CHAPTER

5

Bracketing Methods

This chapter on roots of equations deals with methods that exploit the fact that a function typically changes sign in the vicinity of a root. These techniques are called *bracketing methods* because two initial guesses for the root are required. As the name implies, these guesses must "bracket," or be on either side of, the root. The particular methods described herein employ different strategies to systematically reduce the width of the bracket and, hence, home in on the correct answer.

As a prelude to these techniques, we will briefly discuss graphical methods for depicting functions and their roots. Beyond their utility for providing rough guesses, graphical techniques are also useful for visualizing the properties of the functions and the behavior of the various numerical methods.

5.1 GRAPHICAL METHODS

A simple method for obtaining an estimate of the root of the equation $f(x) = 0$ is to make a plot of the function and observe where it crosses the x axis. This point, which represents the x value for which $f(x) = 0$, provides a rough approximation of the root.

EXAMPLE 5.1 The Graphical Approach

Problem Statement. Use the graphical approach to determine the drag coefficient c needed for a parachutist of mass $m = 68.1$ kg to have a velocity of 40 m/s after free-falling for time $t = 10$ s. *Note:* The acceleration due to gravity is 9.81 m/s^2.

Solution. This problem can be solved by determining the root of Eq. (PT2.4) using the parameters $t = 10$, $g = 9.81$, $v = 40$, and $m = 68.1$:

$$f(c) = \frac{9.81(68.1)}{c}(1 - e^{-(c/68.1)10}) - 40$$

or

$$f(c) = \frac{668.06}{c}(1 - e^{-0.146843c}) - 40 \qquad \text{(E5.1.1)}$$

Various values of c can be substituted into this equation to compute the following points:

c	f(c)
4	34.190
8	17.712
12	6.114
16	−2.230
20	−8.368

These points are plotted in Fig. 5.1. The resulting curve crosses the c axis between 12 and 16. Visual inspection of the plot provides a rough estimate of the root of 14.75. The validity of the graphical estimate can be checked by substituting it into Eq. (E5.1.1) to yield

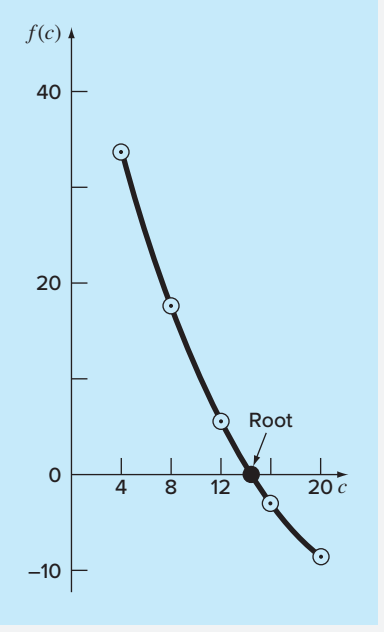
$$f(14.75) = \frac{668.06}{14.75}(1 - e^{-0.146843(14.75)}) - 40 = 0.100$$

which is close to zero. It can also be checked by substituting it into Eq. (PT2.3) along with the parameter values from this example to give

$$v = \frac{9.81(68.1)}{14.75}(1 - e^{-(14.75/68.1)10}) = 40.100$$

which is very close to the desired fall velocity of 40 m/s.

FIGURE 5.1
The graphical approach for determining the roots of an equation.



9.1.2 Determinants and Cramer's Rule

Cramer's rule is another solution technique that is best suited to small numbers of equations. Before describing this method, we will briefly introduce the concept of the determinant, which is used to implement Cramer's rule. In addition, the determinant has relevance to the evaluation of the ill-conditioning of a matrix.

Determinants. The determinant can be illustrated for a set of three equations:

$$[A]\{X\} = \{B\}$$

where $[A]$ is the coefficient matrix:

$$[A] = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

The determinant D of this system is formed from the coefficients of the equations, as in

$$D = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} \tag{9.2}$$

Although the determinant D and the coefficient matrix $[A]$ are composed of the same elements, they are completely different mathematical concepts. That is why they are distinguished visually by using brackets to enclose the matrix and straight lines to enclose the determinant. In contrast to a matrix, the determinant is a single number. For example, the value of the second-order determinant

$$D = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}$$

is calculated by

$$D = a_{11}a_{22} - a_{12}a_{21} \tag{9.3}$$

For the third-order case [Eq. (9.2)], a single numerical value for the determinant can be computed as

$$D = a_{11}\begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix} - a_{12}\begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix} + a_{13}\begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix} \tag{9.4}$$

where the 2 by 2 determinants are called ***minors.***

EXAMPLE 9.2 **Determinants**

Problem Statement. Compute values for the determinants of the systems represented in Figs. 9.1 and 9.2.

Solution. For Fig. 9.1:

$$D = \begin{vmatrix} 3 & 2 \\ -1 & 2 \end{vmatrix} = 3(2) - 2(-1) = 8$$

For Fig. 9.2*a*:

$$D = \begin{vmatrix} -1/2 & 1 \\ -1/2 & 1 \end{vmatrix} = \frac{-1}{2}(1) - 1\left(\frac{-1}{2}\right) = 0$$

For Fig. 9.2*b*:

$$D = \begin{vmatrix} -1/2 & 1 \\ -1 & 2 \end{vmatrix} = \frac{-1}{2}(2) - 1(-1) = 0$$

For Fig. 9.2*c*:

$$D = \begin{vmatrix} -1/2 & 1 \\ -2.3/5 & 1 \end{vmatrix} = \frac{-1}{2}(1) - 1\left(\frac{-2.3}{5}\right) = -0.04$$

In the foregoing example, the singular systems had zero determinants. Additionally, the results suggest that the system that is almost singular (Fig. 9.2*c*) has a determinant that is close to zero. These ideas will be pursued further in our subsequent discussion of ill-conditioning (Sec. 9.3.3).

Cramer's Rule. This rule states that each unknown in a system of linear algebraic equations may be expressed as a fraction of two determinants with denominator D and with the numerator obtained from D by replacing the column of coefficients of the unknown in question by the constants $b_1, b_2, \ldots, b_n$. For example, x_1 would be computed as

$$x_1 = \frac{\begin{vmatrix} b_1 & a_{12} & a_{13} \\ b_2 & a_{22} & a_{23} \\ b_3 & a_{32} & a_{33} \end{vmatrix}}{D} \tag{9.5}$$

EXAMPLE 9.3 Cramer's Rule

Problem Statement. Use Cramer's rule to solve

$$0.3x_1 + 0.52x_2 + x_3 = -0.01$$
$$0.5x_1 + x_2 + 1.9x_3 = 0.67$$
$$0.1x_1 + 0.3x_2 + 0.5x_3 = -0.44$$

Solution. The determinant D can be written as [Eq. (9.2)]

$$D = \begin{vmatrix} 0.3 & 0.52 & 1 \\ 0.5 & 1 & 1.9 \\ 0.1 & 0.3 & 0.5 \end{vmatrix}$$

The minors are [Eq. (9.3)]

$$A_1 = \begin{vmatrix} 1 & 1.9 \\ 0.3 & 0.5 \end{vmatrix} = 1(0.5) - 1.9(0.3) = -0.07$$

$$A_2 = \begin{vmatrix} 0.5 & 1.9 \\ 0.1 & 0.5 \end{vmatrix} = 0.5(0.5) - 1.9(0.1) = 0.06$$

$$A_3 = \begin{vmatrix} 0.5 & 1 \\ 0.1 & 0.3 \end{vmatrix} = 0.5(0.3) - 1(0.1) = 0.05$$

These can be used to evaluate the determinant, as in [Eq. (9.4)]

$$D = 0.3(-0.07) - 0.52(0.06) + 1(0.05) = -0.0022$$

Applying Eq. (9.5), the solution is

$$x_1 = \frac{\begin{vmatrix} -0.01 & 0.52 & 1 \\ 0.67 & 1 & 1.9 \\ -0.44 & 0.3 & 0.5 \end{vmatrix}}{-0.0022} = \frac{0.03278}{-0.0022} = -14.9$$

$$x_2 = \frac{\begin{vmatrix} 0.3 & -0.01 & 1 \\ 0.5 & 0.67 & 1.9 \\ 0.1 & -0.44 & 0.5 \end{vmatrix}}{-0.0022} = \frac{0.0649}{-0.0022} = -29.5$$

$$x_3 = \frac{\begin{vmatrix} 0.3 & 0.52 & -0.01 \\ 0.5 & 1 & 0.67 \\ 0.1 & 0.3 & -0.44 \end{vmatrix}}{-0.0022} = \frac{-0.04356}{-0.0022} = 19.8$$

For more than three equations, Cramer's rule becomes impractical because, as the number of equations increases, the determinants are time-consuming to evaluate by hand (or by computer). Consequently, more efficient alternatives are used. Some of these alternatives are based on the last noncomputer solution technique, covered in the next section—the elimination of unknowns.

9.1.3 The Elimination of Unknowns

The elimination of unknowns by combining equations is an algebraic approach that can be illustrated for a set of two equations:

$$a_{11}x_1 + a_{12}x_2 = b_1 \tag{9.6}$$

$$a_{21}x_1 + a_{22}x_2 = b_2 \tag{9.7}$$

The basic strategy is to multiply the equations by constants so that one of the unknowns will be eliminated when the two equations are combined. The result is a single equation that can be solved for the remaining unknown. This value can then be substituted into either of the original equations to compute the other variable.

For example, Eq. (9.6) might be multiplied by a_{21} and Eq. (9.7) by a_{11} to give

$$a_{11}a_{21}x_1 + a_{12}a_{21}x_2 = b_1a_{21} \tag{9.8}$$

$$a_{21}a_{11}x_1 + a_{22}a_{11}x_2 = b_2a_{11} \tag{9.9}$$

9.2 NAIVE GAUSS ELIMINATION

In the previous section, the elimination of unknowns was used to solve a pair of simultaneous equations. The procedure consisted of two steps:

1. The equations were manipulated to eliminate one of the unknowns from the equations. The result of this *elimination* step was that we had one equation with one unknown.
2. Consequently, this equation could be solved directly and the result *back-substituted* into one of the original equations to solve for the remaining unknown.

This basic approach can be extended to large sets of equations by developing a systematic scheme or algorithm to eliminate unknowns and to back-substitute. *Gauss elimination* is the most basic of these schemes.

This section includes the systematic techniques for forward elimination and back substitution that comprise Gauss elimination. Although these techniques are ideally suited for implementation on computers, some modifications will be required to obtain a reliable algorithm. In particular, the computer program must avoid division by zero. The following method is called *"naive" Gauss elimination* because it does not avoid this problem. Subsequent sections will deal with the additional features required for an effective computer program.

The approach is designed to solve a general set of n equations:

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n = b_1 \tag{9.12a}$$

$$a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n = b_2 \tag{9.12b}$$

$$\begin{matrix} \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \end{matrix}$$

$$a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \cdots + a_{nn}x_n = b_n \tag{9.12c}$$

As was the case with the solution of two equations, the technique for n equations consists of two phases: elimination of unknowns and solution through back substitution.

Forward Elimination of Unknowns. The first phase is designed to reduce the set of equations to an upper triangular system (Fig. 9.3). The initial step will be to eliminate the first unknown, x_1, from the second through the nth equations. To do this, multiply Eq. (9.12a) by a_{21}/a_{11} to give

$$a_{21}x_1 + \frac{a_{21}}{a_{11}}a_{12}x_2 + \cdots + \frac{a_{21}}{a_{11}}a_{1n}x_n = \frac{a_{21}}{a_{11}}b_1 \tag{9.13}$$

Now, this equation can be subtracted from Eq. (9.12b) to give

$$\left(a_{22} - \frac{a_{21}}{a_{11}}a_{12}\right)x_2 + \cdots + \left(a_{2n} - \frac{a_{21}}{a_{11}}a_{1n}\right)x_n = b_2 - \frac{a_{21}}{a_{11}}b_1$$

or

$$a'_{22}x_2 + \cdots + a'_{2n}x_n = b'_2$$

where the prime indicates that the elements have been changed from their original values.

The procedure is then repeated for the remaining equations. For instance, Eq. (9.12a) can be multiplied by a_{31}/a_{11} and the result subtracted from the third equation. Repeating

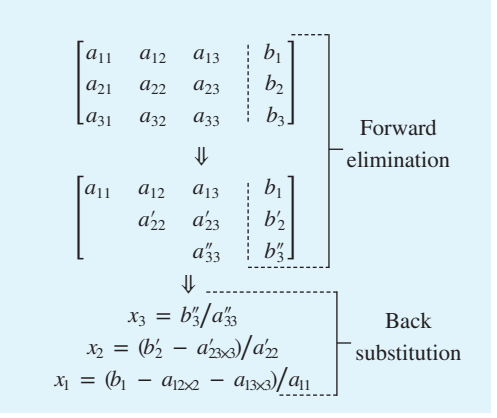


FIGURE 9.3
The two phases of Gauss elimination: forward elimination and back substitution. The primes indicate the number of times that the coefficients and constants have been modified.

the procedure for the remaining equations results in the following modified system:

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n = b_1 \tag{9.14a}$$
$$a'_{22}x_2 + a'_{23}x_3 + \cdots + a'_{2n}x_n = b'_2 \tag{9.14b}$$
$$a'_{32}x_2 + a'_{33}x_3 + \cdots + a'_{3n}x_n = b'_3 \tag{9.14c}$$
$$\vdots \qquad\qquad \vdots$$
$$a'_{n2}x_2 + a'_{n3}x_3 + \cdots + a'_{nn}x_n = b'_n \tag{9.14d}$$

For the foregoing steps, Eq. (9.12*a*) is called the *pivot equation* and a_{11} is called the *pivot coefficient* or *element.* Note that the process of multiplying the first row by a_{21}/a_{11} is equivalent to dividing it by a_{11} and multiplying it by a_{21}. Sometimes the division operation is referred to as normalization. We make this distinction because a zero pivot element can interfere with normalization by causing a division by zero. We will return to this important issue after we complete our description of naive Gauss elimination.

Now repeat the above to eliminate the second unknown from Eq. (9.14*c*) through (9.14*d*). To do this multiply Eq. (9.14*b*) by a'_{32}/a'_{22} and subtract the result from Eq. (9.14*c*). Perform a similar elimination for the remaining equations to yield

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n = b_1$$
$$a'_{22}x_2 + a'_{23}x_3 + \cdots + a'_{2n}x_n = b'_2$$
$$a''_{33}x_3 + \cdots + a''_{3n}x_n = b''_2$$
$$\vdots \qquad\qquad \vdots$$
$$a''_{n3}x_3 + \cdots + a''_{nn}x_n = b''_n$$

where the double prime indicates that the elements have been modified twice.

The procedure can be continued using the remaining pivot equations. The final manipulation in the sequence is to use the $(n-1)$th equation to eliminate the x_{n-1} term from the nth equation. At this point, the system will have been transformed to an upper triangular system (recall Box PT3.1):

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n = b_1 \tag{9.15a}$$

$$a'_{22}x_2 + a'_{23}x_3 + \cdots + a'_{2n}x_n = b'_2 \tag{9.15b}$$

$$a''_{33}x_3 + \cdots + a''_{3n}x_n = b''_3 \tag{9.15c}$$

$$\vdots \qquad\qquad \vdots$$

$$a_{nn}^{(n-1)}x_n = b_n^{(n-1)} \tag{9.15d}$$

Pseudocode to implement forward elimination is presented in Fig. 9.4*a*. Notice that three nested loops provide a concise representation of the process. The outer loop moves down the matrix from one pivot row to the next. The middle loop moves below the pivot row to each of the subsequent rows where elimination is to take place. Finally, the innermost loop progresses across the columns to eliminate or transform the elements of a particular row.

Back Substitution. Equation (9.15*d*) can now be solved for x_n:

$$x_n = \frac{b_n^{(n-1)}}{a_{nn}^{(n-1)}} \tag{9.16}$$

This result can be back-substituted into the $(n-1)$th equation to solve for x_{n-1}. The procedure, which is repeated to evaluate the remaining x's, can be represented by the following formula:

$$x_i = \frac{b_i^{(i-1)} - \sum_{j=i+1}^{n} a_{ij}^{(i-1)}x_j}{a_{ii}^{(i-1)}} \qquad \text{for } i = n-1, n-2, \ldots, 1 \tag{9.17}$$

FIGURE 9.4
Pseudocode to perform (*a*) forward elimination and (*b*) back substitution.

(*a*)

```
DOFOR k = 1, n − 1
  DOFOR i = k + 1, n
    factor = a_{i,k} / a_{k,k}
    DOFOR j = k + 1 to n
      a_{i,j} = a_{i,j} − factor · a_{k,j}
    END DO
    b_i = b_i − factor · b_k
  END DO
END DO
```

(*b*)

```
x_n = b_n / a_{n,n}
DOFOR i = n − 1, 1, −1
  sum = b_i
  DOFOR j = i + 1, n
    sum = sum − a_{i,j} · x_j
  END DO
  x_i = sum / a_{i,i}
END DO
```

Pseudocode to implement Eqs. (9.16) and (9.17) is presented in Fig. 9.4*b*. Notice the similarity between this pseudocode and that in Fig. PT3.4 for matrix multiplication. As with Fig. PT3.4, a temporary variable, `sum`, is used to accumulate the summation from Eq. (9.17). This results in a somewhat faster execution time than if the summation were accumulated in b_i. More importantly, it allows efficient improvement in precision if the variable `sum` is declared in double precision.

EXAMPLE 9.5 **Naive Gauss Elimination**

Problem Statement. Use Gauss elimination to solve

$$3x_1 - 0.1x_2 - 0.2x_3 = 7.85 \tag{E9.5.1}$$
$$0.1x_1 + 7x_2 - 0.3x_3 = -19.3 \tag{E9.5.2}$$
$$0.3x_1 - 0.2x_2 + 10x_3 = 71.4 \tag{E9.5.3}$$

Carry six significant figures during the computation.

