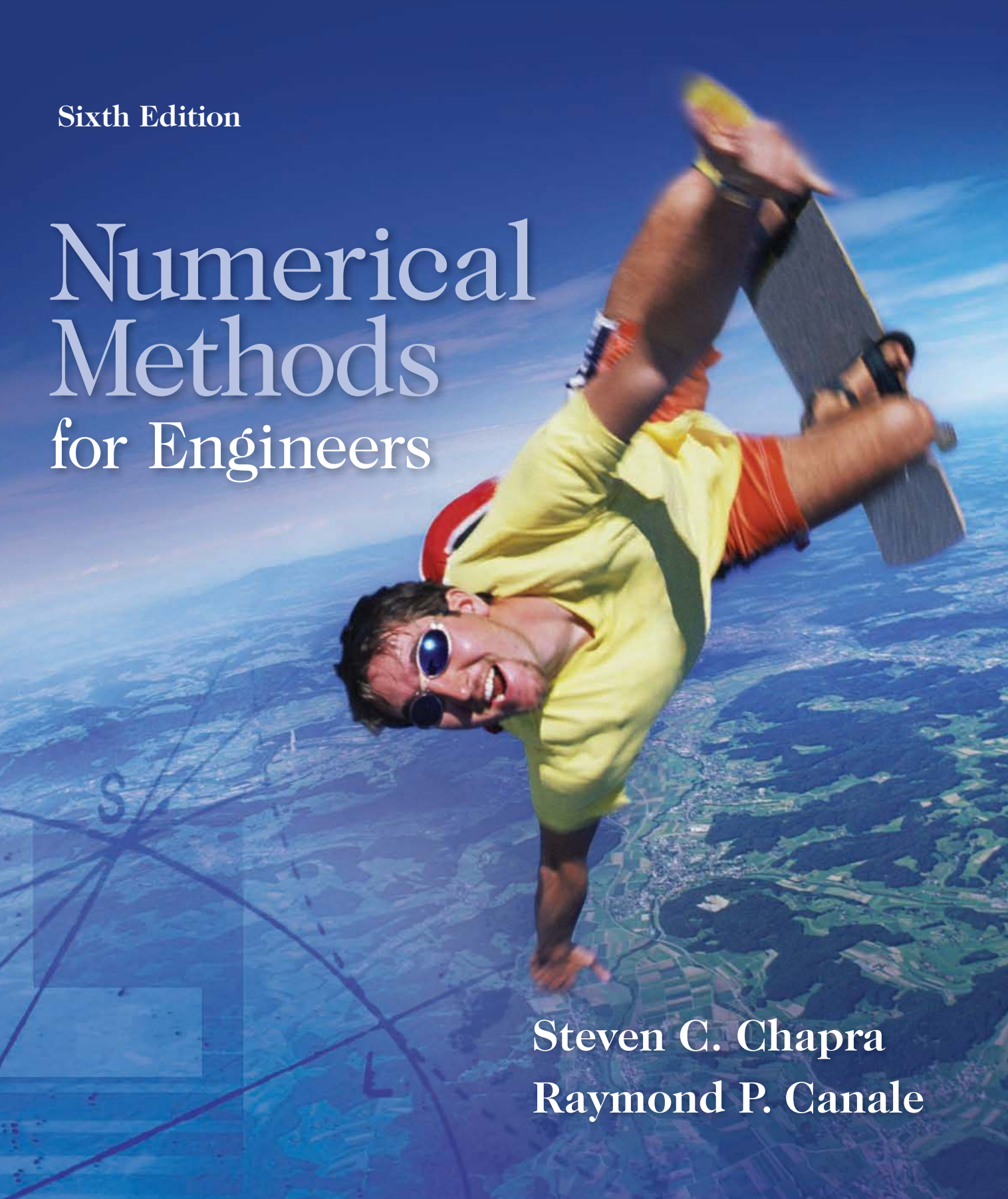
Sixth Edition

Numerical Methods for Engineers



Steven C. Chapra
Raymond P. Canale

CHAPTER 25

Runge-Kutta Methods

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

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

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

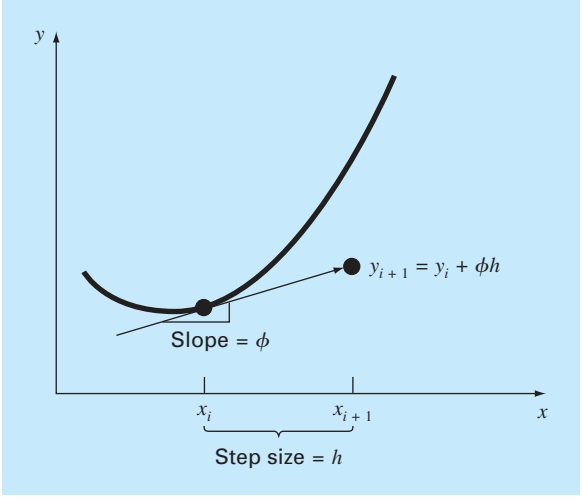
$$\text{New value} = \text{old value} + \text{slope} \times \text{step size}$$

or, in mathematical terms,

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

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

FIGURE 25.1
Graphical depiction of a one-step method.



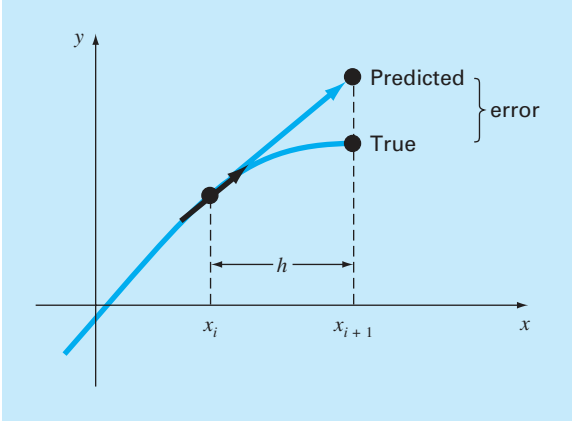


FIGURE 25.2
Euler's method.

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

25.1 EULER'S METHOD

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

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

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

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

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

EXAMPLE 25.1 Euler's Method

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

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

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

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

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

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

