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:

$$x_l = 12 \qquad f(x_l) = 6.0699$$
$$x_u = 16 \qquad f(x_u) = -2.2688$$
$$x_r = 16 - \frac{-2.2688(12 - 16)}{6.0669 - (-2.2688)} = 14.9113$$

which has a true relative error of 0.89 percent.

Second iteration:

$$f(x_l) f(x_r) = -1.5426$$

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

$$x_l = 12 \qquad f(x_l) = 6.0699$$
$$x_u = 14.9113 \qquad f(x_u) = -0.2543$$
$$x_r = 14.9113 - \frac{-0.2543(12 - 14.9113)}{6.0669 - (-0.2543)} = 14.7942$$

which has true and approximate relative errors of 0.09 and 0.79 percent. 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

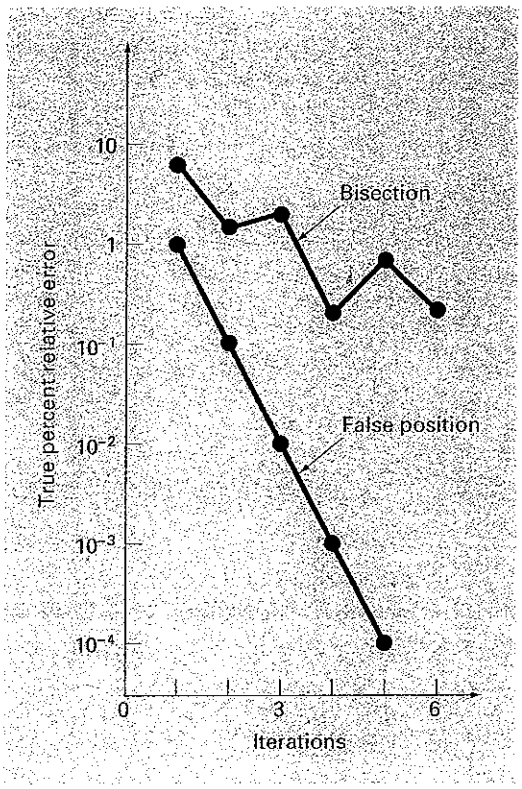


FIGURE 5.13
Comparison of the relative errors of the bisection and the false-position methods.

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 5.6 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 5.6 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 5.6, 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

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 percent. 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

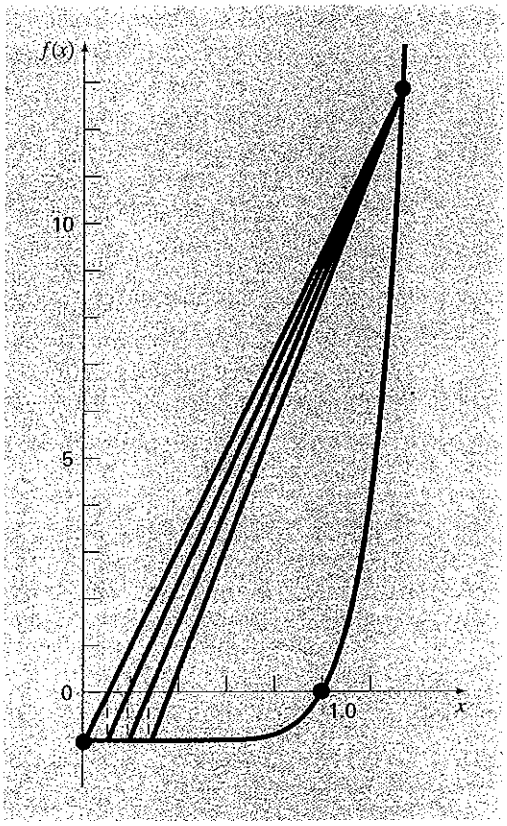


FIGURE 5.14
Plot of $f(x) = x^{10} - 1$, illustrating slow convergence of the false-position method.

