5

Numerical Integration

Dynamic systems may frequently be modeled by systems of *ordinary differential equations* (or ODEs) of the form

$$\dot{x}(t) = f(x,t) \quad x(t_0) = x_0 \tag{5.1}$$

where $x(t) \in R^n$ is a time-varying function that depends on the initial condition x_0. Such problems are often referred to as "initial value problems." A system of nonlinear differential equations cannot typically be solved analytically. In other words, a closed form expression for $x(t)$ cannot be found directly from Equation (5.1), but rather must be solved numerically.

In the numerical solution of Equation (5.1), a sequence of points $x_0, x_1, x_2 \ldots$, is computed that approximates the true solution at a set of time points $t_0, t_1, t_2, \ldots$. The time interval between adjacent time points is called the *time step* and an integration algorithm advances the numerical solution by one time step with each application. The time step $h_{n+1} = t_{n+1} - t_n$ may be constant for all time intervals over the entire integration interval $t \in [t_0, t_N]$ or may vary at each step.

The basic integration algorithm advances the solution from t_n to t_{n+1} with integration step size h_{n+1} based on a calculation that involves previously computed values $x_n, x_{n-1}, \ldots$ and functions $f(x_n, t_n), f(x_{n-1}, t_{n-1}), \ldots$. Each practical integration algorithm must satisfy certain criteria concerning

1. numerical accuracy,
2. numerical stability, and
3. numerical efficiency.

Numerical accuracy ensures that the numerical error incurred at each step of the integration remains bounded. The global error of an integration error is the total error accrued over a given time interval. The global error at time t_n is given by

$$\text{global error } = \|x(t_n) - x_n\|$$

where $x(t_n)$ is the exact solution to Equation (5.1) at time t_n and x_n is the approximated solution. Of course, it is impossible to determine the global error exactly if the solution $x(t)$ is not known analytically, but it is possible to establish bounds on the error incurred at each step of the integration method.

The numerical stability of an integration algorithm implies that errors incurred at each step do not propagate to future times. Numerical efficiency is a function of the amount of computation required at each time step and the size of the steps between adjacent time intervals. Each of these criteria will be discussed in greater detail later in this chapter after an introduction to several different forms of integration algorithms.

5.1 One-Step Methods

The basic form of an integration algorithm is one that advances the solution from x_n to x_{n+1} using only the information currently available. This type of solution is called a *one-step* method, in that only information from one step of the integration algorithm is used. The family of one-step methods has the advantage of conserving memory, since only the previous solution must be retained. Several well-known methods fall into this category.

5.1.1 Taylor Series-Based Methods

One important class of integration methods is derived from using the Taylor series expansion of Equation (5.1). Let $\hat{x}(t)$ denote the exact solution to Equation (5.1). Expanding $\hat{x}(t)$ in a Taylor series about $t = t_n$ and evaluating the series at $t = t_{n+1}$ yields the following series expansion for $\hat{x}\left(t_{n+1}\right)$:

$$\hat{x}\left(t_{n+1}\right) = \hat{x}\left(t_n\right) + \dot{x}\left(t_n\right)\left(t_{n+1} - t_n\right) + \frac{1}{2!}\ddot{x}\left(t_n\right)\left(t_{n+1} - t_n\right)^2 + \ldots + \frac{1}{p!}x^{(p)}\left(t_n\right)\left(t_{n+1} - t_n\right)^p + h.o.t.$$

where *h.o.t.* stands for *higher-order terms* of the expansion. If the time step $h = t_{n+1} - t_n$, then

$$\hat{x}\left(t_{n+1}\right) = \hat{x}\left(t_n\right) + h\dot{x}\left(t_n\right) + \frac{h^2}{2!}\ddot{x}\left(t_n\right) + \ldots + \frac{h^p}{p!}x^{(p)}\left(t_n\right) + h.o.t.$$

From Equation (5.1), $\dot{x}(t) = f(x, t)$; therefore,

$$\hat{x}\left(t_{n+1}\right) - h.o.t. = \hat{x}\left(t_n\right) + hf\left(x\left(t_n\right), t_n\right) + \frac{h^2}{2!}f'\left(x\left(t_n\right), t_n\right) + \ldots + \frac{h^p}{p!}f^{(p-1)}\left(x\left(t_n\right), t_n\right) \quad (5.2)$$

If the higher-order terms are small, then a good approximation x_{n+1} to $\hat{x}\left(t_{n+1}\right)$ is given by the right-hand side of Equation (5.2).

In general, the Taylor series-based integration methods can be expressed as

$$x_{n+1} = x_n + hT_p(x_n) \tag{5.3}$$

where

$$T_p(x_n) = f(x(t_n), t_n) + \frac{h^2}{2!} f'(x(t_n), t_n) + \ldots + \frac{h^p}{p!} f^{(p-1)}(x(t_n), t_n)$$

and the integer p is called the *order* of the integration method. This method is very accurate for large p, but is not computationally efficient for large p since it requires a large number of function derivatives and evaluations.

5.1.2 Forward Euler Method

For $p = 1$, the Taylor series-based integration algorithm is given by:

$$x_{n+1} = x_n + hf(x_n, t_n) \tag{5.4}$$

which is also the well-known *Euler* or *forward Euler* method.

5.1.3 Runge–Kutta Methods

A second order Taylor method can be derived for $p = 2$.

$$\begin{aligned} x_{n+1} &= x_n + hT_2(x_n, t_n) \\ &= x_n + hf(x_n, t_n) + \frac{h^2}{2} f'(x_n, t_n) \end{aligned}$$

As the order of the Taylor method increases, so does the number of derivatives and partial derivatives. In many cases, the analytic derivation of the derivatives can be replaced by a numerical approximation. One of the most commonly known higher-order Taylor series-based integration methods is the Runge–Kutta method, where the derivatives are replaced by approximations. The fourth-order Runge–Kutta method is given by

$$x_{n+1} = x_n + hK_4(x_n, t_n) \tag{5.5}$$

where K_4 is an approximation to T_4:

$$\begin{aligned} K_4 &= \frac{1}{6}[k_1 + 2k_2 + 2k_3 + k_4] \\ k_1 &= f(x_n, t_n) \\ k_2 &= f\left(x_n + \frac{h}{2}k_1, t_n + \frac{h}{2}\right) \\ k_3 &= f\left(x_n + \frac{h}{2}k_2, t_n + \frac{h}{2}\right) \\ k_4 &= f(x_n + hk_3, t_n + h) \end{aligned}$$

where each k_i represents the slope (derivative) of the function at four different points. The slopes are then weighted $\left[\frac{1}{6}\ \frac{2}{6}\ \frac{2}{6}\ \frac{1}{6}\right]$ to approximate the T_4 function.

The advantages of Taylor series-based methods is that the method is straightforward to program and only depends on the previous time step. These methods (especially the Runge–Kutta methods) suffer from difficult error analysis, however, since the derivatives are approximated and not found analytically. Therefore, the integration step size is typically chosen conservatively (small), and computational efficiency may be lost.

5.2 Multistep Methods

Another approach to approximating the solution $x(t)$ of Equation (5.1) is to approximate the nonlinear function as a polynomial of degree k such that

$$\hat{x}(t) = \alpha_0 + \alpha_1 t + \alpha_2 t^2 + \ldots + \alpha_k t^k \tag{5.6}$$

where the coefficients $\alpha_0, \alpha_1, \ldots, \alpha_k$ are constant. It can be proven that any function can be approximated arbitrarily closely (within a predetermined ε) with a polynomial of sufficiently high degree on a finite interval $[t_0, t_N]$. The polynomial approximation can be related to the solution of Equation (5.1) through the introduction of *multistep* methods. A multistep method is one in which the approximation x_{n+1} can be a function of any number of previous numerical approximations $x_n, x_{n-1}, \ldots$ and corresponding functions $f(x_n, t_n), f(x_{n-1}, t_{n-1}), \ldots$ unlike one-step methods (such as the Runge–Kutta), which depend only on the information from the immediately previous step. In general,

$$\begin{aligned} x_{n+1} &= a_0 x_n + a_1 x_{n-1} + \ldots + a_p x_{n-p} + h\left[b_{-1} f(x_{n+1}, t_{n+1}) + b_0 f(x_n, t_n)\right. \\ &\quad \left. + b_1 f(x_{n-1}, t_{n-1}) + \ldots + b_p f(x_{n-p}, t_{n-p})\right] \qquad (5.7) \\ &= \sum_{i=0}^{p} a_i x_{n-i} + h \sum_{i=-1}^{p} b_i f(x_{n-i}, t_{n-i}) \qquad (5.8) \end{aligned}$$

To relate the integration method to the polynomial approximation, a relationship between the coefficients must be determined. A k-degree polynomial is uniquely determined by $k+1$ coefficients $(\alpha_0, \ldots, \alpha_k)$. The numerical integration method has $2p+3$ coefficients; therefore, the coefficients must be chosen such that

$$2p + 3 \geq k + 1 \tag{5.9}$$

The order of the numerical integration method is the highest degree k of a polynomial in t for which the numerical solution coincides with the exact

solution. The coefficients may be determined by selecting a set of linear basis functions $[\phi_1(t)\ \phi_2(t),\ \ldots,\ \phi_k(t)]$ such that

$$\phi_j(t) = t^j \quad j = 0, 1, \ldots, k$$

and solving the set of multistep equations

$$\phi_j\left(t_{n+1}\right) = \sum_{i=0}^{p} a_i \phi_j\left(t_{n-i}\right) + h_{n+1}\left[\sum_{i=-1}^{p} b_i \dot{\phi}\left(t_{n-i}\right)\right]$$

for all $j = 0, 1, \ldots, k$.

This method can be applied to derive several first-order numerical integration methods. Consider the case where $p = 0$ and $k = 1$. This satisfies the constraint of Equation (5.9); thus it is possible to determine multistep coefficients that will result in an exact polynomial of degree 1. The set of basis functions for $k = 1$ is

$$\phi_0(t) = 1 \tag{5.10}$$
$$\phi_1(t) = t \tag{5.11}$$

which lead to the derivatives

$$\dot{\phi}_0(t) = 0 \tag{5.12}$$
$$\dot{\phi}_1(t) = 1 \tag{5.13}$$

and the multistep equation

$$x_{n+1} = a_0 x_n + b_{-1} h_{n+1} f\left(x_{n+1}, t_{n+1}\right) + b_0 h_{n+1} f\left(x_n, t_n\right) \tag{5.14}$$

Representing the multistep method of Equation (5.14) in terms of basis functions yields the following two equations:

$$\phi_0\left(t_{n+1}\right) = a_0 \phi_0(t_n) + b_{-1} h_{n+1} \dot{\phi}_0\left(t_{n+1}\right) + b_0 h_{n+1} \dot{\phi}_0\left(t_n\right) \tag{5.15}$$
$$\phi_1\left(t_{n+1}\right) = a_0 \phi_1(t_n) + b_{-1} h_{n+1} \dot{\phi}_1\left(t_{n+1}\right) + b_0 h_{n+1} \dot{\phi}_1\left(t_n\right) \tag{5.16}$$

Substituting the choice of basis functions of Equations (5.10) and (5.11) into Equations (5.15) and (5.16) results in

$$1 = a_0(1) + b_{-1} h_{n+1}(0) + h_{n+1} b_0(0) \tag{5.17}$$
$$t_{n+1} = a_0 t_n + b_{-1} h_{n+1}(1) + b_0 h_{n+1}(1) \tag{5.18}$$

From Equation (5.17), the coefficient $a_0 = 1$. Recalling that $t_{n+1} - t_n = h_{n+1}$, Equation (5.18) yields

$$b_{-1} + b_0 = 1 \tag{5.19}$$

This choice of order and degree leads to two equations in three unknowns; therefore, one of them may be chosen arbitrarily. By choosing $a_0 = 1, b_{-1} = 0$, and $b_0 = 1$, Euler's method is once again obtained:

$$x_{n+1} = x_n + h_{n+1} f\left(x_n, t_n\right)$$

However, if $a_0 = 1, b_{-1} = 1$, and $b_0 = 0$, a different integration method is obtained:

$$x_{n+1} = x_n + h_{n+1} f\left(x_{n+1}, t_{n+1}\right) \tag{5.20}$$

This particular integration method is frequently called the *backward Euler* method. Note that, in this method, the coefficient b_{-1} is not identically zero; thus the expression for x_{n+1} depends implicitly on the function $f\left(x_{n+1}, t_{n+1}\right)$. Methods in which $b_{-1} \neq 0$ are referred to as *implicit* methods; otherwise they are *explicit*. Since there is an implicit (and often nonlinear) dependence on x_{n+1}, implicit integration methods must usually be solved iteratively at each time interval.

Consider now the case where $p = 0$, and $k = 2$. In this case, $2p + 3 = k + 1$ and the coefficients can be uniquely determined. Choosing the basis functions as previously with $\phi_2(t) = t^2$ and $\dot{\phi}_2(t) = 2t$ yields the following three equations:

$$1 = a_0(1) + b_{-1} h_{n+1}(0) + h_{n+1} b_0(0) \tag{5.21}$$

$$t_{n+1} = a_0 t_n + b_{-1} h_{n+1}(1) + b_0 h_{n+1}(1) \tag{5.22}$$

$$t_{n+1}^2 = a_0 t_n^2 + h_{n+1}\left(b_{-1}\left(2t_{n+1}\right) + b_0\left(2t_n\right)\right) \tag{5.23}$$

If $t_n = 0$, then $t_{n+1} = h_{n+1}$, and Equations (5.21) through (5.23) yield $a_0 = 1, b_{-1} = \frac{1}{2}$, and $b_0 = \frac{1}{2}$; thus

$$x_{n+1} = x_n + \frac{1}{2} h_{n+1}\left[f\left(x_{n+1}, t_{n+1}\right) + f\left(x_n, t_n\right)\right] \tag{5.24}$$

This second-order integration method is called the *trapezoidal* method and it is also implicit. This formula is called the trapezoidal method since the second term of Equation (5.24) can be interpreted as being the area under a trapezoid.

Example 5.1

Numerically solve

$$\ddot{x}(t) = -x(t) \quad x(0) = 1 \tag{5.25}$$

using the Euler, backward Euler, trapezoidal, and Runge–Kutta methods for different fixed step sizes.

Solution 5.1 This second-order differential equation must first be converted to ODE format by defining $x_1 = x$ and $x_2 = \dot{x}$. Then

$$\dot{x}_1 = x_2 = f_1\left(x_1, x_2\right) \quad x_1(0) = 1 \tag{5.26}$$

$$\dot{x}_2 = -x_1 = f_2\left(x_1, x_2\right) \tag{5.27}$$

By inspection, the analytic solution to this set of equations is

$$x_1(t) = \cos t \tag{5.28}$$

$$x_2(t) = -\sin t \tag{5.29}$$

Typically, it is not possible to find the exact solution, but in this example, the exact solution will be used to compare the numerical solutions against.