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

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

Therefore,

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

The true solution at $x = 0.5$ is

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

Thus, the error is

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

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

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

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

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

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

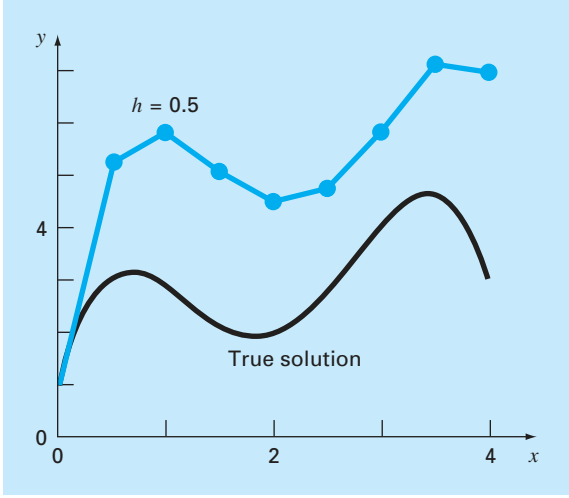


FIGURE 25.3
Comparison of the true solution with a numerical solution using Euler's method for the integral of $y' = -2x^3 + 12x^2 - 20x + 8.5$ from $x = 0$ to $x = 4$ with a step size of 0.5. The initial condition at $x = 0$ is $y = 1$.

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

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

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

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

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

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

25.1.1 Error Analysis for Euler's Method

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

or

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

where E_a = the approximate local truncation error.

