic carrying capacity [mg/L]; $K_s =$ the Michaelis-Menten half-saturation constant [mg/L], $k_d =$ death rate [/d]; $k_e =$ excretion rate [/d]; and $k_h =$ hydrolysis rate [/d]. Simulate the concentrations from $t = 0$ to 100 d, given the initial conditions $X(0) = 1$ mg/L, $S(0) =$ 100 mg/L and $C(0) = 0$ mg/L. Employ the following parameters in your calculation: $\mu_{\max} = 10$/d, $K = 10$ mg/L, $K_s = 10$ mg/L, $k_d =$ 0.1/d, $k_e = 0.1$/d, and $k_h = 0.1$/d.

**Civil/Environmental Engineering**

**28.17** Perform the same computation for the Lotka-Volterra system in Sec. 28.2, but use **(a)** Euler's method, **(b)** Heun's method (without iterating the corrector), **(c)** the fourth-order RK method, and **(d)** the MATLAB `ode45` function. In all cases use single-precision variables, a step size of 0.1, and simulate from $t = 0$ to 20. Develop phase-plane plots for all cases.

**28.18** Perform the same computation for the Lorenz equations in Sec. 28.2, but use **(a)** Euler's method, **(b)** Heun's method (without iterating the corrector), **(c)** the fourth-order RK method, and **(d)** the MATLAB `ode45` function. In all cases use single-precision variables and a step size of 0.1 and simulate from $t = 0$ to 20. Develop phase-plane plots for all cases.

**28.19** The following equation can be used to model the deflection of a sailboat mast subject to a wind force:

$$\frac{d^2y}{dz^2} = \frac{f}{2EI}(L - z)^2$$

where $f$ = wind force, $E$ = modulus of elasticity, $L$ = mast length, and $I$ = moment of inertia. Calculate the deflection if $y = 0$ and $dy/dz = 0$ at $z = 0$. Use parameter values of $f = 60$, $L = 30$, $E = 1.25 \times 10^8$, and $I = 0.05$ for your computation.

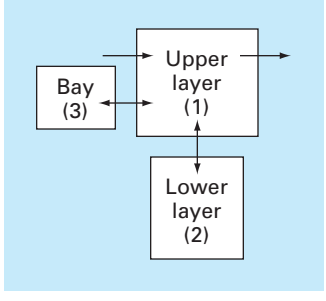
**28.20** Perform the same computation as in Prob. 28.19, but rather than using a constant wind force, employ a force that varies with height according to (recall Sec. 24.2)

$$f(z) = \frac{200z}{5 + z}e^{-2z/30}$$

**28.21** An environmental engineer is interested in estimating the mixing that occurs between a stratified lake and an adjacent embayment (Fig. P28.21). A conservative tracer is instantaneously mixed with the bay water, and then the tracer concentration is monitored over the ensuing period in all three segments. The values are

| $t$ | 0 | 2 | 4 | 6 | 8 | 12 | 16 | 20 |
|---|---|---|---|---|---|---|---|---|
| $c_1$ | 0 | 15 | 11 | 7 | 6 | 3 | 2 | 1 |
| $c_2$ | 0 | 3 | 5 | 7 | 7 | 6 | 4 | 2 |
| $c_3$ | 100 | 48 | 26 | 16 | 10 | 4 | 3 | 2 |

**Figure P28.21**




Using mass balances, the system can be modeled as the following simultaneous ODEs:

$$V_1\frac{dc_1}{dt} = -Qc_1 + E_{12}(c_2 - c_1) + E_{13}(c_3 - c_1)$$

$$V_2\frac{dc_2}{dt} = E_{12}(c_1 - c_2)$$

$$V_3\frac{dc_3}{dt} = E_{13}(c_1 - c_3)$$

where $V_i$ = volume of segment $i$, $Q$ = flow, and $E_{ij}$ = diffusive mixing rate between segments $i$ and $j$. Use the data and the differential equations to estimate the $E$'s if $V_1 = 1 \times 10^7$, $V_2 = 8 \times 10^6$, $V_3 = 5 \times 10^6$, and $Q = 4 \times 10^6$. Employ Euler's method with a step size of 0.1 for your analysis.

**28.22** Population-growth dynamics are important in a variety of planning studies for areas such as transportation and water-resource engineering. One of the simplest models of such growth incorporates the assumption that the rate of change of the population $p$ is proportional to the existing population at any time $t$:

$$\frac{dp}{dt} = Gp \tag{P28.22}$$

where $G$ = a growth rate (per year). This model makes intuitive sense because the greater the population, the greater the number of potential parents. At time $t = 0$, an island has a population of 6000 people. If $G = 0.075$ per year, employ Heun's method (without iteration) to predict the population at $t = 20$ years, using a step size of 0.5 year. Plot $p$ versus $t$ on standard and semilog graph paper. Determine the slope of the line on the semilog plot. Discuss your results.

**28.23** Although the model in Prob. 28.22 works adequately when population growth is unlimited, it breaks down when factors such as food shortages, pollution, and lack of space inhibit growth. In such cases, the growth rate itself can be thought of as being inversely proportional to population. One model of this relationship is

$$G = G'(p_{\max} - p) \tag{P28.23}$$

where $G'$ = a population-dependent growth rate (per people-year) and $p_{\max}$ = the maximum sustainable population. Thus, when population is small ($p \ll p_{\max}$), the growth rate will be at a high constant rate of $G'p_{\max}$. For such cases, growth is unlimited and Eq. (P28.23) is essentially identical to Eq. (P28.22). However, as population grows (that is, $p$ approaches $p_{\max}$), $G$ decreases until at $p = p_{\max}$ it is zero. Thus, the model predicts that, when the population reaches the maximum sustainable level, growth is nonexistent, and the system is at a steady state. Substituting Eq. (P28.23) into Eq. (P28.22) yields

$$\frac{dp}{dt} = G'(p_{\max} - p)p$$

For the same island studied in Prob. 28.22, employ Heun's method (without iteration) to predict the population at $t = 20$ years, using a step size of 0.5 year. Employ values of $G' = 10^{-5}$ per people-year and $p_{max} = 20{,}000$ people. At time $t = 0$, the island has a population of 6000 people. Plot $p$ versus $t$ and interpret the shape of the curve.