Forward Euler
Applying the forward Euler method to the ODEs yields

$$\begin{aligned} x_{1,n+1} &= x_{1,n} + hf_1\left(x_{1,n}, x_{2,n}\right) && (5.30)\\ &= x_{1,n} + hx_{2,n} && (5.31)\\ x_{2,n+1} &= x_{2,n} + hf_2\left(x_{1,n}, x_{2,n}\right) && (5.32)\\ &= x_{2,n} - hx_{1,n} && (5.33) \end{aligned}$$

or in matrix form

$$\begin{bmatrix} x_{1,n+1} \\ x_{2,n+1} \end{bmatrix} = \begin{bmatrix} 1 & h \\ -h & 1 \end{bmatrix} \begin{bmatrix} x_{1,n} \\ x_{2,n} \end{bmatrix} \tag{5.34}$$

Backward Euler
Applying the backward Euler method to the ODEs yields

$$\begin{aligned} x_{1,n+1} &= x_{1,n} + hf_1\left(x_{1,n+1}, x_{2,n+1}\right) && (5.35)\\ &= x_{1,n} + hx_{2,n+1} && (5.36)\\ x_{2,n+1} &= x_{2,n} + hf_2\left(x_{1,n+1}, x_{2,n+1}\right) && (5.37)\\ &= x_{2,n} - hx_{1,n+1} && (5.38) \end{aligned}$$

or in matrix form

$$\begin{bmatrix} x_{1,n+1} \\ x_{2,n+1} \end{bmatrix} = \begin{bmatrix} 1 & -h \\ h & 1 \end{bmatrix}^{-1} \begin{bmatrix} x_{1,n} \\ x_{2,n} \end{bmatrix} \tag{5.39}$$

In the solution of Equation (5.39), the inverse of the matrix is not found explicitly, but rather the equations would be solved using LU factorization.

Trapezoidal
Applying the trapezoidal method to the ODEs yields

$$\begin{aligned} x_{1,n+1} &= x_{1,n} + \frac{1}{2}h\left[f_1\left(x_{1,n}, x_{2,n}\right) + f_1\left(x_{1,n+1}, x_{2,n+1}\right)\right] && (5.40)\\ &= x_{1,n} + \frac{1}{2}h\left[x_{2,n} + x_{2,n+1}\right] && (5.41)\\ x_{2,n+1} &= x_{2,n} + \frac{1}{2}h\left[f_2\left(x_{1,n}, x_{2,n}\right) + f_2\left(x_{1,n+1}, x_{2,n+1}\right)\right] && (5.42)\\ &= x_{2,n} - \frac{1}{2}h\left[x_{1,n} + x_{1,n+1}\right] && (5.43) \end{aligned}$$

or in matrix form

$$\begin{bmatrix} x_{1,n+1} \\ x_{2,n+1} \end{bmatrix} = \begin{bmatrix} 1 & -\frac{1}{2}h \\ \frac{1}{2}h & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1 & \frac{1}{2}h \\ -\frac{1}{2}h & 1 \end{bmatrix} \begin{bmatrix} x_{1,n} \\ x_{2,n} \end{bmatrix} \tag{5.44}$$

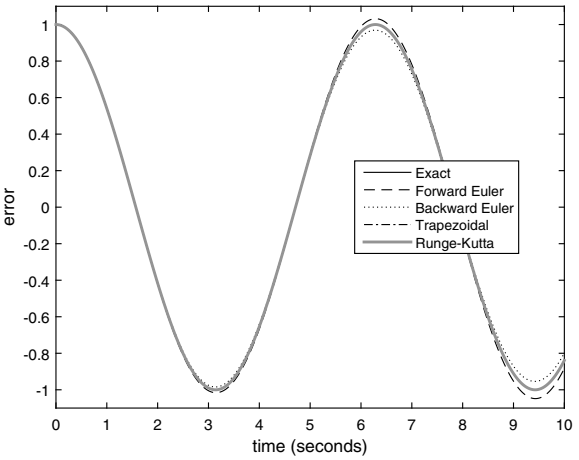

FIGURE 5.1
Numerical solutions for Example 5.1

Runge–Kutta
Applying the Runge–Kutta method to the ODEs yields

$$\begin{array}{ll} k_{11} = x_{2,n} & k_{21} = -x_{1,n} \\ k_{12} = x_{2,n} + \frac{h}{2}k_{11} & k_{22} = -x_{1,n} - \frac{h}{2}k_{21} \\ k_{13} = x_{2,n} + \frac{h}{2}k_{12} & k_{23} = -x_{1,n} - \frac{h}{2}k_{22} \\ k_{14} = x_{2,n} + hk_{13} & k_{24} = -x_{1,n} - hk_{23} \end{array}$$

and

$$x_{1,n+1} = x_{1,n} + \frac{h}{6}\left(k_{11} + 2k_{12} + 2k_{13} + k_{14}\right) \tag{5.45}$$

$$x_{2,n+1} = x_{2,n} + \frac{h}{6}\left(k_{21} + 2k_{22} + 2k_{23} + k_{24}\right) \tag{5.46}$$

The numerical solution of Equation (5.25) for each of the methods is shown in Figure 5.1 including the exact solution $\cos t$. Note that the trapezoidal and Runge–Kutta methods are nearly indistinguishable from the exact solution. Since the forward and backward Euler methods are first-order methods, they are not as accurate as the higher order trapezoidal and Runge–Kutta methods. Note that the forward Euler method generates a numerical solution whose magnitude is slightly larger than the exact solution and is increasing with time. Conversely, the backward Euler method generates a numerical solution whose magnitude is slightly less than the exact solution and is decreasing with time. Both properties are due to the *local truncation error* of the methods. The forward Euler method has a tendency to generate numerical solutions that increase with time (underdamped), whereas the backward Euler method

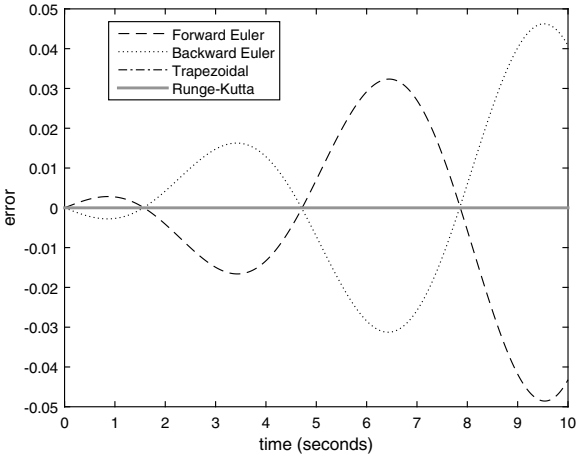

FIGURE 5.2
Error in numerical solutions for Example 5.1

tends to add damping to the numerical solution. Therefore, caution must be used when using either of these first-order methods for numerical integration.

Figure 5.2 shows the global error with time for each of the numerical methods. Note that the errors for the forward and backward Euler methods are equal in magnitude, but opposite in sign. This relationship will be further discussed later in this chapter. The numerical errors for the trapezoidal and the Runge–Kutta methods are reproduced in Figure 5.3 on a magnified scale. Note that even in this case, the Runge–Kutta error is much smaller than the trapezoidal method error. This is because the Runge–Kutta method is a fourth-order Taylor method whereas the trapezoidal method is a second-order polynomial approximation method. Section 5.3 will further explore the development of expressions for estimating the error of various integration methods. ■

When implicit methods, such as the trapezoidal method, are used to solve nonlinear systems of differential equations, the system of equations must be solved iteratively at each time step. For example, consider the following nonlinear system of equations:

$$\dot{x} = f(x(t), t) \quad x_0 = x(t_0) \tag{5.47}$$

Applying the trapezoidal method to numerically integrate this system results in the following discretized system:

$$x_{n+1} = x_n + \frac{h}{2}\left[f\left(x_n, t_n\right) + f\left(x_{n+1}, t_{n+1}\right)\right] \tag{5.48}$$

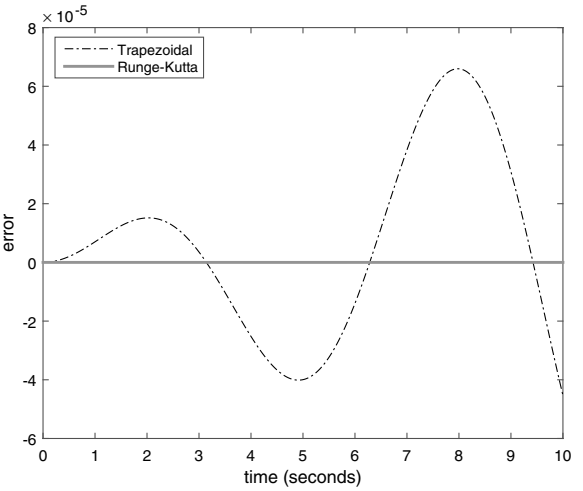

FIGURE 5.3
Error in trapezoidal and Runge–Kutta numerical solutions for Example 5.1

Since this nonlinear expression is implicit in x_{n+1}, it must be solved numerically:

$$x_{n+1}^{k+1} = x_{n+1}^{k} - \left[I - \frac{h}{2}\frac{\partial f}{\partial x}\right]^{-1}\Bigg|_{x_{n+1}^{k}} \left(x_{n+1}^{k} - x_n - \frac{h}{2}\left[f\left(x_n\right) + f\left(x_{n+1}^{k}\right)\right]\right) \tag{5.49}$$

where k is the Newton–Raphson iteration index, I is the identity matrix, and x_n is the converged value from the previous time step.

5.2.1 Adams Methods

Recall that the general class of multistep methods may be represented by

$$x_{n+1} = \sum_{i=0}^{p} a_i x_{n-i} + h \sum_{i=-1}^{p} b_i f\left(x_{n-i}, t_{n-i}\right) \tag{5.50}$$

A numerical multistep algorithm will give the exact value for x_{n+1} if $x(t)$ is a polynomial of degree less than or equal to k if the following *exactness constraints* are satisfied:

$$\sum_{i=0}^{p} a_i = 1 \tag{5.51}$$

$$\sum_{i=1}^{p} (-i)^p a_i + j \sum_{i=-1}^{p} (-i)^{(j-1)} b_i = 1 \quad \text{for } j = 1, 2, \ldots, k \tag{5.52}$$

The exactness constraint of Equation (5.51) is frequently referred to as the *consistency* constraint. Numerical multistep integration algorithms that satisfy Equation (5.51) are said to be "consistent." For a desired polynomial of degree k, these constraints can be satisfied by a wide variety of possibilities. Several families of methods have been developed by predefining some of the relationships between the coefficients. The family of *Adams* methods is defined by setting the coefficients $a_1 = a_2 = \ldots = a_p = 0$. By the consistency constraint, the coefficient a_0 must therefore equal 1.0. Thus, the Adams methods are reduced to

$$x_{n+1} = x_n + h \sum_{i=-1}^{p} b_i f\left(x_{n-i}, t_{n-i}\right) \tag{5.53}$$

where $p = k - 1$. The Adams methods can be further classified by the choice of implicit or explicit integration. The explicit class, frequently referred to as the "Adams–Bashforth" methods, is specified by setting $b_{-1} = 0$ and applying the second exactness constraint as

$$\sum_{i=0}^{k-1} (-i)^{(j-1)} b_i = \frac{1}{j} \quad j = 1, \ldots, k \tag{5.54}$$

In matrix form, Equation (5.54) becomes

$$\begin{bmatrix} 1 & 1 & 1 & \ldots & 1 \\ 0 & -1 & -2 & \ldots & -(k-1) \\ 0 & 1 & 4 & \ldots & (-(k-1))^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & (-1)^{(k-1)} & (-2)^{(k-1)} & \ldots & (-(k-1))^{(k-1)} \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_{k-1} \end{bmatrix} = \begin{bmatrix} 1 \\ \frac{1}{2} \\ \frac{1}{3} \\ \vdots \\ \frac{1}{k} \end{bmatrix} \tag{5.55}$$

By choosing the desired degree k (and subsequently the order p), the remaining b_i coefficients may be found from solving Equation (5.55).

Example 5.2
Find the third-order Adams–Bashforth integration method.

Solution 5.2 Setting $k = 3$ yields the following linear system:

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & -1 & -2 \\ 0 & 1 & 4 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} 1 \\ \frac{1}{2} \\ \frac{1}{3} \end{bmatrix}$$

Solving this system yields

$$b_0 = \frac{23}{12}$$
$$b_1 = -\frac{16}{12}$$
$$b_2 = \frac{5}{12}$$

Thus the third-order Adams–Bashforth method is given by

$$x_{n+1} = x_n + \frac{1}{12}h\left[23f\left(x_n, t_n\right) - 16f\left(x_{n-1}, t_{n-1}\right) + 5f\left(x_{n-2}, t_{n-2}\right)\right] \quad (5.56)$$

When implementing this algorithm, the values of x_n, x_{n-1}, and x_{n-2} must be saved in memory. ■

The implicit versions of the Adams methods have $b_{-1} \neq 0$, $p = (k-2)$, are called the "Adams–Moulton" methods, and are given by

$$x_{n+1} = x_n + h\sum_{i=-1}^{k-2} b_i f\left(x_{n-i}, t_{n-i}\right) \quad (5.57)$$

The second exactness constraint yields

$$\sum_{i=-1}^{k-2} (-i)^{(j-1)} b_i = \frac{1}{j} \quad j = 1, \ldots, k \quad (5.58)$$

or in matrix form

$$\begin{bmatrix} 1 & 1 & 1 & 1 & \ldots & 1 \\ 1 & 0 & -1 & -2 & \ldots & -(k-2) \\ 1 & 0 & 1 & 4 & \ldots & (-(k-2))^2 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & (-1)^{(k-2)} & (-2)^{(k-2)} & \ldots & (-(k-2))^{(k-2)} \end{bmatrix} \begin{bmatrix} b_{-1} \\ b_0 \\ b_1 \\ \vdots \\ b_{k-2} \end{bmatrix} = \begin{bmatrix} 1 \\ \frac{1}{2} \\ \frac{1}{3} \\ \vdots \\ \frac{1}{k} \end{bmatrix} \quad (5.59)$$

Example 5.3
Find the third-order Adams–Moulton integration method.

Solution 5.3 Setting $k = 3$ yields the following linear system:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & -1 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} b_{-1} \\ b_0 \\ b_1 \end{bmatrix} = \begin{bmatrix} 1 \\ \frac{1}{2} \\ \frac{1}{3} \end{bmatrix}$$

Solving this system yields

$$b_{-1} = \frac{5}{12}$$
$$b_0 = \frac{8}{12}$$
$$b_1 = -\frac{1}{12}$$

Thus the third-order Adams–Moulton method is given by

$$x_{n+1} = x_n + \frac{1}{12}h\left[5f\left(x_{n+1}, t_{n+1}\right) + 8f\left(x_n, t_n\right) - f\left(x_{n-1}, t_{n-1}\right)\right] \quad (5.60)$$

When implementing this algorithm, the values of x_n and x_{n-1} must be saved in memory and the equations must be solved iteratively if the function $f(x)$ is nonlinear. ■

The Adams–Moulton method is implicit and must be solved iteratively using the Newton–Raphson method (or other similar method), as shown in Equation (5.49). Iterative methods require an initial value for the iterative process to reduce the number of required iterations. The explicit Adams–Bashforth method is frequently used to estimate the initial value for the implicit Adams–Moulton method. If sufficiently high-order predictor methods are used, the Adams–Moulton method iteration will typically converge in only one iteration. This process is often called a *predictor corrector* approach; the Adams–Bashforth method predicts the solution and the implicit Adams–Moulton corrects the solution.

Another implementation issue for multistep methods is how to start up the integration at the beginning of the simulation since a high-order method requires several previous values. The usual procedure is to use a high order one-step method or to increase the number of steps of the method with each time step to generate the required number of values for the desired multistep method.