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

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

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

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

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

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

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

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

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

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

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

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

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

For the third-order term:

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

and the fourth-order term:

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

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

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

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

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

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

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

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

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

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

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

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

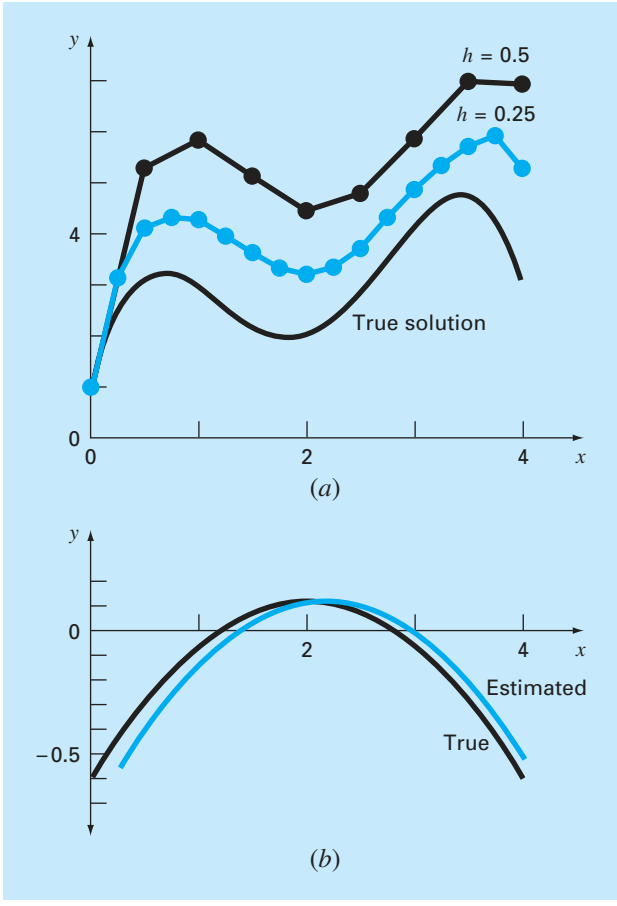


FIGURE 25.4
(*a*) Comparison of two numerical solutions with Euler's method using step sizes of 0.5 and 0.25. (*b*) Comparison of true and estimated local truncation error for the case where the step size is 0.5. Note that the "estimated" error is based on Eq. (E25.2.5).

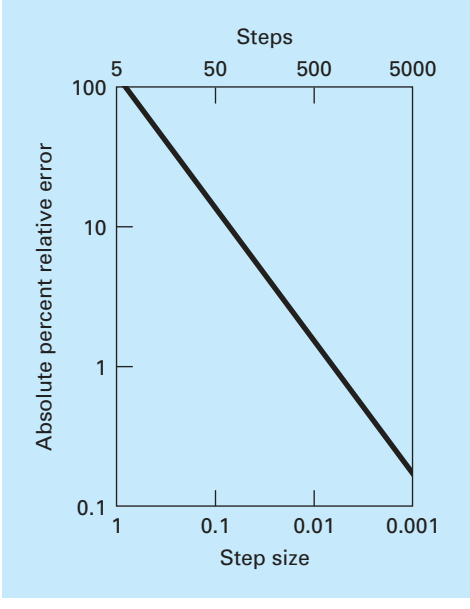
Solution. The computation is repeated, and the results are compiled in Fig. 25.4*a*. Halving the step size reduces the absolute value of the average global error to 40 percent and the absolute value of the local error to 6.4 percent. This is compared to global and local errors for Example 25.1 of 90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

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

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

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

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



25.1.2 Algorithm for Euler's Method

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

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

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

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

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

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

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

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

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

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

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

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

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

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

(*a*) Main or "Driver" Program

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

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

(*b*) Routine to Take One Output Step

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

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

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

(*d*) Routine to Determine Derivative

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

EXAMPLE 25.4 Solving ODEs with the Computer

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

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

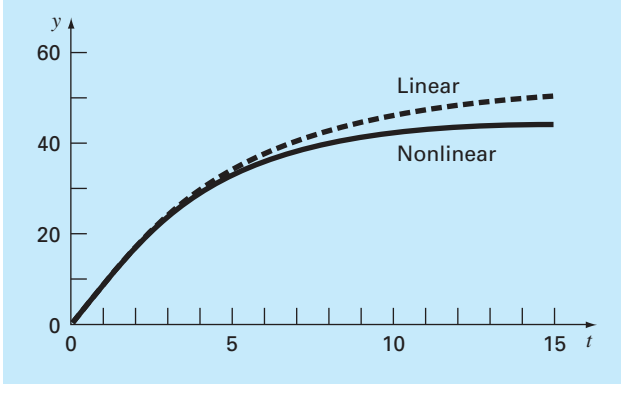
This differential equation was solved both analytically (Example 1.1) and numerically using Euler's method (Example 1.2). These solutions were for the case where $g = 9.8$, $c = 12.5$, $m = 68.1$, and $v = 0$ at $t = 0$.