**28.24** Isle Royale National Park is a 210-square-mile archipelago composed of a single large island and many small islands in Lake Superior. Moose arrived around 1900 and by 1930, their population approached 3000, ravaging vegetation. In 1949, wolves crossed an ice bridge from Ontario. Since the late 1950s, the numbers of the moose and wolves have been tracked. (Dash indicates no data.)

| Year | Moose | Wolves | Year | Moose | Wolves |
|---|---|---|---|---|---|
| 1960 | 700 | 22 | 1972 | 836 | 23 |
| 1961 | — | 22 | 1973 | 802 | 24 |
| 1962 | — | 23 | 1974 | 815 | 30 |
| 1963 | — | 20 | 1975 | 778 | 41 |
| 1964 | — | 25 | 1976 | 641 | 43 |
| 1965 | — | 28 | 1977 | 507 | 33 |
| 1966 | 881 | 24 | 1978 | 543 | 40 |
| 1967 | — | 22 | 1979 | 675 | 42 |
| 1968 | 1000 | 22 | 1980 | 577 | 50 |
| 1969 | 1150 | 17 | 1981 | 570 | 30 |
| 1970 | 966 | 18 | 1982 | 590 | 13 |
| 1971 | 674 | 20 | 1983 | 811 | 23 |

**(a)** Integrate the Lotka-Volterra equations from 1960 through 2020. Determine the coefficient values that yield an optimal fit. Compare your simulation with the data using a time-series approach, and comment on the results.

**(b)** Plot the simulation of **(a)**, but use a phase-plane approach.

**(c)** After 1993, suppose that the wildlife managers trap one wolf per year and transport it off the island. Predict how the populations of both the wolves and moose would evolve to the year 2020. Present your results as both time-series and phase-plane plots. For this case, as well as for **(d)**, use the following coefficients: $a = 0.3$, $b = 0.01111$, $c = 0.2106$, $d = 0.0002632$.

**(d)** Suppose that in 1993, some poachers snuck onto the island and killed 50% of the moose. Predict how the populations of both the wolves and moose would evolve to the year 2020. Present your results as both time-series and phase-plane plots.

**28.25** A cable is hanging from two supports at A and B (Fig. P28.25). The cable is loaded with a distributed load whose magnitude varies with $x$ as

$$w = w_o\left[1 + \sin\left(\frac{\pi x}{2l_A}\right)\right]$$

where $w_o = 1000$ lbs/ft. The slope of the cable $(dy/dx) = 0$ at $x = 0$, which is the lowest point for the cable. It is also the point where the tension in the cable is a minimum of $T_o$. The differential equation that governs the cable is

$$\frac{d^2y}{dx^2} = \frac{w_o}{T_o}\left[1 + \sin\left(\frac{\pi x}{2l_A}\right)\right]$$

Solve this equation using a numerical method and plot the shape of the cable ($y$ versus $x$). For the numerical solution, the value of $T_o$ is unknown, so the solution must use an iterative technique, similar to the shooting method, to converge on a correct value of $h_A$ for various values of $T_o$.

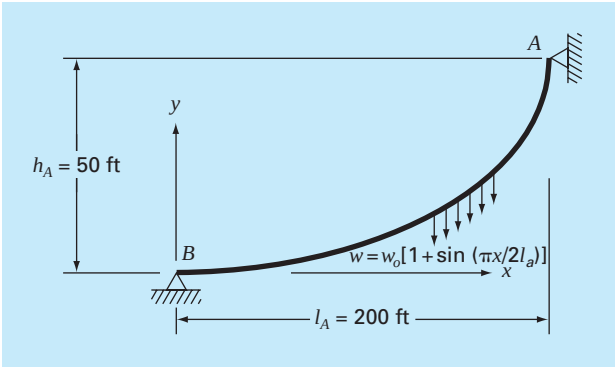



**Figure P28.25**

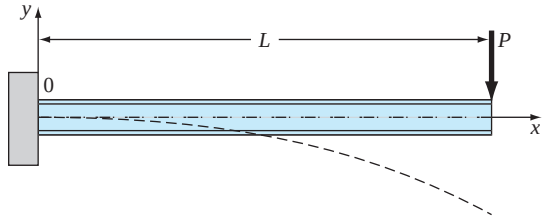



**Figure P28.26**

**28.26** The basic differential equation of the elastic curve for a cantilever beam (Fig. P28.26) is given as

$$EI\frac{d^2y}{dx^2} = -P(L - x)$$

where $E$ = the modulus of elasticity and $I$ = the moment of inertia. Solve for the deflection of the beam using a numerical method. The following parameter values apply: $E = 30{,}000$ ksi, $I = 800$ in$^4$,