Solution. The first part of the procedure is forward elimination. Multiply Eq. (E9.5.1) by 0.1/3 and subtract the result from Eq. (E9.5.2) to give

$$7.00333x_2 - 0.293333x_3 = -19.5617$$

Then multiply Eq. (E9.5.1) by 0.3/3 and subtract it from Eq. (E9.5.3) to eliminate x_1. After these operations, the set of equations is

$$3x_1 - 0.1x_2 - 0.2x_3 = 7.85 \tag{E9.5.4}$$
$$7.00333x_2 - 0.293333x_3 = -19.5617 \tag{E9.5.5}$$
$$-0.190000x_2 + 10.0200x_3 = 70.6150 \tag{E9.5.6}$$

To complete the forward elimination, x_2 must be removed from Eq. (E9.5.6). To accomplish this, multiply Eq. (E9.5.5) by −0.190000/7.00333 and subtract the result from Eq. (E9.5.6). This eliminates x_2 from the third equation and reduces the system to an upper triangular form, as in

$$3x_1 - 0.1x_2 - 0.2x_3 = 7.85 \tag{E9.5.7}$$
$$7.00333x_2 - 0.293333x_3 = -19.5617 \tag{E9.5.8}$$
$$10.0120x_3 = 70.0843 \tag{E9.5.9}$$

We can now solve these equations by back substitution. First, Eq. (E9.5.9) can be solved for

$$x_3 = \frac{70.0843}{10.0120} = 7.0000 \tag{E9.5.10}$$

This result can be back-substituted into Eq. (E9.5.8):

$$7.00333x_2 - 0.293333(7.0000) = -19.5617$$

which can be solved for

$$x_2 = \frac{-19.5617 + 0.293333(7.0000)}{7.00333} = -2.50000 \tag{E9.5.11}$$

Finally, Eqs. (E9.5.10) and (E9.5.11) can be substituted into Eq. (E9.5.4):

$$3x_1 - 0.1(-2.50000) - 0.2(7.0000) = 7.85$$

which can be solved for

$$x_1 = \frac{7.85 + 0.1(-2.50000) + 0.2(7.0000)}{3} = 3.00000$$

The results are identical to the exact solution of $x_1 = 3$, $x_2 = -2.5$, and $x_3 = 7$. This can be verified by substituting the results into the original equation set:

$$3(3) - 0.1(-2.5) - 0.2(7) = 7.85$$
$$0.1(3) + 7(-2.5) - 0.3(7) = -19.3$$
$$0.3(3) - 0.2(-2.5) + 10(7) = 71.4$$

9.2.1 Operation Counting

The execution time of Gauss elimination depends on the amount of *floating-point operations* (or *flops*) involved in the algorithm. On modern computers using math coprocessors, the time consumed to perform addition/subtraction and multiplication/division is about the same. Therefore, totaling up these operations provides insight into which parts of the algorithm are most time-consuming and how computation time increases as the system gets larger.

Before analyzing naive Gauss elimination, we will first define some quantities that facilitate operation counting:

$$\sum_{i=1}^{m} cf(i) = c\sum_{i=1}^{m} f(i) \qquad \sum_{i=1}^{m} f(i) + g(i) = \sum_{i=1}^{m} f(i) + \sum_{i=1}^{m} g(i) \tag{9.18a,b}$$

$$\sum_{i=1}^{m} 1 = 1 + 1 + 1 + \cdots + 1 = m \qquad \sum_{i=k}^{m} 1 = m - k + 1 \tag{9.18c,d}$$

$$\sum_{i=1}^{m} i = 1 + 2 + 3 + \cdots + m = \frac{m(m+1)}{2} = \frac{m^2}{2} + O(m) \tag{9.18e}$$

$$\sum_{i=1}^{m} i^2 = 1^2 + 2^2 + 3^2 + \cdots + m^2 = \frac{m(m+1)(2m+1)}{6} = \frac{m^3}{3} + O(m^2) \tag{9.18f}$$

where $O(m^n)$ means "terms of order m^n and lower."

Now let us examine the naive Gauss elimination algorithm (Fig. 9.4*a*) in detail. We will first count the flops in the elimination stage. On the first pass through the outer loop, $k = 1$. Therefore, the limits on the middle loop are from $i = 2$ to n. According to Eq. (9.18*d*), this means that the number of iterations of the middle loop will be

$$\sum_{i=2}^{n} 1 = n - 2 + 1 = n - 1 \tag{9.19}$$

For every one of these iterations, there is one division to define the factor. The interior loop then performs a single multiplication and subtraction for each iteration from $j = 2$ to n. Finally, there is one additional multiplication and subtraction for the right-hand-side value.

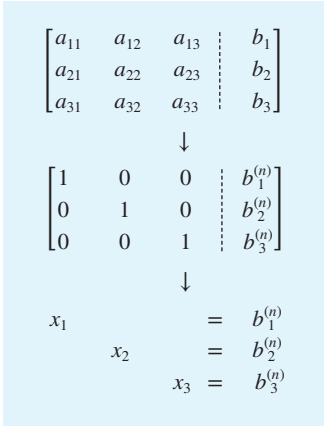


FIGURE 9.9
Graphical depiction of the Gauss-Jordan method. Compare with Fig. 9.3 to understand the differences between this technique and Gauss elimination. The superscript (n) means that the elements of the right-hand-side vector have been modified n times (for this case, $n = 3$).

Finally, the function values at i can be expressed as

$$\{F_i\}^T = \lfloor f_{1,i} \quad f_{2,i} \quad \cdots \quad f_{n,i} \rfloor$$

Using these relationships, Eq. (9.33) can be represented concisely as

$$[Z]\{X_{i+1}\} = -\{F_i\} + [Z]\{X_i\} \tag{9.35}$$

Equation (9.35) can be solved using a technique such as Gauss elimination. This process can be repeated iteratively to obtain refined estimates in a fashion similar to the two-equation case in Sec. 6.6.2.

It should be noted that there are two major shortcomings to the foregoing approach. First, Eq. (9.34) is often inconvenient to evaluate. Therefore, variations of the Newton-Raphson approach have been developed to circumvent this dilemma. As might be expected, most are based on using finite-difference approximations for the partial derivatives that comprise $[Z]$.

The second shortcoming of the multiequation Newton-Raphson method is that excellent initial guesses are usually required to ensure convergence. Because these are often difficult to obtain, alternative approaches that are slower than Newton-Raphson but which have better convergence behavior have been developed. One common approach is to reformulate the nonlinear system as a single function

$$F(x) = \sum_{i=1}^{n} [f_i(x_1, x_2, \ldots, x_n)]^2 \tag{9.36}$$

where $f_i(x_1, x_2, \ldots, x_n)$ is the ith member of the original system of Eq. (9.31). The values of x that minimize this function also represent the solution of the nonlinear system. As we will see in Chap. 17, this reformulation is used for a class of problems called *nonlinear regression*. As such, it can be approached with a number of optimization techniques such as the ones described later in this text (Part Four and specifically Chap. 14).

9.7 GAUSS-JORDAN

The Gauss-Jordan method is a variation of Gauss elimination. The major difference is that when an unknown is eliminated in the Gauss-Jordan method, it is eliminated from all other equations rather than just the subsequent ones. In addition, all rows are normalized by dividing them by their pivot elements. Thus, the elimination step results in an identity matrix rather than a triangular matrix (Fig. 9.9). Consequently, it is not necessary to employ back substitution to obtain the solution. The method is best illustrated by an example.

EXAMPLE 9.12 Gauss-Jordan Method

Problem Statement. Use the Gauss-Jordan technique to solve the same system as in Example 9.5:

$$3x_1 - 0.1x_2 - 0.2x_3 = 7.85$$
$$0.1x_1 + 7x_2 - 0.3x_3 = -19.3$$
$$0.3x_1 - 0.2x_2 + 10x_3 = 71.4$$

Solution. First, express the coefficients and the constants on the right-hand side as an augmented matrix:

$$\begin{bmatrix} 3 & -0.1 & -0.2 & 7.85 \\ 0.1 & 7 & -0.3 & -19.3 \\ 0.3 & -0.2 & 10 & 71.4 \end{bmatrix}$$

Then normalize the first row by dividing it by the pivot element, 3, to yield

$$\begin{bmatrix} 1 & -0.0333333 & -0.066667 & 2.61667 \\ 0.1 & 7 & -0.3 & -19.3 \\ 0.3 & -0.2 & 10 & 71.4 \end{bmatrix}$$

The x_1 term can be eliminated from the second row by subtracting 0.1 times the first row from the second row. Similarly, subtracting 0.3 times the first row from the third row will eliminate the x_1 term from the third row:

$$\begin{bmatrix} 1 & -0.0333333 & -0.066667 & 2.61667 \\ 0 & 7.00333 & -0.293333 & -19.5617 \\ 0 & -0.190000 & 10.0200 & 70.6150 \end{bmatrix}$$

Next, normalize the second row by dividing it by 7.00333:

$$\begin{bmatrix} 1 & -0.0333333 & -0.066667 & 2.61667 \\ 0 & 1 & -0.0418848 & -2.79320 \\ 0 & -0.190000 & 10.0200 & 70.6150 \end{bmatrix}$$

Reduction of the x_2 terms from the first and third equations gives

$$\begin{bmatrix} 1 & 0 & -0.0680629 & 2.52356 \\ 0 & 1 & -0.0418848 & -2.79320 \\ 0 & 0 & 10.01200 & 70.0843 \end{bmatrix}$$

The third row is then normalized by dividing it by 10.0120:

$$\begin{bmatrix} 1 & 0 & -0.0680629 & 2.52356 \\ 0 & 1 & -0.0418848 & -2.79320 \\ 0 & 0 & 1 & 7.0000 \end{bmatrix}$$

Finally, the x_3 terms can be reduced from the first and the second equations to give

$$\begin{bmatrix} 1 & 0 & 0 & 3.0000 \\ 0 & 1 & 0 & -2.5000 \\ 0 & 0 & 1 & 7.0000 \end{bmatrix}$$

Thus, as depicted in Fig. 9.9, the coefficient matrix has been transformed to the identity matrix, and the solution is obtained in the right-hand-side vector. Notice that no back substitution was required to obtain the solution.

All the material in this chapter regarding the pitfalls and improvements in Gauss elimination also applies to the Gauss-Jordan method. For example, a similar pivoting strategy can be used to avoid division by zero and to reduce round-off error.

An algorithm to implement Gauss-Jordan for an augmented matrix (i.e., with the right-hand-side constants appended as an additional column to the coefficient matrix) without partial pivoting is displayed in Fig. 9.10. Although the Gauss-Jordan technique and Gauss elimination might appear almost identical, the former requires more work. Using a similar approach to Sec. 9.2.1, it can be determined that the number of flops involved in naive Gauss-Jordan is

$$n^3 + 3n^2 - 2n \xrightarrow{\text{as } n \text{ increases}} n^3 + O(n^2) \tag{9.37}$$

Thus, Gauss-Jordan involves approximately 50 percent more operations than Gauss elimination [compare with Eq. (9.23)]. Therefore, Gauss elimination is the simple elimination method of preference for obtaining solutions of linear algebraic equations. One of the primary reasons that we have introduced the Gauss-Jordan, however, is that it is still used in engineering as well as in some numerical algorithms.

FIGURE 9.10
Pseudocode to implement the Gauss-Jordan algorithm without partial pivoting.

```
SUB GaussJordan(aug, m, n, x)
DOFOR k = 1, m
  d = aug(k, k)
  DOFOR j = 1, n
    aug(k, j) = aug(k, j)/d
  END DO
  DOFOR i = 1, m
     IF i ≠ k THEN
       d = aug(i, k)
       DOFOR j = k, n
         aug(i, j) = aug(i, j) – d*aug(k, j)
       END DO
     END IF
  END DO
END DO
DOFOR k = 1, m
  x(k) = aug(k, n)
END DO
END GaussJordan
```

9.8 SUMMARY

In summary, we have devoted most of this chapter to Gauss elimination, the most fundamental method for solving simultaneous linear algebraic equations. Although it is one of the earliest techniques developed for this purpose, it is nevertheless an extremely effective algorithm for obtaining solutions for many engineering problems. Aside from this practical utility, focusing on this method also provided a context for our discussion of

general issues such as round-off, scaling, and conditioning. In addition, we briefly presented material on the Gauss-Jordan method, as well as complex and nonlinear systems.

Answers obtained using Gauss elimination may be checked by substituting them into the original equations. However, this does not always represent a reliable check for ill-conditioned systems. Therefore, some measure of condition, such as the determinant of the scaled system, should be computed if round-off error is suspected. Using partial pivoting and more significant figures in the computation are two options for mitigating round-off error. In the next chapter, we will return to the topic of system condition when we discuss the matrix inverse.

PROBLEMS

9.1

(a) Write the following set of equations in matrix form:

$$40 = 5x_3 + 2x_1$$
$$10 - x_2 = x_3$$
$$4x_2 + 8x_1 = 21$$

(b) Multiply the matrix of coefficients by its transpose; i.e., $[A][A]^T$.

9.2 A number of matrices are defined as

$$[A] = \begin{bmatrix} 4 & 5 \\ 1 & 2 \\ 5 & 6 \end{bmatrix} \quad [B] = \begin{bmatrix} 4 & 3 & 7 \\ 1 & 2 & 6 \\ 2 & 0 & 4 \end{bmatrix}$$

$$\{C\} = \begin{Bmatrix} 3 \\ 5 \\ 1 \end{Bmatrix} \quad [D] = \begin{bmatrix} 9 & 4 & 3 & -6 \\ 2 & -1 & 6 & 5 \end{bmatrix}$$

$$[E] = \begin{bmatrix} 1 & 5 & 9 \\ 7 & 2 & 3 \\ 4 & 0 & 6 \end{bmatrix}$$

$$[F] = \begin{bmatrix} 2 & 0 & 1 \\ 1 & 7 & 3 \end{bmatrix} \quad \lfloor G \rfloor = \lfloor 7 \quad 5 \quad 4 \rfloor$$

Answer the following questions regarding these matrices:

(a) What are the dimensions of the matrices?

(b) Identify the square, column, and row matrices.

(c) What are the values of the elements: a_{12}, b_{23}, d_{32}, e_{22}, f_{12}, g_{12}?

(d) Perform the following operations:

(1) $[E] + [B]$
(2) $[A] \times [F]$
(3) $[B] - [E]$
(4) $8 \times [B]$
(5) $[A] \times [B]$
(6) $\{C\}^T$
(7) $[B] \times [A]$
(8) $[D]^T$
(9) $[A] \times \{C\}$
(10) $[I] \times [B]$
(11) $[E]^T[E]$
(12) $\{C\}^T\{C\}$

9.3 Three matrices are defined as

$$[A] = \begin{bmatrix} 1 & 5 \\ 3 & 10 \\ -4 & 3 \end{bmatrix} \quad [B] = \begin{bmatrix} 4 & 3 \\ 0.5 & 2 \end{bmatrix} \quad [C] = \begin{bmatrix} 2 & -2 \\ -3 & 5 \end{bmatrix}$$

(a) Perform all possible multiplications that can be computed between pairs of these matrices.

(b) Use the method in Box PT3.2 to justify why the remaining pairs cannot be multiplied.

(c) Use the results of **(a)** to illustrate why the order of multiplication is important.

9.4 Use the graphical method to solve

$$2x_1 - 6x_2 = -18$$
$$-x_1 + 8x_2 = 40$$

Check your results by substituting them back into the equations.

9.5 Given the system of equations

$$0.77x_1 + x_2 = 14.25$$
$$1.2x_1 + 1.7x_2 = 20$$

(a) Solve graphically and check your results by substituting them back into the equations.

(b) On the basis of the graphical solution, what do you expect regarding the condition of the system?

(c) Compute the determinant.

(d) Solve by the elimination of unknowns.

9.6 For the set of equations

$$2x_2 + 5x_3 = 1$$
$$2x_1 + x_2 + x_3 = 1$$
$$3x_1 + x_2 = 2$$

(a) Compute the determinant.

(b) Use Cramer's rule to solve for the x's.

(c) Substitute your results back into the original equations to check your results.

CHAPTER

17

Least-Squares Regression

Where substantial error is associated with data, polynomial interpolation is inappropriate and may yield unsatisfactory results when used to predict intermediate values. Experimental data are often of this type. For example, Fig. 17.1*a* shows seven experimentally derived data points exhibiting significant variability. Visual inspection of these data suggests a positive relationship between y and x. That is, the overall trend indicates that higher values of y are associated with higher values of x. Now, if a sixth-order interpolating polynomial is fitted to these data (Fig. 17.1*b*), it will pass exactly through all of the points. However, because of the variability in these data, the curve oscillates widely in the interval between the points. In particular, the interpolated values at $x = 1.5$ and $x = 6.5$ appear to be well beyond the range suggested by these data.

A more appropriate strategy for such cases is to derive an approximating function that fits the shape or general trend of the data without necessarily matching the individual points. Figure 17.1*c* illustrates how a straight line can be used to generally characterize the trend of these data without passing through any particular point.

One way to determine the line in Fig. 17.1*c* is to visually inspect the plotted data and then sketch a "best" line through the points. Although such "eyeball" approaches have commonsense appeal and are valid for "back-of-the-envelope" calculations, they are deficient because they are arbitrary. That is, unless the points define a perfect straight line (in which case, interpolation would be appropriate), different analysts would draw different lines.

To remove this subjectivity, some criterion must be devised to establish a basis for the fit. One way to do this is to derive a curve that minimizes the discrepancy between the data points and the curve. A technique for accomplishing this objective, called *least-squares regression,* will be discussed in the present chapter.

17.1 LINEAR REGRESSION

The simplest example of a least-squares approximation is fitting a straight line to a set of paired observations: (x_1, y_1), (x_2, y_2), . . . , (x_n, y_n). The mathematical expression for the straight line is

$$y = a_0 + a_1 x + e \tag{17.1}$$

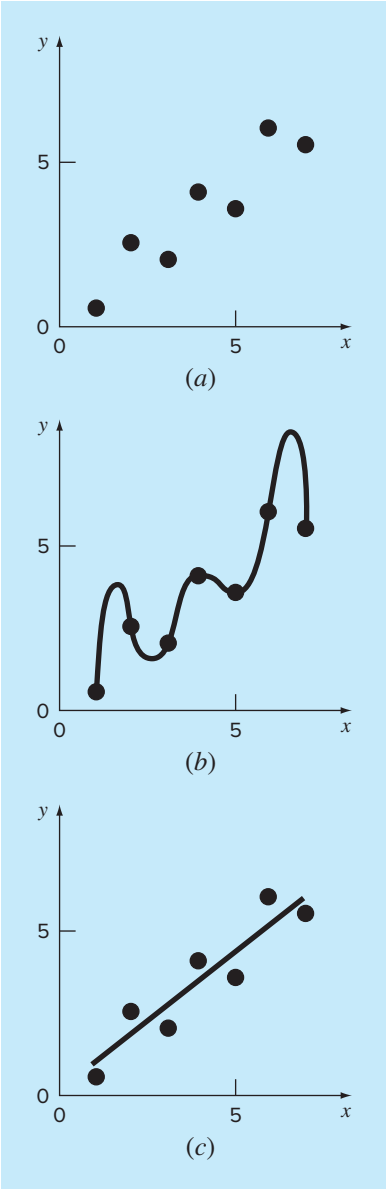


FIGURE 17.1
(*a*) Data exhibiting significant error. (*b*) Polynomial fit oscillating beyond the range of the data. (*c*) More satisfactory result using the least-squares fit.

where a_0 and a_1 are coefficients representing the intercept and the slope, respectively, and e is the error, or residual, between the model and the observations, which can be represented by rearranging Eq. (17.1) as

$$e = y - a_0 - a_1 x$$

Thus, the error, or *residual,* is the discrepancy between the true value of y and the approximate value, $a_0 + a_1 x$, predicted by the linear equation.