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

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

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

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



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

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

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

25.1.3 Higher-Order Taylor Series Methods

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

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

with a local truncation error of

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

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

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

The second derivative is

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

Higher-order derivatives become increasingly more complicated.

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

25.2 IMPROVEMENTS OF EULER'S METHOD

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

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

25.2.1 Heun's Method

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

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

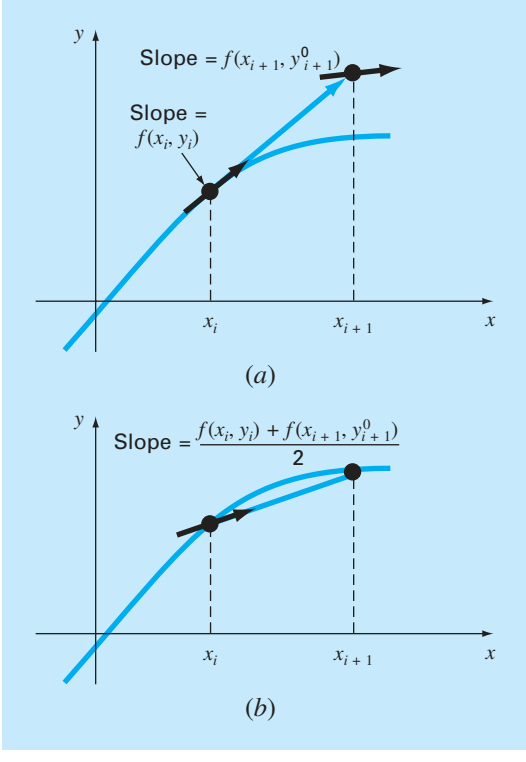
$$y_i' = f(x_i, y_i) \tag{25.12}$$

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

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

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

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



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

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

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

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

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

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

which is called a *corrector equation.*

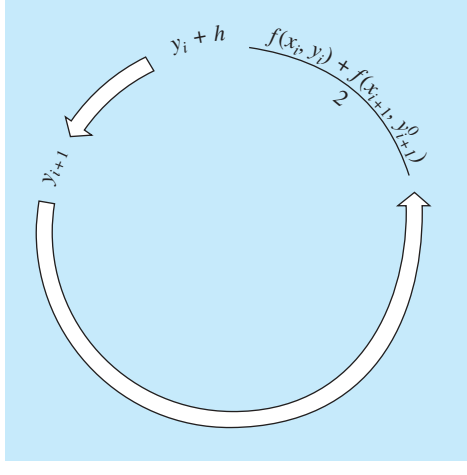
The Heun method is a *predictor-corrector approach.* All the multistep methods to be discussed subsequently in Chap. 26 are of this type. The Heun method is the only one-step predictor-corrector method described in this book. As derived above, it can be expressed concisely as

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

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

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

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



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

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

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

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

EXAMPLE 25.5 Heun's Method

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

This equation can be solved for y by integration:

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

which yields

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

or

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

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

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

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

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

which is equivalent to Eq. (25.18).

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

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

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

25.2.2 The Midpoint (or Improved Polygon) Method

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

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

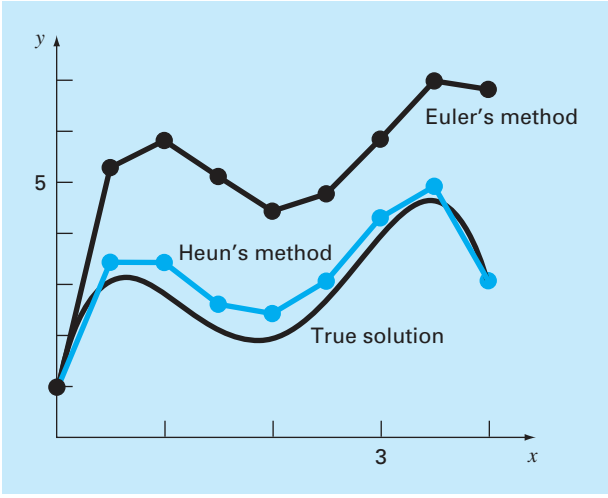


FIGURE 25.11
Comparison of the true solution with a numerical solution using Euler's and Heun's methods for the integral of $y' = -2x^3 + 12x^2 - 20x + 8.5$.