5.2.2 Gear's Methods

Another well-known family of multistep methods is Gear's methods [15]. This family of methods is particularly well suited for the numerical solution of stiff systems. As opposed to the Adams family of methods, where all the a_i coefficients except a_0 are zero, Gear's methods are identified by having all of the b_i coefficients equal to zero except b_{-1}. Obviously, since $b_{-1} \neq 0$, all Gear's methods are implicit methods. The kth order Gear's algorithm is obtained by setting $p = k - 1$ and $b_0 = b_1 = \ldots = 0$, yielding

$$x_{n+1} = a_0 x_n + a_1 x_{n-1} + \ldots + a_{k-1} x_{n-k+1} + h b_{-1} f\left(x_{n+1}, t_{n+1}\right) \tag{5.61}$$

The $k + 1$ coefficients can be calculated explicitly by applying the exactness constraints, as illustrated with the Adams methods

$$\begin{bmatrix} 1 & 1 & 1 \ldots & 1 & 0 \\ 0 & -1 & -2 \ldots & -(k-1) & 1 \\ 0 & 1 & 4 \ldots & [-(k-1)]^2 & 2 \\ \vdots & \vdots & \vdots \ddots & \vdots & \vdots \\ 0 & (-1)^k & (-2)^k \ldots & [-(k-1)]^k & k \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ b_{-1} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \tag{5.62}$$

The solution of Equation (5.62) uniquely determines the $k + 1$ coefficients of the kth order Gear's method.

Example 5.4
Find the third-order Gear's integration method.

Solution 5.4 Setting $k = 3$ yields the following linear system:

$$\begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & -1 & -2 & 1 \\ 0 & 1 & 4 & 2 \\ 0 & -1 & -8 & 3 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ b_{-1} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

Solving this system yields

$$\begin{aligned} b_{-1} &= \frac{6}{11} \\ a_0 &= \frac{18}{11} \\ a_1 &= -\frac{9}{11} \\ a_2 &= \frac{2}{11} \end{aligned}$$

Thus the third-order Gear's method is given by

$$x_{n+1} = \frac{18}{11}x_n - \frac{9}{11}x_{n-1} + \frac{2}{11}x_{n-2} + \frac{6}{11}hf\left(x_{n+1}, t_{n+1}\right) \tag{5.63}$$

When implementing this algorithm, the values of x_n through x_{n-2} must be saved in memory and the equations must be solved iteratively if the function $f(x)$ is nonlinear. ■

5.3 Accuracy and Error Analysis

The accuracy of numerical integration methods is impacted by two primary causes: computer round-off error and truncation error. Computer round-off error occurs as a result of the finite precision of the computer upon which the algorithm is implemented, and little can be done to reduce this error short of using a computer with greater precision. A double precision word length is normally used for scientific computation. The difference between the exact solution and the calculated solution is dominated by truncation error, which arises from the truncation of the Taylor series or polynomial being used to approximate the solution.

In the implementation of numerical integration algorithms, the most effective methods are those methods which require the least amount of calculation to yield the most accurate results. In general, higher-order methods produce

the most accurate results, but also require the greatest amount of computation. Therefore, it is desirable to compute the solution as infrequently as possible by taking the largest time step possible between intervals. Several factors impact the size of the time step, including the error introduced at each step by the numerical integration process itself. This error is the *local truncation error* (LTE) and arises from the truncation of the polynomial approximation and/or the truncation of the Taylor series expansion, depending on the method used. The term *local* emphasizes that the error is introduced locally and is not residual global error from earlier time steps. The error introduced at a single step of an integration method is given by

$$\varepsilon_T \triangleq x\left(t_{n+1}\right) - x_{n+1} \tag{5.64}$$

where $x\left(t_{n+1}\right)$ is the exact solution at time t_{n+1} and x_{n+1} is the numerical approximation. This definition assumes that this is the error *introduced in one step*; therefore, $x(t_n) = x_n$. The local truncation error is shown graphically in Figure 5.4. To compute the error, the solution $x\left(t_{n-i}\right)$ is expanded about t_{n+1}:

$$x_{n-i} = x\left(t_{n-i}\right) = \sum_{j=0}^{\infty} \frac{\left(t_{n-i} - t_{n+1}\right)^j}{j!} \frac{d^{(j)}}{dt} x\left(t_{n+1}\right) \tag{5.65}$$

Recall that

$$\begin{aligned} f\left(x_{n-i}, t_{n-i}\right) &= \dot{x}\left(t_{n-i}\right) \\ &= \sum_{j=0}^{\infty} \frac{\left(t_{n-i} - t_{n+1}\right)^j}{j!} \frac{d^{(j+1)}}{dt} x\left(t_{n+1}\right) \end{aligned} \tag{5.66}$$

Solving for $x\left(t_{n+1}\right) - x_{n+1}$ yields

$$\begin{aligned} \varepsilon_T = {} & C_0 x\left(t_n\right) + C_1 x\left(t_{n-1}\right) \\ & + C_2 x\left(t_{n-2}\right) + \ldots + C_k x\left(t_{n-k}\right) + C_{k+1} x\left(t_{n-k-1}\right) + \ldots \end{aligned} \tag{5.67}$$

If the order of this method is k, then the first k coefficients are equal to zero and the local truncation error is given by

$$\varepsilon_T = C_{k+1} h^{k+1} x^{(k+1)}\left(t_{n+1}\right) + O\left(h^{k+2}\right) \tag{5.68}$$

where $O\left(h^{k+2}\right)$ indicates an error on the order of h^{k+2}.

Example 5.5

Find expressions for the local truncation error of the forward Euler, backward Euler, and trapezoidal integration methods.

Solution 5.5

Forward Euler

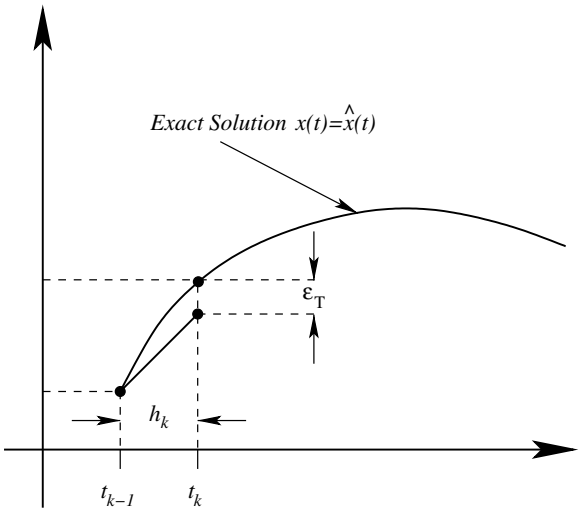

FIGURE 5.4
Graphical depiction of local truncation error

Recall that the expression for the forward Euler integration algorithm is

$$x_{n+1} = x_n + hf\left(x_n, t_n\right)$$

If $x_n = x\left(t_n\right)$ (by the definition of the local truncation error), then

$$x_n = x\left(t_{n+1}\right) - h\dot{x}\left(t_{n+1}\right) + \frac{1}{2!}h^2\ddot{x}\left(t_{n+1}\right) + \ldots \tag{5.69}$$

and

$$f\left(x_n, t_n\right) = \dot{x}\left(t_n\right) = \dot{x}\left(t_{n+1}\right) - h\ddot{x}\left(t_{n+1}\right) + \ldots \tag{5.70}$$

Thus

$$\begin{aligned}
\varepsilon_T &= x\left(t_{n+1}\right) - x_{n+1} && (5.71)\\
&= x\left(t_{n+1}\right) - x_n - hf\left(x_n, t_n\right) && (5.72)\\
&= x\left(t_{n+1}\right) - \left[x\left(t_{n+1}\right) - h\dot{x}\left(t_{n+1}\right) + \frac{1}{2!}h^2\ddot{x}\left(t_{n+1}\right) + \ldots\right]\\
&\quad -h\left[\dot{x}\left(t_{n+1}\right) - h\ddot{x}\left(t_{n+1}\right) + \ldots\right] && (5.73)\\
&= \frac{h^2}{2}\ddot{x}\left(t_{n+1}\right) + O\left(h^3\right) && (5.74)
\end{aligned}$$

Backward Euler
The expression for the backward Euler integration algorithm is

$$x_{n+1} = x_n + hf\left(x_{n+1}, t_{n+1}\right)$$

Following the same approach as the forward Euler method, but using

$$f\left(x_{n+1}, t_{n+1}\right) = \dot{x}\left(t_{n+1}\right) \tag{5.75}$$

then

$$\begin{aligned}
\varepsilon_T &= x\left(t_{n+1}\right) - x_{n+1} && (5.76)\\
&= x\left(t_{n+1}\right) - x_n - hf\left(x_{n+1}, t_{n+1}\right) && (5.77)\\
&= x\left(t_{n+1}\right) - \left[x\left(t_{n+1}\right) - h\dot{x}\left(t_{n+1}\right) + \frac{1}{2!}h^2\ddot{x}\left(t_{n+1}\right) + \ldots\right] \\
&\quad -h\dot{x}\left(t_{n+1}\right) && (5.78)\\
&= -\frac{h^2}{2}\ddot{x}\left(t_{n+1}\right) - O\left(h^3\right) && (5.79)
\end{aligned}$$

Note that the local truncation errors for the forward and backward Euler methods are equal, but opposite in sign. This property is consistent with the results of Example 5.1, shown in Figure 5.2, where the respective errors were identical except in sign.

Trapezoidal
The expression for the second-order trapezoidal integration algorithm is

$$x_{n+1} = x_n + \frac{1}{2}h\left[f\left(x_{n+1}, t_{n+1}\right) + f\left(x_n, t_n\right)\right]$$

Following the same approach as the previous methods using similar substitutions,

$$\begin{aligned}
\varepsilon_T &= x\left(t_{n+1}\right) - x_{n+1} && (5.80)\\
&= x\left(t_{n+1}\right) - x_n - \frac{1}{2}hf\left(x_n, t_n\right) - \frac{1}{2}hf\left(x_{n+1}, t_{n+1}\right) && (5.81)\\
&= x\left(t_{n+1}\right) - \left[x\left(t_{n+1}\right) - h\dot{x}\left(t_{n+1}\right) + \frac{h^2}{2}\ddot{x}\left(t_{n+1}\right) - \frac{h^3}{3!}x^{(3)}\left(t_{n+1}\right) + \ldots\right] \\
&\quad -\frac{h}{2}\left[\dot{x}\left(t_{n+1}\right) - h\ddot{x}\left(t_{n+1}\right) + \frac{h^2}{2}x^{(3)}\left(t_{n+1}\right) + \ldots\right] - \frac{h}{2}\dot{x}\left(t_{n+1}\right) && (5.82)\\
&= \frac{h^3}{6}x^{(3)}\left(t_{n+1}\right) - \frac{h^3}{4}x^{(3)}\left(t_{n+1}\right) + O\left(h^4\right) && (5.83)\\
&= -\frac{1}{12}h^3x^{(3)}\left(t_{n+1}\right) + O\left(h^4\right) && (5.84)
\end{aligned}$$

■

Both the first-order Euler methods had errors on the order of h^2, whereas the second-order method (trapezoidal) had an error on the order of h^3. Both methods are implicit and must be solved iteratively at each time step. Consider the iterative solution of the trapezoidal method repeated here from Equation (5.49):

$$x_{n+1}^{k+1} = x_{n+1}^{k} - \left[I - \frac{h}{2}\frac{\partial f}{\partial x}\right]^{-1}\Bigg|_{x_{n+1}^{k}} \left(x_{n+1}^{k} - x_n - \frac{h}{2}\left[f\left(x_n\right) + f\left(x_{n+1}^{k}\right)\right]\right) \tag{5.85}$$

Similarly, the iterative solution of the backward Euler method is given by

$$x_{n+1}^{k+1} = x_{n+1}^{k} - \left[I - h\frac{\partial f}{\partial x}\right]^{-1}\Bigg|_{x_{n+1}^{k}} \left(x_{n+1}^{k} - x_n - h\left[f\left(x_{n+1}^{k}\right)\right]\right) \tag{5.86}$$

Note that both methods require the same function evaluations and comparable computational effort, yet the trapezoidal method yields a much smaller local truncation error for the same time step size h. For this reason, the trapezoidal method is a more widely used general purpose implicit numerical integration algorithm than the backward Euler method.

For multistep methods, a generalized expression for the local truncation error has been developed [8]. For a multistep method

$$x_{n+1} = \sum_{i=0}^{p} a_i x_{n-i} + h\sum_{i=-1}^{p} b_i f\left(x_{n-i}, t_{n-i}\right) \tag{5.87}$$

which is exact for a polynomial solution of degree less than or equal to k, the local truncation error is given by

$$\varepsilon_T = C_k x^{(k+1)}(\tau) h^{k+1} = O\left(h^{k+1}\right) \tag{5.88}$$

where $-ph \leq \tau \leq h$ and

$$C_k \triangleq \frac{1}{(k+1)!}\left\{(p+1)^{k+1} - \left[\sum_{i=0}^{p-1} a_i\left(p-i\right)^{k+1} + (k+1)\sum_{i=-1}^{p-1} b_i\left(p-i\right)^{k}\right]\right\} \tag{5.89}$$

This expression provides an approach for approximating the local truncation error at each time step as a function of h and x.

5.4 Numerical Stability Analysis

From the previous discussion, it was shown that the choice of integration step size directly impacts the numerical accuracy of the solution. Less obvious is how the choice of step size impacts the numerical stability of the integration method. Numerical stability guarantees that the global truncation error remains bounded. This guarantees that the error introduced at each time step

does not accrue with time, but rather dissipates such that the choice of step size can be made by considering the local truncation error only. To analyze the effect of step size on the numerical stability of integration methods, consider the simple, scalar ODE

$$\dot{x} = f(x) = \lambda x(t) \quad x_0 = x(t_0) \tag{5.90}$$

By inspection, the solution to this equation is

$$x(t) = x_0 e^{(\lambda t)} \tag{5.91}$$

If $\lambda < 0$, then $x(t)$ approaches zero as t goes to infinity. Conversely, if $\lambda > 0$, then $x(t)$ approaches infinity as t goes to infinity. Numerical stability ensures that the global behavior of the estimated system matches that of the actual system. Consider the forward Euler method applied to the scalar system of Equation (5.90):

$$\begin{aligned} x_{n+1} &= x_n + h\lambda x_n \\ &= (1 + h\lambda)x_n \end{aligned}$$

thus

$$\begin{aligned} x_1 &= (1 + h\lambda)x_0 \\ x_2 &= (1 + h\lambda)x_1 = (1 + h\lambda)^2 x_0 \\ &\vdots \\ x_n &= (1 + h\lambda)^n x_0 \end{aligned}$$

If $\lambda < 0$, then $x(t)$ should approach zero as t goes to infinity. This will be achieved only if

$$|1 + h\lambda| < 1 \tag{5.92}$$

Therefore, this system is stable for $\lambda < 0$ only if $h\lambda$ lies within the unit circle centered at $(-1, 0)$, shown in Figure 5.5. Thus the larger the value of λ, the smaller the integration step size must be.

Similarly, consider the backward Euler integration method applied to the same scalar ODE system:

$$\begin{aligned} x_{n+1} &= x_n + h\lambda x_{n+1} \\ &= \frac{x_n}{(1 - h\lambda)} \end{aligned}$$

thus

$$\begin{aligned} x_1 &= \frac{x_0}{(1 - h\lambda)} \\ x_2 &= \frac{x_1}{(1 - h\lambda)} = \frac{x_0}{(1 - h\lambda)^2} \\ &\vdots \\ x_n &= \frac{x_0}{(1 - h\lambda)^n} \end{aligned}$$

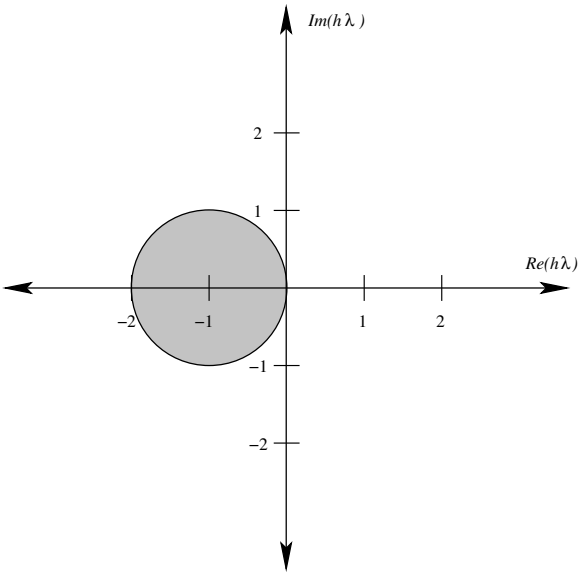

FIGURE 5.5
Region of absolute stability of the forward Euler method

If $\lambda < 0$, then $x(t)$ should approach zero as t goes to infinity. This will be achieved only if

$$|1 - h\lambda| > 1 \tag{5.93}$$

Therefore, this system is stable for $\lambda < 0$ only if $h\lambda$ does not lie within the unit circle centered at (1, 0), shown in Figure 5.6. This implies that, for all $\lambda < 0$, the backward Euler method is numerically stable. Thus, if the ODE system is stable, the integration step size may be chosen arbitrarily large without affecting the numerical stability of the solution. Thus the selection of integration step size will be dependent only on the local truncation error. Note that, if $h\lambda$ is large, then x_n will rapidly approach zero. This characteristic manifests itself as a tendency to overdamp the numerical solution. This characteristic was illustrated in Figure 5.1.