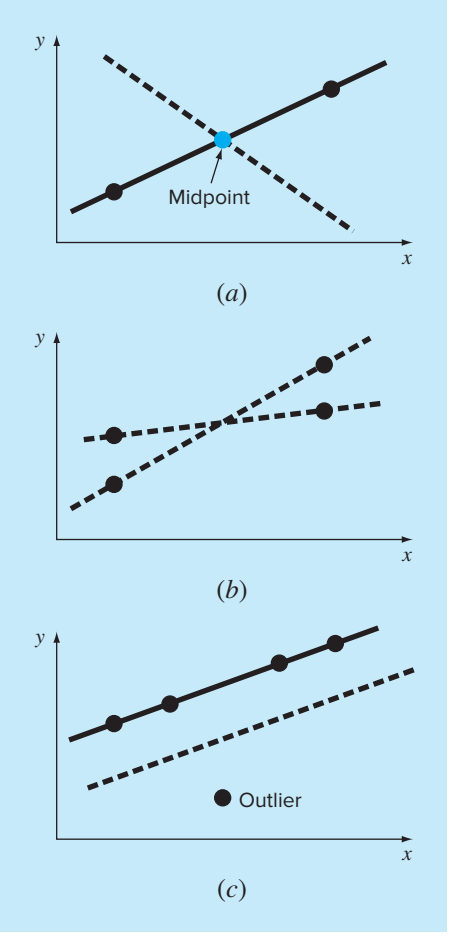
17.1.1 Criteria for a "Best" Fit

One strategy for fitting a "best" line through the data would be to minimize the sum of the residual errors for all the available data, as in

$$\sum_{i=1}^{n} e_i = \sum_{i=1}^{n} (y_i - a_0 - a_1 x_i) \tag{17.2}$$

where n = total number of points. However, this is an inadequate criterion, as illustrated by Fig. 17.2*a* which depicts the fit of a straight line to two points. Obviously, the best

FIGURE 17.2
Examples of some criteria for "best fit" that are inadequate for regression: (*a*) minimizes the sum of the residuals, (*b*) minimizes the sum of the absolute values of the residuals, and (*c*) minimizes the maximum error of any individual point.



fit is the line connecting the points. However, any straight line passing through the midpoint of the connecting line (except a perfectly vertical line) results in a minimum value of Eq. (17.2) equal to zero because the errors cancel.

Therefore, another logical criterion might be to minimize the sum of the absolute values of the discrepancies, as in

$$\sum_{i=1}^{n} |e_i| = \sum_{i=1}^{n} |y_i - a_0 - a_1 x_i|$$

Figure 17.2*b* demonstrates why this criterion is also inadequate. For the four points shown, any straight line falling within the dashed lines will minimize the sum of the absolute values. Thus, this criterion also does not yield a unique best fit.

A third strategy for fitting a best line is the *minimax* criterion. In this technique, the line is chosen that minimizes the maximum distance that an individual point falls from the line. As depicted in Fig. 17.2*c*, this strategy is ill-suited for regression because it gives undue influence to an outlier, that is, a single point with a large error. It should be noted that the minimax principle is sometimes well-suited for fitting a simple function to a complicated function (Carnahan, Luther, and Wilkes 1969).

A strategy that overcomes the shortcomings of the aforementioned approaches is to minimize the sum of the squares of the residuals between the measured y and the y calculated with the linear model,

$$S_r = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_{i,\text{measured}} - y_{i,\text{model}})^2 = \sum_{i=1}^{n} (y_i - a_0 - a_1 x_i)^2 \tag{17.3}$$

This criterion has a number of advantages, including the fact that it yields a unique line for a given set of data. Before discussing these properties, we will present a technique for determining the values of a_0 and a_1 that minimize the result of Eq. (17.3).

17.1.2 Least-Squares Fit of a Straight Line

To determine values for a_0 and a_1, Eq. (17.3) is differentiated with respect to each coefficient:

$$\frac{\partial S_r}{\partial a_0} = -2 \sum (y_i - a_0 - a_1 x_i)$$

$$\frac{\partial S_r}{\partial a_1} = -2 \sum [(y_i - a_0 - a_1 x_i) x_i]$$

Note that we have simplified the summation symbols; unless otherwise indicated, all summations are from $i = 1$ to n. Setting these derivatives equal to zero will result in a minimum S_r. If this is done, the equations can be expressed as

$$0 = \sum y_i - \sum a_0 - \sum a_1 x_i$$

$$0 = \sum y_i x_i - \sum a_0 x_i - \sum a_1 x_i^2$$

Now, realizing that $\Sigma a_0 = na_0$, we can express the equations as a set of two simultaneous linear equations with two unknowns (a_0 and a_1):

$$na_0 + \left(\sum x_i\right)a_1 = \sum y_i \tag{17.4}$$

$$\left(\sum x_i\right)a_0 + \left(\sum x_i^2\right)a_1 = \sum x_i y_i \tag{17.5}$$

These are called the *normal equations*. They can be solved simultaneously

$$a_1 = \frac{n\Sigma x_i y_i - \Sigma x_i \Sigma y_i}{n\Sigma x_i^2 - (\Sigma x_i)^2} \tag{17.6}$$

This result can then be used in conjunction with Eq. (17.4) to solve for

$$a_0 = \bar{y} - a_1\bar{x} \tag{17.7}$$

where $\bar{y}$ and $\bar{x}$ are the means of y and x, respectively.

EXAMPLE 17.1 **Linear Regression**

Problem Statement. Fit a straight line to the x and y values in the first two columns of Table 17.1.

Solution. The following quantities can be computed:

$$n = 7 \qquad \sum x_i y_i = 119.5 \qquad \sum x_i^2 = 140$$

$$\sum x_i = 28 \qquad \bar{x} = \frac{28}{7} = 4$$

$$\sum y_i = 24 \qquad \bar{y} = \frac{24}{7} = 3.428571$$

Using Eqs. (17.6) and (17.7),

$$a_1 = \frac{7(119.5) - 28(24)}{7(140) - (28)^2} = 0.8392857$$

$$a_0 = 3.428571 - 0.8392857(4) = 0.07142857$$

TABLE 17.1 Computations for an error analysis of the linear fit.

x_i	y_i	$(y_i - \bar{y})$	$(y_i - a_0 - a_1x_i)^2$
1	0.5	8.5765	0.1687
2	2.5	0.8622	0.5625
3	2.0	2.0408	0.3473
4	4.0	0.3265	0.3265
5	3.5	0.0051	0.5896
6	6.0	6.6122	0.7972
7	5.5	4.2908	0.1993
Σ	24.0	22.7143	2.9911

Therefore, the least-squares fit is

$$y = 0.07142857 + 0.8392857x$$

The line, along with the data, is shown in Fig. 17.1*c*.

17.1.3 Quantification of Error of Linear Regression

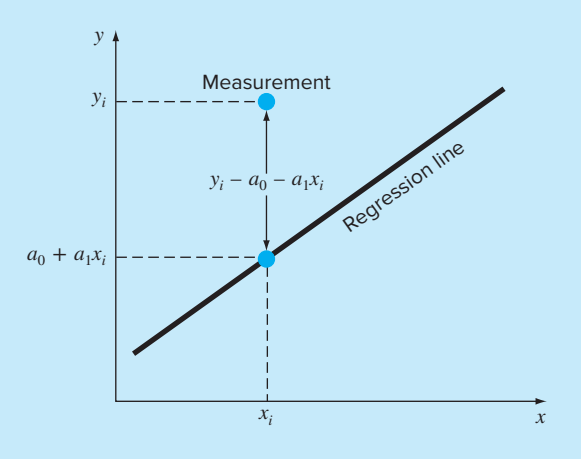
Any line other than the one computed in Example 17.1 results in a larger sum of the squares of the residuals. Thus, the line is unique and in terms of our chosen criterion is a "best" line through the points. A number of additional properties of this fit can be elucidated by examining more closely the way in which residuals were computed. Recall that the sum of the squares is defined as [Eq. (17.3)]

$$S_r = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (y_i - a_0 - a_1 x_i)^2 \tag{17.8}$$

Notice the similarity between Eqs. (PT5.3) and (17.8). In the former case, the square of the residual represented the square of the discrepancy between the data and a single estimate of the measure of central tendency—the mean. In Eq. (17.8), the square of the residual represents the square of the vertical distance between the data and another measure of central tendency—the straight line (Fig. 17.3).

The analogy can be extended further for cases where (1) the spread of the points around the line is of similar magnitude along the entire range of the data and (2) the distribution of these points about the line is normal. It can be demonstrated that if these criteria are met, least-squares regression will provide the best (that is, the most likely) estimates of a_0 and a_1 (Draper and Smith 1981). This is called the *maximum likelihood*

FIGURE 17.3
The residual in linear regression represents the vertical distance between a data point and the straight line.



principle in statistics. In addition, if these criteria are met, a "standard deviation" for the regression line can be determined as [compare with Eq. (PT5.2)]

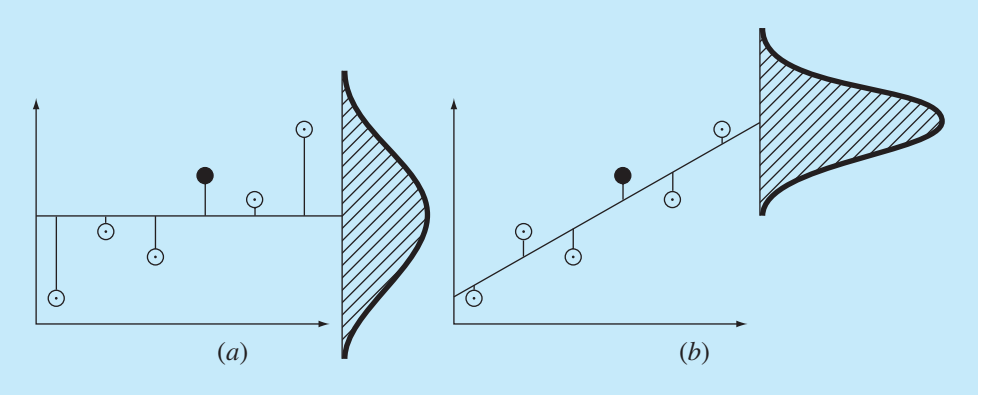
$$s_{y/x} = \sqrt{\frac{S_r}{n-2}} \tag{17.9}$$

where $s_{y/x}$ is called the *standard error of the estimate*. The subscript notation "y/x" designates that the error is for a predicted value of y corresponding to a particular value of x. Also, notice that we now divide by $n - 2$ because two data-derived estimates—a_0 and a_1—were used to compute S_r; thus, we have lost two degrees of freedom. As in our discussion of the standard deviation in PT5.2.1, another justification for dividing by $n - 2$ is that there is no such thing as the "spread of data" around a straight line connecting two points. Thus, for the case where $n = 2$, Eq. (17.9) yields a meaningless result of infinity.

Just as was the case with the standard deviation, the standard error of the estimate quantifies the spread of the data. However, $s_{y/x}$ quantifies the spread *around the regression line,* as shown in Fig. 17.4*b*, in contrast to the original standard deviation s_y, which quantified the spread *around the mean* (Fig. 17.4*a*).

The above concepts can be used to quantify the "goodness" of our fit. This is particularly useful for comparison of several regressions (Fig. 17.5). To do this, we return to the original data and determine the *total sum of the squares* around the mean for the dependent variable (in our case, y). As was the case for Eq. (PT5.3), this quantity is designated S_t. This is the magnitude of the residual error associated with the dependent variable prior to regression. After performing the regression, we can compute S_r, the sum of the squares of the residuals around the regression line. This characterizes the residual error that remains after the regression. It is, therefore, sometimes called the unexplained

FIGURE 17.4
Regression data showing (*a*) the spread of the data around the mean of the dependent variable and (*b*) the spread of the data around the best-fit line. The reduction in the spread in going from (*a*) to (*b*), as indicated by the bell-shaped curves at the right, represents the improvement due to linear regression.



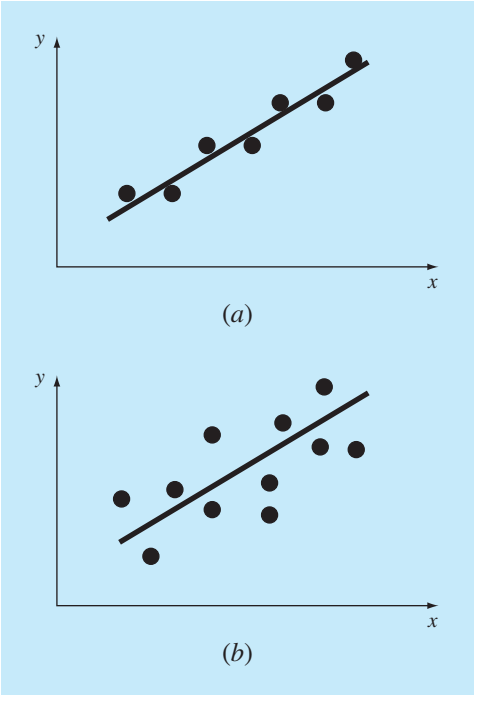


FIGURE 17.5
Examples of linear regression with (*a*) small and (*b*) large residual errors.

sum of the squares. The difference between the two quantities, $S_t - S_r$, quantifies the improvement, or error reduction, due to describing the data in terms of a straight line rather than as an average value. Because the magnitude of this quantity is scale-dependent, the difference is normalized to S_t to yield

$$r^2 = \frac{S_t - S_r}{S_t} \tag{17.10}$$

where r^2 is called the *coefficient of determination* and r is the *correlation coefficient* $(=\sqrt{r^2})$. For a perfect fit, $S_r = 0$ and $r = r^2 = 1$, signifying that the line explains 100 percent of the variability of the data. For $r = r^2 = 0$, $S_r = S_t$ and the fit represents no improvement. An alternative formulation for r that is more convenient for computer implementation is

$$r = \frac{n\Sigma x_i y_i - (\Sigma x_i)(\Sigma y_i)}{\sqrt{n\Sigma x_i^2 - (\Sigma x_i)^2}\sqrt{n\Sigma y_i^2 - (\Sigma y_i)^2}} \tag{17.11}$$

Thus, the intercept, $\log \alpha_2$, equals -0.300, and therefore, by taking the antilogarithm, we get $\alpha_2 = 10^{-0.3} = 0.5$. The slope is $\beta_2 = 1.75$. Consequently, the power equation is

$$y = 0.5x^{1.75}$$

This curve, as plotted in Fig. 17.10*a*, indicates a good fit.

17.1.6 General Comments on Linear Regression

Before proceeding to curvilinear and multiple linear regression, we must emphasize the introductory nature of the foregoing material on linear regression. We have focused on the simple derivation and practical use of equations to fit data. You should be cognizant of the fact that there are theoretical aspects of regression that are of practical importance but are beyond the scope of this book. For example, some statistical assumptions that are inherent in the linear least-squares procedures are

1. Each x has a fixed value; it is not random and is known without error.
2. The y values are independent random variables and all have the same variance.
3. The y values for a given x must be normally distributed.

Such assumptions are relevant to the proper derivation and use of regression. For example, the first assumption means that (1) the x values must be error-free and (2) the regression of y versus x is not the same as x versus y (try Prob. 17.4 at the end of the chapter). You are urged to consult other references such as Draper and Smith (1981) to appreciate aspects and nuances of regression that are beyond the scope of this book.

17.2 POLYNOMIAL REGRESSION

In Sec. 17.1, a procedure was developed to derive the equation of a straight line using the least-squares criterion. Some engineering data, although exhibiting a marked pattern such as seen in Fig. 17.8, are poorly represented by a straight line. For these cases, a curve would be better suited to fit these data. As discussed in the previous section, one method to accomplish this objective is to use transformations. Another alternative is to fit polynomials to the data using *polynomial regression.*

The least-squares procedure can be readily extended to fit the data to a higher-order polynomial. For example, suppose that we fit a second-order polynomial or quadratic:

$$y = a_0 + a_1 x + a_2 x^2 + e$$

For this case the sum of the squares of the residuals is [compare with Eq. (17.3)]

$$S_r = \sum_{i=1}^{n} (y_i - a_0 - a_1 x_i - a_2 x_i^2)^2 \tag{17.18}$$

Following the procedure of the previous section, we take the derivative of Eq. (17.18) with respect to each of the unknown coefficients of the polynomial, as in

$$\frac{\partial S_r}{\partial a_0} = -2\sum (y_i - a_0 - a_1 x_i - a_2 x_i^2)$$

$$\frac{\partial S_r}{\partial a_1} = -2\sum x_i(y_i - a_0 - a_1x_i - a_2x_i^2)$$

$$\frac{\partial S_r}{\partial a_2} = -2\sum x_i^2(y_i - a_0 - a_1x_i - a_2x_i^2)$$

These equations can be set equal to zero and rearranged to develop the following set of normal equations:

$$\begin{aligned}(n)a_0 + \left(\sum x_i\right)a_1 + \left(\sum x_i^2\right)a_2 &= \sum y_i \\ \left(\sum x_i\right)a_0 + \left(\sum x_i^2\right)a_1 + \left(\sum x_i^3\right)a_2 &= \sum x_iy_i \\ \left(\sum x_i^2\right)a_0 + \left(\sum x_i^3\right)a_1 + \left(\sum x_i^4\right)a_2 &= \sum x_i^2y_i\end{aligned} \tag{17.19}$$

where all summations are from $i = 1$ through n. Note that the above three equations are linear and have three unknowns: a_0, a_1, and a_2. The coefficients of the unknowns can be calculated directly from the observed data.

For this case, we see that the problem of determining a least-squares second-order polynomial is equivalent to solving a system of three simultaneous linear equations. Techniques to solve such equations were discussed in Part Three.

The two-dimensional case can be easily extended to an mth-order polynomial:

$$y = a_0 + a_1x + a_2x^2 + \cdots + a_mx^m + e$$

The foregoing analysis can be easily extended to this more general case. Thus, we can recognize that determining the coefficients of an mth-order polynomial is equivalent to solving a system of $m + 1$ simultaneous linear equations. For this case, the standard error is formulated as

$$s_{y/x} = \sqrt{\frac{S_r}{n - (m + 1)}} \tag{17.20}$$

This quantity is divided by $n - (m + 1)$ because $(m + 1)$ data-derived coefficients—a_0, a_1, . . . , a_m—were used to compute S_r; thus, we have lost $m + 1$ degrees of freedom. In addition to the standard error, a coefficient of determination can also be computed for polynomial regression with Eq. (17.10).

EXAMPLE 17.5 Polynomial Regression

Problem Statement. Fit a second-order polynomial to the data in the first two columns of Table 17.4.