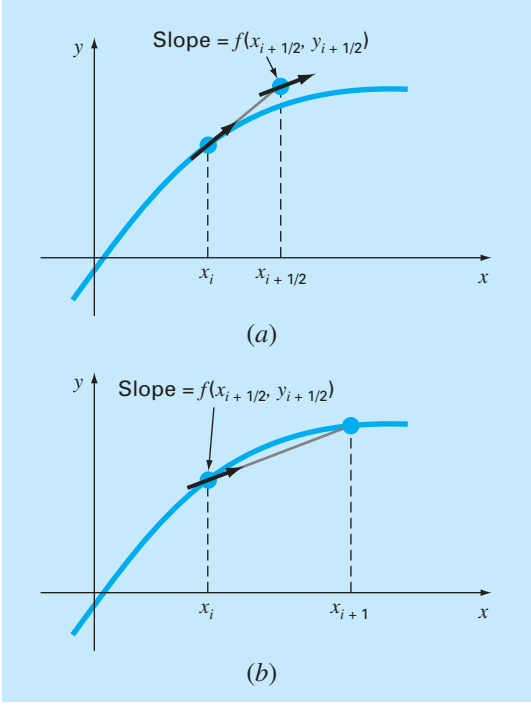


FIGURE 25.12
Graphical depiction of the midpoint method. (*a*) Eq. (25.25) and (*b*) Eq. (25.27).

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

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

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

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

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

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

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

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

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

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

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

25.2.3 Computer Algorithms for Heun and Midpoint Methods

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

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

25.2.4 Summary

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

(a) Simple Heun without Corrector

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

(b) Midpoint Method

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

(c) Heun with Corrector

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

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

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

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

25.3 RUNGE-KUTTA METHODS

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

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

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

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

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

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

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

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

$$\vdots$$

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

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

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

25.3.1 Second-Order Runge-Kutta Methods

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

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

where

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

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

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

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

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

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

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

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

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

where

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

and

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

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

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

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

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

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

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

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

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

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

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

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

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

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

or, by collecting terms,

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

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

$$a_1 + a_2 = 1$$

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

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

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

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

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

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

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

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

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

where

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

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

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

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

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

where

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

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

This is the midpoint method.

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

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

where

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

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

EXAMPLE 25.6 Comparison of Various Second-Order RK Schemes

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

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

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

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

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

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

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

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

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

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

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

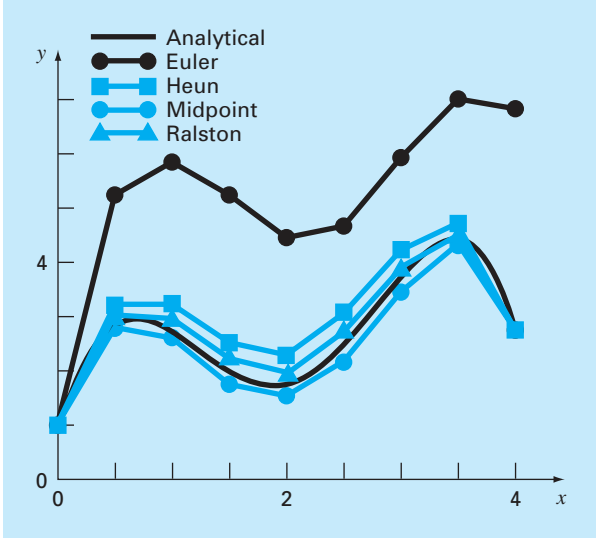


TABLE 25.3 Comparison of true and approximate values of the integral of $y' = -2x^3 + 12x^2 - 20x + 8.5$, with the initial condition that $y = 1$ at $x = 0$. The approximate values were computed using three versions of second-order RK methods with a step size of 0.5.