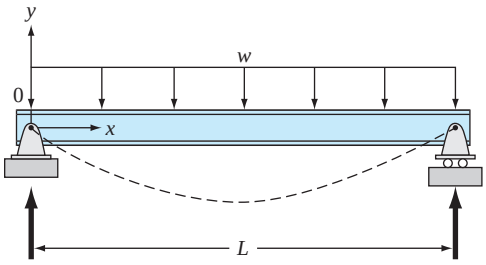



**Figure P28.27**

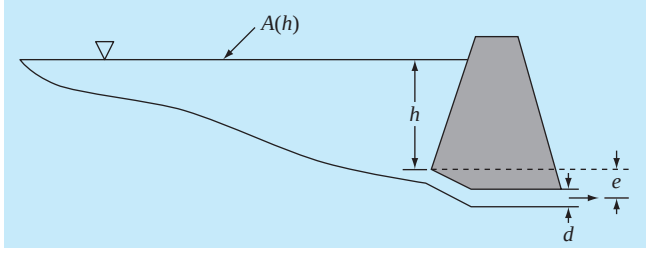



**Figure P28.28**

$P = 1$ kip, $L = 10$ ft. Compare your numerical results to the analytical solution,

$$y = -\frac{PLx^2}{2EI} + \frac{Px^3}{6EI}$$

**28.27** The basic differential equation of the elastic curve for a uniformly loaded beam (Fig. P28.27) is given as

$$EI\frac{d^2y}{dx^2} = \frac{wLx}{2} - \frac{wx^2}{2}$$

where $E$ = the modulus of elasticity and $I$ = the moment of inertia. Solve for the deflection of the beam using **(a)** the finite-difference approach ($\Delta x = 2$ ft) and **(b)** the shooting method. The following parameter values apply: $E = 30{,}000$ ksi, $I = 800$ in$^4$, $w = 1$ kip/ft, $L = 10$ ft. Compare your numerical results to the analytical solution,

$$y = \frac{wLx^3}{12EI} - \frac{wx^4}{24EI} - \frac{wL^3x}{24EI}$$

**28.28** A pond drains through a pipe as shown in Fig. P28.28. Under a number of simplifying assumptions, the following differential equation describes how depth changes with time:

$$\frac{dh}{dt} = -\frac{\pi d^2}{4A(h)}\sqrt{2g(h+e)}$$

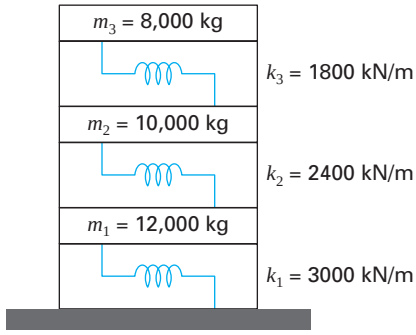



**Figure P28.29**

where $h$ = depth (m), $t$ = time (s), $d$ = pipe diameter (m), $A(h)$ = pond surface area as a function of depth (m$^2$), $g$ = gravitational constant (= 9.81 m/s$^2$), and $e$ = depth of pipe outlet below the pond bottom (m). Based on the following area-depth table, solve this differential equation to determine how long it takes for the pond to empty given that $h(0) = 6$ m, $d = 0.25$ m, $e = 1$ m.

| $h$, m | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| $A(h)$, $10^4$ m$^2$ | 1.17 | 0.97 | 0.67 | 0.45 | 0.32 | 0.18 | 0 |

**28.29** Engineers and scientists use mass-spring models to gain insight into the dynamics of structures under the influence of disturbances such as earthquakes. Figure P28.29 shows such a representation for a three-story building. For this case, the analysis is limited to horizontal motion of the structure. Force balances can be developed for this system as

$$\left(\frac{k_1+k_2}{m_1} - \omega^2\right)X_1 - \frac{k_2}{m_1}X_2 = 0$$

$$-\frac{k_2}{m_2}X_1 + \left(\frac{k_2+k_3}{m_2} - \omega^2\right)X_2 - \frac{k_3}{m_2}X_3 = 0$$

$$-\frac{k_3}{m_3}X_2 + \left(\frac{k_3}{m_3} - \omega^2\right)X_3 = 0$$

Determine the eigenvalues and eigenvectors and graphically represent the modes of vibration for the structure by displaying the amplitudes versus height for each of the eigenvectors. Normalize the amplitudes so that the displacement of the third floor is one.

**28.30** Under a number of simplifying assumptions, the steady-state height of the water table in a one-dimensional, unconfined groundwater aquifer (Fig. P28.30) can be modeled with the following 2nd-order ODE,

$$K\bar{h}\frac{d^2h}{dx^2} + N = 0$$

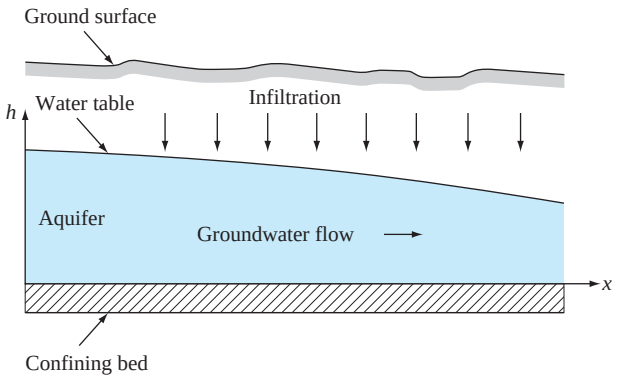



**Figure P28.30**
An unconfined or "phreatic" aquifer

where $x$ = distance (m), $K$ = hydraulic conductivity [m/d], $h$ = height of the water table [m], $\bar{h}$ = the average height of the water table [m], and $N$ = infiltration rate [m/d].

Solve for the height of the water table for $x = 0$ to 1000 m where $h(0) = 10$ m and $h(1000) = 5$ m. Use the following parameters for the calculation: $K = 1$ m/d and $N = 0.1$ m/d. Set the average height of the water table as the average of the boundary conditions. Obtain your solution with (**a**) the shooting method and (**b**) the finite-difference method ($\Delta x = 100$ m).