Solution. From the given data,

$$\begin{aligned} m &= 2 & \sum x_i &= 15 & \sum x_i^4 &= 979 \\ n &= 6 & \sum y_i &= 152.6 & \sum x_iy_i &= 585.6 \\ \bar{x} &= 2.5 & \sum x_i^2 &= 55 & \sum x_i^2y_i &= 2488.8 \\ \bar{y} &= 25.433 & \sum x_i^3 &= 225 \end{aligned}$$

TABLE 17.4 Computations for an error analysis of the quadratic least-squares fit.

x_i	y_i	$(y_i - \bar{y})^2$	$(y_i - a_0 - a_1x_i - a_2x_i^2)^2$
0	2.1	544.44	0.14332
1	7.7	314.47	1.00286
2	13.6	140.03	1.08158
3	27.2	3.12	0.80491
4	40.9	239.22	0.61951
5	61.1	1272.11	0.09439
Σ	152.6	2513.39	3.74657

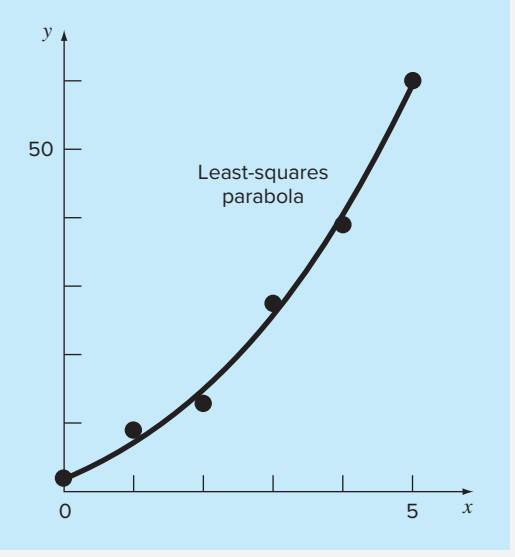


FIGURE 17.11
Fit of a second-order polynomial.

Therefore, the simultaneous linear equations are

$$\begin{bmatrix} 6 & 15 & 55 \\ 15 & 55 & 225 \\ 55 & 225 & 979 \end{bmatrix} \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \end{Bmatrix} = \begin{Bmatrix} 152.6 \\ 585.6 \\ 2488.8 \end{Bmatrix}$$

Solving these equations through a technique such as Gauss elimination gives $a_0 = 2.47857$, $a_1 = 2.35929$, and $a_2 = 1.86071$. Therefore, the least-squares quadratic equation for this case is

$$y = 2.47857 + 2.35929x + 1.86071x^2$$

The standard error of the estimate based on the regression polynomial is [Eq. (17.20)]

$$s_{y/x} = \sqrt{\frac{3.74657}{6 - 3}} = 1.12$$

The coefficient of determination is

$$r^2 = \frac{2513.39 - 3.74657}{2513.39} = 0.99851$$

and the correlation coefficient is $r = 0.99925$.

These results indicate that 99.851% of the original uncertainty has been explained by the model. This result supports the conclusion that the quadratic equation represents an excellent fit, as is also evident from Fig. 17.11.

17.2.1 Algorithm for Polynomial Regression

An algorithm for polynomial regression is delineated in Fig. 17.12. Note that the primary task is the generation of the coefficients of the normal equations [Eq. (17.19)]. (Pseudocode for accomplishing this is presented in Fig. 17.13.) Then, techniques from Part Three can be applied to solve these simultaneous equations for the coefficients.

A potential problem associated with implementing polynomial regression on the computer is that the normal equations tend to be ill-conditioned. This is particularly true for higher-order versions. For these cases, the computed coefficients may be highly susceptible to round-off error, and consequently, the results can be inaccurate. Among other things, this problem is related to the structure of the normal equations and to the fact that for higher-order polynomials the normal equations can have very large and very small coefficients. This is because the coefficients are summations of the data raised to powers.

Although the strategies for mitigating round-off error discussed in Part Three, such as pivoting, can help to partially remedy this problem, a simpler alternative is to use a computer with higher precision. Fortunately, most practical problems are limited to lower-order polynomials for which round-off is usually negligible. In situations where higher-order versions are required, other alternatives are available for certain types of data. However, these techniques (such as orthogonal polynomials) are beyond the scope of this book. The reader should consult texts on regression, such as Draper and Smith (1981), for additional information regarding the problem and possible alternatives.

FIGURE 17.12
Algorithm for implementation of polynomial and multiple linear regression.

Step 1: Input order of polynomial to be fit, m.
Step 2: Input number of data points, n.
Step 3: If $n < m + 1$, print out an error message that regression is impossible and terminate the process. If $n \geq m + 1$, continue.
Step 4: Compute the elements of the normal equation in the form of an augmented matrix.
Step 5: Solve the augmented matrix for the coefficients $a_0, a_1, a_2, \ldots, a_m$ using an elimination method.
Step 6: Print out the coefficients.

However, they are utilized for evaluating improper integrals and for the solution of ordinary differential equations. This chapter emphasizes the closed forms. However, material on open Newton-Cotes formulas is briefly introduced at the end of this chapter.

21.1 THE TRAPEZOIDAL RULE

The *trapezoidal rule* is the first of the Newton-Cotes closed integration formulas. It corresponds to the case where the polynomial in Eq. (21.1) is first-order:

$$I = \int_a^b f(x)\,dx \cong \int_a^b f_1(x)\,dx$$

Recall from Chap. 18 that a straight line can be represented as [Eq. (18.2)]

$$f_1(x) = f(a) + \frac{f(b) - f(a)}{b - a}(x - a) \tag{21.2}$$

The area under this straight line is an estimate of the integral of $f(x)$ between the limits a and b:

$$I = \int_a^b \left[f(a) + \frac{f(b) - f(a)}{b - a}(x - a) \right] dx$$

The result of the integration (see Box 21.1 for details) is

$$I = (b - a)\frac{f(a) + f(b)}{2} \tag{21.3}$$

which is called the *trapezoidal rule.*

Box 21.1 Derivation of Trapezoidal Rule

Before integration, Eq. (21.2) can be expressed as

$$f_1(x) = \frac{f(b) - f(a)}{b - a}x + f(a) - \frac{af(b) - af(a)}{b - a}$$

Grouping the last two terms gives

$$f_1(x) = \frac{f(b) - f(a)}{b - a}x + \frac{bf(a) - af(a) - af(b) + af(a)}{b - a}$$

or

$$f_1(x) = \frac{f(b) - f(a)}{b - a}x + \frac{bf(a) - af(b)}{b - a}$$

which can be integrated between $x = a$ and $x = b$ to yield

$$I = \frac{f(b) - f(a)}{b - a}\frac{x^2}{2} + \frac{bf(a) - af(b)}{b - a}x\Big|_a^b$$

This result can be evaluated to give

$$I = \frac{f(b) - f(a)}{b - a}\frac{(b^2 - a^2)}{2} + \frac{bf(a) - af(b)}{b - a}(b - a)$$

Now, since $b^2 - a^2 = (b - a)(b + a)$,

$$I = [f(b) - f(a)]\frac{b + a}{2} + bf(a) - af(b)$$

Multiplying and collecting terms yields

$$I = (b - a)\frac{f(a) + f(b)}{2}$$

which is the formula for the trapezoidal rule.

Geometrically, the trapezoidal rule is equivalent to approximating the area of the trapezoid under the straight line connecting $f(a)$ and $f(b)$ in Fig. 21.4. Recall from geometry that the formula for computing the area of a trapezoid is the height times the average of the bases (Fig. 21.5*a*). In our case, the concept is the same but the trapezoid is on its side (Fig. 21.5*b*). Therefore, the integral estimate can be represented as

$$I \cong \text{width} \times \text{average height} \tag{21.4}$$

FIGURE 21.4
Graphical depiction of the trapezoidal rule.

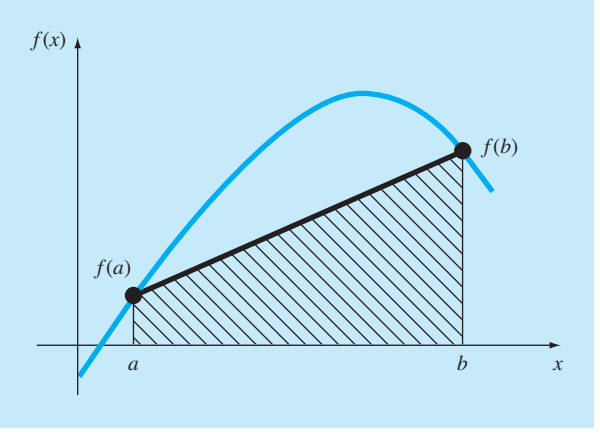
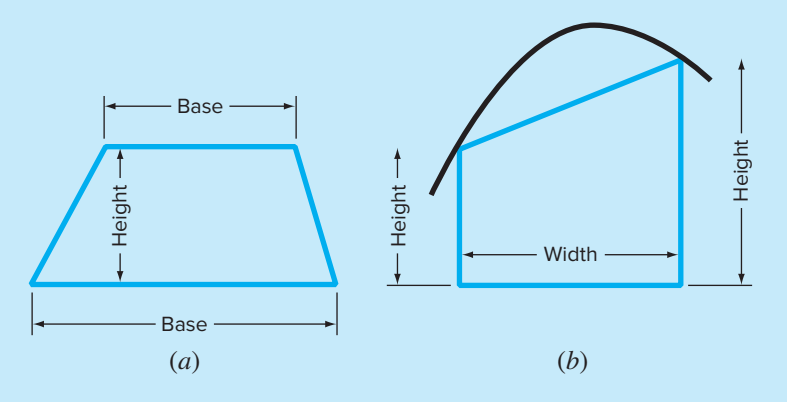


FIGURE 21.5
(*a*) The formula for computing the area of a trapezoid: height times the average of the bases. (*b*) For the trapezoidal rule, the concept is the same but the trapezoid is on its side.



or

$$I \cong (b - a) \times \text{average height} \tag{21.5}$$

where, for the trapezoidal rule, the average height is the average of the function values at the end points, or $[f(a) + f(b)]/2$.

All the Newton-Cotes closed formulas can be expressed in the general format of Eq. (21.5). In fact, they differ only with respect to the formulation of the average height.

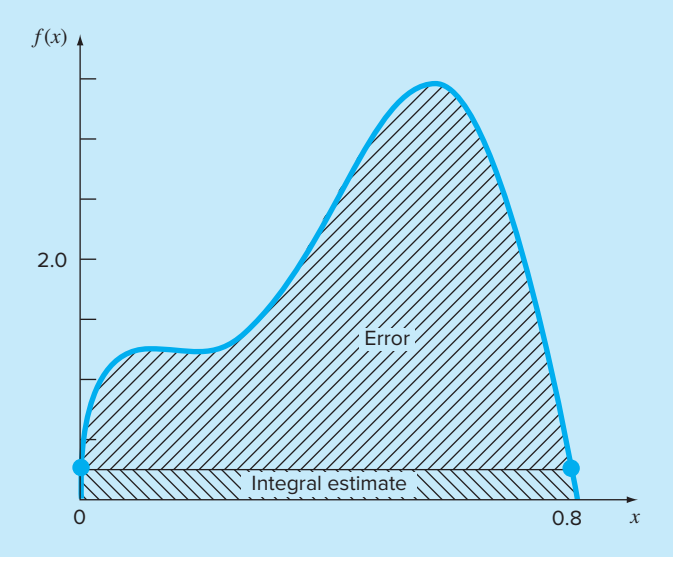
21.1.1 Error of the Trapezoidal Rule

When we employ the integral under a straight-line segment to approximate the integral under a curve, we obviously can incur an error that may be substantial (Fig. 21.6). An estimate for the local truncation error of a single application of the trapezoidal rule is (Box. 21.2)

$$E_t = -\frac{1}{12} f''(\xi)(b - a)^3 \tag{21.6}$$

where ξ lies somewhere in the interval from a to b. Equation (21.6) indicates that if the function being integrated is linear, the result from the trapezoidal rule will be exact. Otherwise, for functions with second- and higher-order derivatives (that is, with curvature), some error can occur.

FIGURE 21.6
Graphical depiction of the use of a single application of the trapezoidal rule to approximate the integral of $f(x) = 0.2 + 25x - 200x^2 + 675x^3 - 900x^4 + 400x^5$ from $x = 0$ to 0.8.



Box 21.2 Derivation and Error Estimate of the Trapezoidal Rule

An alternative derivation of the trapezoidal rule is possible by integrating the forward Newton-Gregory interpolating polynomial. Recall that for the first-order version with an error term, the integral would be (Box 18.2)

$$I = \int_a^b \left[f(a) + \Delta f(a)\alpha + \frac{f''(\xi)}{2}\alpha(\alpha - 1)h^2 \right] dx \quad \text{(B21.2.1)}$$

To simplify the analysis, realize that because $\alpha = (x - a)/h$,

$$dx = h\, d\alpha$$

Inasmuch as $h = b - a$ (for the one-segment trapezoidal rule), the limits of integration a and b correspond to 0 and 1, respectively. Therefore, Eq. (B21.2.1) can be expressed as

$$I = h\int_0^1 \left[f(a) + \Delta f(a)\alpha + \frac{f''(\xi)}{2}\alpha(\alpha - 1)h^2 \right] d\alpha$$

If it is assumed that, for small h, the term $f''(\xi)$ is approximately constant, this equation can be integrated:

$$I = h\left[\alpha f(a) + \frac{\alpha^2}{2}\Delta f(a) + \left(\frac{\alpha^3}{6} - \frac{\alpha^2}{4}\right) f''(\xi)h^2 \right]_0^1$$

and evaluated as

$$I = h\left[f(a) + \frac{\Delta f(a)}{2} \right] - \frac{1}{12} f''(\xi)h^3$$

Because $\Delta f(a) = f(b) - f(a)$, the result can be written as

$$I = \underbrace{h\frac{f(a) + f(b)}{2}}_{\text{Trapezoidal rule}} \underbrace{- \frac{1}{12} f''(\xi)h^3}_{\text{Truncation error}}$$

Thus, the first term is the trapezoidal rule and the second is an approximation for the error.

EXAMPLE 21.1 Single Application of the Trapezoidal Rule

Problem Statement. Use Eq. (21.3) to numerically integrate

$$f(x) = 0.2 + 25x - 200x^2 + 675x^3 - 900x^4 + 400x^5$$

from $a = 0$ to $b = 0.8$. Recall from Sec. PT6.2 that the exact value of the integral can be determined analytically to be 1.640533.

Solution. The function values

$$f(0) = 0.2$$
$$f(0.8) = 0.232$$

can be substituted into Eq. (21.3) to yield

$$I \cong 0.8\frac{0.2 + 0.232}{2} = 0.1728$$

which represents an error of

$$E_t = 1.640533 - 0.1728 = 1.467733$$

which corresponds to a percent relative error of $\varepsilon_t = 89.5\%$. The reason for this large error is evident from the graphical depiction in Fig. 21.6. Notice that the area under the straight line neglects a significant portion of the integral lying above the line.

In actual situations, we would have no foreknowledge of the true value. Therefore, an approximate error estimate is required. To obtain this estimate, the function's second

derivative over the interval can be computed by differentiating the original function twice to give

$$f''(x) = -400 + 4050x - 10{,}800x^2 + 8000x^3$$

The average value of the second derivative can be computed using Eq. (PT6.4):

$$\bar{f}''(x) = \frac{\int_0^{0.8} (-400 + 4050x - 10{,}800x^2 + 8000x^3)\,dx}{0.8 - 0} = -60$$

which can be substituted into Eq. (21.6) to yield

$$E_a = -\frac{1}{12}(-60)(0.8)^3 = 2.56$$

which is of the same order of magnitude and sign as the true error. A discrepancy does exist, however, because of the fact that for an interval of this size, the average second derivative is not necessarily an accurate approximation of $f''(\xi)$. Thus, we denote that the error is approximate by using the notation E_a, rather than using E_t.

21.1.2 The Multiple-Application Trapezoidal Rule

One way to improve the accuracy of the trapezoidal rule is to divide the integration interval from a to b into a number of segments and apply the method to each segment (Fig. 21.7). The areas of individual segments can then be added to yield the integral for the entire interval. The resulting equations are called *multiple-application,* or *composite, integration formulas*.

Figure 21.8 shows the general format and nomenclature we will use to characterize multiple-application integrals. There are $n + 1$ equally spaced base points (x_0, x_1, x_2, . . . , x_n). Consequently, there are n segments of equal width:

$$h = \frac{b - a}{n} \tag{21.7}$$

If a and b are designated as x_0 and x_n, respectively, the total integral can be represented as

$$I = \int_{x_0}^{x_1} f(x)\,dx + \int_{x_1}^{x_2} f(x)\,dx + \cdots + \int_{x_{n-1}}^{x_n} f(x)\,dx$$

Substituting the trapezoidal rule for each integral yields

$$I = h\frac{f(x_0) + f(x_1)}{2} + h\frac{f(x_1) + f(x_2)}{2} + \cdots + h\frac{f(x_{n-1}) + f(x_n)}{2} \tag{21.8}$$

or, grouping terms,

$$I = \frac{h}{2}\left[f(x_0) + 2\sum_{i=1}^{n-1} f(x_i) + f(x_n)\right] \tag{21.9}$$

- Finally, round-off errors can limit our ability to determine integrals. This is due both to the machine precision as well as to the numerous computations involved in simple techniques like the multiple-segment trapezoidal rule.

We now turn to one way in which efficiency is improved: by using higher-order polynomials to approximate the integral.

21.2 SIMPSON'S RULES

Aside from applying the trapezoidal rule with finer segmentation, another way to obtain a more accurate estimate of an integral is to use higher-order polynomials to connect the points. For example, if there is an extra point midway between $f(a)$ and $f(b)$, the three points can be connected with a parabola (Fig. 21.10*a*). If there are two points equally spaced between $f(a)$ and $f(b)$, the four points can be connected with a third-order polynomial (Fig. 21.10*b*). The formulas that result from taking the integrals under these polynomials are called *Simpson's rules*.

21.2.1 Simpson's 1/3 Rule

Simpson's 1/3 rule results when a second-order interpolating polynomial is substituted into Eq. (21.1):

$$I = \int_a^b f(x)\,dx \cong \int_a^b f_2(x)\,dx$$

If a and b are designated as x_0 and x_2 and $f_2(x)$ is represented by a second-order Lagrange polynomial [Eq. (18.23)], the integral becomes

$$I = \int_{x_0}^{x_2} \left[\frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} f(x_0) + \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} f(x_1) + \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} f(x_2) \right] dx$$

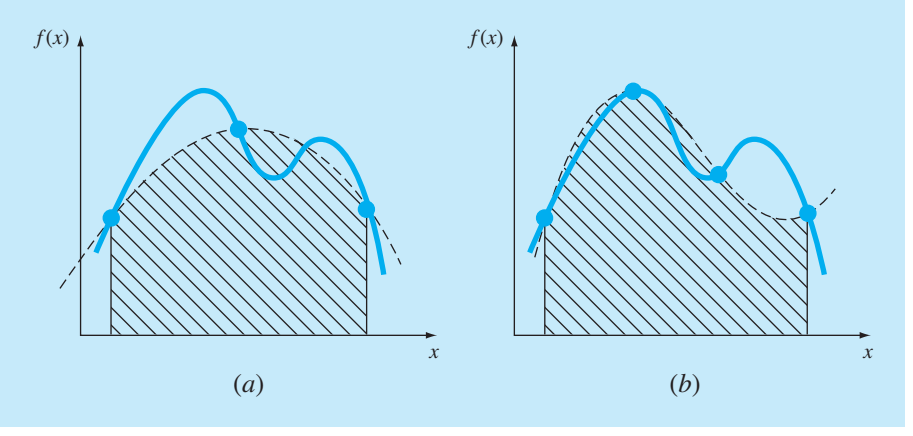


FIGURE 21.10
(*a*) Graphical depiction of Simpson's 1/3 rule: It consists of taking the area under a parabola connecting three points. (*b*) Graphical depiction of Simpson's 3/8 rule: It consists of taking the area under a cubic equation connecting four points.