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

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

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

The average slope is computed by

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

which can be used to predict

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

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

25.3.2 Third-Order Runge-Kutta Methods

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

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

where

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

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

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

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

25.3.3 Fourth-Order Runge-Kutta Methods

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

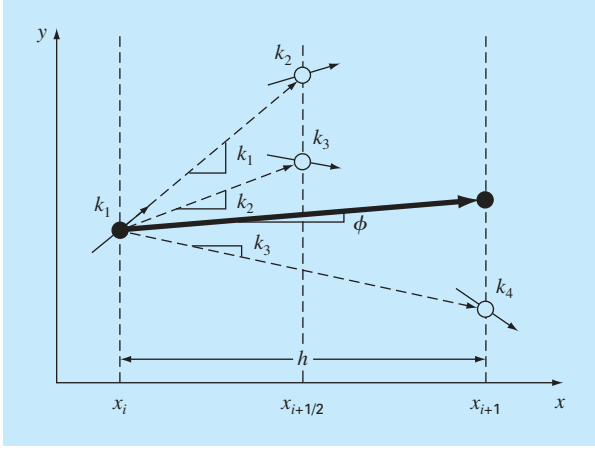
$$y_{i+1} = y_i + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)h \tag{25.40}$$

where

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

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

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



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

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

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

EXAMPLE 25.7 Classical Fourth-Order RK Method

Problem Statement.

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

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

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

(b) Similarly, integrate

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

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

Solution.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

which compares favorably with the true solution of 3.751521.

25.3.4 Higher-Order Runge-Kutta Methods

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

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

where

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

$$k_2 = f\left(x_i + \frac{1}{4}h, y_i + \frac{1}{4}k_1h\right) \tag{25.41b}$$

$$k_3 = f\left(x_i + \frac{1}{4}h, y_i + \frac{1}{8}k_1h + \frac{1}{8}k_2h\right) \tag{25.41c}$$

$$k_4 = f\left(x_i + \frac{1}{2}h, y_i - \frac{1}{2}k_2h + k_3h\right) \tag{25.41d}$$

$$k_5 = f\left(x_i + \frac{3}{4}h, y_i + \frac{3}{16}k_1h + \frac{9}{16}k_4h\right) \tag{25.41e}$$

$$k_6 = f\left(x_i + h, y_i - \frac{3}{7}k_1h + \frac{2}{7}k_2h + \frac{12}{7}k_3h - \frac{12}{7}k_4h + \frac{8}{7}k_5h\right) \tag{25.41f}$$

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

EXAMPLE 25.8 Comparison of Runge-Kutta Methods

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

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

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

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

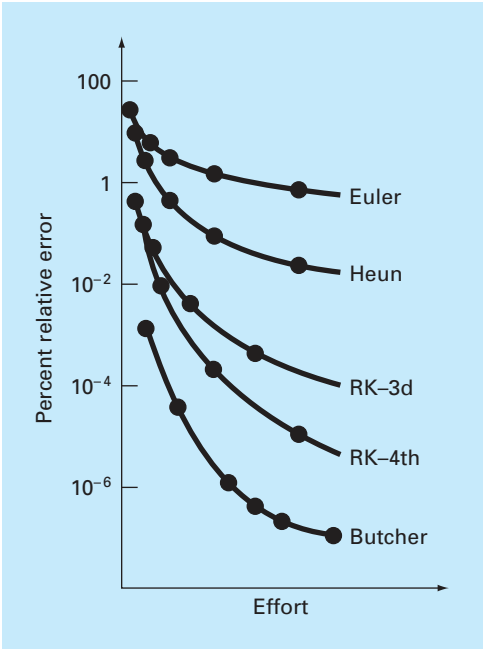


FIGURE 25.16
Comparison of percent relative error versus computational effort for first- through fifth-order RK methods.