**28.31** In Prob. 28.30, a linearized groundwater model was used to simulate the height of the water table for an unconfined aquifer. A more realistic result can be obtained by using the following nonlinear ODE:

$$\frac{d}{dx}\left(Kh\frac{dh}{dx}\right) + N = 0$$

where $x$ = distance (m), $K$ = hydraulic conductivity [m/d], $h$ = height of the water table [m], and $N$ = infiltration rate [m/d]. Solve for the height of the water table for the same case as in Prob. 28.30. That is solve from $x = 0$ to 1000 m with $h(0) = 10$ m, $h(1000) = 5$ m, $K = 1$ m/d, and $N = 0.1$ m/d. Obtain your solution with (a) the shooting method and (b) the finite-difference method ($\Delta x = 100$ m).

**28.32** The Lotka-Volterra equations described in Sec. 28.2 have been refined to include additional factors that impact predator-prey dynamics. For example, over and above predation, prey population can be limited by other factors such as space. Space limitation can be incorporated into the model as a carrying capacity (recall the logistic model described in Prob. 28.16) as in

$$\frac{dx}{dt} = a\left(1 - \frac{x}{K}\right)x - bxy$$

$$\frac{dy}{dt} = -cy + dxy$$

where $K$ = the carrying capacity. Use the same parameter values and initial conditions as in Sec. 28.2 to integrate these equations from $t = 0$ to 100 using `ode45`.

**(a)** Employ a very large value of $K = 10^8$ to validate that you obtain the same results as in Sec. 28.2.

**(b)** Compare **(a)** with the more realistic carrying capacity of $K =$ 20,000. Discuss your results.

**28.33** The growth of floating, unicellular algae below a sewage treatment plant discharge can be modeled with the following simultaneous ODEs:

$$\frac{da}{dt} = \left[k_g(n, p) - k_d - k_s\right]a$$

$$\frac{dn}{dt} = r_{nc}k_h c - r_{na}k_g(n, p)a$$

$$\frac{dp}{dt} = r_{pc}k_h c - r_{pa}k_g(n, p)a$$

$$\frac{dc}{dt} = r_{ca}k_d a - k_h c$$

where $t$ = travel time [d], $a$ = algal chlorophyll concentration [$\mu$gA/L], $n$ = inorganic nitrogen concentration [$\mu$gN/L], $p$ = inorganic phosphorus concentration [$\mu$gP/L], $c$ = detritus concentration [$\mu$gC/L], $k_d$ = algal death rate [/d], $k_s$ = algal settling rate [/d], $k_h$ = detrital hydrolysis rate [/d], $r_{nc}$ = nitrogen-to-carbon ratio [$\mu$gN/$\mu$gC], $r_{pc}$ = phosphorus-to-carbon ratio [$\mu$gP/$\mu$gC], $r_{na}$ = nitrogen-to-chlorophyll ratio [$\mu$gN/$\mu$gA], $r_{pa}$ = phosphorus-to-chlorophyll ratio [$\mu$gP/$\mu$gA], and $k_g(n, p)$ = algal growth rate [/d], which can be computed with

$$k_g(n, p) = k_g \min\left\{\frac{p}{k_{sp} + p}, \frac{n}{k_{sn} + n}\right\}$$

where $k_g$ = the algal growth rate at excess nutrient levels [/d], $k_{sp}$ = the phosphorus half-saturation constant [$\mu$gP/L], and $k_{sn}$ = the nitrogen half-saturation constant [$\mu$gN/L]. Use the `ode45` and `ode15s` functions to solve these equations from $t = 0$ to 20 d given the initial conditions $a = 1$, $n = 4000$, $p = 400$, and $c = 0$. Note that the parameters are $k_d = 0.1$, $k_s = 0.15$, $k_h = 0.025$, $r_{nc} =$ 0.18, $r_{pc} = 0.08333$, $r_{na} = 7.2$, $r_{pa} = 1$, $k_g = 0.5$, $k_{sp} = 2$, and $k_{sn} = 15$. Develop plots of both solutions and interpret the results.

**28.34** The following ODEs have been proposed as a model of an epidemic:

$$\frac{dS}{dt} = -aSI$$

$$\frac{dI}{dt} = aSI - rI$$

$$\frac{dR}{dt} = rI$$

where $S$ = the susceptible individuals, $I$ = the infected, $R$ = the recovered, $a$ = the infection rate, and $r$ = the recovery rate. A city has 10,000 people, all of whom are susceptible.

**(a)** If a single infectious individual enters the city at $t = 0$, compute the progression of the epidemic until the number of infected individuals falls below 10. Use the following parameters: $a = 0.002$/(person·week) and $r = 0.15$/d. Develop time-series plots of all the state variables. Also generate a phase-plane plot of $S$ versus $I$ versus $R$.

**(b)** Suppose that after recovery, there is a loss of immunity that causes recovered individuals to become susceptible. This reinfection mechanism can be computed as $\rho R$, where $\rho$ = the reinfection rate. Modify the model to include this mechanism and repeat the computations in **(a)** using $\rho = 0.015$/d.

**Electrical Engineering**

**28.35** Perform the same computation as in the first part of Sec. 28.3, but with $R = 0.025\ \Omega$.

**28.36** Solve the ODE in the first part of Sec. 8.3 from $t = 0$ to 0.5 using numerical techniques if $q = 0.1$ and $i = -3.281515$ at $t = 0$. Use an $R = 50$ along with the other parameters from Sec. 8.3.

**28.37** For a simple RL circuit, Kirchhoff's voltage law requires that (if Ohm's law holds)

$$L\frac{di}{dt} + Ri = 0$$

where $i$ = current, $L$ = inductance, and $R$ = resistance. Solve for $i$, if $L = 1$, $R = 1.5$, and $i(0) = 0.5$. Solve this problem analytically and with a numerical method. Present your results graphically.