After integration and algebraic manipulation, the following formula results:

$$I \cong \frac{h}{3}[f(x_0) + 4f(x_1) + f(x_2)] \tag{21.14}$$

where, for this case, $h = (b - a)/2$. This equation is known as *Simpson's 1/3 rule*. It is the second Newton-Cotes closed integration formula. The label "1/3" stems from the fact that h is divided by 3 in Eq. (21.14). An alternative derivation is shown in Box 21.3 where the Newton-Gregory polynomial is integrated to obtain the same formula.

Simpson's 1/3 rule can also be expressed using the format of Eq. (21.5):

$$I \cong \underbrace{(b - a)}_{\text{Width}} \underbrace{\frac{f(x_0) + 4f(x_1) + f(x_2)}{6}}_{\text{Average height}} \tag{21.15}$$

Box 21.3 Derivation and Error Estimate of Simpson's 1/3 Rule

As was done in Box 21.2 for the trapezoidal rule, Simpson's 1/3 rule can be derived by integrating the forward Newton-Gregory interpolating polynomial (Box 18.2):

$$I = \int_{x_0}^{x_2} \left[f(x_0) + \Delta f(x_0)\alpha + \frac{\Delta^2 f(x_0)}{2}\alpha(\alpha - 1) + \frac{\Delta^3 f(x_0)}{6}\alpha(\alpha - 1)(\alpha - 2) + \frac{f^{(4)}(\xi)}{24}\alpha(\alpha - 1)(\alpha - 2)(\alpha - 3)h^4 \right] dx$$

Notice that we have written the polynomial up to the fourth-order term rather than the third-order term as would be expected. The reason for this will be apparent shortly. Also notice that the limits of integration are from x_0 to x_2. Therefore, when the simplifying substitutions are made (recall Box 21.2), the integral is from $\alpha = 0$ to 2:

$$I = h\int_0^2 \left[f(x_0) + \Delta f(x_0)\alpha + \frac{\Delta^2 f(x_0)}{2}\alpha(\alpha - 1) + \frac{\Delta^3 f(x_0)}{6}\alpha(\alpha - 1)(\alpha - 2) + \frac{f^{(4)}(\xi)}{24}\alpha(\alpha - 1)(\alpha - 2)(\alpha - 3)h^4 \right] d\alpha$$

which can be integrated to yield

$$I = h\left[\alpha f(x_0) + \frac{\alpha^2}{2}\Delta f(x_0) + \left(\frac{\alpha^3}{6} - \frac{\alpha^2}{4}\right)\Delta^2 f(x_0) + \left(\frac{\alpha^4}{24} - \frac{\alpha^3}{6} + \frac{\alpha^2}{6}\right)\Delta^3 f(x_0) + \left(\frac{\alpha^5}{120} - \frac{\alpha^4}{16} + \frac{11\alpha^3}{72} - \frac{\alpha^2}{8}\right) f^{(4)}(\xi)h^4 \right]_0^2$$

and evaluated for the limits to give

$$I = h\left[2f(x_0) + 2\Delta f(x_0) + \frac{\Delta^2 f(x_0)}{3} + (0)\Delta^3 f(x_0) - \frac{1}{90} f^{(4)}(\xi)h^4 \right] \tag{B21.3.1}$$

Notice the significant result that the coefficient of the third divided difference is zero. Because $\Delta f(x_0) = f(x_1) - f(x_0)$ and $\Delta^2 f(x_0) = f(x_2) - 2f(x_1) + f(x_0)$, Eq. (B21.3.1) can be rewritten as

$$I = \underbrace{\frac{h}{3}[f(x_0) + 4f(x_1) + f(x_2)]}_{\text{Simpson's 1/3}} - \underbrace{\frac{1}{90} f^{(4)}(\xi)h^5}_{\text{Truncation error}}$$

Thus, the first term is Simpson's 1/3 rule and the second is the truncation error. Because the third divided difference dropped out, we obtain the significant result that the formula is third-order accurate.

where $a = x_0$, $b = x_2$, and x_1 = the point midway between a and b, which is given by $(b + a)/2$. Notice that, according to Eq. (21.15), the middle point is weighted by two-thirds and the two end points by one-sixth.

It can be shown that a single-segment application of Simpson's 1/3 rule has a truncation error of (Box 21.3)

$$E_t = -\frac{1}{90}h^5 f^{(4)}(\xi)$$

or, because $h = (b - a)/2$,

$$E_t = -\frac{(b-a)^5}{2880} f^{(4)}(\xi) \tag{21.16}$$

where ξ lies somewhere in the interval from a to b. Thus, Simpson's 1/3 rule is more accurate than the trapezoidal rule. However, comparison with Eq. (21.6) indicates that it is more accurate than expected. Rather than being proportional to the third derivative, the error is proportional to the fourth derivative. This is because, as shown in Box 21.3, the coefficient of the third-order term goes to zero during the integration of the interpolating polynomial. Consequently, Simpson's 1/3 rule is third-order accurate even though it is based on only three points. In other words, it yields exact results for cubic polynomials even though it is derived from a parabola!

EXAMPLE 21.4 Single Application of Simpson's 1/3 Rule

Problem Statement. Use Eq. (21.15) to integrate

$$f(x) = 0.2 + 25x - 200x^2 + 675x^3 - 900x^4 + 400x^5$$

from $a = 0$ to $b = 0.8$. Recall that the exact value is 1.640533.

Solution.

$$f(0) = 0.2 \qquad f(0.4) = 2.456 \qquad f(0.8) = 0.232$$

Therefore, Eq. (21.15) can be used to compute

$$I \cong 0.8\frac{0.2 + 4(2.456) + 0.232}{6} = 1.367467$$

which represents an exact error of

$$E_t = 1.640533 - 1.367467 = 0.2730667 \qquad \varepsilon_t = 16.6\%$$

which is approximately five times more accurate than for a single application of the trapezoidal rule (Example 21.1).

The estimated error is [Eq. (21.16)]

$$E_a = -\frac{(0.8)^5}{2880}(-2400) = 0.2730667$$

where −2400 is the average fourth derivative for the interval, as obtained using Eq. (PT6.4). As was the case in Example 21.1, the error is approximate (E_a) because the average fourth

derivative is not an exact estimate of $f^{(4)}(\xi)$. However, because this case deals with a fifth-order polynomial, the result matches.

21.2.2 The Multiple-Application Simpson's 1/3 Rule

Just as with the trapezoidal rule, Simpson's rule can be improved by dividing the integration interval into a number of segments of equal width (Fig. 21.11):

$$h = \frac{b - a}{n} \tag{21.17}$$

The total integral can be represented as

$$I = \int_{x_0}^{x_2} f(x)\,dx + \int_{x_2}^{x_4} f(x)\,dx + \cdots + \int_{x_{n-2}}^{x_n} f(x)\,dx$$

Substituting Simpson's 1/3 rule for the individual integrals yields

$$I \cong 2h\frac{f(x_0) + 4f(x_1) + f(x_2)}{6} + 2h\frac{f(x_2) + 4f(x_3) + f(x_4)}{6} + \cdots + 2h\frac{f(x_{n-2}) + 4f(x_{n-1}) + f(x_n)}{6}$$

or, combining terms and using Eq. (21.17),

$$I \cong \underbrace{(b - a)}_{\text{Width}} \underbrace{\frac{f(x_0) + 4\sum_{i=1,3,5}^{n-1} f(x_i) + 2\sum_{j=2,4,6}^{n-2} f(x_j) + f(x_n)}{3n}}_{\text{Average height}} \tag{21.18}$$

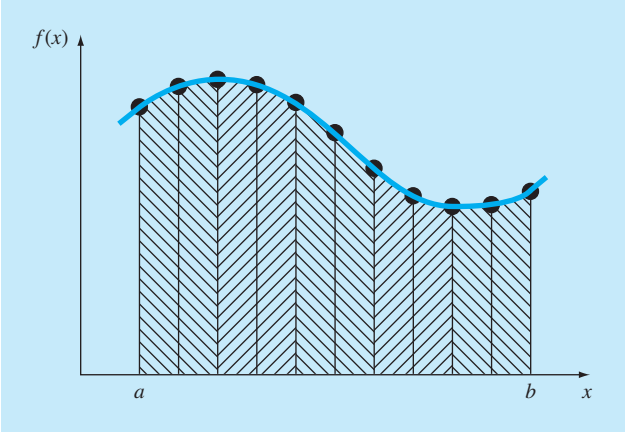


FIGURE 21.11
Graphical representation of the multiple application of Simpson's 1/3 rule. Note that the method can be employed only if the number of segments is even.

as Simpson's 3/8 rule is used in conjunction with the 1/3 rule to permit evaluation of both even and odd numbers of segments.

21.2.3 Simpson's 3/8 Rule

In a similar manner to the derivation of the trapezoidal and Simpson's 1/3 rule, a third-order Lagrange polynomial can be fit to four points and integrated:

$$I = \int_a^b f(x)\,dx \cong \int_a^b f_3(x)\,dx$$

to yield

$$I \cong \frac{3h}{8}[f(x_0) + 3f(x_1) + 3f(x_2) + f(x_3)]$$

where $h = (b - a)/3$. This equation is called *Simpson's 3/8 rule* because h is multiplied by 3/8. It is the third Newton-Cotes closed integration formula. The 3/8 rule can also be expressed in the form of Eq. (21.5):

$$I \cong \underbrace{(b - a)}_{\text{Width}} \underbrace{\frac{f(x_0) + 3f(x_1) + 3f(x_2) + f(x_3)}{8}}_{\text{Average height}} \tag{21.20}$$

Thus, the two interior points are given weights of three-eighths, whereas the end points are weighted with one-eighth. Simpson's 3/8 rule has an error of

$$E_t = -\frac{3}{80}h^5 f^{(4)}(\xi)$$

or, because $h = (b - a)/3$,

$$E_t = -\frac{(b - a)^5}{6480}f^{(4)}(\xi) \tag{21.21}$$

Because the denominator of Eq. (21.21) is larger than that of Eq. (21.16), the 3/8 rule is somewhat more accurate than the 1/3 rule.

Simpson's 1/3 rule is usually the method of preference because it attains third-order accuracy with three points rather than the four points required for the 3/8 version. However, the 3/8 rule has utility when the number of segments is odd. For instance, in Example 21.5 we used Simpson's rule to integrate the given function for four segments. Suppose that you desired an estimate for five segments. One option would be to use a multiple-application version of the trapezoidal rule as was done in Examples 21.2 and 21.3. This may not be advisable, however, because of the large truncation error associated with this method. An alternative would be to apply Simpson's 1/3 rule to the first two segments and Simpson's 3/8 rule to the last three (Fig. 21.12). In this way, you could obtain an estimate with third-order accuracy across the entire interval.

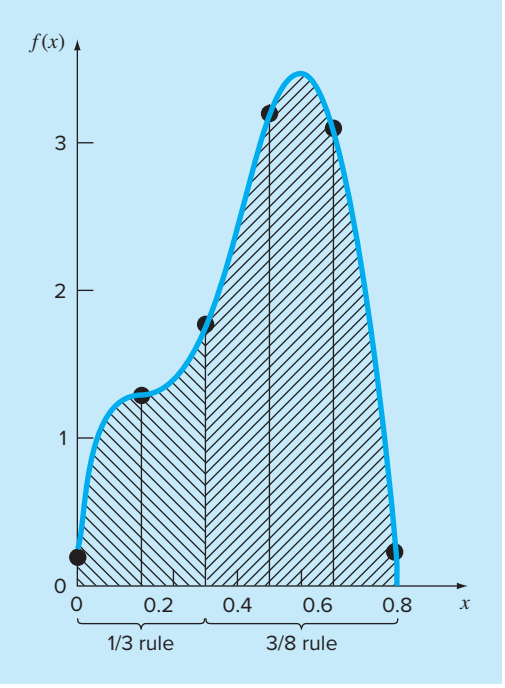


FIGURE 21.12
Illustration of how Simpson's 1/3 and 3/8 rules can be applied in tandem to handle multiple applications with odd numbers of intervals.

EXAMPLE 21.6 Simpson's 3/8 Rule

Problem Statement.

(**a**) Use Simpson's 3/8 rule to integrate

$$f(x) = 0.2 + 25x - 200x^2 + 675x^3 - 900x^4 + 400x^5$$

from $a = 0$ to $b = 0.8$.

(**b**) Use it in conjunction with Simpson's 1/3 rule to integrate the same function for five segments.

Solution.

(**a**) A single application of Simpson's 3/8 rule requires four equally spaced points:

$$f(0) = 0.2 \qquad f(0.2667) = 1.432724$$
$$f(0.5333) = 3.487177 \qquad f(0.8) = 0.232$$

Using Eq. (21.20),

$$I \cong 0.8\frac{0.2 + 3(1.432724 + 3.487177) + 0.232}{8} = 1.519170$$

$$E_t = 1.640533 - 1.519170 = 0.1213630 \qquad \varepsilon_t = 7.4\%$$

$$E_a = -\frac{(0.8)^5}{6480}(-2400) = 0.1213630$$

(**b**) The data needed for a five-segment application ($h = 0.16$) is

$$f(0) = 0.2 \qquad f(0.16) = 1.296919$$
$$f(0.32) = 1.743393 \qquad f(0.48) = 3.186015$$
$$f(0.64) = 3.181929 \qquad f(0.80) = 0.232$$

The integral for the first two segments is obtained using Simpson's 1/3 rule:

$$I \cong 0.32\frac{0.2 + 4(1.296919) + 1.743393}{6} = 0.3803237$$

For the last three segments, the 3/8 rule can be used to obtain

$$I \cong 0.48\frac{1.743393 + 3(3.186015 + 3.181929) + 0.232}{8} = 1.264754$$

The total integral is computed by summing the two results:

$$I = 0.3803237 + 1.264753 = 1.645077$$
$$E_t = 1.640533 - 1.645077 = -0.00454383 \qquad \varepsilon_t = -0.28\%$$

21.2.4 Computer Algorithms for Simpson's Rules

Pseudocodes for a number of forms of Simpson's rule are outlined in Fig. 21.13. Note that all are designed for data that are in tabulated form. A general program should have the capability to evaluate known functions or equations as well. We will illustrate how functions are handled in Chap. 22.

Notice that the program in Fig. 21.13*d* is set up so that either an even or odd number of segments may be used. For the even case, Simpson's 1/3 rule is applied to each pair of segments, and the results are summed to compute the total integral. For the odd case, Simpson's 3/8 rule is applied to the last three segments, and the 1/3 rule is applied to all the previous segments.

21.2.5 Higher-Order Newton-Cotes Closed Formulas

As noted previously, the trapezoidal rule and both of Simpson's rules are members of a family of integrating equations known as the Newton-Cotes closed integration formulas. Some of the formulas are summarized in Table 21.2 along with their truncation-error estimates.

Notice that, as was the case with Simpson's 1/3 and 3/8 rules, the five- and six-point formulas have errors of the same order. This general characteristic holds for the higher-point formulas and leads to the result that the even-segment–odd-point formulas (for example, 1/3 rule and *Boole's rule*) are usually the methods of preference.

However, it must also be stressed that, in engineering practice, the higher-order (that is, greater than four-point) formulas are rarely used. Simpson's rules are sufficient for most applications. Accuracy can be improved by using the multiple-application version. Furthermore, when the function is known and high accuracy is required, methods such

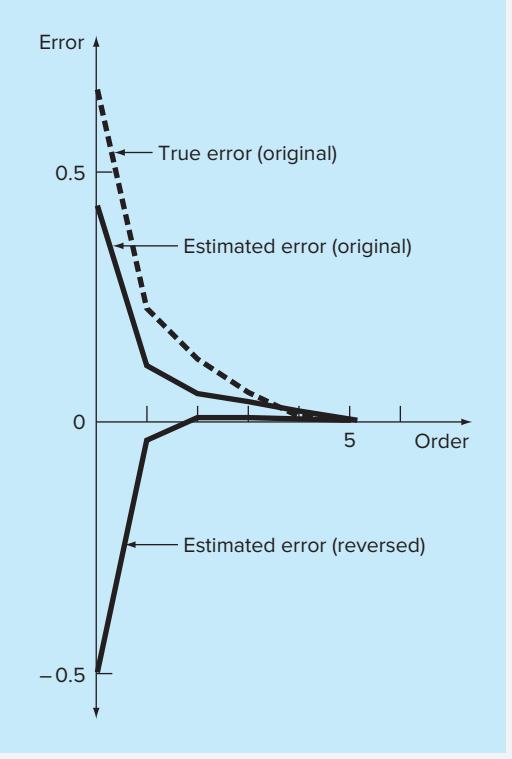


FIGURE 18.9
Percent relative errors for the prediction of ln 2 as a function of the order of the interpolating polynomial.

The foregoing example illustrates the importance of the choice of base points. As should be intuitively obvious, the points should be centered around and as close as possible to the unknown. This observation is also supported by direct examination of the error equation [Eq. (18.17)]. If we assume that the finite divided difference does not vary markedly along the range of these data, the error is proportional to the product: $(x - x_0)(x - x_1) \cdots (x - x_n)$. Obviously, the closer the base points are to x, the smaller the magnitude of this product.

18.2 LAGRANGE INTERPOLATING POLYNOMIALS

The *Lagrange interpolating polynomial* is simply a reformulation of the Newton polynomial that avoids the computation of divided differences. It can be represented concisely as

$$f_n(x) = \sum_{i=0}^{n} L_i(x) f(x_i) \tag{18.20}$$

where

$$L_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j} \tag{18.21}$$

and Π designates the "product of." For example, the linear version ($n = 1$) is

$$f_1(x) = \frac{x - x_1}{x_0 - x_1} f(x_0) + \frac{x - x_0}{x_1 - x_0} f(x_1) \tag{18.22}$$

and the second-order version is

$$f_2(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} f(x_0) + \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} f(x_1) + \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} f(x_2) \tag{18.23}$$

Equation (18.20) can be derived directly from Newton's polynomial (Box 18.1). However, the rationale underlying the Lagrange formulation can be grasped directly by realizing that each term $L_i(x)$ will be 1 at $x = x_i$ and 0 at all other sample points (Fig. 18.10). Thus, each product $L_i(x)f(x_i)$ takes on the value of $f(x_i)$ at the sample point x_i. Consequently, the summation of all the products designated by Eq. (18.20) is the unique nth-order polynomial that passes exactly through all $n + 1$ data points.