Extending this approach to the general family of multistep methods yields

$$x_{n+1} = \sum_{i=0}^{p} a_i x_{n-i} + h\lambda \sum_{i=-1}^{p} b_i x_{n-i} \tag{5.94}$$

Rearranging the terms of the multistep method gives

$$x_{n+1} = \frac{(a_0 + h\lambda b_0)}{(1 - h\lambda b_{-1})} x_n + \frac{(a_1 + h\lambda b_1)}{(1 - h\lambda b_{-1})} x_{n-1} + \ldots$$

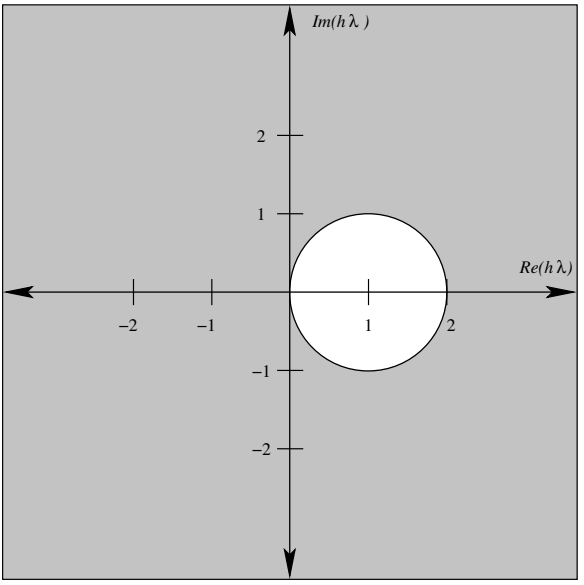

FIGURE 5.6
Region of absolute stability of the backward Euler method

$$+\frac{(a_p + h\lambda b_p)}{(1 - h\lambda b_{-1})} x_{n-p} \tag{5.95}$$

$$= \gamma_0 x_n + \gamma_1 x_{n-1} + \ldots + \gamma_p x_{n-p} \tag{5.96}$$

This relationship specifies the characteristic equation

$$P(z, h\lambda) = z^{p+1} + \gamma_0 z^p + \ldots + \gamma_p = 0 \tag{5.97}$$

where $z_1, z_2, \ldots, z_{p+1}$ are the (complex) roots of Equation (5.97). Therefore,

$$x_{n+1} = \sum_{i=1}^{p+1} C_i z_i^{n+1} \tag{5.98}$$

If $\lambda < 0$, the solution x_{n+1} will go to zero as n goes to infinity only if $|z_j| < 1$ for all $j = 1, 2, \ldots, p+1$. Thus a multistep method is said to be *absolutely stable* for a given value of $h\lambda$ if the roots of $P(z, h\lambda) = 0$ satisfy $|z_i| < 1$ for $i = 1, \ldots, k$. Absolute stability implies that the global error decreases with increasing n. The region of absolute stability is defined to be the region in the complex $h\lambda$ plane where the roots of $P(z, h\lambda) = 0$ satisfy $|z_i| < 1$ for $i = 1, \ldots, k$. Let

$$P(z, h\lambda) = P_a(z) + h\lambda P_b(z) = 0$$

where

$$P_a(z) \triangleq z^{p+1} - a_0 z^p - a_1 z^{p-1} - \ldots - a_p$$
$$P_b(z) \triangleq b_{-1} z^{p+1} + b_0 z^p + b_1 z^{p-1} + \ldots + b_p$$

then

$$h\lambda = -\frac{P_a(z)}{P_b(z)} \tag{5.99}$$

Since z is a complex number, it can also be represented as

$$z = e^{(j\theta)}$$

The boundary of the region can be mapped by plotting $h\lambda$ in the complex plane as θ varies from 0 through 2π where

$$h\lambda(\theta) = -\frac{e^{j(p+1)\theta} - a_0 e^{jp\theta} - a_1 e^{j(p-1)\theta} - \ldots - a_{p-1} e^{j\theta} - a_p}{b_{-1} e^{j(p+1)\theta} + b_0 e^{jp\theta} + b_1 e^{j(p-1)\theta} + \ldots + b_{p-1} e^{j\theta} + b_p} \tag{5.100}$$

Example 5.6
Plot the regions of absolute stability of Gear's third-order and the Adams third-order (both implicit and explicit) methods.

Solution 5.6

Gear's

The region of stability of a Gear's method can be found by plotting $h\lambda$ in the complex plane from Equation (5.100) by setting $p = k - 1$ and $b_0, b_1, \ldots = 0$.

$$h\lambda(\theta) = \frac{e^{jk\theta} - a_0 e^{j(k-1)\theta} - \ldots - a_{k-1}}{b_{-1} e^{jk\theta}} \tag{5.101}$$

Substituting in the coefficients for the third-order Gear's method yields the following expression in θ:

$$h\lambda(\theta) = \frac{e^{j3\theta} - \frac{18}{11} e^{j2\theta} + \frac{9}{11} e^{j\theta} - \frac{2}{11}}{\frac{6}{11} e^{j3\theta}} \tag{5.102}$$

By varying θ from zero to 2π, the region of absolute stability of Gear's third-order method is shown as the shaded region of Figure 5.7.

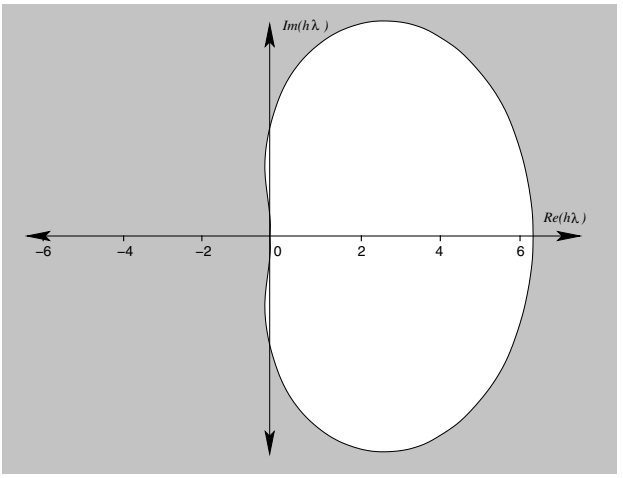

FIGURE 5.7
Region of absolute stability for Gear's third-order method

Adams–Moulton

The region of absolute stability of the Adams–Moulton methods can be developed from Equation (5.100) by setting $p = k - 1$, and $a_1, a_2, \ldots = 0$:

$$h\lambda(\theta) = \frac{e^{jk\theta} - a_0 e^{j(k-1)\theta}}{b_{-1}e^{jk\theta} + b_0 e^{j(k-1)\theta} + b_1 e^{j(k-2)\theta} + \ldots + b_{k-2}e^{j\theta}} \tag{5.103}$$

After substituting in the third-order coefficients, the expression for the region of absolute stability as a function of θ is given by

$$h\lambda(\theta) = \frac{e^{j3\theta} - e^{j2\theta}}{\frac{5}{12}e^{j3\theta} + \frac{8}{12}e^{j2\theta} - \frac{1}{12}e^{j\theta}} \tag{5.104}$$

The region of absolute stability of the Adams–Moulton third-order method is shown as the shaded region of Figure 5.8.

Adams–Bashforth

The stability of the family of Adams–Bashforth methods can be derived from Equation (5.100) by setting $p = k - 1$, $b_{-1} = 0$ and $a_1, a_2, \ldots, = 0$:

$$h\lambda(\theta) = \frac{e^{jk\theta} - a_0 e^{j(k-1)\theta}}{b_0 e^{j(k-1)\theta} + b_1 e^{j(k-2)\theta} + \ldots + b_{k-1}} \tag{5.105}$$

$$h\lambda(\theta) = \frac{e^{j3\theta} - e^{j2\theta}}{\frac{23}{12}e^{j2\theta} - \frac{16}{12}e^{j\theta} + \frac{5}{12}} \tag{5.106}$$

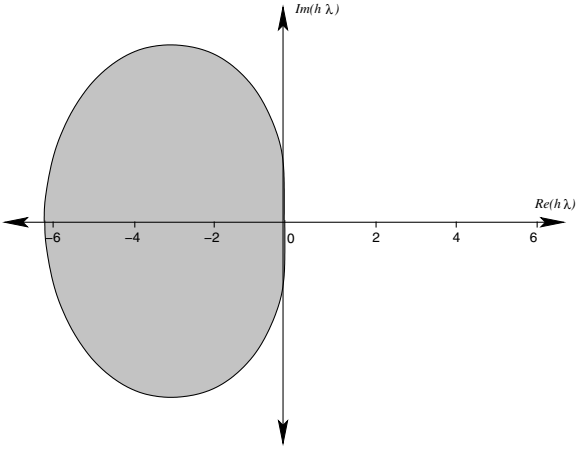

FIGURE 5.8
Region of absolute stability for the Adams–Moulton third-order method

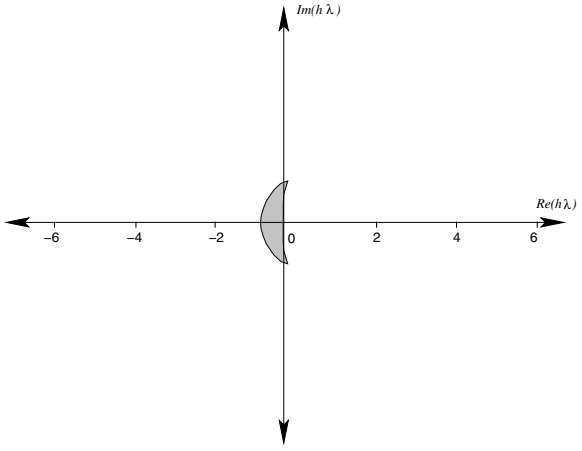

FIGURE 5.9
Region of absolute stability for the Adams–Bashforth third-order method

The region of absolute stability of the Adams–Bashforth method is shown as the shaded region in Figure 5.9.

This example illustrates one of the primary differences between implicit and explicit methods. For the same order, the two implicit methods (Gear's and Adams–Moulton) exhibit much larger regions of absolute stability than does the explicit method (Adams–Bashforth). Gear's region of absolute stability contains nearly the entire left half $h\lambda$ plane; thus, for any stable dynamic system, the step size can be chosen as large as desired without any consideration for numerical stability. Even the Adams–Moulton region of absolute stability is quite large compared to the Adams–Bashforth. Typically, the region of stability of explicit methods is much smaller than the region of absolute stability of corresponding order implicit methods. For this reason, implicit methods are frequently used in commercial integration packages so that the integration step size can be chosen based purely on the local truncation error criteria. The region of absolute stability shrinks as the order of the method increases, whereas the accuracy increases. This is a trade-off between accuracy, stability, and numerical efficiency in integration algorithms. ■

5.5 Stiff Systems

Gear's methods were originally developed for the solution of systems of *stiff* ordinary differential equations. Stiff systems are systems that exhibit a wide range of time varying dynamics from "very fast" to "very slow." A stiff linear system exhibits eigenvalues that span several orders of magnitude. A nonlinear system is stiff if its associated Jacobian matrix exhibits widely separated eigenvalues when evaluated at the operating points of interest. For efficient and accurate solutions of stiff differential equations, it is desirable for a multistep method to be "stiffly stable." An appropriate integration algorithm will allow the step size to be varied over a wide range of values and yet will remain numerically stable. A stiffly stable method exhibits the three stability regions shown in Figure 5.10 such that:

1. Region I is a region of absolute stability
2. Region II is a region of accuracy and stability
3. Region III is a region of accuracy and relative stability

Only during the initial period of the solution of the ODE do the large negative eigenvalues significantly impact the solution, yet they must be accounted for throughout the whole solution. Large negative eigenvalues ($\lambda < 0$) will decay rapidly by a factor of $1/e$ in time $1/\lambda$. If $h\lambda = \gamma + j\beta$, then the change in magnitude in one step is e^{γ}. If $\gamma \leq \delta \leq 0$, where δ defines the interface

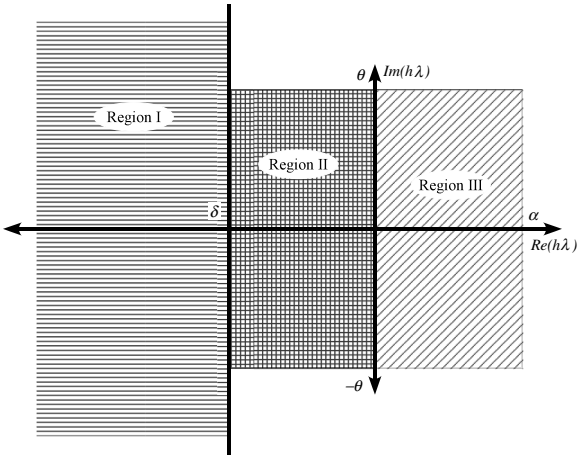

FIGURE 5.10
Regions required for stiff stability

between Regions I and II, then the component is reduced by at least e^{δ} in one step. After a finite number of steps, the impact of the fast components is negligible and their numerical accuracy is unimportant. Therefore, the integration method is required to be absolutely stable in Region I.

Around the origin, numerical accuracy becomes more significant and relative or absolute stability is required. A region of *relative stability* consists of those values of $h\lambda$ for which the extraneous eigenvalues of the characteristic polynomial of Equation (5.97) are less in magnitude than the principal eigenvalue. The principal eigenvalue is the eigenvalue which governs the system response most closely. If the method is relatively stable in Region III, then the system response will be dominated by the principal eigenvalue in that region. If $\gamma > \alpha > 0$, one component of the system response is increasing by at least e^{α} one step. This increase must be limited by choosing the step sizes small enough to track this change.

If $\|\beta\| > \theta$, there are at least $\theta/2\pi$ complete cycles of oscillation in one step. Except in Region I, where the response is rapidly decaying and where $\gamma > \alpha$ is not used, the oscillatory responses must be captured. In practice, it is customary to have eight or more time points per cycle (to accurately capture the magnitude and frequency of the oscillation); thus θ is chosen to be bounded by $\pi/4$ in Region II.