**28.38** In contrast to Prob. 28.37, real resistors may not always obey Ohm's law. For example, the voltage drop may be nonlinear and the circuit dynamics is described by a relationship such as

$$L\frac{di}{dt} + R\left[\frac{i}{I} - \left(\frac{i}{I}\right)^3\right] = 0$$

where all other parameters are as defined in Prob. 28.37 and $I$ is a known reference current equal to 1. Solve for $i$ as a function of time under the same conditions as specified in Prob. 28.37.

**28.39** Develop an eigenvalue problem for an $LC$ network similar to the one in Fig. 28.14, but with only two loops. That is, omit the $i_3$ loop. Draw the network, illustrating how the currents oscillate in their primary modes.

**28.40** Just as Fourier's law and the heat balance can be employed to characterize temperature distribution, analogous relationships are available to model field problems in other areas of engineering. For example, electrical engineers use a similar approach when modeling electrostatic fields. Under a number of simplifying assumptions, an analog of Fourier's law can be represented in one-dimensional form as

$$D = -\varepsilon\frac{dV}{dx}$$

where $D$ is called the electric flux density vector, $\varepsilon$ = permittivity of the material, and $V$ = electrostatic potential. Similarly, a Poisson equation for electrostatic fields can be represented in one dimension as

$$\frac{d^2V}{dx^2} = -\frac{\rho_v}{\varepsilon}$$

where $\rho_v$ = charge density. Use the finite-difference technique with $\Delta x = 2$ to determine $V$ for a wire where $V(0) = 1000$, $V(20) = 0$, $\varepsilon = 2$, $L = 20$, and $\rho_v = 30$.

**Mechanical/Aerospace Engineering**

**28.41** Perform the same computation as in Sec. 28.4 but for a 1-m-long pendulum.

**28.42** The rate of cooling of a body can be expressed as

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

where $T$ = temperature of the body (°C), $T_a$ = temperature of the surrounding medium (°C), and $k$ = the proportionality constant ($\text{min}^{-1}$). Thus, this equation specifies that the rate of cooling is proportional to the difference in temperature between the body and the surrounding medium. If a metal ball heated to 90°C is dropped into water that is held at a constant value of $T_a = 20$°C, use a numerical method to compute how long it takes the ball to cool to 40°C if $k = 0.25\ \text{min}^{-1}$.

**28.43** The rate of heat flow (conduction) between two points on a cylinder heated at one end is given by

$$\frac{dQ}{dt} = \lambda A\frac{dT}{dx}$$

where $\lambda$ = a constant, $A$ = the cylinder's cross-sectional area, $Q$ = heat flow, $T$ = temperature, $t$ = time, and $x$ = distance from the heated end. Because the equation involves two derivatives, we will simplify this equation by letting

$$\frac{dT}{dx} = \frac{100(L - x)(20 - t)}{100 - xt}$$

where $L$ is the length of the rod. Combine the two equations and compute the heat flow for $t = 0$ to 25 s. The initial condition is $Q(0) = 0$ and the parameters are $\lambda = 0.5$ cal · cm/s, $A = 12\ \text{cm}^2$, $L = 20$ cm, and $x = 2.5$ cm. Plot your results.

**28.44** Repeat the falling parachutist problem (Example 1.2), but with the upward force due to drag as a second-order rate:

$$F_u = -cv^2$$

where $c = 0.225$ kg/m. Solve for $t = 0$ to 30, plot your results, and compare with those of Example 1.2.

**28.45** Suppose that, after falling for 13 s, the parachutist from Examples 1.1 and 1.2 pulls the rip cord. At this point, assume that the drag coefficient is instantaneously increased to a constant value of 55 kg/s. Compute the parachutist's velocity from $t = 0$ to 30 s with Heun's method (without iteration of the corrector) using a step-size of 2 s. Plot $v$ versus $t$ for $t = 0$ to 30 s.

**28.46** The following ordinary differential equation describes the motion of a damped spring-mass system (Fig. P28.46):

$$m\frac{d^2x}{dt^2} + a\left|\frac{dx}{dt}\right|\frac{dx}{dt} + bx^3 = 0$$

where $x$ = displacement from the equilibrium position, $t$ = time, $m = 1$ kg mass, and $a = 5$ N/(m/s)$^2$. The damping term is nonlinear and represents air damping.

The spring is a cubic spring and is also nonlinear with $b = 5$ N/m$^3$. The initial conditions are

| | |
|---|---|
| *Initial velocity* | $\frac{dx}{dt} = 0.5$ m/s |
| *Initial displacement* | $x = 1$ m |

Solve this equation using a numerical method over the time period $0 \le t \le 8$ s. Plot the displacement and velocity versus time and plot the phase-plane portrait (velocity versus displacement) for all the following cases:

**(a)** A similar linear equation

$$m\frac{d^2x}{dt^2} + 2\frac{dx}{dt} + 5x = 0$$

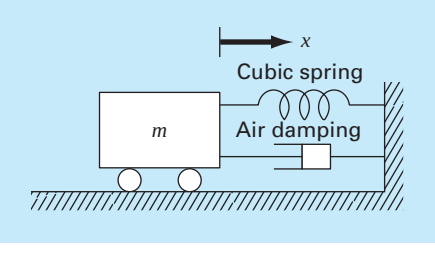
**(b)** The nonlinear equation with only a nonlinear spring term