EXAMPLE 18.6 **Lagrange Interpolating Polynomials**

Problem Statement. Use a Lagrange interpolating polynomial of the first and second order to evaluate ln 2 on the basis of the data given in Example 18.2:

$$x_0 = 1 \qquad f(x_0) = 0$$
$$x_1 = 4 \qquad f(x_1) = 1.386294$$
$$x_2 = 6 \qquad f(x_2) = 1.791760$$

Solution. The first-order polynomial [Eq. (18.22)] can be used to obtain the estimate at $x = 2$,

$$f_1(2) = \frac{2 - 4}{1 - 4} 0 + \frac{2 - 1}{4 - 1} 1.386294 = 0.4620981$$

In a similar fashion, the second-order polynomial is developed as [Eq. (18.23)]

$$f_2(2) = \frac{(2 - 4)(2 - 6)}{(1 - 4)(1 - 6)} 0 + \frac{(2 - 1)(2 - 6)}{(4 - 1)(4 - 6)} 1.386294 + \frac{(2 - 1)(2 - 4)}{(6 - 1)(6 - 4)} 1.791760 = 0.5658444$$

As expected, both these results agree with those previously obtained using Newton's interpolating polynomial.

Box 18.1 Derivation of the Lagrange Form Directly from Newton's Interpolating Polynomial

The Lagrange interpolating polynomial can be derived directly from Newton's formulation. We will do this for the first-order case only [Eq. (18.2)]. To derive the Lagrange form, we reformulate the divided differences. For example, the first finite divided difference,

$$f[x_1, x_0] = \frac{f(x_1) - f(x_0)}{x_1 - x_0} \tag{B18.1.1}$$

can be reformulated as

$$f[x_1, x_0] = \frac{f(x_1)}{x_1 - x_0} + \frac{f(x_0)}{x_0 - x_1} \tag{B18.1.2}$$

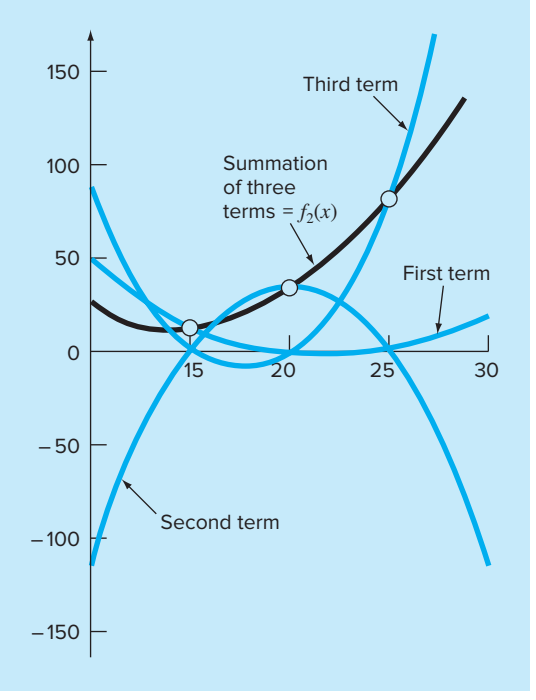
which is referred to as the *symmetric form*. Substituting Eq. (B18.1.2) into Eq. (18.2) yields

$$f_1(x) = f(x_0) + \frac{x - x_0}{x_1 - x_0} f(x_1) + \frac{x - x_0}{x_0 - x_1} f(x_0)$$

Finally, grouping similar terms and simplifying yields the Lagrange form,

$$f_1(x) = \frac{x - x_1}{x_0 - x_1} f(x_0) + \frac{x - x_0}{x_1 - x_0} f(x_1)$$

FIGURE 18.10
A visual depiction of the rationale behind the Lagrange polynomial. This figure shows a second-order case. Each of the three terms in Eq. (18.23) passes through one of the data points and is zero at the other two. The summation of the three terms must, therefore, be the unique second-order polynomial $f_2(x)$ that passes exactly through the three points.



tangent to the x axis. A more detailed picture of the behavior of $f(x)$ is obtained by changing the plotting range from $x = 3$ to $x = 5$, as shown in Fig. 5.4*b*. Finally, in Fig. 5.4*c*, the vertical scale is narrowed further to $f(x) = -0.15$ to $f(x) = 0.15$ and the horizontal scale is narrowed to $x = 4.2$ to $x = 4.3$. This plot shows clearly that a double root does not exist in this region and that in fact there are two distinct roots at about $x = 4.23$ and $x = 4.26$.

Computer graphics will have great utility in your studies of numerical methods. This capability will also find many other applications in your other classes and professional activities as well.

5.2 THE BISECTION METHOD

When applying the graphical technique in Example 5.1, you have observed (Fig. 5.1) that $f(x)$ changed sign on opposite sides of the root. In general, if $f(x)$ is real and continuous in the interval from x_l to x_u and $f(x_l)$ and $f(x_u)$ have opposite signs, that is,

$$f(x_l)\,f(x_u) < 0 \tag{5.1}$$

then there is at least one real root between x_l and x_u.

Incremental search methods capitalize on this observation by locating an interval where the function changes sign. Then the location of the sign change (and consequently, the root) is identified more precisely by dividing the interval into a number of subintervals. Each of these subintervals is searched to locate the sign change. The process is repeated and the root estimate refined by dividing the subintervals into finer increments. We will return to the general topic of incremental searches in Sec. 5.4.

The *bisection method,* which is alternatively called binary chopping, interval halving, or Bolzano's method, is one type of incremental search method in which the interval is always divided in half. If a function changes sign over an interval, the function value at the midpoint is evaluated. The location of the root is then determined as lying at the midpoint of the subinterval within which the sign change occurs. The process is repeated to obtain refined estimates. A simple algorithm for the bisection calculation is listed in Fig. 5.5, and a graphical depiction of the method is provided in Fig. 5.6. The following example goes through the actual computations involved in the method.

FIGURE 5.5

Step 1: Choose lower x_l and upper x_u guesses for the root such that the function changes sign over the interval. This can be checked by ensuring that $f(x_l)f(x_u) < 0$.

Step 2: An estimate of the root x_r is determined by

$$x_r = \frac{x_l + x_u}{2}$$

Step 3: Make the following evaluations to determine in which subinterval the root lies:

(*a*) If $f(x_l)f(x_r) < 0$, the root lies in the lower subinterval. Therefore, set $x_u = x_r$ and return to step 2.

(*b*) If $f(x_l)f(x_r) > 0$, the root lies in the upper subinterval. Therefore, set $x_l = x_r$ and return to step 2.

(*c*) If $f(x_l)f(x_r) = 0$, the root equals x_r; terminate the computation.

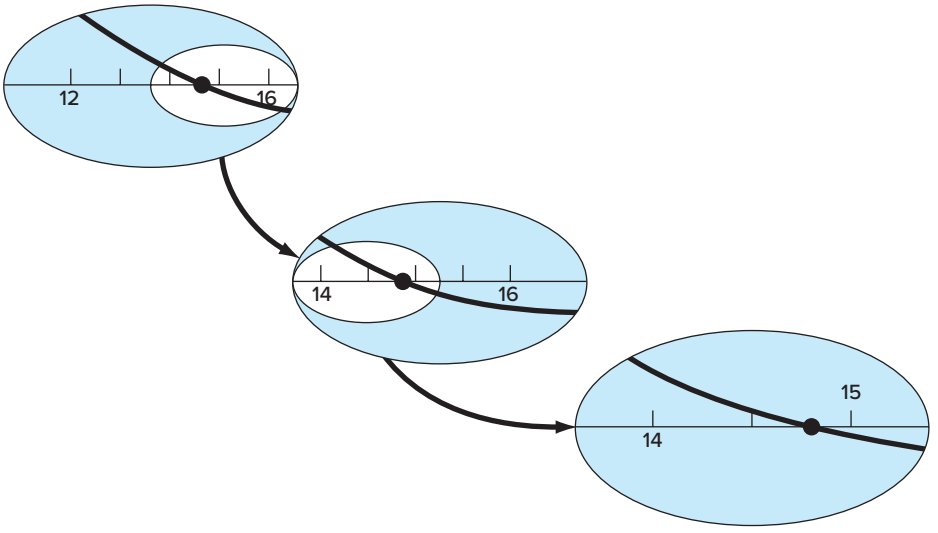


FIGURE 5.6
A graphical depiction of the bisection method. This plot conforms to the first three iterations from Example 5.3.

EXAMPLE 5.3 Bisection

Problem Statement. Use bisection to solve the same problem approached graphically in Example 5.1.

Solution. The first step in bisection is to guess two values of the unknown (in the present problem, c) that give values for $f(c)$ with different signs. From Fig. 5.1, we can see that the function changes sign between values of 12 and 16. Therefore, the initial estimate of the root x_r lies at the midpoint of the interval

$$x_r = \frac{12 + 16}{2} = 14$$

This estimate represents a true percent relative error of $\varepsilon_t = 5.3\%$ (note that the true value of the root is 14.8011). Next we compute the product of the function value at the lower bound and at the midpoint:

$$f(12)\,f(14) = 6.114\,(1.611) = 9.850$$

which is greater than zero, and hence no sign change occurs between the lower bound and the midpoint. Consequently, the root must be located between 14 and 16. Therefore, we create a new interval by redefining the lower bound as 14 and determining a revised root estimate as

$$x_r = \frac{14 + 16}{2} = 15$$

which represents a true percent error of $\varepsilon_t = 1.3\%$. The process can be repeated to obtain refined estimates. For example,

$$f(14)\,f(15) = 1.611(-0.384) = -0.619$$

Therefore, the root is between 14 and 15. The upper bound is redefined as 15, and the root estimate for the third iteration is calculated as

$$x_r = \frac{14 + 15}{2} = 14.5$$

which represents a percent relative error of $\varepsilon_t = 2.0\%$. The method can be repeated until the result is accurate enough to satisfy your needs.

In the previous example, you may have noticed that the true error does not decrease with each iteration. However, the interval within which the root is located is halved with each step in the process. As discussed in the next section, the interval width provides an exact estimate of the upper bound of the error for the bisection method.

5.2.1 Termination Criteria and Error Estimates

We ended Example 5.3 with the statement that the method could be continued to obtain a refined estimate of the root. We must now develop an objective criterion for deciding when to terminate the method.

An initial suggestion might be to end the calculation when the true error falls below some prespecified level. For instance, in Example 5.3, the relative error dropped to 2.0 percent during the course of the computation. We might decide that we should terminate when the error drops below, say, 0.1 percent. This strategy is flawed because the error estimates in the example were based on knowledge of the true root of the function. This would not be the case in an actual situation because there would be no point in using the method if we already knew the root.

Therefore, we require an error estimate that is not contingent on foreknowledge of the root. As developed previously in Sec. 3.3, an approximate percent relative error ε_a can be calculated, as in [recall Eq. (3.5)]

$$\varepsilon_a = \left| \frac{x_r^{\text{new}} - x_r^{\text{old}}}{x_r^{\text{new}}} \right| 100\% \tag{5.2}$$

where x_r^{new} is the root for the present iteration and x_r^{old} is the root from the previous iteration. The absolute value is used because we are usually concerned with the magnitude of ε_a rather than with its sign. When ε_a becomes less than a prespecified stopping criterion ε_s, the computation is terminated.

EXAMPLE 5.4 **Error Estimates for Bisection**

Problem Statement. Continue Example 5.3 until the approximate error falls below a stopping criterion of $\varepsilon_s = 0.5\%$. Use Eq. (5.2) to compute the errors.

Solution. The results of the first two iterations for Example 5.3 were 14 and 15. Substituting these values into Eq. (5.2) yields

$$|\varepsilon_a| = \left| \frac{15 - 14}{15} \right| 100\% = 6.667\%$$

the root estimate is calculated. Previously calculated values are saved and merely reassigned as the bracket shrinks. Thus, $n + 1$ function evaluations are performed, rather than $2n$.

5.3 THE FALSE-POSITION METHOD

Although bisection is a perfectly valid technique for determining roots, its "brute-force" approach is relatively inefficient. False position is an alternative based on a graphical insight.

A shortcoming of the bisection method is that, in dividing the interval from x_l to x_u into equal halves, no account is taken of the magnitudes of $f(x_l)$ and $f(x_u)$. For example, if $f(x_l)$ is much closer to zero than $f(x_u)$, it is likely that the root is closer to x_l than to x_u (Fig. 5.12). An alternative method that exploits this graphical insight is to join $f(x_l)$ and $f(x_u)$ by a straight line. The intersection of this line with the x axis represents an improved estimate of the root. The fact that the replacement of the curve by a straight line gives a "false position" of the root is the origin of the name, *method of false position,* or in Latin, *regula falsi.* It is also called the *linear interpolation method.*

Using similar triangles (Fig. 5.12), the intersection of the straight line with the x axis can be estimated as

$$\frac{f(x_l)}{x_r - x_l} = \frac{f(x_u)}{x_r - x_u} \tag{5.6}$$

which can be solved for (see Box 5.1 for details).

$$x_r = x_u - \frac{f(x_u)(x_l - x_u)}{f(x_l) - f(x_u)} \tag{5.7}$$

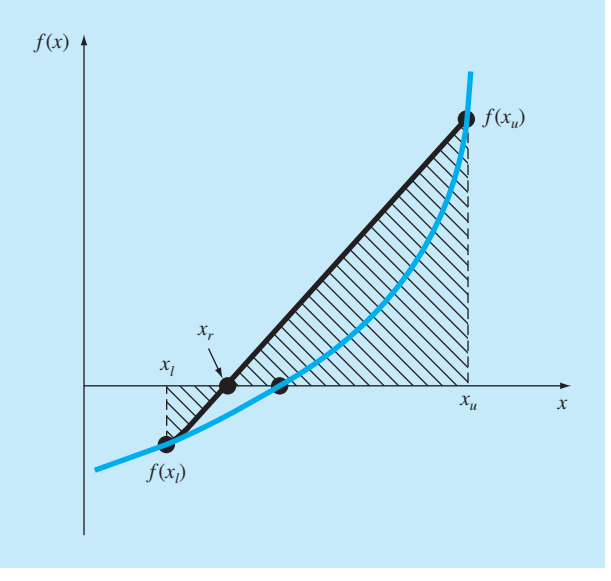


FIGURE 5.12
A graphical depiction of the method of false position. Similar triangles used to derive the formula for the method are shaded.

Box 5.1 Derivation of the Method of False Position

Cross-multiply Eq. (5.6) to yield

$$f(x_l)(x_r - x_u) = f(x_u)(x_r - x_l)$$

Collect terms and rearrange:

$$x_r[f(x_l) - f(x_u)] = x_u f(x_l) - x_l f(x_u)$$

Divide by $f(x_l) - f(x_u)$:

$$x_r = \frac{x_u f(x_l) - x_l f(x_u)}{f(x_l) - f(x_u)} \tag{B5.1.1}$$

This is one form of the method of false position. Note that it allows the computation of the root x_r as a function of the lower and upper guesses x_l and x_u. It can be put in an alternative form by expanding it:

$$x_r = \frac{x_u f(x_l)}{f(x_l) - f(x_u)} - \frac{x_l f(x_u)}{f(x_l) - f(x_u)}$$

then adding and subtracting x_u on the right-hand side:

$$x_r = x_u + \frac{x_u f(x_l)}{f(x_l) - f(x_u)} - x_u - \frac{x_l f(x_u)}{f(x_l) - f(x_u)}$$

Collecting terms yields

$$x_r = x_u + \frac{x_u f(x_u)}{f(x_l) - f(x_u)} - \frac{x_l f(x_u)}{f(x_l) - f(x_u)}$$

or

$$x_r = x_u - \frac{f(x_u)(x_l - x_u)}{f(x_l) - f(x_u)}$$

which is the same as Eq. (5.7). We use this form because it involves one less function evaluation and one less multiplication than Eq. (B5.1.1). In addition, it is directly comparable with the secant method, which will be discussed in Chap. 6.

This is the *false-position formula.* The value of x_r computed with Eq. (5.7) then replaces whichever of the two initial guesses, x_l or x_u, yields a function value with the same sign as $f(x_r)$. In this way, the values of x_l and x_u always bracket the true root. The process is repeated until the root is estimated adequately. The algorithm is identical to the one for bisection (Fig. 5.5) with the exception that Eq. (5.7) is used for step 2. In addition, the same stopping criterion [Eq. (5.2)] is used to terminate the computation.

EXAMPLE 5.5 False Position

Problem Statement. Use the false-position method to determine the root of the same equation investigated in Example 5.1 [Eq. (E5.1.1)].

Solution. As in Example 5.3, initiate the computation with guesses of $x_l = 12$ and $x_u = 16$.

First iteration:

$$x_l = 12 \qquad f(x_l) = 6.1139$$
$$x_u = 16 \qquad f(x_u) = -2.2303$$
$$x_r = 16 - \frac{-2.2303(12 - 16)}{6.1139 - (-2.2303)} = 14.309$$

which has a true relative error of 0.88%.

Second iteration:

$$f(x_l) f(x_r) = -1.5376$$

Therefore, the root lies in the first subinterval, and x_r becomes the upper limit for the next iteration, $x_u = 14.9113$:

$$x_l = 12 \qquad f(x_l) = 6.1139$$
$$x_u = 14.9309 \qquad f(x_u) = -0.2515$$
$$x_r = 14.9309 - \frac{-0.2515(12 - 14.9309)}{6.1139 - (-0.2515)} = 14.8151$$

which has true and approximate relative errors of 0.09 and 0.78%. Additional iterations can be performed to refine the estimate of the roots.

A feeling for the relative efficiency of the bisection and false-position methods can be appreciated by referring to Fig. 5.13, where we have plotted the true percent relative errors for Examples 5.4 and 5.5. Note how the error for false position decreases much faster than for bisection because of the more efficient scheme for root location in the false-position method.

Recall in the bisection method that the interval between x_l and x_u grew smaller during the course of a computation. The interval, as defined by $\Delta x/2 = |x_u - x_l|/2$ for the first iteration, therefore provided a measure of the error for this approach. This is not the case

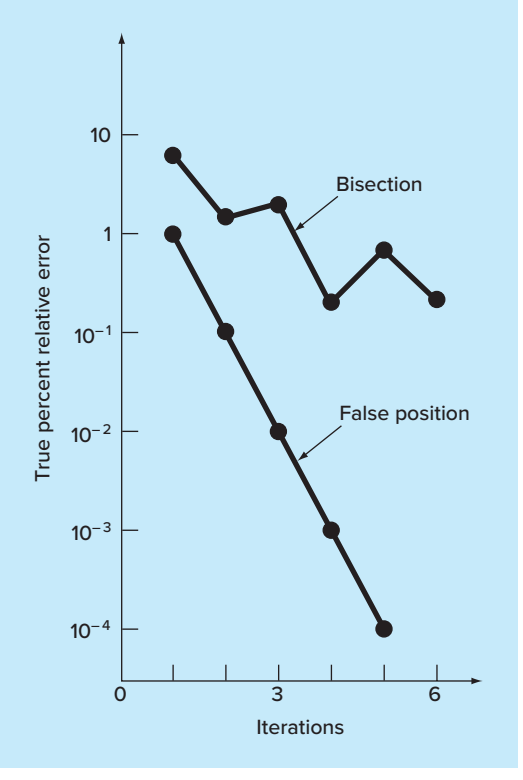


FIGURE 5.13
Comparison of the relative errors of the bisection and the false-position methods.

for the method of false position because one of the initial guesses may stay fixed throughout the computation as the other guess converges on the root. For instance, in Example 5.5 the lower guess x_l remained at 12 while x_u converged on the root. For such cases, the interval does not shrink but rather approaches a constant value.