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

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

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

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

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

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

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

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

25.3.5 Computer Algorithms for Runge-Kutta Methods

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

25.4 SYSTEMS OF EQUATIONS

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

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

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

25.4.1 Euler's Method

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

EXAMPLE 25.9 Solving Systems of ODEs Using Euler's Method

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

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

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

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

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

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

25.4.2 Runge-Kutta Methods

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

which can be used to compute the endpoint slopes,

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

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

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

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

Proceeding in a like manner for the remaining steps yields

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

25.4.3 Computer Algorithm for Solving Systems of ODEs

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

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

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

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

EXAMPLE 25.11 Solving Systems of ODEs with the Computer

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

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

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

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

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

(a) Main or "Driver" Program

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

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

(b) Routine to Take One Output Step

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

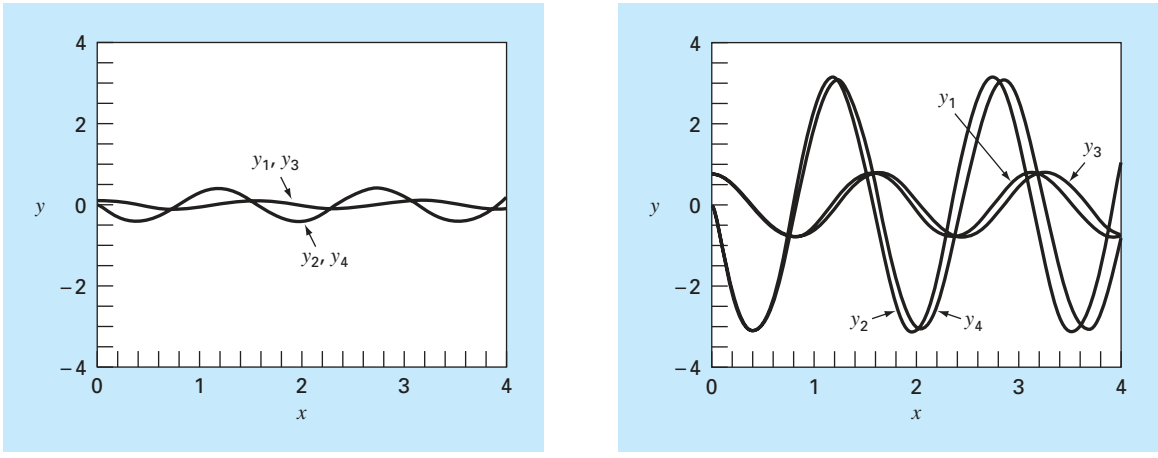
(c) Fourth-Order RK Method for a System of ODEs

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

(d) Routine to Determine Derivatives

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

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



(*a*) (*b*)

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

Solution.

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

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

25.5 ADAPTIVE RUNGE-KUTTA METHODS

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

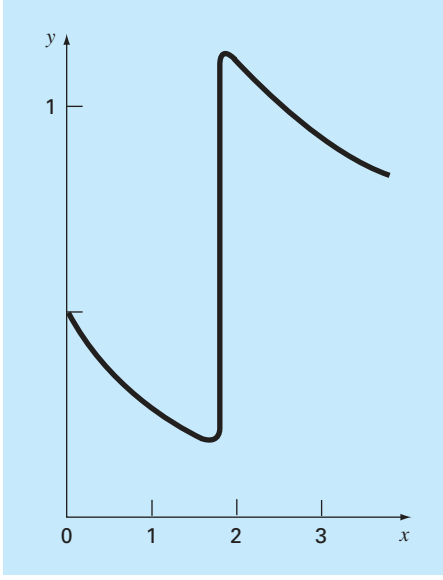


FIGURE 25.20
An example of a solution of an ODE that exhibits an abrupt change. Automatic step-size adjustment has great advantages for such cases.

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

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

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

is equivalent to solving the differential equation

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

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

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

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

25.5.1 Adaptive RK or Step-Halving Method

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

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

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

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

This estimate is fifth-order accurate.

EXAMPLE 25.12 Adaptive Fourth-Order RK Method

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

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

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

The two half-step predictions are

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

and

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

Therefore, the approximate error is

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

which compares favorably with the true error of

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

The error estimate can also be used to correct the prediction

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

which has an $E_t = -0.00235$.