$$\frac{d^2x}{dt^2} + 2\frac{dx}{dt} + bx^3 = 0$$

**(c)** The nonlinear equation with only a nonlinear damping term

$$m\frac{d^2x}{dt^2} + a\left|\frac{dx}{dt}\right|\frac{dx}{dt} + 5x = 0$$

**Figure P28.46**



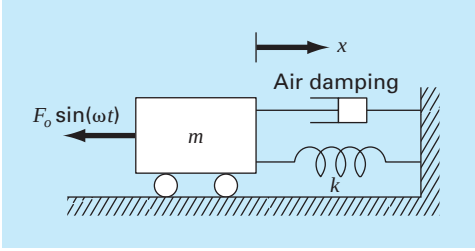




**Figure P28.47**

**(d)** The full nonlinear equation where both the damping and spring terms are nonlinear

$$m\frac{d^2x}{dt^2} + a\left|\frac{dx}{dt}\right|\frac{dx}{dt} + bx^3 = 0$$

**28.47** A forced damped spring-mass system (Fig. P28.47) has the following ordinary differential equation of motion:

$$m\frac{d^2x}{dt^2} + a\left|\frac{dx}{dt}\right|\frac{dx}{dt} + kx = F_o \sin(\omega t)$$

where $x$ = displacement from the equilibrium position, $t$ = time, $m = 2$ kg mass, $a = 5$ N/(m/s)$^2$, and $k = 6$ N/m. The damping term is nonlinear and represents air damping. The forcing function $F_o \sin(\omega t)$ has values of $F_o = 2.5$ N and $\omega = 0.5$ rad/sec. The initial conditions are

| | |
|---|---|
| *Initial velocity* | $\frac{dx}{dt} = 0$ m/s |
| *Initial displacement* | $x = 1$ m |

Solve this equation using a numerical method over the time period $0 \le t \le 15$ s. Plot the displacement and velocity versus time, and plot the forcing function on the same curve. Also, develop a separate plot of velocity versus displacement.

**28.48** The temperature distribution in a tapered conical cooling fin (Fig. P28.48) is described by the following differential equation, which has been nondimensionalized

$$\frac{d^2u}{dx^2} + \left(\frac{2}{x}\right)\left(\frac{du}{dx} - pu\right) = 0$$

where $u$ = temperature $(0 \le u \le 1)$, $x$ = axial distance $(0 \le x \le 1)$, and $p$ is a nondimensional parameter that describes the heat transfer and geometry

$$p = \frac{hL}{k}\sqrt{1 + \frac{4}{2m^2}}$$

where $h$ = a heat transfer coefficient, $k$ = thermal conductivity, $L$ = the length or height of the cone, and $m$ = the slope of the cone wall. The equation has the boundary conditions

$$u(x = 0) = 0 \qquad u(x = 1) = 1$$

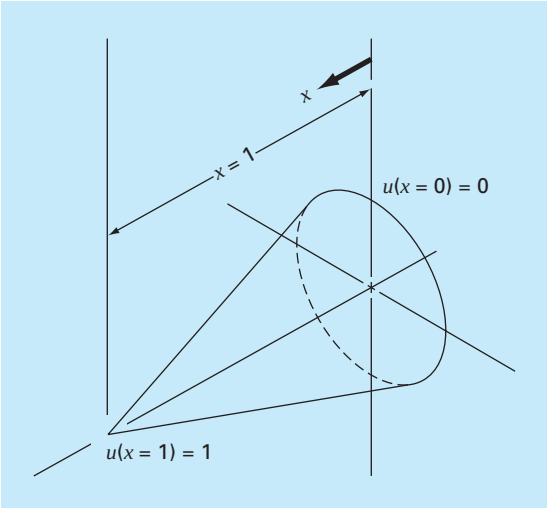



**Figure P28.48**

Solve this equation for the temperature distribution using finite difference methods. Use second-order accurate finite difference analogues for the derivatives. Write a computer program to obtain the solution and plot temperature versus axial distance for various values of $p = 10$, 20, 50, and 100.

**28.49** The dynamics of a forced spring-mass-damper system can be represented by the following second-order ODE:

$$m\frac{d^2x}{dt^2} + c\frac{dx}{dt} + k_1x + k_3x^3 = P\cos(\omega t)$$

where $m = 1$ kg, $c = 0.4$ N · s/m, $P = 0.5$ N, and $\omega = 0.5$/s. Use a numerical method to solve for displacement ($x$) and velocity ($v = dx/dt$) as a function of time with the initial conditions $x = v = 0$. Express your results graphically as time-series plots ($x$ and $v$ versus $t$) and a phase plane plot ($v$ versus $x$). Perform simulations for both **(a)** linear ($k_1 = 1$; $k_3 = 0$) and **(b)** nonlinear ($k_1 = 1$; $k_3 = 0.5$) springs.

**28.50** The differential equation for the velocity of a bungee jumper is different depending on whether the jumper has fallen to a distance where the cord is fully extended and begins to stretch. Thus, if the distance fallen is less than the cord length, the jumper is only subject to gravitational and drag forces. Once the cord begins to stretch, the spring and dampening forces of the cord must also be included. These two conditions can be expressed by the following equations:

$$\frac{dv}{dt} = g - \text{sign}(v)\frac{c_d}{m}v^2 \qquad x \le L$$

$$\frac{dv}{dt} = g - \text{sign}(v)\frac{c_d}{m}v^2 - \frac{k}{m}(x - L) - \frac{\gamma}{m}v \qquad x > L$$

where $v$ = velocity (m/s), $t$ = time (s), $g$ = gravitational constant (= 9.81 m/s$^2$), sign($x$) = function that returns −1, 0, and 1 for negative, zero and positive $x$, respectively, $c_d$ = second-order drag coefficient (kg/m), $m$ = mass (kg), $k$ = cord spring constant (N/m), $\gamma$ = cord dampening coefficient (N · s/m), and $L$ = cord length (m). Determine the position and velocity of the jumper given the following parameters: $L = 30$ m, $m = 68.1$ kg, $c_d = 0.25$ kg/m, $k = 40$ N/m, and $\gamma = 8$ kg/s. Perform the computation from $t = 0$ to 50 s and assume that the initial conditions are $x(0) = v(0) = 0$.