Example 5.5 suggests that Eq. (5.2) represents a very conservative error criterion. In fact, Eq. (5.2) actually constitutes an approximation of the discrepancy of the previous iteration. This is because for a case such as Example 5.5, where the method is converging quickly (for example, the error is being reduced nearly an order of magnitude per iteration), the root for the present iteration x_r^{new} is a much better estimate of the true value than the result of the previous iteration x_r^{old}. Thus, the quantity in the numerator of Eq. (5.2) actually represents the discrepancy of the previous iteration. Consequently, we are assured that satisfaction of Eq. (5.2) ensures that the root will be known with greater accuracy than the prescribed tolerance. However, as described in the next section, there are cases where false position converges slowly. For these cases, Eq. (5.2) becomes unreliable, and an alternative stopping criterion must be developed.

5.3.1 Pitfalls of the False-Position Method

Although the false-position method would seem to always be the bracketing method of preference, there are cases where it performs poorly. In fact, as in the following example, there are certain cases where bisection yields superior results.

EXAMPLE 5.6 A Case Where Bisection Is Preferable to False Position

Problem Statement. Use bisection and false position to locate the root of

$$f(x) = x^{10} - 1$$

between $x = 0$ and 1.3.

Solution. Using bisection, the results can be summarized as follows:

Iteration	x_l	x_u	x_r	ε_a (%)	ε_t (%)
1	0	1.3	0.65	100.0	35
2	0.65	1.3	0.975	33.3	2.5
3	0.975	1.3	1.1375	14.3	13.8
4	0.975	1.1375	1.05625	7.7	5.6
5	0.975	1.05625	1.015625	4.0	1.6

Thus, after five iterations, the true error is reduced to less than 2%. For false position, a very different outcome is obtained:

Iteration	x_l	x_u	x_r	ε_a (%)	ε_t (%)
1	0	1.3	0.09430		90.6
2	0.09430	1.3	0.18176	48.1	81.8
3	0.18176	1.3	0.26287	30.9	73.7
4	0.26287	1.3	0.33811	22.3	66.2
5	0.33811	1.3	0.40788	17.1	59.2

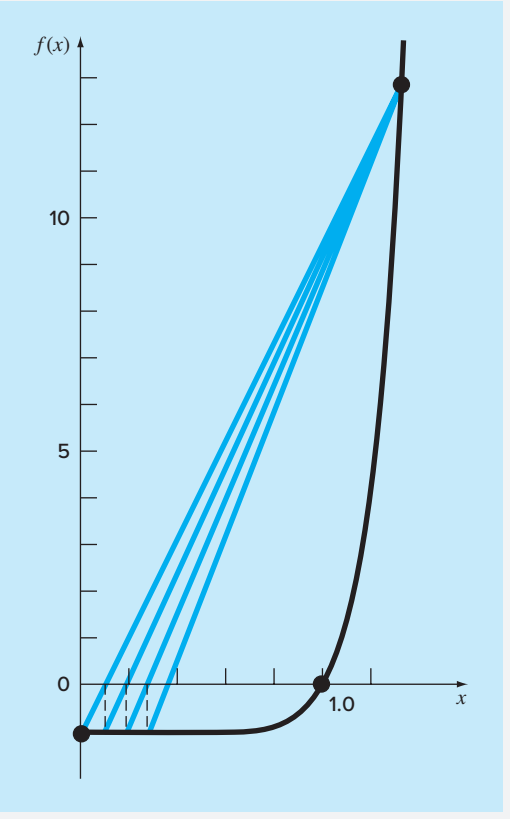


FIGURE 5.14
Plot of $f(x) = x^{10} - 1$, illustrating slow convergence of the false-position method.

After five iterations, the true error has only been reduced to about 59 percent. In addition, note that $\varepsilon_a < \varepsilon_t$. Thus, the approximate error is misleading. Insight into these results can be gained by examining a plot of the function. As in Fig. 5.14, the curve violates the premise upon which false position was based—that is, if $f(x_l)$ is much closer to zero than $f(x_u)$, then the root is closer to x_l than to x_u (recall Fig. 5.12). Because of the shape of the present function, the opposite is true.

The forgoing example illustrates that blanket generalizations regarding root-location methods are usually not possible. Although a method such as false position is often superior to bisection, there are invariably cases that violate this general conclusion. Therefore, in addition to using Eq. (5.2), the results should always be checked by substituting the root estimate into the original equation and determining whether the result is close to zero. Such a check should be incorporated into all computer programs for root location.

The example also illustrates a major weakness of the false-position method: its one-sidedness. That is, as iterations are proceeding, one of the bracketing points will tend to

stay fixed. This can lead to poor convergence, particularly for functions with significant curvature. The following section provides a remedy.

5.3.2 Modified False Position

One way to mitigate the "one-sided" nature of false position is to have the algorithm detect when one of the bounds is stuck. If this occurs, the function value at the stagnant bound can be divided in half. This is called the *modified false-position method.*

The algorithm in Fig. 5.15 implements this strategy. Notice how counters are used to determine when one of the bounds stays fixed for two iterations. If this occurs, the function value at this stagnant bound is halved.

The effectiveness of this algorithm can be demonstrated by applying it to Example 5.6. If a stopping criterion of 0.01% is used, the bisection and standard false-position

FIGURE 5.15
Pseudocode for the modified false-position method.

```
FUNCTION ModFalsePos(xl, xu, es, imax, xr, iter, ea)
  iter = 0
  fl = f(xl)
  fu = f(xu)
  DO
    xrold = xr
    xr = xu - fu * (xl - xu) / (fl - fu)
    fr = f(xr)
    iter = iter + 1
    IF xr <> 0 THEN
      ea = Abs((xr - xrold) / xr) * 100
    END IF
    test = fl * fr
    IF test < 0 THEN
      xu = xr
      fu = f(xu)
      iu = 0
      il = il + 1
      IF il ≥ 2 THEN fl = fl / 2
    ELSE IF test > 0 THEN
      xl = xr
      fl = f(xl)
      il = 0
      iu = iu + 1
      IF iu ≥ 2 THEN fu = fu / 2
    ELSE
      ea = 0
    END IF
    IF ea < es OR iter ≥ imax THEN EXIT
  END DO
  ModFalsePos = xr
END MODFALSEPOS
```

errors of the same order, modifiers of the type listed in Fig. 26.5 can be incorporated to improve accuracy and allow step-size control. Box 26.1 provides general equations for these modifiers. In this section, we present two of the most common higher-order multistep approaches: Milne's method and the fourth-order Adams method.

Milne's Method. Milne's method is the most common multistep method based on Newton-Cotes integration formulas. It uses the three-point Newton-Cotes open formula as a predictor:

$$y_{i+1}^0 = y_{i-3}^m + \frac{4h}{3}(2f_i^m - f_{i-1}^m + 2f_{i-2}^m) \tag{26.40}$$

and the three-point Newton-Cotes closed formula (Simpson's 1/3 rule) as a corrector:

$$y_{i+1}^j = y_{i-1}^m + \frac{h}{3}(f_{i-1}^m + 4f_i^m + f_{i+1}^{j-1}) \tag{26.41}$$

where j is an index representing the number of iterations of the modifier. The predictor and corrector modifiers for Milne's method can be developed from the formulas in Box 26.1 and the error coefficients in Tables 21.2 and 21.4:

$$E_p = \frac{28}{29}(y_i^m - y_i^0) \tag{26.42}$$

$$E_c \cong -\frac{1}{29}(y_{i+1}^m - y_{i+1}^0) \tag{26.43}$$

EXAMPLE 26.5 Milne's Method

Problem Statement. Use Milne's method to integrate $y' = 4e^{0.8x} - 0.5y$ from $x = 0$ to $x = 4$ using a step size of 1. The initial condition at $x = 0$ is $y = 2$. Because we are dealing with a multistep method, previous points are required. In an actual application, a one-step method such as a fourth-order RK would be used to compute the required points. For the present example, we will use the analytical solution [recall Eq. (E25.5.1) from Example 25.5] to compute exact values at $x_{i-3} = -3$, $x_{i-2} = -2$, and $x_{i-1} = -1$: $y_{i-3} = -4.547302$, $y_{i-2} = -2.306160$, and $y_{i-1} = -0.3929953$, respectively.

Solution. The predictor [Eq. (26.40)] is used to calculate a value at $x = 1$:

$$y_1^0 = -4.54730 + \frac{4(1)}{3}[2(3) - 1.99381 + 2(1.96067)] = 6.02272 \qquad \varepsilon_t = 2.8\%$$

The corrector [Eq. (26.41)] is then employed to compute

$$y_1^1 = -0.3929953 + \frac{1}{3}[1.99381 + 4(3) + 5.890802] = 6.235210 \qquad \varepsilon_t = -0.66\%$$

This result can be substituted back into Eq. (26.41) to iteratively correct the estimate. This process converges on a final corrected value of 6.204855 ($\varepsilon_t = -0.17\%$).

This value is more accurate than the comparable estimate of 6.360865 ($\varepsilon_t = -2.68\%$) obtained previously with the non-self-starting Heun method (Examples 26.2 through 26.4). The results for the remaining steps are $y(2) = 14.86031$ ($\varepsilon_t = -0.11\%$), $y(3) = 33.72426$ ($\varepsilon_t = -0.14\%$), and $y(4) = 75.43295$ ($\varepsilon_t = -0.12\%$).

CHAPTER

4

Truncation Errors and the Taylor Series

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

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

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

4.1 THE TAYLOR SERIES

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

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

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

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

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

Box 4.1 Taylor's Theorem

Taylor's Theorem

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

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

where the remainder R_n is defined as

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

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

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

First Theorem of Mean for Integrals

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

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

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

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

Second Theorem of Mean for Integrals

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

where the remainder term is now

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

EXAMPLE 4.1 **Taylor Series Approximation of a Polynomial**

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

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

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

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

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

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

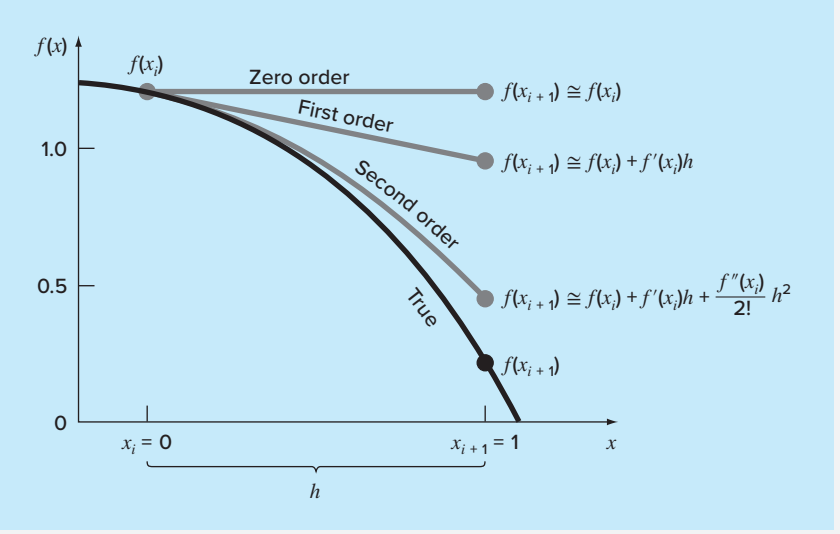


FIGURE 4.1
The approximation of $f(x) = -0.1x^4 - 0.15x^3 - 0.5x^2 - 0.25x + 1.2$ at $x = 1$ by zero-order, first-order, and second-order Taylor series expansions.

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

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

at $x = 1$.

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

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

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

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

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

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

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

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

Therefore, according to Eq. (4.4),

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

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

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

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

where the remainder term is

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

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

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

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

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

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

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

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

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

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.

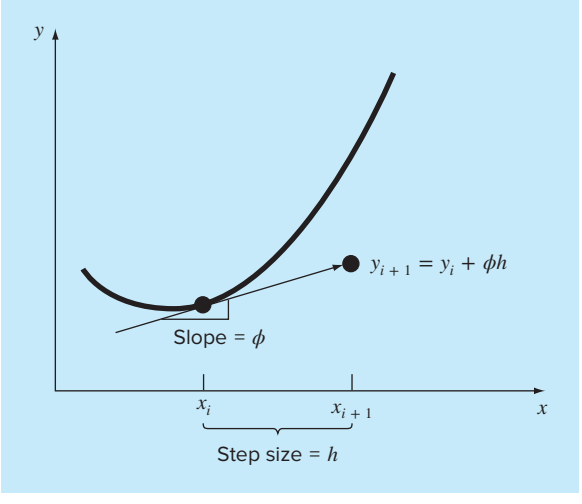


FIGURE 25.1
Graphical depiction of a one-step method.

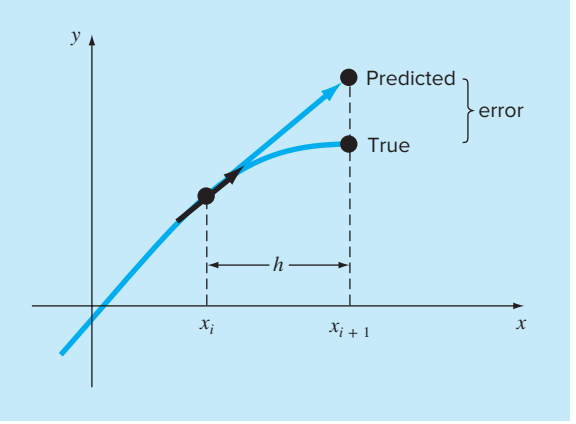


FIGURE 25.2
Euler's method.

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.

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	−28.1
1.5	2.21875	5.12500	−131.0	−1.4
2.0	2.00000	4.50000	−125.0	20.3
2.5	2.71875	4.75000	−74.7	17.2
3.0	4.00000	5.87500	−46.9	3.9
3.5	4.71875	7.12500	−51.0	−11.3
4.0	3.00000	7.00000	−133.3	−53.1

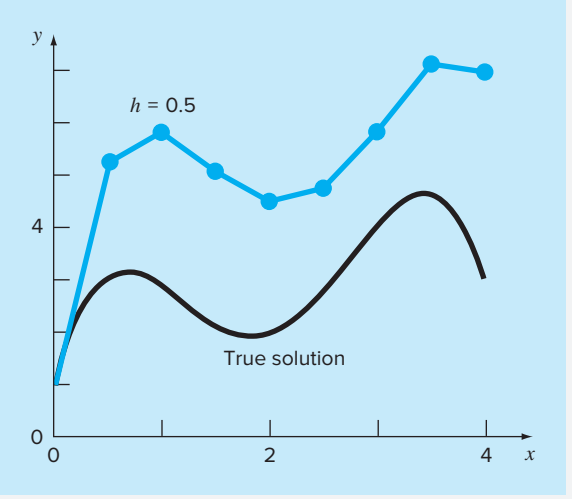


FIGURE 25.3
Comparison of the true solution with a numerical solution using Euler's method for the integral of $y' = -2x^3 + 12x^2 - 20x + 8.5$ from $x = 0$ to $x = 4$ with a step size of 0.5. The initial condition at $x = 0$ is $y = 1$.

Note that, 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.

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 performing 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,

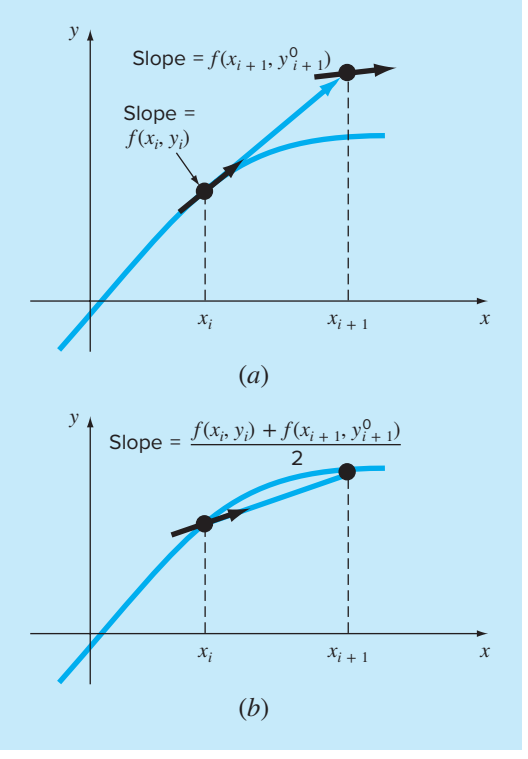
$$y_i' = f(x_i, y_i) \tag{25.12}$$

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

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



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

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

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

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

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

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

This is called a *corrector equation.*

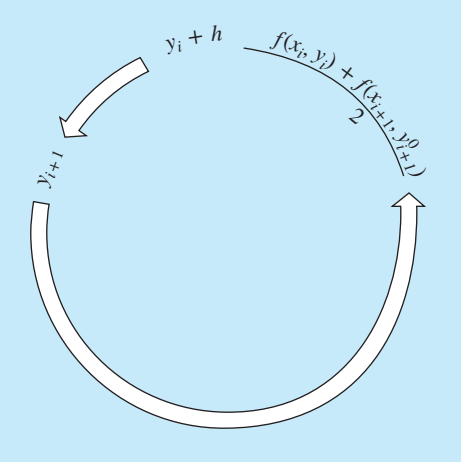
The Heun method is a *predictor-corrector approach.* All the multistep methods to be discussed subsequently in Chap. 26 are of this type. The Heun method is the only one-step predictor-corrector method described in this book. As derived above, it can be expressed concisely as

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*): $$y_{i+1} = y_i + \frac{f(x_i, y_i) + f(x_{i+1}, y^0_{i+1})}{2} h \tag{25.16}$$

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.



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

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

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

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

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

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

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

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

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

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

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:

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

which represents an $|\varepsilon_t|$ of 3.03%. Notice how the errors can 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*):

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

```
FUNCTION Fixpt(x0, es, imax, iter, ea)
  xr = x0
  iter = 0
  DO
    xrold = xr
    xr = g(xrold)
    iter = iter + 1
    IF xr ≠ 0 THEN
      ea = |(xr − xrold)/xr| · 100
    END IF
    IF ea < es OR iter ≥ imax EXIT
  END DO
  Fixpt = xr
END Fixpt
```

FIGURE 6.4
Pseudocode for fixed-point iteration. Note that other open methods can be cast in this general format.

presents pseudocode for the algorithm. Other open methods can be programmed in a similar way, the major modification being to change the iterative formula that is used to compute the new root estimate.