25.5.2 Runge-Kutta Fehlberg

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

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

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

along with the fifth-order formula:

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

where

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

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

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

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

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

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

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

EXAMPLE 25.13 Runge-Kutta Fehlberg Method

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

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

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

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

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

along with a fifth-order formula:

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

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

$$E_a = 14.83677 - 14.83192 = 0.004842$$

25.5.3 Step-Size Control

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

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

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

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

A more general way to handle such cases is to determine Δ_{new} as

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

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

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

This is the version we will use in our algorithm.

25.5.4 Computer Algorithm

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

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

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

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

(a) Driver Program

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

(b) Adaptive Step Routine

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

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

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

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

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

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

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

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

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

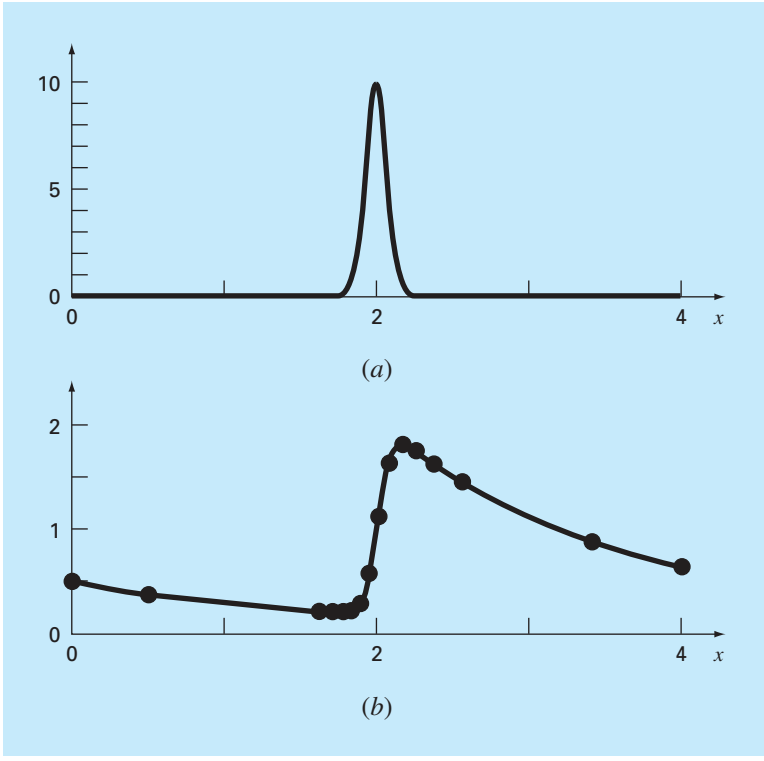


FIGURE 25.23
(*a*) A bell-shaped forcing function that induces an abrupt change in the solution of an ODE [Eq. (E25.14.1)]. (*b*) The solution. The points indicate the predictions of an adaptive step-size routine.

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

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

PROBLEMS

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

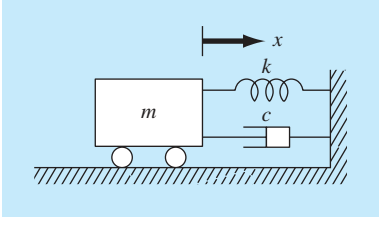
25.15 Develop a user-friendly computer program for systems of equations using the fourth-order RK method. Use this program to duplicate the computation in Example 25.10.

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

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

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

Figure P25.16



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

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

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

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

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

where

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

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

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

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

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

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

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

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

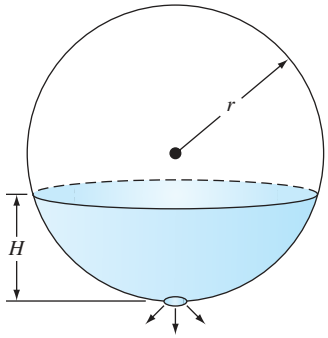


Figure P25.20
A spherical tank.

25.21 The logistic model is used to simulate population as in

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

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

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

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

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

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

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

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

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