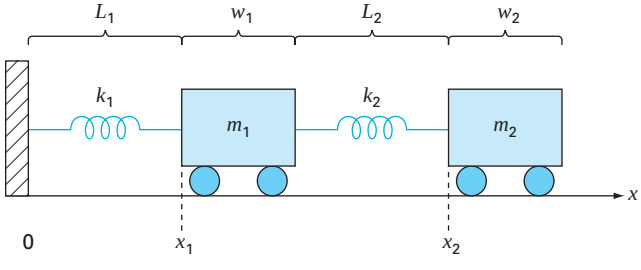
**28.51** Two masses are attached to a wall by linear springs (Fig. P28.51). Force balances based on Newton's second law can be written as

$$\frac{d^2x_1}{dt^2} = -\frac{k_1}{m_1}(x_1 - L_1) + \frac{k_2}{m_1}(x_2 - x_1 - w_1 - L_2)$$

$$\frac{d^2x_2}{dt^2} = -\frac{k_2}{m_2}(x_2 - x_1 - w_1 - L_2)$$

where $k$ = the spring constants, $m$ = mass, $L$ = the length of the unstretched spring, and $w$ = the width of the mass. Compute the positions of the masses as a function of time using the following parameter values: $k_1 = k_2 = 5$, $m_1 = m_2 = 2$, $w_1 = w_2 = 5$, and $L_1 = L_2 = 2$. Set the initial conditions as $x_1 = L_1$ and $x_2 = L_1 + w_1 + L_2 + 6$. Perform the simulation from $t = 0$ to 20. Construct time-series plots of both the displacements and the velocities. In addition, produce a phase-plane plot of $x_1$ versus $x_2$.

**Figure P28.51**



# EPILOGUE: PART SEVEN

## PT7.4 TRADE-OFFS

Table PT7.3 contains trade-offs associated with numerical methods for the solution of initial-value ordinary differential equations. The factors in this table must be evaluated by the engineer when selecting a method for each particular applied problem.

Simple self-starting techniques such as Euler's method can be used if the problem requirements involve a short range of integration. In this case, adequate accuracy may be obtained using small step sizes to avoid large truncation errors, and the round-off errors may be acceptable. Euler's method may also be appropriate for cases where the mathematical model has an inherently high level of uncertainty or has coefficients or forcing functions with significant errors as might arise during a measurement process. In this case, the accuracy of the model itself simply does not justify the effort involved to employ a

**TABLE PT7.3** Comparison of the characteristics of alternative methods for the numerical solution of ODEs. The comparisons are based on general experience and do not account for the behavior of special functions.

| Method | Starting Values | Iterations Required | Global Error | Ease of Changing Step Size | Programming Effort | Comments |
|---|---|---|---|---|---|---|
| One step | | | | | | |
| Euler's | 1 | No | $O(h)$ | Easy | Easy | Good for quick estimates |
| Heun's | 1 | Yes | $O(h^2)$ | Easy | Moderate | — |
| Midpoint | 1 | No | $O(h^2)$ | Easy | Moderate | — |
| Second-order Ralston | 1 | No | $O(h^2)$ | Easy | Moderate | The second-order RK method that minimizes truncation error |
| Fourth-order RK | 1 | No | $O(h^4)$ | Easy | Moderate | Widely used |
| Adaptive fourth-order RK or RK-Fehlberg | 1 | No | $O(h^5)$* | Easy | Moderate to difficult | Error estimate allows step-size adjustment |
| Multistep | | | | | | |
| Non-self-starting Heun | 2 | Yes | $O(h^3)$* | Difficult | Moderate to difficult† | Simple multistep method |
| Milne's | 4 | Yes | $O(h^5)$* | Difficult | Moderate to difficult† | Sometimes unstable |
| Fourth-order Adams | 4 | Yes | $O(h^5)$* | Difficult | Moderate to difficult† | |

*Provided the error estimate is used to modify the solution.

†With variable step size.

more complicated numerical method. Finally, the simpler techniques may be best when the problem or simulation need only be performed a few times. In these applications, it is probably best to use a simple method that is easy to program and understand despite the fact that the method may be computationally inefficient and relatively time-consuming to run on the computer.

If the range of integration of the problem is long enough to involve a large number of steps, then it may be necessary and appropriate to use a more accurate technique than Euler's method. The fourth-order RK method is popular and reliable for many engineering problems. In these cases, it may also be advisable to estimate the truncation error for each step as a guide to selecting the best step size. This can be accomplished with the adaptive RK or fourth-order Adams approaches. If the truncation errors are extremely small, it may be wise to increase the step size to save computer time. On the other hand, if the truncation error is large, the step size should be decreased to avoid accumulation of error. Milne's method should be avoided if significant stability problems are expected. The Runge-Kutta method is simple to program and convenient to use but may be less efficient than the multistep methods. However, the Runge-Kutta method is usually employed in any event to obtain starting values for the multistep methods.

A large number of engineering problems may fall into an intermediate range of integration interval and accuracy requirement. In these cases, the second-order RK and the non-self-starting Heun methods are simple to use and are relatively efficient and accurate.

*Stiff systems* involve equations with slowly and rapidly varying components. Special techniques are usually required for the adequate solution of stiff equations. For example, implicit approaches are often used. You can consult Enright et al. (1975), Gear (1971), and Shampine and Gear (1979) for additional information regarding these techniques.

A variety of techniques are available for solving eigenvalue problems. For small systems or where only a few of the smallest or largest eigenvalues are required, simple approaches such as the polynomial and the power methods are available. For symmetric systems, Jacobi's, Given's, or Householder's method can be employed. Finally, the QR method represents a general approach for finding all the eigenvalues of symmetric and nonsymmetric matrices.