6.2 THE NEWTON-RAPHSON METHOD

Perhaps the most widely used of all root-locating formulas is the Newton-Raphson equation (Fig. 6.5). If the initial guess at the root is x_i, a tangent can be extended from the point $[x_i, f(x_i)]$. The point where this tangent crosses the x axis usually represents an improved estimate of the root.

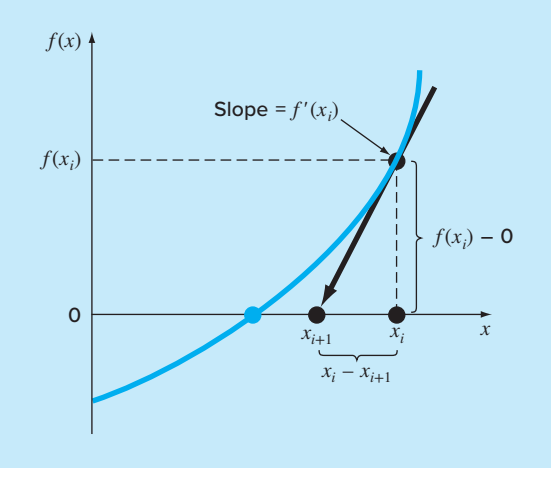


FIGURE 6.5
Graphical depiction of the Newton-Raphson method. A tangent to the function of x_i [that is, $f'(x_i)$] is extrapolated down to the x axis to provide an estimate of the root at x_{i+1}.

The Newton-Raphson method can be derived on the basis of this geometrical interpretation (an alternative method based on the Taylor series is described in Box 6.2). As in Fig. 6.5, the first derivative at x is equivalent to the slope:

$$f'(x_i) = \frac{f(x_i) - 0}{x_i - x_{i+1}} \tag{6.5}$$

which can be rearranged to yield

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \tag{6.6}$$

which is called the *Newton-Raphson formula.*

EXAMPLE 6.3 Newton-Raphson Method

Problem Statement. Use the Newton-Raphson method to estimate the root of $f(x) = e^{-x} - x$, employing an initial guess of $x_0 = 0$.

Solution. The first derivative of the function can be evaluated as

$$f'(x) = -e^{-x} - 1$$

which can be substituted along with the original function into Eq. (6.6) to give

$$x_{i+1} = x_i - \frac{e^{-x_i} - x_i}{-e^{-x_i} - 1}$$

Starting with an initial guess of $x_0 = 0$, this iterative equation can be applied to compute

i	x_i	ϵ_t (%)	E_t
0	0.000000000	100	0.567143290
1	0.500000000	11.8	0.067143290
2	0.566311003	0.147	0.000832287
3	0.567143165	0.0000221	0.000000125
4	0.567143290	$< 10^{-8}$	0.000000000

Thus, the approach rapidly converges on the true root. Notice that the true percent relative error at each iteration decreases much faster than it does in simple fixed-point iteration (compare with Example 6.1).

6.2.1 Termination Criteria and Error Estimates

As with other root-location methods, Eq. (3.5) can be used as a termination criterion. In addition, however, the Taylor series derivation of the method (Box 6.2) provides theoretical insight regarding the rate of convergence as expressed by $E_{i+1} = O(E_i^2)$. Thus the error should be roughly proportional to the square of the previous error. In other words,

Box 6.2 Derivation and Error Analysis of the Newton-Raphson Method

Aside from the geometric derivation [Eqs. (6.5) and (6.6)], the Newton-Raphson method may also be developed from the Taylor series expansion. This alternative derivation is useful in that it also provides insight into the rate of convergence of the method.

Recall from Chap. 4 that the Taylor series expansion can be represented as

$$f(x_{i+1}) = f(x_i) + f'(x_i)(x_{i+1} - x_i) + \frac{f''(\xi)}{2!}(x_{i+1} - x_i)^2 \tag{B6.2.1}$$

where ξ lies somewhere in the interval from x_i to x_{i+1}. An approximate version is obtainable by truncating the series after the first derivative term:

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

At the intersection with the x axis, $f(x_{i+1})$ would be equal to zero, or

$$0 = f(x_i) + f'(x_i)(x_{i+1} - x_i) \tag{B6.2.2}$$

which can be solved for

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

which is identical to Eq. (6.6). Thus, we have derived the Newton-Raphson formula using a Taylor series.

Aside from the derivation, the Taylor series can also be used to estimate the error of the formula. This can be done by realizing that if the complete Taylor series were employed, an exact result would be obtained. For this situation $x_{i+1} = x_r$, where x is the true value of the root. Substituting this value along with $f(x_r) = 0$ into Eq. (B6.2.1) yields

$$0 = f(x_i) + f'(x_i)(x_r - x_i) + \frac{f''(\xi)}{2!}(x_r - x_i)^2 \tag{B6.2.3}$$

Equation (B6.2.2) can be subtracted from Eq. (B6.2.3) to give

$$0 = f'(x_i)(x_r - x_{i+1}) + \frac{f''(\xi)}{2!}(x_r - x_i)^2 \tag{B6.2.4}$$

Now, realize that the error is equal to the discrepancy between x_{i+1} and the true value x_r, as in

$$E_{t,i+1} = x_r - x_{i+1}$$

and Eq. (B6.2.4) can be expressed as

$$0 = f'(x_i)E_{t,i+1} + \frac{f''(\xi)}{2!}E_{t,i}^2 \tag{B6.2.5}$$

If we assume convergence, both x_i and ξ should eventually be approximated by the root x_r, and Eq. (B6.2.5) can be rearranged to yield

$$E_{t,i+1} \cong \frac{-f''(x_r)}{2f'(x_r)}E_{t,i}^2 \tag{B6.2.6}$$

According to Eq. (B6.2.6), the error is roughly proportional to the square of the previous error. This means that the number of correct decimal places approximately doubles with each iteration. Such behavior is referred to as *quadratic convergence*. Example 6.4 manifests this property.

the number of significant figures of accuracy approximately doubles with each iteration. This behavior is examined in the following example.

EXAMPLE 6.4 Error Analysis of Newton-Raphson Method

Problem Statement. As derived in Box 6.2, the Newton-Raphson method is quadratically convergent. That is, the error is roughly proportional to the square of the previous error, as in

$$E_{t,i+1} \cong \frac{-f''(x_r)}{2f'(x_r)}E_{t,i}^2 \tag{E6.4.1}$$

Examine this formula and see if it applies to the results of Example 6.3.

Solution. The first derivative of $f(x) = e^{-x} - x$ is

$$f'(x) = -e^{-x} - 1$$

1. A plotting routine should be included in the program.
2. At the end of the computation, the final root estimate should always be substituted into the original function to compute whether the result is close to zero. This check partially guards against those cases where slow or oscillating convergence may lead to a small value of ε_a while the solution is still far from a root.
3. The program should always include an upper limit on the number of iterations to guard against oscillating, slowly convergent, or divergent solutions that could persist interminably.
4. The program should alert the user and take account of the possibility that $f'(x)$ might equal zero at any time during the computation.

6.3 THE SECANT METHOD

A potential problem in implementing the Newton-Raphson method is the evaluation of the derivative. Although this is not inconvenient for polynomials and many other functions, there are certain functions whose derivatives may be extremely difficult or inconvenient to evaluate. For these cases, the derivative can be approximated by a backward finite divided difference, as in (Fig. 6.7)

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

This approximation can be substituted into Eq. (6.6) to yield the following iterative equation:

$$x_{i+1} = x_i - \frac{f(x_i)(x_{i-1} - x_i)}{f(x_{i-1}) - f(x_i)} \tag{6.7}$$

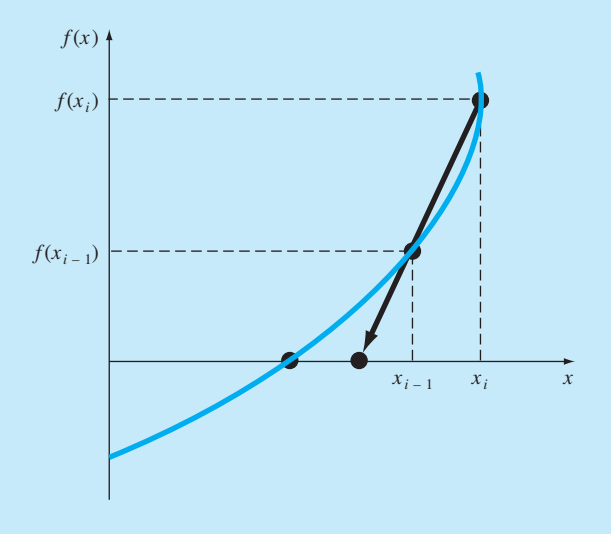


FIGURE 6.7
Graphical depiction of the secant method. This technique is similar to the Newton-Raphson technique (Fig. 6.5) in the sense that an estimate of the root is predicted by extrapolating a tangent of the function to the *x* axis. However, the secant method uses a difference rather than a derivative to estimate the slope.

Equation (6.7) is the formula for the *secant method*. Notice that the approach requires two initial estimates of x. However, because $f(x)$ is not required to change signs between the estimates, it is not classified as a bracketing method.

EXAMPLE 6.6 The Secant Method

Problem Statement. Use the secant method to estimate the root of $f(x) = e^{-x} - x$. Start with initial estimates of $x_{-1} = 0$ and $x_0 = 1.0$.

Solution. Recall that the true root is 0.56714329. . . .

First iteration:

$$x_{-1} = 0 \qquad f(x_{-1}) = 1.00000$$
$$x_0 = 1 \qquad f(x_0) = -0.63212$$
$$x_1 = 1 - \frac{-0.63212(0 - 1)}{1 - (-0.63212)} = 0.61270 \qquad \varepsilon_t = 8.0\%$$

Second iteration:

$$x_0 = 1 \qquad f(x_0) = -0.63212$$
$$x_1 = 0.61270 \qquad f(x_1) = -0.07081$$

(Note that both estimates are now on the same side of the root.)

$$x_2 = 0.61270 - \frac{-0.07081(1 - 0.61270)}{-0.63212 - (-0.07081)} = 0.56384 \qquad \varepsilon_t = 0.58\%$$

Third iteration:

$$x_1 = 0.61270 \qquad f(x_1) = -0.07081$$
$$x_2 = 0.56384 \qquad f(x_2) = 0.00518$$
$$x_3 = 0.56384 - \frac{0.00518(0.61270 - 0.56384)}{-0.07081 - (-0.00518)} = 0.56717 \qquad \varepsilon_t = 0.0048\%$$

6.3.1 The Difference Between the Secant and False-Position Methods

Note the similarity between the secant method and the false-position method. For example, Eqs. (6.7) and (5.7) are identical on a term-by-term basis. Both use two initial estimates to compute an approximation of the slope of the function that is used to project to the x axis for a new estimate of the root. However, a critical difference between the methods is how one of the initial values is replaced by the new estimate. Recall that in the false-position method the latest estimate of the root replaces whichever of the original values yielded a function value with the same sign as $f(x_r)$. Consequently, the two estimates always bracket the root. Therefore, for all practical purposes, the method always converges because the root is kept within the bracket. In contrast, the secant method replaces the values in strict sequence, with the new value x_{i+1} replacing x_i and x_i replacing x_{i-1}. As a result, the two values can sometimes lie on the same side of the root. For certain cases, this can lead to divergence.

Starting with an initial guess of $x_0 = 0$, this iterative equation can be applied to compute

i	x_i	ε_a (%)	ε_t (%)
0	0		100.0
1	1.000000	100.0	76.3
2	0.367879	171.8	35.1
3	0.692201	46.9	22.1
4	0.500473	38.3	11.8
5	0.606244	17.4	6.89
6	0.545396	11.2	3.83
7	0.579612	5.90	2.20
8	0.560115	3.48	1.24
9	0.571143	1.93	0.705
10	0.564879	1.11	0.399

Thus, each iteration brings the estimate closer to the true value of the root: 0.56714329.

6.1.1 Convergence

Notice that the true percent relative error for each iteration of Example 6.1 is roughly proportional (by a factor of about 0.5 to 0.6) to the error from the previous iteration. This property, called *linear convergence,* is characteristic of fixed-point iteration.

Aside from the "rate" of convergence, we must comment at this point about the "possibility" of convergence. The concepts of convergence and divergence can be depicted graphically. Recall that in Sec. 5.1, we graphed a function to visualize its structure and behavior (Example 5.1). Such an approach is employed in Fig. 6.2*a* for the function $f(x) = e^{-x} - x$. An alternative graphical approach is to separate the equation into two component parts, as in

$$f_1(x) = f_2(x)$$

Then the two equations

$$y_1 = f_1(x) \tag{6.3}$$

and

$$y_2 = f_2(x) \tag{6.4}$$

can be plotted separately (Fig. 6.2*b*). The x values corresponding to the intersections of these functions represent the roots of $f(x) = 0$.

EXAMPLE 6.2 **The Two-Curve Graphical Method**

Problem Statement. Separate the equation $e^{-x} - x = 0$ into two parts and determine its root graphically.

Box 6.1 Convergence of Fixed-Point Iteration

From studying Fig. 6.3, it should be clear that fixed-point iteration converges if, in the region of interest, $|g'(x)| < 1$. In other words, convergence occurs if the magnitude of the slope of $g(x)$ is less than the slope of the line $f(x) = x$. This observation can be demonstrated theoretically. Recall that the iterative equation is

$$x_{i+1} = g(x_i)$$

Suppose that the true solution is

$$x_r = g(x_r)$$

Subtracting these equations yields

$$x_r - x_{i+1} = g(x_r) - g(x_i) \tag{B6.1.1}$$

The *derivative mean-value theorem* (recall Sec. 4.1.1) states that if a function $g(x)$ and its first derivative are continuous over an interval $a \le x \le b$, then there exists at least one value of $x = \xi$ within the interval such that

$$g'(\xi) = \frac{g(b) - g(a)}{b - a} \tag{B6.1.2}$$

The right-hand side of this equation is the slope of the line joining $g(a)$ and $g(b)$. Thus, the mean-value theorem states that there is at least one point between a and b that has a slope, designated by $g'(\xi)$, which is parallel to the line joining $g(a)$ and $g(b)$ (recall Fig. 4.3).

Now, if we let $a = x_i$ and $b = x_r$, the right-hand side of Eq. (B6.1.1) can be expressed as

$$g(x_r) - g(x_i) = (x_r - x_i)g'(\xi)$$

where ξ is somewhere between x_i and x_r. This result can then be substituted into Eq. (B6.1.1) to yield

$$x_r - x_{i+1} = (x_r - x_i)g'(\xi) \tag{B6.1.3}$$

If the true error for iteration i is defined as

$$E_{t,i} = x_r - x_i$$

then Eq. (B6.1.3) becomes

$$E_{t,i+1} = g'(\xi)E_{t,i}$$

Consequently, if $|g'(x)| < 1$, the errors decrease with each iteration. For $|g'(x)| > 1$, the errors grow. Notice also that if the derivative is positive, the errors will be positive, and hence, the iterative solution will be monotonic (Fig. 6.3*a* and *c*). If the derivative is negative, the errors will oscillate (Fig. 6.3*b* and *d*).

An offshoot of the analysis is that it also demonstrates that when the method converges, the error is roughly proportional to and less than the error of the previous step. For this reason, simple fixed-point iteration is said to be *linearly convergent*.

Thus, in both the equation and in the plot, a starting value of x_0 is used to obtain an estimate of x_1. The next iteration consists of moving to $[x_1, g(x_1)]$ and then to (x_2, x_2). This iteration is equivalent to the equation

$$x_2 = g(x_1)$$

The solution in Fig. 6.3*a* is *convergent* because the estimates of x move closer to the root with each iteration. The same is true for Fig. 6.3*b*. However, this is not the case for Fig. 6.3*c* and *d*, where the iterations diverge from the root. Notice that convergence seems to occur only when the absolute value of the slope of $y_2 = g(x)$ is less than the slope of $y_1 = x$, that is, when $|g'(x)| < 1$. Box 6.1 provides a theoretical derivation of this result.

6.1.2 Algorithm for Fixed-Point Iteration

The computer algorithm for fixed-point iteration is extremely simple. It consists of a loop to iteratively compute new estimates until the termination criterion has been met. Figure 6.4

30.2.1 Convergence and Stability

Convergence means that as Δx and Δt approach zero, the results of the finite-difference technique approach the true solution. *Stability* means that errors at any stage of the computation are not amplified but are attenuated as the computation progresses. It can be shown (Carnahan et al., 1969) that the explicit method is both convergent and stable if $\lambda \leq 1/2$, or

$$\Delta t \leq \frac{1}{2}\frac{\Delta x^2}{k} \tag{30.6}$$

In addition, it should be noted that setting $\lambda \leq 1/2$ could result in a solution in which errors do not grow, but oscillate. Setting $\lambda \leq 1/4$ ensures that the solution will not oscillate. It is also known that setting $\lambda = 1/6$ tends to minimize truncation error (Carnahan et al., 1969).

Figure 30.5 is an example of instability caused by violating Eq. (30.6). This plot is for the same case as in Example 30.1 but with $\lambda = 0.735$, which is considerably greater than 0.5. As in Fig. 30.5, the solution undergoes progressively increasing oscillations. This situation will continue to deteriorate as the computation continues.

Although satisfaction of Eq. (30.6) will alleviate the instabilities of the sort manifested in Fig. 30.5, it also places a strong limitation on the explicit method. For example, suppose that Δx is halved to improve the approximation of the spatial second derivative. According to Eq. (30.6), the time step must be quartered to maintain convergence and stability. Thus, to perform comparable computations, the time steps must be increased by a factor of 4. Furthermore, the computation for each of these time steps will take twice as long because halving Δx doubles the total number of nodes for which equations must be written. Consequently, for the one-dimensional case, halving Δx results in an eightfold increase in the number of calculations. Thus, the computational burden may be large to attain acceptable accuracy. As will be described shortly, other techniques are available that do not suffer from such severe limitations.

30.2.2 Derivative Boundary Conditions

As was the case for elliptic PDEs (recall Sec. 29.3.1), derivative boundary conditions can be readily incorporated into parabolic equations. For a one-dimensional rod, this necessitates adding two equations to characterize the heat balance at the end nodes. For example, the node at the left end ($i = 0$) would be represented by

$$T_0^{l+1} = T_0^l + \lambda(T_1^l - 2T_0^l + T_{-1}^l)$$

Thus, an imaginary point is introduced at $i = -1$ (recall Fig. 29.7). However, as with the elliptic case, this point provides a vehicle for incorporating the derivative boundary condition into the analysis. Problem 30.2 at the end of the chapter deals with this exercise.

30.2.3 Higher-Order Temporal Approximations

The general idea of re-expressing the PDE as a system of ODEs is sometimes called the *method of lines*. Obviously, one way to improve on the Euler approach used above would be to employ a more accurate integration scheme for solving the ODEs. For example, the Heun method can be employed to obtain second-order temporal accuracy. An example of