Examination of the family of Adams–Bashforth methods shows that they all fail to satisfy the criteria to be stiffly stable and are not suitable for integrating stiff systems. Only the first- and second-order Adams–Moulton (backward Euler and trapezoidal, respectively) satisfy the stiffly stable criteria. Gear's algorithms, on the other hand, were developed specifically to address stiff

system integration [15]. Gear's algorithms up to order six satisfy the stiff properties with the following choice of δ [8]:

Order	δ
1	0
2	0
3	0.1
4	0.7
5	2.4
6	6.1

Example 5.7
Compare the application of the third-order Adams–Bashforth, Adams–Moulton, and Gear's method, to the integration of the following system:

$$\dot{x}_1 = 48x_1 + 98x_2 \quad x_1(0) = 1 \tag{5.107}$$
$$\dot{x}_2 = -49x_1 - 99x_2 \quad x_2(0) = 0 \tag{5.108}$$

Solution 5.7 The exact solution to Example 5.7 is

$$x_1(t) = 2e^{-t} - e^{-50t} \tag{5.109}$$
$$x_2(t) = -e^{-t} + e^{-50t} \tag{5.110}$$

This solution is shown in Figure 5.11. Both states contain both fast and slow components, with the fast component dominating the initial response and the slow component dominating the longer-term dynamics. Since Gear's, Adams–Bashforth, and Adams–Moulton methods are multistep methods, each method is initialized using the absolutely stable trapezoidal method for the first two to three steps using a small step size.

The Adams–Bashforth algorithm with a time step of 0.0111 seconds is shown in Figure 5.12. Note that, even with a small step size of 0.0111 seconds, the inherent error in the integration algorithm eventually causes the system response to exhibit numerical instabilities. The step size can be decreased to increase the stability properties, but this requires more time steps in the integration window ($t \in [0, 2]$) than is computationally necessary.

The Adams–Moulton response to the stiff system for an integration step size of 0.15 seconds is shown in Figure 5.13. Although a much larger time step can be used for integration as compared to the Adams–Bashforth algorithm, the Adams–Moulton algorithm does not exhibit numerical absolute stability. For an integration step size of $h = 0.15$ seconds, the Adams–Moulton algorithm exhibits numerical instability. Note that the solution is oscillating with growing magnitude around the exact solution.

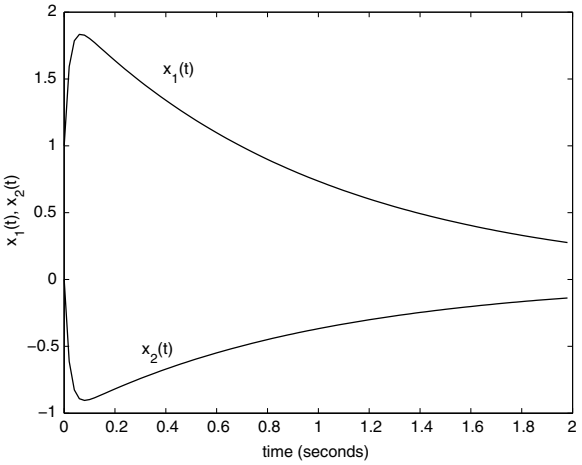

FIGURE 5.11
Stiff system response

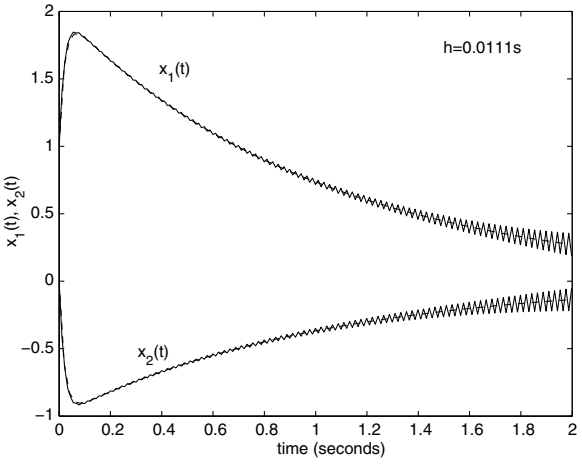

FIGURE 5.12
Adams–Bashforth stiff system response with $h = 0.0111$ s step size

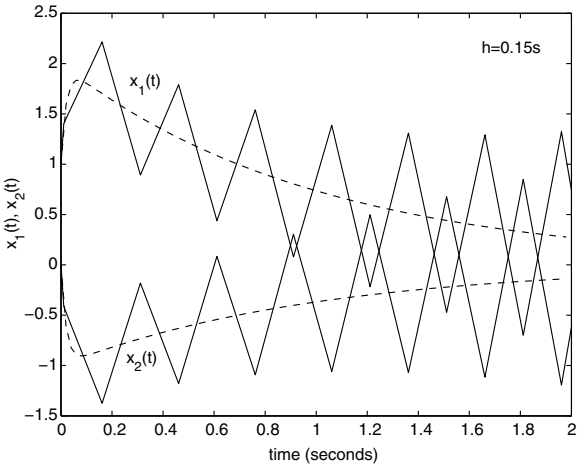

FIGURE 5.13
Adams–Moulton stiff system response with $h = 0.15$ s step size

Figure 5.14 shows the Gear's method numerical solution to the stiff system using an integration step size of 0.15 s, which is the same step size as used in the Adams–Moulton response of Figure 5.13. Gear's method is numerically stable since the global error decreases with increasing time.

The comparison of the three integration methods supports the necessity of using integration algorithms developed specifically for stiff systems. Even though both the third-order Adams–Bashforth and Adams–Moulton have regions of absolute stability, these regions are not sufficient to ensure accurate stiff system integration. ■

5.6 Step Size Selection

For computational efficiency, it is desirable to choose the largest integration step size possible while satisfying a predetermined level of accuracy. The level of accuracy can be maintained by constant vigilance of the local truncation error of the method if the chosen method is numerically stable. If the function $x(t)$ is varying rapidly, the step size should be chosen small enough to capture the significant dynamics. Conversely, if the function $x(t)$ is not varying significantly (nearly linear) over a finite time interval, then the integration time steps can be chosen quite large over that interval while still maintaining numerical accuracy. The true challenge to a numerical integration algorithm is when the dynamic response of $x(t)$ has intervals of both rapid variance and latency. In this case, it is desirable for the integration step size to have

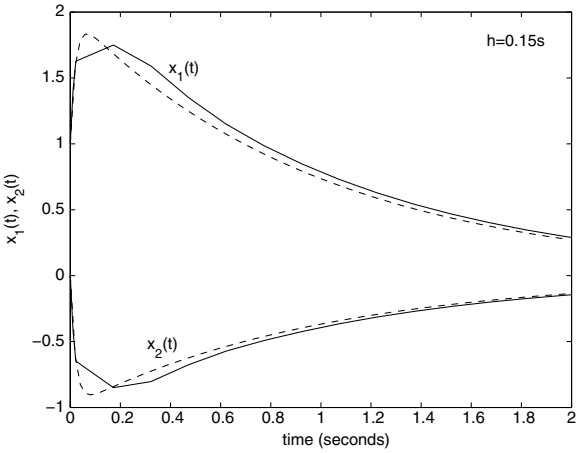

FIGURE 5.14
Gear's stiff system response with $h = 0.15$ s step size

the ability to increase or decrease throughout the simulation interval. This can be accomplished by choosing the integration step size based on the local truncation error bounds.

Consider the trapezoidal method absolute local truncation error:

$$\varepsilon_T = \frac{1}{12} h^3 x^{(3)}(\tau) \tag{5.111}$$

The local truncation error is dependent on the integration step size h and the third derivative of the function $x^{(3)}(\tau)$. If the local truncation error is chosen to be in the interval

$$B_L \leq \varepsilon \leq B_U \tag{5.112}$$

where B_L and B_U represent the prespecified lower and upper bounds, respectively, then the integration step size can be bounded by

$$h \leq \sqrt[3]{\frac{12 B_U}{x^{(3)}(\tau)}} \tag{5.113}$$

If $x(t)$ is rapidly varying, the $x^{(3)}(\tau)$ will be large and h must be chosen small to satisfy $\varepsilon \leq B_U$, whereas, if $x(t)$ is not varying rapidly, then $x^{(3)}(\tau)$ will be small and h can be chosen relatively large while still satisfying $\varepsilon \leq B_U$. This leads to the following procedure for calculating the integration step size:

Integration Step Size Selection

Attempt an integration step size h_{n+1} to calculate x_{n+1} from $x_n, x_{n-1}, \ldots$

1. Using x_{n+1}, calculate the local truncation error ε_T. If the size of x is greater than 1, then
$$\varepsilon_T = \max_i \left(|\varepsilon_{T,i}|\right)$$
2. If $B_L \leq \varepsilon_T \leq B_U$, then PASS, accept $h_{n+1}, h_{next} = h_{n+1}$, and continue.
3. If $\varepsilon > B_U$, then FAIL (h_{n+1} is too large), set $h_{n+1} = \alpha h_{n+1}$, repeat integration for x_{n+1}.
4. If $\varepsilon \leq B_U$, then PASS, accept h_{n+1}, set $h_{next} = \alpha h_{n+1}$, and continue.

where
$$\alpha = \left[\frac{B_{avg}}{\varepsilon_T}\right]^{\frac{1}{k+1}} \tag{5.114}$$
where $B_L \leq B_{avg} \leq B_U$ and k is the degree of the method.

Commercial integration packages may implement an algorithm that is slightly different. In these packages, if the local truncation error is smaller than the lower bound, then the attempted integration step size also FAILS, and the integration step is reattempted with a larger integration step size. Once again, there is a trade off between the time spent in recalculating x_{n+1} with a larger step size and the additional computational effort acquired by simply accepting the current value and continuing on.

The difficulty in implementing this step size selection approach is the calculation of the higher-order derivatives of $x(t)$. Since $x(t)$ is not known analytically, the derivatives must be calculated numerically. One common approach is to use *difference methods* to approximate the derivatives. The $(k+1)$st derivative to $x(\tau)$ is approximated by
$$x^{(k+1)}(\tau) \approx (k+1)! \nabla_{k+1} x_{n+1} \tag{5.115}$$
where $\nabla_{k+1} x_{n+1}$ is found recursively:
$$\nabla_1 x_{n+1} = \frac{x_{n+1} - x_n}{t_{n+1} - t_n} \tag{5.116}$$
$$\nabla_1 x_n = \frac{x_n - x_{n-1}}{t_n - t_{n-1}} \tag{5.117}$$
$$\vdots \tag{5.118}$$
$$\nabla_2 x_{n+1} = \frac{\nabla_1 x_{n+1} - \nabla_1 x_n}{t_{n+1} - t_{n-1}} \tag{5.119}$$
$$\nabla_2 x_n = \frac{\nabla_1 x_n - \nabla_1 x_{n-1}}{t_n - t_{n-2}} \tag{5.120}$$

$$\vdots \tag{5.121}$$

$$\nabla_{k+1} x_{n+1} = \frac{\nabla_k x_{n+1} - \nabla_k x_n}{t_{n+1} - t_{n-k}} \tag{5.122}$$

Example 5.8
Find an expression for step size selection for the trapezoidal integration method.

Solution 5.8
The LTE for the trapezoidal method is

$$|\varepsilon_T| = \frac{1}{12} h^3 x^{(3)}(\tau) \tag{5.123}$$

$$= \frac{1}{2} h^3 \nabla_3 x_{n+1} \tag{5.124}$$

where

$$\nabla_3 x_{n+1} = \frac{\nabla_2 x_{n+1} - \nabla_2 x_n}{t_{n+1} - t_{n-2}} \tag{5.125}$$

$$= \frac{\nabla_2 x_{n+1} - \nabla_2 x_n}{h_{n+1} + h_n + h_{n-1}} \tag{5.126}$$

$$= \frac{1}{h_{n+1} + h_n + h_{n-1}} \left\{ \frac{1}{h_{n+1} + h_n} \left[\frac{x_{n+1} - x_n}{h_{n+1}} - \frac{x_n - x_{n-1}}{h_n} \right] - \frac{1}{h_n + h_{n-1}} \left[\frac{x_n - x_{n-1}}{h_n} - \frac{x_{n-1} - x_{n-2}}{h_{n-1}} \right] \right\} \tag{5.127}$$

■

Example 5.9
Numerically solve

$$\dot{x}(t) = \begin{bmatrix} -1 & 1 & 1 \\ 1 & -25 & 98 \\ 1 & -49 & -60 \end{bmatrix} x, \quad \text{with } x(0) = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$

using a variable integration step size with the following parameters:

$$h_0 = 0.01 \quad B_L = 10^{-6}$$
$$B_U = 10^{-4} \quad B_{avg} = 10^{-5}$$

Solution 5.9 This is a stiff system with the following eigenvalues:

$$-0.9788$$
$$-42.5106 + j67.0420$$
$$-42.5106 - j67.0420$$

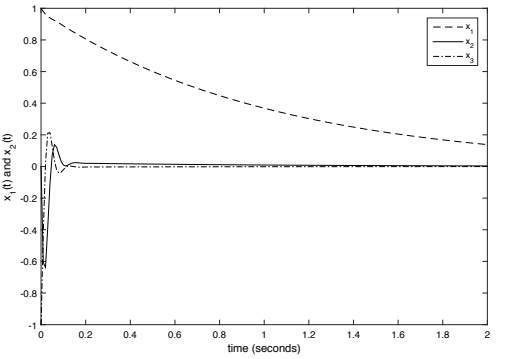

FIGURE 5.15
Numerical solution for Example 5.9 with constant step size $h = 0.01$ s

A constant step size integration yields the numerical solution in Figure 5.15 with local truncation error shown in Figure 5.16.

Note that the local truncation error continually decreases throughout the simulation. The LTE is initially dominated by the LTE of x_2 and x_3 due to the influence of the oscillatory response resulting from the complex eigenvalues. As these states decay as a result of the large negative real part of the complex eigenvalues, the LTE of the real eigenvalue x_1 begins to dominate. However, the LTE decreases throughout the simulation interval as the rate of change of all states decreases.