## PT7.5 IMPORTANT RELATIONSHIPS AND FORMULAS

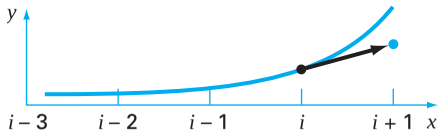
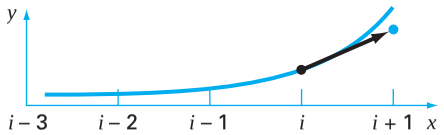
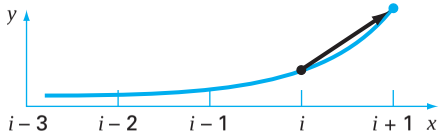
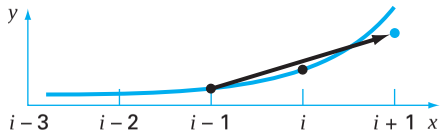
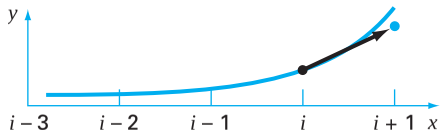
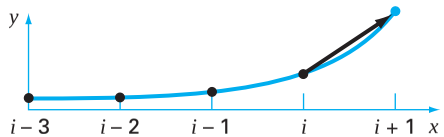
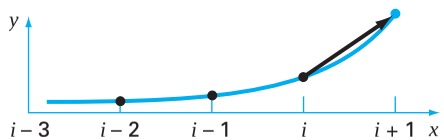
Table PT7.4 summarizes important information that was presented in Part Seven. This table can be consulted to quickly access important relationships and formulas.

## PT7.6 ADVANCED METHODS AND ADDITIONAL REFERENCES

Although we have reviewed a number of techniques for solving ordinary differential equations there is additional information that is important in engineering practice. The question of *stability* was introduced in Sec. 26.2.4. This topic has general relevance to all methods for solving ODEs. Further discussion of the topic can be pursued in Carnahan, Luther, and Wilkes (1969), Gear (1971), and Hildebrand (1974).

In Chap. 27, we introduced methods for solving *boundary-value* problems. Isaacson and Keller (1966), Keller (1968), Na (1979), and Scott and Watts (1976) can be consulted for additional information on standard boundary-value problems. Additional material on

**TABLE PT7.4** Summary of important information presented in Part Seven.

| Method | Formulation | Graphic Interpretation | Errors |
|---|---|---|---|
| Euler (First-order RK) | $y_{i+1} = y_i + hk_1$<br>$k_1 = f(x_i, y_i)$ |  | Local error $\simeq O(h^2)$<br>Global error $\simeq O(h)$ |
| Ralston's second-order RK | $y_{i+1} = y_i + h(\frac{1}{3}k_1 + \frac{2}{3}k_2)$<br>$k_1 = f(x_i, y_i)$<br>$k_2 = f(x_i + \frac{3}{4}h, y_i + \frac{3}{4}hk_1)$ |  | Local error $\simeq O(h^3)$<br>Global error $\simeq O(h^2)$ |
| Classic fourth-order RK | $y_{i+1} = y_i + h(\frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4)$<br>$k_1 = f(x_i, y_i)$<br>$k_2 = f(x_i + \frac{1}{2}h, y_i + \frac{1}{2}hk_1)$<br>$k_3 = f(x_i + \frac{1}{2}h, y_i + \frac{1}{2}hk_2)$<br>$k_4 = f(x_i + h, y_i + hk_3)$ |  | Local error $\simeq O(h^5)$<br>Global error $\simeq O(h^4)$ |
| Non-self-starting Heun | Predictor: (midpoint method)<br>$y^0_{i+1} = y^m_{i-1} + 2hf(x_i, y^m_i)$ |  | Predictor modifier:<br>$E_p \simeq \frac{4}{5}(y^m_{i,u} + y^0_{i,u})$ |
| | Corrector: (trapezoidal rule)<br>$y^j_{i+1} = y^m_i + h\,\dfrac{f(x_i, y^m_i) + f(x_{i+1}, y^{j-1}_{i+1})}{2}$ |  | Corrector modifier:<br>$E_c \simeq -\dfrac{y^m_{i+1,u} - y^0_{i+1,u}}{5}$ |
| Fourth-order Adams | Predictor: (fourth Adams-Bashforth)<br>$y^0_{i+1} = y^m_i + h(\frac{55}{24}f^m_i - \frac{59}{24}f^m_{i-1} + \frac{37}{24}f^m_{i-2} - \frac{9}{24}f^m_{i-3})$ |  | Predictor modifier:<br>$E_p \simeq \frac{251}{270}(y^m_{i,u} - y^0_{i,u})$ |
| | Corrector: (fourth Adams-Moulton)<br>$y^j_{i+1} = y^m_i + h(\frac{9}{24}f^{j-1}_{i+1} + \frac{19}{24}f^m_i - \frac{5}{24}f^m_{i-1} + \frac{1}{24}f^m_{i-2})$ |  | Corrector modifier:<br>$E_c \simeq -\frac{19}{270}(y^m_{i+1,u} - y^0_{i+1,u})$ |

eigenvalues can be found in Ralston and Rabinowitz (1978), Wilkinson (1965), Fadeev and Fadeeva (1963), and Householder (1953, 1964).

In summary, the foregoing is intended to provide you with avenues for deeper exploration of the subject. Additionally, all the above references provide descriptions of the basic techniques covered in Part Seven. We urge you to consult these alternative sources to broaden your understanding of numerical methods for the solution of differential equations.