After implementing the step size selection process based on local truncation error, the variable step size and corresponding LTE are shown in Figures 5.17 and 5.18. Note that using the variable step size, the LTE is held between the specified upper and lower bounds. This is accomplished by increasing the step size when the LTE becomes too small in response to the decay of the response rates of the underlying states. Figure 5.19 shows the numerical results of the states using the variable step size. Note that the simulation results differ very little from the original results of Figure 5.15 because the error is not allowed to exceed the upper bound. ■

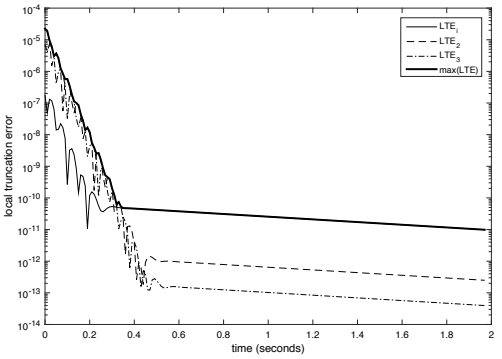

FIGURE 5.16
LTE solution for Example 5.9 with constant step size $h = 0.01$ s

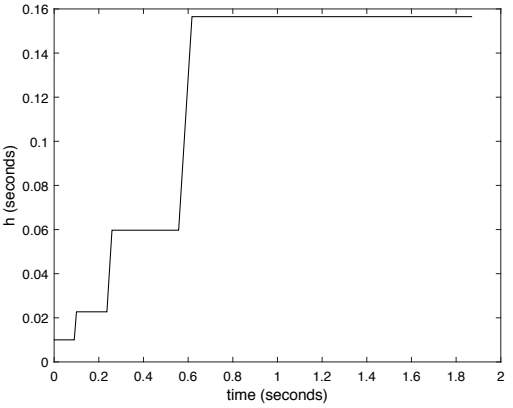

FIGURE 5.17
Step sizes as varied through the step size selection process

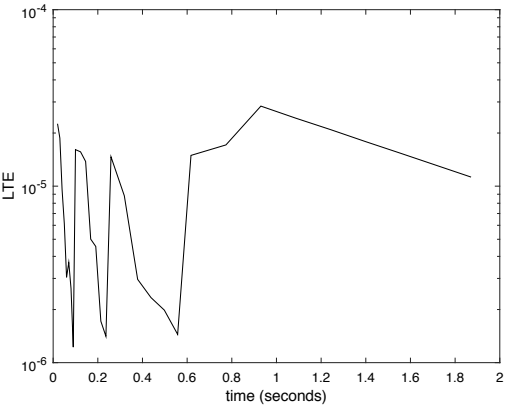

FIGURE 5.18
LTE solution for Example 5.9 with variable step size

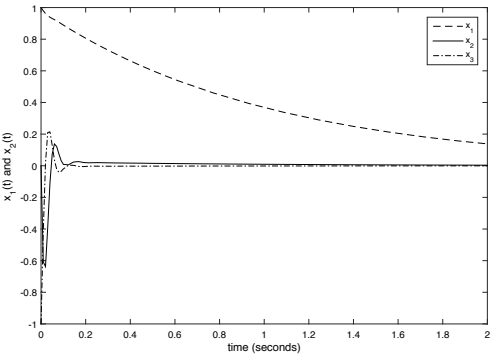

FIGURE 5.19
Numerical solution for Example 5.9 with variable step sizes

5.7 Differential-Algebraic Equations

Many classes of systems can be generically written in the form

$$F\left(t, y(t), y'(t)\right) = 0 \tag{5.128}$$

where F and $y \in R^m$. In some cases, Equation (5.128) can be rewritten as

$$F\left(t, x(t), \dot{x}(t), y(t)\right) = 0 \tag{5.129}$$

$$g\left(t, x(t), y(t)\right) = 0 \tag{5.130}$$

In this form, the system of equations is typically known as a system of *differential-algebraic equations*, or DAEs [7]. This form of DAEs may be considered to be a system of differential equations (Equation (5.129)) constrained to an algebraic manifold (Equation (5.130)). Often the set of equations (5.130) are invertible (i.e., $y(t)$ can be obtained from $y(t) = g^{-1}\left(t, x(t)\right)$) and Equation (5.129) can be rewritten as an ordinary differential equation:

$$F\left(t, x(t), \dot{x}(t), g^{-1}\left(t, x(t)\right)\right) = f\left(t, x(t), \dot{x}(t)\right) = 0 \tag{5.131}$$

Although the set of ODEs may be conceptually simpler to solve, there are usually several compelling reasons to leave the system in its original DAE form. Many DAE models are derived from physical problems in which each variable has individual characteristics and physical significance. Converting the DAE into an ODE may result in a loss of physical information in the solution. Furthermore, it may be more computationally expensive to obtain the solution of the system of ODEs since inherent sparsity may have been lost. Solving the original DAE directly provides greater information regarding the behavior of the system.

A special case of DAEs is the semi-explicit DAE:

$$\dot{x} = f(x, y, t) \quad x \in R^n \tag{5.132}$$

$$0 = g(x, y, t) \quad y \in R^m \tag{5.133}$$

where y has the same dimension as g. DAE systems that can be written in semi-explicit form are often referred to as *index 1* DAE systems [7]. The first concerted effort to solve semi-explicit DAEs was proposed in [16] and later refined in [17], and consisted of replacing $\dot{x}(t)$ by a k-step backwards difference formula (BDF) approximation

$$\dot{x}(t) \approx \frac{\rho_n x_n}{h_n} = \frac{1}{h_n} \sum_{i=0}^{k} \alpha_i x_{n-i} \tag{5.134}$$

and then solving the resulting equations

$$\rho_n x_n = h_n f\left(x_n, y_n, t_n\right) \tag{5.135}$$

$$0 = g\left(x_n, y_n, t_n\right) \tag{5.136}$$

for approximations to x_n and y_n.

Various other numerical integration techniques have been studied for application to DAE systems. Variable step size/fixed formula code has been proposed to solve the system of DAEs [48]. Specifically, a classic fourth-order Runge–Kutta method was used to solve for x and a third-order BDF to solve for y.

In general, however, many DAE systems may be solved by any multistep numerical integration method which is convergent when applied to ODEs [22]. The application of multistep integration methods to a DAE system is straightforward. A general multistep method of the form of Equation (5.8) applied to Equations (5.132) and (5.133) yields

$$x_{n+1} = \sum_{i=0}^{p} a_i x_{n-i} + h \sum_{i=-1}^{p} b_i f\left(x_{n-i}, y_{n-i}, t_{n-i}\right) \tag{5.137}$$

$$0 = g\left(x_{n+1}, y_{n+1}, t_{n+1}\right) \tag{5.138}$$

Multistep methods applied to semi-explicit index 1 DAEs are stable and convergent to the same order of accuracy for the DAE as for standard nonstiff ODEs [7].

There are two basic approaches to solving Equations (5.137) and (5.138). One is the *iterative* approach in which Equation (5.138) is solved for y_{n+1}, which is then substituted into Equation (5.138). This equation is then solved for x_{n+1}. This process is repeated at t_{n+1} until the values for x_{n+1} and y_{n+1} converge, and then the solution is advanced to the next time step.

Another approach is the *simultaneous* approach, in which both sets of equations are solved simultaneously for x_{n+1} and y_{n+1} using a nonlinear solver such as the Newton–Raphson method. In this case, the system of equations is recast as

$$\begin{aligned} 0 &= F\left(x_{n+1}, y_{n+1}, t_{n+1}\right) \\ &= x_{n+1} - \sum_{i=0}^{p} a_i x_{n-i} - h \sum_{i=-1}^{p} b_i f\left(x_{n-i}, y_{n-i}, t_{n-i}\right) \end{aligned} \tag{5.139}$$

$$0 = g\left(x_{n+1}, y_{n+1}, t_{n+1}\right) \tag{5.140}$$

and the Newton–Raphson Jacobian becomes

$$J_{xy} = \begin{bmatrix} I_n - hb_{-1}\frac{\partial f}{\partial x} & -hb_{-1}\frac{\partial f}{\partial y} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} \end{bmatrix} \tag{5.141}$$

Solving the full $n + m$ system of equations yields both x_{n+1} and y_{n+1} simultaneously.

The solvability of a DAE system implies that there exists a unique solution for sufficiently different inputs and *consistent* initial conditions, unlike the ODE system, which has a unique solution for *arbitrary* initial conditions. For

a DAE system, there exists only one set of initial conditions to the nonstate (algebraic) variables. The initial values that are consistent with the system input are called *admissible* initial values. The admissible initial values are those which satisfy

$$y_0 = g^{-1}(x_0, t_0) \tag{5.142}$$

5.8 Power System Applications

Power systems in general are considered to be large scale, often involving several hundred equations to describe the behavior of an interconnected power system during and following a fault on the system. As power system operations become increasingly complex, it becomes necessary to be able to perform analyses of voltage conditions and system stability. A medium-sized power company serving a mixed urban and rural population of two to three million people operates a network that may typically contain hundreds of buses and thousands of transmission lines, excluding the distribution system [13]. Under certain assumptions, such as instantaneous transmission lines, interconnected power systems are often modeled in DAE form with over 1000 differential equations and 10,000 algebraic constraint equations. One traditional approach to solving large-scale systems of this type has been to replace the full system model with a reduced-order state space model.

5.8.1 Transient Stability Analysis

The "classical model" of a synchronous machine is often used to study the transient stability of a power system during the period of time in which the system dynamics depend largely on the stored kinetic energy in the rotating masses. This is usually on the order of a second or two. The classical model is derived under several simplifying assumptions [2]:

1. Mechanical power input, P_m, is constant.

2. Damping is negligible.

3. The constant voltage behind the transient reactance model for the synchronous machines is valid.

4. The rotor angle of the machine coincides with the voltage behind the transient reactance angle.

5. Loads are represented as constant impedances.

The equations of motion are given by

$$\dot{\omega}_i = \frac{1}{M_i}\left(P_{m_i} - E_i^2 G_{ii} - E_i \sum_{j \neq i}^{n} E_j \left(B_{ij}\sin\delta_{ij} + G_{ij}\cos\delta_{ij}\right)\right) \quad (5.143)$$

$$\dot{\delta}_i = \omega_i - \omega_s \quad i = 1, \ldots, n \quad (5.144)$$

where n is the number of machines, ω_s is the synchronous angular frequency, $\delta_{ij} = \delta_i - \delta_j$, $M_i = \frac{2H_i}{\omega_s}$, and H_i is the inertia constant in seconds. B_{ij} and G_{ij} are elements of the reduced admittance matrix Y at the internal nodes of the machine. The loads are modeled as constant impedances, which are then absorbed into the admittance matrix. The classical model is appropriate for frequency studies that result from faults on the transmission system for the first or second swing of the rotor angle. The procedure for setting up a transient stability analysis is given below.

Transient Stability Analysis

1. Perform a load flow analysis to obtain system voltages, angles, active and reactive power generation.

2. For each generator $i, \ldots, n$, in the system, calculate the internal voltage and initial rotor angle $E\angle\delta_0$:

$$I^*_{gen} = \frac{(P_{gen} + jQ_{gen})}{V_T\angle\theta_T} \quad (5.145)$$

$$E\angle\delta_0 = jx'_d I_{gen} + V_T\angle\theta_T \quad (5.146)$$

   where $P_{gen}+jQ_{gen}$ are the generated active and reactive power obtained from the power flow solution and $V_T\angle\theta_T$ is the generator terminal voltage.

3. For each load $1, \ldots, m$ in the system, convert the active and reactive power loads to admittances:

$$Y_L = G_L + jB_L = \frac{I_L}{V_L\angle\theta_L} \quad (5.147)$$

$$= \frac{S^*_L}{V_L^2} \quad (5.148)$$

$$= \frac{P_L - jQ_L}{V_L^2} \quad (5.149)$$

   Add the shunt admittance Y_L to the corresponding diagonal of the admittance matrix.

4. For each generator in the system, augment the admittance matrix by adding an internal bus connected to the terminal bus with the transient reactance x'_d, as shown in Figure 5.20.

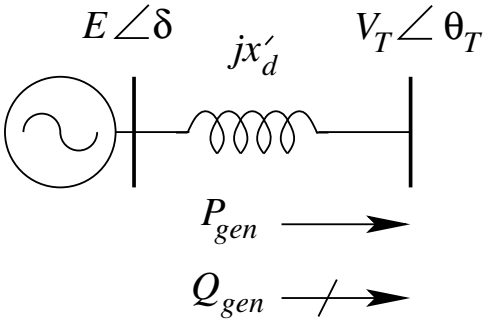

FIGURE 5.20
Voltage behind transient reactance

Then let

$$Y_{nn} = \begin{bmatrix} jx'_{d_1} & 0 & 0 & \dots & 0 \\ 0 & jx'_{d_2} & 0 & \dots & 0 \\ 0 & 0 & jx'_{d_3} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & jx'_{d_n} \end{bmatrix}$$

and

$$Y_{nm} = \left[[-Y_{nn}] \, [0_{n\times m}] \right]$$
$$Y_{mn} = Y_{nm}^T$$

where $[0_{n\times m}]$ is an $(n \times m)$ matrix of zeros.

Further, let

$$Y_{mm} = \left[Y_{original} + \begin{bmatrix} Y_{nn} & 0_{n\times(m-n)} \\ 0_{(m-n)\times n} & 0_{(m-n)\times(m-n)} \end{bmatrix} \right]$$

This matrix layout assumes that the systems buses have been ordered such that the generators are numbered $1, \dots, n$, with the remaining load buses numbered $n+1, \dots, m$.

5. The reduced admittance matrix Y_{red} is found by

$$Y_{red} = \left[Y_{nn} - Y_{nm} Y_{mm}^{-1} Y_{mn} \right] \tag{5.150}$$
$$= G_{red} + jB_{red} \tag{5.151}$$

The reduced admittance matrix is now $n \times n$ where n is the number of generators, whereas the original admittance matrix was $m \times m$.

6. Repeat steps 4 and 5 to calculate the fault-on reduced admittance matrix and the postfault reduced admittance matrix (if different from the prefault reduced admittance matrix).

7. For $0 < t \leq t_{apply}$, integrate the transient stability equations (5.143) and (5.144) with the integration method of choice using the prefault reduced admittance matrix, where t_{apply} is the time the system fault is applied. In many applications, $t_{apply} = 0$.

8. For $t_{apply} < t \leq t_{clear}$, integrate the transient stability equations (5.143) and (5.144) with the integration method of choice using the fault-on reduced admittance matrix, where t_{clear} is the time the system fault is cleared.

9. For $t_{clear} < t \leq t_{max}$, integrate the transient stability equations (5.143) and (5.144) with the integration method of choice using the postfault reduced admittance matrix, where t_{max} is end of the simulation interval.

At the end of the simulation, the state variables (δ_i, ω_i) for each of the generators may be plotted against time. The rotor angle responses are in radians and may be converted to degrees if desired. The rotor angular frequency is in radians per second and may be converted to hertz (cycles per second) if desired. These waveforms may then be analyzed to determine whether or not system stability is maintained. If the system responses diverge from one another or exhibit growing oscillations, then the system is most probably unstable.

Example 5.10

For the three-machine, nine-bus system shown in Figure 5.21, a solid three-phase fault occurs at bus 8 at 0.1 seconds. The fault is cleared by opening line 8-9. Determine whether or not the system will remain stable for a clearing time of 0.12 seconds after the application of the fault.

Solution 5.10 Following the procedure for transient stability analysis outlined previously, the first step is to perform a power flow analysis of the system. The line and bus data are given in Figure 5.21. The load flow results are

i	V	θ	P_{gen}	Q_{gen}
1	1.0400	0	0.7164	0.2685
2	1.0253	9.2715	1.6300	0.0669
3	1.0254	4.6587	0.8500	–0.1080
4	1.0259	–2.2165		
5	1.0128	–3.6873		
6	1.0327	1.9625		
7	1.0162	0.7242		
8	1.0261	3.7147		
9	0.9958	–3.9885		

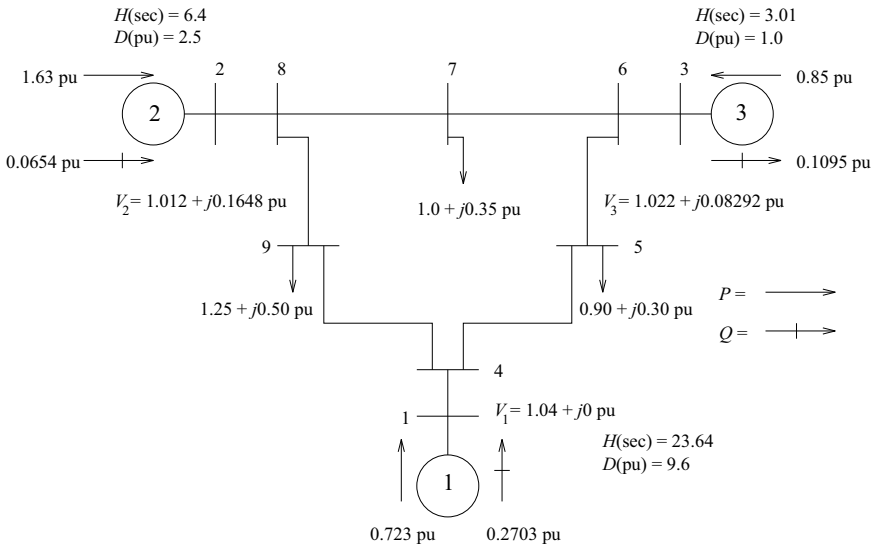

FIGURE 5.21
Three-machine, nine-bus system of Example 5.10

where the bus angles are given in degrees and all other data are in per unit. The admittance matrix for this system is given in Figure 5.22.

The generator data for this system are

i	x'_d	H
1	0.0608	23.64
2	0.1198	6.40
3	0.1813	3.01

The internal voltages and rotor angles for each generator are computed using the generated active and reactive powers, voltage magnitudes, and angles as:

$$I_1^* = \frac{(0.7164 + j0.2685)}{1.0400\angle 0^\circ} = 0.6888 + j0.2582$$
$$E_1\angle\delta_1 = (j0.0608)(0.6888 - j0.2582) + 1.0400\angle 0^\circ = 1.0565\angle 2.2718^\circ$$

$$I_2^* = \frac{(1.6300 + j0.0669)}{1.0253\angle 9.2715^\circ} = 1.5795 - j0.1918$$
$$E_2\angle\delta_2 = (j0.1198)(1.5795 + j0.1918) + 1.0253\angle 9.2715^\circ = 1.0505\angle 19.7162^\circ$$

$$I_3^* = \frac{(0.8500 - j0.1080)}{1.0254\angle 4.6587^\circ} = 0.8177 - j0.1723$$

$$
[Y_{bus}] = \begin{bmatrix}
17.3611\angle -90.00^{\circ} & 0 & 0 & 17.3611\angle 90.00^{\circ} & 0 & 0 & 0 & 0 & 0 \\
0 & 16.0000\angle -90.00^{\circ} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 17.0648\angle -90.00^{\circ} & 0 & 0 & 17.0648\angle 90.00^{\circ} & 0 & 0 & 0 \\
17.3611\angle 90.00^{\circ} & 0 & 0 & 39.4478\angle -85.19^{\circ} & 10.6886\angle 100.47^{\circ} & 0 & 0 & 0 & 11.6841\angle 96.71^{\circ} \\
0 & 0 & 0 & 33.8085\angle -90.00^{\circ} & 16.1657\angle -78.50^{\circ} & 5.7334\angle 102.92^{\circ} & 0 & 0 & 0 \\
0 & 0 & 17.0648\angle 90.00^{\circ} & 0 & 5.7334\angle 102.92^{\circ} & 32.2461\angle -85.67^{\circ} & 9.8522\angle 96.73^{\circ} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 9.8522\angle 96.73^{\circ} & 23.4676\angle -83.22^{\circ} & 13.7931\angle 96.73^{\circ} & 0 \\
0 & 16.0000\angle 90.00^{\circ} & 0 & 0 & 0 & 0 & 13.7931\angle 96.73^{\circ} & 35.5564\angle -85.48^{\circ} & 6.0920\angle 101.24^{\circ} \\
0 & 0 & 0 & 11.6841\angle 96.71^{\circ} & 0 & 0 & 0 & 6.0920\angle 101.24^{\circ} & 17.5252\angle -81.62^{\circ}
\end{bmatrix}
$$

FIGURE 5.22
Admittance matrix for Example 5.10

$$E_3\angle\delta_3 = (j0.1813)(0.8177 + j0.1723) + 1.0254\angle 4.6587^\circ = 1.0174\angle 13.1535^\circ$$

The next step is to convert the loads to equivalent impedances:

$$G_5 + jB_5 = \frac{(0.90 - j0.30)}{1.0128^2} = 0.8773 - j0.2924$$
$$G_7 + jB_7 = \frac{(1.00 - j0.35)}{1.0162^2} = 0.9684 - j0.3389$$
$$G_9 + jB_9 = \frac{(1.25 - j0.50)}{0.9958^2} = 1.2605 - j0.5042$$

These values are added to the diagonal of the original admittance matrix.

The reduced admittance matrices can now be computed as outlined in steps 4 and 5 above. The prefault admittance matrix is

$$Y_{red}^{prefault} = \begin{bmatrix} 0.8453 - j2.9881 & 0.2870 + j1.5131 & 0.2095 + j1.2257 \\ 0.2870 + j1.5131 & 0.4199 - j2.7238 & 0.2132 + j1.0880 \\ 0.2095 + j1.2257 & 0.2132 + j1.0880 & 0.2769 - j2.3681 \end{bmatrix}$$

The fault-on matrix is found similarly, except that the Y_{mm} is altered to reflect the fault on bus 8. The solid three-phase fault is modeled by shorting the bus to ground. In the admittance matrix, the row and column corresponding to bus 8 are removed. The lines between bus 8 and adjacent buses are now connected to ground; thus they will still appear in the original admittance diagonals. The column of Y_{nm} and the row of Y_{mn} corresponding to bus 8 must also be removed. The matrix Y_{nn} remains unchanged. The fault-on reduced admittance matrix is

$$Y_{red}^{fault\text{-}on} = \begin{bmatrix} 0.6567 - j3.8159 & 0 & 0.0701 + j0.6306 \\ 0 & 0 - j5.4855 & 0 \\ 0.0701 + j0.6306 & 0 & 0.1740 - j2.7959 \end{bmatrix}$$

The postfault reduced admittance matrix is computed in much the same way, except that line 8-9 is removed from Y_{mm}. The elements of Y_{mm} are updated to reflect the removal of the line:

$$Y_{mm}(8,8) = Y_{mm}(8,8) + Y_{mm}(8,9)$$
$$Y_{mm}(8,9) = Y_{mm}(9,9) + Y_{mm}(8,9)$$
$$Y_{mm}(8,9) = 0$$
$$Y_{mm}(9,8) = 0$$

Note that the diagonals must be updated before the off-diagonals are zeroed out. The postfault reduced admittance is then computed:

$$Y_{red}^{postfault} = \begin{bmatrix} 1.1811 - j2.2285 & 0.1375 + j0.7265 & 0.1909 + j1.0795 \\ 0.1375 + j0.7265 & 0.3885 - j1.9525 & 0.1987 + j1.2294 \\ 0.1909 + j1.0795 & 0.1987 + j1.2294 & 0.2727 - j2.3423 \end{bmatrix}$$

These admittance matrices are then ready to be substituted into the transient stability equations at the appropriate time in the simulation.

Applying the trapezoidal algorithm to the transient stability equations yields the following system of equations:

$$\delta_1(n+1) = \delta_1(n) + \frac{h}{2}\left[\omega_1(n+1) - \omega_s + \omega_1(n) - \omega_s\right] \tag{5.152}$$

$$\omega_1(n+1) = \omega_1(n) + \frac{h}{2}\left[f_1(n+1) + f_1(n)\right] \tag{5.153}$$

$$\delta_2(n+1) = \delta_2(n) + \frac{h}{2}\left[\omega_2(n+1) - \omega_s + \omega_2(n) - \omega_s\right] \tag{5.154}$$

$$\omega_2(n+1) = \omega_2(n) + \frac{h}{2}\left[f_2(n+1) + f_2(n)\right] \tag{5.155}$$

$$\delta_3(n+1) = \delta_3(n) + \frac{h}{2}\left[\omega_3(n+1) - \omega_s + \omega_3(n) - \omega_s\right] \tag{5.156}$$

$$\omega_3(n+1) = \omega_3(n) + \frac{h}{2}\left[f_3(n+1) + f_3(n)\right] \tag{5.157}$$

where

$$f_i(n+1) = \frac{1}{M_i}\left(P_{m_i} - E_i^2 G_{ii} - E_i \sum_{j \neq i}^{n} E_j \left(B_{ij}\sin\delta_{ij}(n+1) + G_{ij}\cos\delta_{ij}(n+1)\right)\right) \tag{5.158}$$

Since the transient stability equations are nonlinear and the trapezoidal method is an implicit method, they must be solved iteratively using the Newton–Raphson method at each time point. The iterative equations are

$$\left[I - \frac{h}{2}\left[J(n+1)^k\right]\right]\begin{bmatrix} \delta_1(n+1)^{k+1} - \delta_1(n+1)^k \\ \omega_1(n+1)^{k+1} - \omega_1(n+1)^k \\ \delta_2(n+1)^{k+1} - \delta_2(n+1)^k \\ \omega_2(n+1)^{k+1} - \omega_2(n+1)^k \\ \delta_3(n+1)^{k+1} - \delta_3(n+1)^k \\ \omega_3(n+1)^{k+1} - \omega_3(n+1)^k \end{bmatrix} =$$

$$-\left[\begin{bmatrix} \delta_1^k(n+1) \\ \omega_1^k(n+1) \\ \delta_2^k(n+1) \\ \omega_2^k(n+1) \\ \delta_3^k(n+1) \\ \omega_3^k(n+1) \end{bmatrix} - \begin{bmatrix} \delta_1(n) \\ \omega_1(n) \\ \delta_2(n) \\ \omega_2(n) \\ \delta_3(n) \\ \omega_3(n) \end{bmatrix} - \frac{h}{2}\begin{bmatrix} \omega_1^k(n+1) + \omega_1(n) - 2\omega_s \\ f_1^k(n+1) + f_1(n) \\ \omega_2^k(n+1) + \omega_2(n) - 2\omega_s \\ f_2^k(n+1) + f_2(n) \\ \omega_1^k(n+1) + \omega_1(n) - 2\omega_s \\ f_3^k(n+1) + f_3(n) \end{bmatrix}\right] \tag{5.159}$$

where

$$[J] = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ \frac{\partial f_1}{\partial \delta_1} & 0 & \frac{\partial f_1}{\partial \delta_2} & 0 & \frac{\partial f_1}{\partial \delta_3} & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ \frac{\partial f_2}{\partial \delta_1} & 0 & \frac{\partial f_2}{\partial \delta_2} & 0 & \frac{\partial f_2}{\partial \delta_3} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ \frac{\partial f_3}{\partial \delta_1} & 0 & \frac{\partial f_3}{\partial \delta_2} & 0 & \frac{\partial f_3}{\partial \delta_3} & 0 \end{bmatrix} \tag{5.160}$$

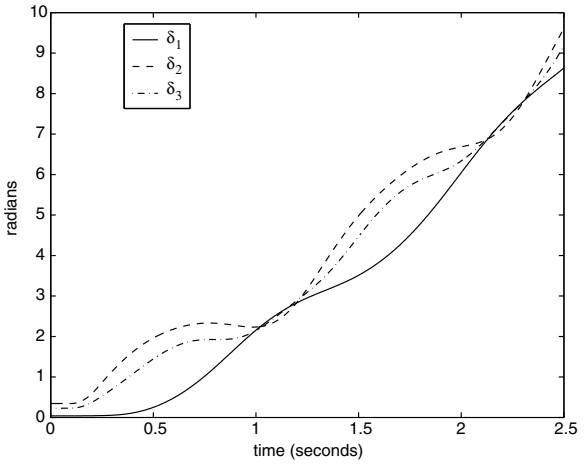

FIGURE 5.23
Rotor angle response for Example 5.10

Note that LU factorization must be employed to solve the discretized equations. These equations are iterated at each time point until convergence of the Newton–Raphson algorithm. The fault-on and postfault matrices are substituted in at the appropriate times in the integration. The simulation results are shown in Figures 5.23 and 5.24 for the rotor angles and angular frequencies, respectively. From the waveforms shown in these figures, it can be concluded that the system remains stable since the waveforms do not diverge during the simulation interval. ■

5.8.2 Midterm Stability Analysis

Reduction processes frequently destroy the natural physical structure and sparsity of the full-order system. Numerical solution algorithms which make use of structure and sparsity for efficiency perform poorly on the reduced-order system even though the reduced-order system is still quite large. After the first few seconds of a power system disturbance, the classical model representation may no longer be valid due to the dynamic behavior of the automatic voltage regulator, the turbine/governor system, under-load-tap-changing transformers, and the dynamic nature of some system loads. For midterm stability analyses, a more detailed model is required to capture a wider range of system behavior. Since the behavior of loads may significantly impact the stability of the system, it is desirable to be able to retain individual load buses during the simulation. This type of model is often referred to as a "structure-preserving" model, since the physical structure of the power system is retained. The inclusion of the load buses requires the solution of the set of power flow equations

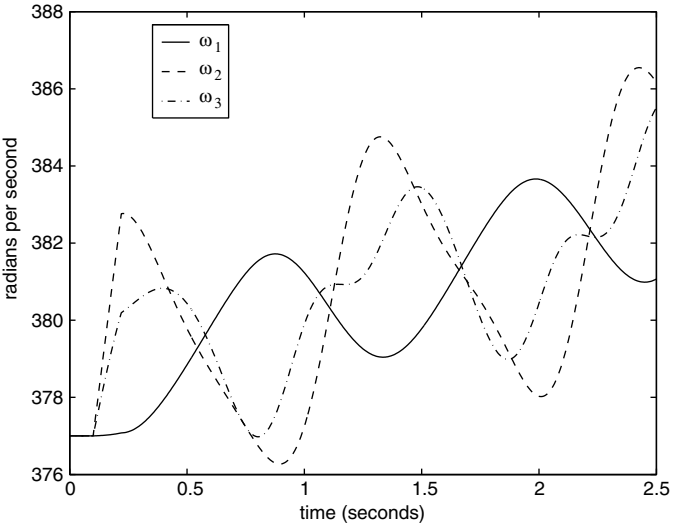

FIGURE 5.24
Angular frequency response for Example 5.10

governing the system network. This constraint leads to the inclusion of the algebraic power flow equations in conjunction with the differential equations describing the states. One example of a structure-preserving DAE model is given below:

$$T_{d0_i}\dot{E}'_{q_i} = -E'_{q_i} - \left(x_{d_i} - x'_{d_i}\right) I_{d_i} + E_{fd_i} \tag{5.161}$$

$$T_{q0_i}\dot{E}'_{d_i} = -E'_{d_i} + \left(x_{q_i} - x'_{q_i}\right) I_{q_i} \tag{5.162}$$

$$\dot{\delta}_i = \omega_i - \omega_s \tag{5.163}$$

$$\frac{2H_i}{\omega_s}\dot{\omega}_i = T_{m_i} - E'_{d_i} I_{d_i} - E'_{q_i} I_{q_i} - \left(x'_{q_i} - x'_{d_i}\right) I_{d_i} I_{q_i} \tag{5.164}$$

$$T_{E_i}\dot{E}_{fd_i} = -\left(K_{E_i} + S_{E_i}\left(E_{fd_i}\right)\right) E_{fd_i} + V_{R_i} \tag{5.165}$$

$$T_{F_i}\dot{R}_{F_i} = -R_{F_i} + \frac{K_{F_i}}{T_{F_i}} E_{fd_i} \tag{5.166}$$

$$T_{A_i}\dot{V}_{R_i} = -V_{R_i} + K_{A_i} R_{F_i} - \frac{K_{A_i} K_{F_i}}{T_{F_i}} E_{fd_i} + K_{A_i}\left(V_{ref_i} - V_{T_i}\right) \tag{5.167}$$

$$T_{RH_i}\dot{T}_{M_i} = -T_{M_i} + \left(1 - \frac{K_{HP_i} T_{RH_i}}{T_{CH_i}}\right) P_{CH_i} + \frac{K_{HP_i} T_{RH_i}}{T_{CH_i}} P_{SV_i} \tag{5.168}$$

$$T_{CH_i}\dot{P}_{CH_i} = -P_{CH_i} + P_{SV_i} \tag{5.169}$$

$$T_{SV_i}\dot{P}_{SV_i} = -P_{SV_i} + P_{C_i} - \frac{1}{R}\frac{\omega_i}{\omega_s} \tag{5.170}$$

and

$$0 = V_i e^{j\theta_i} + \left(r_s + jx'_{d_i}\right)\left(I_{d_i} + jI_{q_i}\right) e^{j\left(\delta_i - \frac{\pi}{2}\right)} - \left[E'_{d_i} + \left(x'_{q_i} - x'_{d_i}\right) I_{q_i} + jE'_{q_i}\right] e^{j\left(\delta_i - \frac{\pi}{2}\right)} \tag{5.171}$$

$$0 = V_i e^{j\theta_i}\left(I_{d_i} - jI_{q_i}\right) e^{-j\left(\delta_i - \frac{\pi}{2}\right)} - \sum_{k=1}^{N} V_i V_k Y_{ik} e^{j(\theta_i - \theta_k - \phi_{ik})} \tag{5.172}$$

$$0 = P_i + jQ_i - \sum_{k=1}^{N} V_i V_k Y_{ik} e^{j(\theta_i - \theta_k - \phi_{ik})} \tag{5.173}$$

These equations describe the behavior of a two-axis generator model, a simple automatic voltage regulator and exciter, a simple turbine/governor, and constant power loads. This set of dynamic equations is listed here for illustration purposes and is not intended to be inclusive of all possible representations. A detailed development of these equations may be found in [47].

These equations may be modeled in a more general form as

$$\dot{x} = f(x, y) \tag{5.174}$$

$$0 = g(x, y) \tag{5.175}$$

where the state vector x contains the dynamic state variables of the generators. The vector y is typically much larger and contains all of the network variables, including bus voltage magnitude and angle. It may also contain generator states such as currents that are not inherently dynamic. There are two typical approaches to solving this set of differential/algebraic equations. The first is the method suggested by Gear whereby all equations are solved simultaneously. The second approach is to solve the differential and algebraic sets of equations separately and iterate between each.

Consider the first approach applied to the DAE system using the trapezoidal integration method:

$$x(n+1) = x(n) + \frac{h}{2}\left[f\left(x(n+1), y(n+1)\right) + f\left(x(n), y(n)\right)\right] \tag{5.176}$$

$$0 = g\left(x(n+1), y(n+1)\right) \tag{5.177}$$

This set of nonlinear equations must then be solved using the Newton–Raphson method for the combined state vector $[x(n+1)\ \ y(n+1)]^T$:

$$\begin{bmatrix} I - \frac{h}{2}\frac{\partial f}{\partial x} & -\frac{h}{2}\frac{\partial f}{\partial y} \\ \frac{\partial g}{\partial x} & \frac{\partial g}{\partial y} \end{bmatrix} \begin{bmatrix} x(n+1)^{k+1} - x(n+1)^k \\ y(n+1)^{k+1} - y(n+1)^k \end{bmatrix} = \begin{bmatrix} x(n+1)^k - x(n) - \frac{h}{2}\left[f\left(x(n+1)^k, y(n+1)^k\right) + f\left(x(n), y(n)\right)\right] \\ g\left(x(n+1)^k, y(n+1)^k\right) \end{bmatrix} \tag{5.178}$$

The advantages of this method are that, since the whole set of system equations is used, the system matrices are quite sparse, and sparse solution techniques can be used efficiently. Second, since the set of equations is solved

simultaneously, the iterations are more likely to converge since the only iteration involved is that of the Newton–Raphson algorithm. Note, however, that, in a system of ODEs, the left-hand side matrix (the matrix to be factored in LU factorization) can be made to be diagonally dominant (and therefore well conditioned) by decreasing the step size h. In DAE systems, however, the left-hand side matrix may be ill conditioned under certain operating points where $\frac{\partial g}{\partial y}$ is ill conditioned, causing difficulty in solving the system of equations. This situation may occur during the simulation of voltage collapse where the states encounter a bifurcation. The subject of bifurcation and voltage collapse is complex and outside the scope of this book; however, several excellent texts have been published that study these phenomena in great detail [25] [47] [60].

The second approach to solving the system of DAEs is to solve each of the subsystems independently and iteratively. The system of differential equations is solved first for $x(n+1)$ while holding $y(n+1)$ constant as an input. After $x(n+1)$ has been found, it is then used as an input to solve the algebraic system. The updated value of $y(n+1)$ is then substituted back into the differential equations, and $x(n+1)$ is recalculated. This back and forth process is repeated until the values $x(n+1)$ and $y(n+1)$ converge. The solution is then advanced to the next time point. The advantage of this method is simplicity in programming, since each subsystem is solved independently and the Jacobian elements $\frac{\partial f}{\partial y}$ and $\frac{\partial g}{\partial x}$ are not used. In some cases, this may speed up the computation, although more iterations may be required to reach convergence.

5.9 Problems

1. Determine a Taylor series expansion for the solution of the equation

$$\dot{x} = x^2 \quad x(0) = 1$$

about the point $\hat{x} = 0$ (a McClaurin series expansion). Use this approximation to compute x for $\hat{x} = 0.2$ and $\hat{x} = 1.2$. Compare with the exact solution and explain the results.

2. Use the following algorithms to solve the initial value problem.

$$\dot{x}_1 = -2x_2 + 2t^2 \quad x_1(0) = -4$$
$$\dot{x}_2 = \frac{1}{2}x_1 + 2t \quad x_2(0) = 0$$

on the interval $0 \leq t \leq 5$ with a fixed integration step of 0.25 seconds.

(a) Backward Euler

(b) Forward Euler

(c) Trapezoidal rule

(d) Fourth-order Runge–Kutta

Compare the answers to the exact solution

$$x_1(t) = -4\cos t$$
$$x_2(t) = -2\sin t + t^2$$

3. Consider the following linear multistep formula.

$$x_{n+1} = a_0 x_n + a_1 x_{n-1} + a_2 x_{n-2} + a_3 x_{n-3} + h b_{-1} f(x_{n+1}, t_{n+1})$$

(a) What are the number of steps in the formula?

(b) What is the maximum order that will enable you to determine all the coefficients of the formula?

(c) Assuming uniform step size h, find all the coefficients of the formula such that its order is the answer of part (b). (This is a Gear's method)

(d) Is the formula implicit or explicit?

(e) Assuming uniform step size h, find the expression for the local truncation error of the formula.

(f) Is the formula absolutely stable?

(g) Is the formula stiffly-stable?

4. Consider the following initial value problem.

$$\dot{x} = 100(\sin(t) - x), \quad x(0) = 0$$

The exact solution is

$$x(t) = \frac{\sin(t) - 0.01\cos(t) + 0.01e^{-100t}}{1.0001}$$

Solve the initial value problem with $h = 0.02$ s using the following integration methods. Plot each of the numerical solutions against the exact solution over the range $t \in [0, 3.0]$ seconds. Plot the global error for each method over the range $t \in [0, 3.0]$ seconds. Discuss your results.

(a) Backward Euler

(b) Forward Euler

(c) Trapezoidal rule

(d) Fourth-order Runge–Kutta

(e) Repeat (a)–(d) with step size $h = 0.03$ s.

5. Consider a simple ecosystem consisting of rabbits that have an infinite food supply and foxes that prey upon the rabbits for their food. A classical mathematical "predator-prey" model due to Volterra describes this system by a pair of nonlinear, first-order differential equations:

$$\begin{aligned} \dot{r} &= \alpha r + \beta r f \quad r(0) = r_0 \\ \dot{f} &= \gamma f + \delta r f \quad f(0) = f_0 \end{aligned}$$

where $r = r(t)$ is the number of rabbits, $f = f(t)$ is the number of foxes. When $\beta = 0$, the two populations do not interact, and so the rabbits multiply and the foxes die off from starvation.

Investigate the behavior of this system for $\alpha = -1, \beta = 0.01, \gamma = 0.25$, and $\delta = -0.01$. Use the trapezoidal integration method with $h = 0.1$ and $T = 50$, and the set of initial conditions $r_0 = 40$ and $f_0 = 70$. Plot (1) r and f vs. t, and (2) r vs. f for each case.

6. The following system of nonlinear equations is known as Duffing's differential equation.

$$\ddot{x} + \delta \dot{x} + \alpha x + \beta x^3 = \gamma \cos \omega t$$

For the following parameters, plot x versus y in two dimensions and x and y versus t for $0 \leq t \leq 200$ seconds.

$$\begin{aligned} \alpha &= 0.1 \\ \beta &= 0.5 \\ \gamma &= -0.8 \\ \delta &= 0.1 \\ \omega &= \frac{\pi}{2} \end{aligned}$$

with $x(0) = 1$ and $\dot{x} = -1$ and $h = 0.01$.

7. The following system of equations is known as the Lorenz equations.

$$\begin{aligned} \dot{x} &= \sigma (y - x) \\ \dot{y} &= \rho x - y - xz \\ \dot{z} &= xy - \beta z \end{aligned}$$

Let $\sigma = 10, \rho = 28$, and $\beta = 4$. Use the trapezoidal method to plot the response of the system for $0 \leq t \leq 10$ seconds using a fixed integration time step of $h = 0.005$ seconds and a Newton–Raphson convergence error of 10^{-5}. Plot x versus y in two dimensions and x vs. y vs. z in three dimensions.

(a) Use $[x(0)\ \ y(0)\ \ z(0)]^T = [20\ \ 20\ \ 20]^T$.

(b) Use $[x(0)\ \ y(0)\ \ z(0)]^T = [21\ \ 20\ \ 20]^T$. Explain the difference.

8. Consider the following system of "stiff" equations.

$$\dot{x}_1 = -2x_1 + x_2 + 100 \quad x_1(0) = 0$$
$$\dot{x}_2 = 10^4 x_1 - 10^4 x_2 + 50 \quad x_2(0) = 0$$

(a) Determine the *maximum step size* $h_{\max}$ for the forward Euler algorithm to remain numerically stable.

(b) Use the forward Euler algorithm with step size $h = \frac{1}{2}h_{\max}$ to solve for $x_1(t)$ and $x_2(t)$ for $t > 0$.

(c) Repeat using a step size $h = 2h_{\max}$.

(d) Use the backward Euler algorithm to solve the stiff equations. Choose the following step sizes in terms of the maximum step size $h_{\max}$.

i. $h = 10h_{\max}$

ii. $h = 100h_{\max}$

iii. $h = 1000h_{\max}$

iv. $h = 10,000h_{\max}$

(e) Repeat using the multistep method (Gear's method) of Problem 4.

9. Consider a multistep method of the form

$$x_{n+2} - x_{n-2} + \alpha\left(x_{n+1} - x_{n-1}\right) = h\left[\beta\left(f_{n+1} + f_{n-1}\right) + \gamma f_n\right]$$

(a) Show that the parameters α, β, and γ can be chosen uniquely so that the method has order $p = 6$.

(b) Discuss the stability properties of this method. For what region is it absolutely stable? Stiffly stable?

10. A power system can be described by the following ODE system.

$$\dot{\delta}_i = \omega_i - \omega_s$$
$$M_i\dot{\omega}_i = P_i - E_i \sum_{k=1}^{n} E_k Y_{ik} \sin\left(\delta_i - \delta_k - \phi_{ik}\right)$$

Using a step size of $h = 0.01$ s and the trapezoidal integration method, determine whether or not this system is stable for a clearing time of 4 cycles.

Let

$$\begin{aligned} E_1 &= 1.0566\angle 2.2717^\circ \\ E_2 &= 1.0502\angle 19.7315^\circ \\ E_3 &= 1.0170\angle 13.1752^\circ \\ P_1 &= 0.716 \\ P_2 &= 1.630 \\ P_3 &= 0.850 \\ H_1 &= 23.64 \\ H_2 &= 6.40 \\ H_3 &= 3.01 \end{aligned}$$

where $M_i = \frac{2H_i}{\omega_s}$.
and the following admittance matrices:

Prefault:

$$\begin{matrix} 0.846 - j2.988 & 0.287 + j1.513 & 0.210 + j1.226 \\ 0.287 + j1.513 & 0.420 - j2.724 & 0.213 + j1.088 \\ 0.210 + j1.226 & 0.213 + j1.088 & 0.277 - j2.368 \end{matrix}$$

Fault-on:

$$\begin{matrix} 0.657 - j3.816 & 0.000 + j0.000 & 0.070 + j0.631 \\ 0.000 + j0.000 & 0.000 - j5.486 & 0.000 + j0.000 \\ 0.070 + j0.631 & 0.000 + j0.000 & 0.174 - j2.796 \end{matrix}$$

Postfault:

$$\begin{matrix} 1.181 - j2.229 & 0.138 + j0.726 & 0.191 + j1.079 \\ 0.138 + j0.726 & 0.389 - j1.953 & 0.199 + j1.229 \\ 0.191 + j1.079 & 0.199 + j1.229 & 0.273 - j2.342 \end{matrix}$$

11. A double pendulum system is shown in Figure 5.25. Masses m_1 and m_2 are connected by massless rods of length r_1 and r_2. The equations of motion of the two masses, expressed in terms of the angles θ_1 and θ_2 as indicated, are

$$\begin{aligned} -(m_1 + m_2)gr_1 \sin\theta_1 &= (m_1 + m_2)r_1^2\ddot{\theta}_1 + m_2 r_1 r_2 \ddot{\theta}_2 \cos(\theta_1 - \theta_2) \\ &\quad + m_2 r_1 r_2 \dot{\theta}_2^2 \sin(\theta_1 - \theta_2) \\ -m_2 g r_2 \sin\theta_2 &= m_2 r_2^2 \ddot{\theta}_2 + m_2 r_1 r_2 \ddot{\theta}_1 \cos(\theta_1 - \theta_2) \\ &\quad - m_2 r_1 r_2 \dot{\theta}_1^2 \sin(\theta_1 - \theta_2) \end{aligned}$$

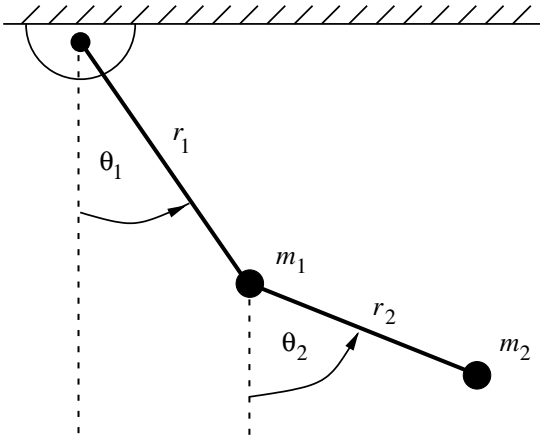

FIGURE 5.25
Double pendulum system

(a) Choose $x_1 = \theta_1, x_2 = \dot{\theta}_1, x_3 = \theta_2, x_4 = \dot{\theta}_2$, and show that $[0; 0; 0; 0]$ is an equilibrium of the system.

(b) Show that the second-order Adams–Bashforth method is given by

$$x_{n+1} = x_n + h\left\{\frac{3}{2}f\left(x_n, t_n\right) - \frac{1}{2}f\left(x_{n-1}, t_{n-1}\right)\right\}$$

with

$$\epsilon_T = \frac{5}{12}\hat{x}^{(3)}(\tau)h^3$$

(c) Show that the third-order Adams–Bashforth method is given by

$$x_{n+1} = x_n + h\left\{\frac{23}{12}f\left(x_n, t_n\right) - \frac{16}{12}f\left(x_{n-1}, t_{n-1}\right) + \frac{5}{12}f\left(x_{n-2}, t_{n-2}\right)\right\}$$

with

$$\epsilon_T = \frac{3}{8}\hat{x}^{(4)}(\tau)h^4$$

(d) Let $r_1 = 1, r_2 = 0.5$, and $g = 10$. Using the second-order Adams–Bashforth method with $h = 0.005$, plot the behavior of the system for an initial displacement of $\theta_1 = 25°$ and $\theta_2 = 10°$, for $T \in [0, 10]$ with

i. $m_1 = 10$ and $m_2 = 10$.
ii. $m_1 = 10$ and $m_2 = 5$.
iii. $m_1 = 10$ and $m_2 = 1$.

(e) Repeat (d) using a variable step method with an upper LTE bound of 0.001 and $B_{avg} = 0.1B_U$. Plot h versus t for each case. Discuss your solution.

(f) Repeat (d) and (e) using a third-order Adams–Bashforth method.