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 methods

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
```

would converge in 14 and 39 iterations, respectively. In contrast, the modified false-position method would converge in 12 iterations. Thus, for this example, it is somewhat more efficient than bisection and is vastly superior to the unmodified false-position method.

5.4 INCREMENTAL SEARCHES AND DETERMINING INITIAL GUESSES

Besides checking an individual answer, you must determine whether all possible roots have been located. As mentioned previously, a plot of the function is usually very useful in guiding you in this task. Another option is to incorporate an incremental search at the beginning of the computer program. This consists of starting at one end of the region of interest and then making function evaluations at small increments across the region. When the function changes sign, it is assumed that a root falls within the increment. The x values at the beginning and the end of the increment can then serve as the initial guesses for one of the bracketing techniques described in this chapter.

A potential problem with an incremental search is the choice of the increment length. If the length is too small, the search can be very time consuming. On the other hand, if the length is too great, there is a possibility that closely spaced roots might be missed (Fig. 5.16). The problem is compounded by the possible existence of multiple roots. A partial remedy for such cases is to compute the first derivative of the function $f'(x)$ at the beginning and the end of each interval. If the derivative changes sign, it suggests that a minimum or maximum may have occurred and that the interval should be examined more closely for the existence of a possible root.

Although such modifications or the employment of a very fine increment can alleviate the problem, it should be clear that brute-force methods such as incremental search are not foolproof. You would be wise to supplement such automatic techniques with any other information that provides insight into the location of the roots. Such information can be found in plotting and in understanding the physical problem from which the equation originated.

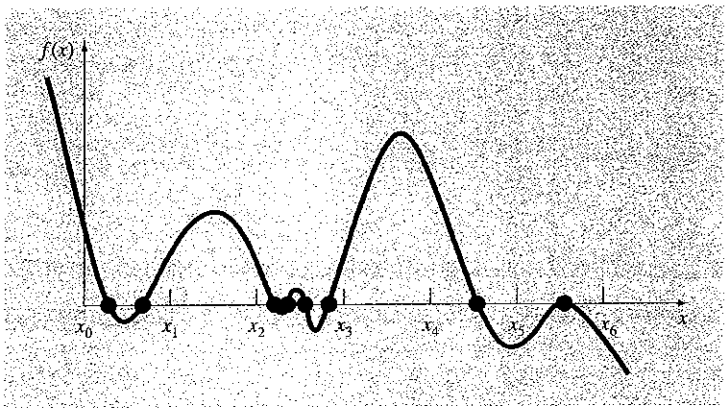


FIGURE 5.16
Cases where roots could be missed because the increment length of the search procedure is too large. Note that the last root on the right is multiple and would be missed regardless of increment length.

PROBLEMS

5.1 Determine the real roots of $f(x) = -0.5x^2 + 2.5x + 4.5$:
(a) Graphically.
(b) Using the quadratic formula.
(c) Using three iterations of the bisection method to determine the highest root. Employ initial guesses of $x_l = 5$ and $x_u = 10$. Compute the estimated error ε_a and the true error ε_t after each iteration.

5.2 Determine the real root of $f(x) = 5x^3 - 5x^2 + 6x - 2$:
(a) Graphically.
(b) Using bisection to locate the root. Employ initial guesses of $x_l = 0$ and $x_u = 1$ and iterate until the estimated error ε_a falls below a level of $\varepsilon_s = 10\%$.

5.3 Determine the real root of $f(x) = -25 + 82x - 90x^2 + 44x^3 - 8x^4 + 0.7x^5$:
(a) Graphically.
(b) Using bisection to determine the root to $\varepsilon_s = 10\%$. Employ initial guesses of $x_l = 0.5$ and $x_u = 1.0$.
(c) Perform the same computation as in **(b)** but use the false-position method and $\varepsilon_s = 0.2\ \%$.

5.4 **(a)** Determine the roots of $f(x) = -12 - 21x + 18x^2 - 2.75x^3$ graphically. In addition, determine the first root of the function with **(b)** bisection, and **(c)** false position. For **(b)** and **(c)** use initial guesses of $x_l = -1$ and $x_u = 0$, and a stopping criterion of 1%.

5.5 Locate the first nontrivial root of $\sin x = x^3$, where x is in radians. Use a graphical technique and bisection with the initial interval from 0.5 to 1. Perform the computation until ε_a is less than $\varepsilon_s = 2\%$. Also perform an error check by substituting your final answer into the original equation.

5.6 Determine the positive real root of $\ln(x^4) = 0.7$ **(a)** graphically, **(b)** using three iterations of the bisection method, with initial guesses of $x_l = 0.5$ and $x_u = 2$, and **(c)** using three iterations of the false-position method, with the same initial guesses as in **(b)**.

5.7 Determine the real root of $f(x) = (0.8 - 0.3x)/x$:
(a) Analytically.
(b) Graphically.
(c) Using three iterations of the false-position method and initial guesses of 1 and 3. Compute the approximate error ε_a and the true error ε_t after each iteration. Is there a problem with the result?

5.8 Find the positive square root of 18 using the false-position method to within $\varepsilon_s = 0.5\%$. Employ initial guesses of $x_l = 4$ and $x_u = 5$.

5.9 Find the smallest positive root of the function (x is in radians) $x^2|\cos\sqrt{x}| = 5$ using the false-position method. To locate the region in which the root lies, first plot this function for values of x between 0 and 5. Perform the computation until ε_a falls below $\varepsilon_s = 1\%$. Check your final answer by substituting it into the original function.

5.10 Find the positive real root of $f(x) = x^4 - 8x^3 - 35x^2 + 450x - 1001$ using the false-position method. Use initial guesses of $x_l = 4.5$ and $x_u = 6$ and performs five iterations. Compute both the true and approximate errors based on the fact that the root is 5.60979. Use a plot to explain your results and perform the computation to within $\varepsilon_s = 1.0\%$.

5.11 Determine the real root of $x^{3.5} = 80$: **(a)** analytically, and **(b)** with the false-position method to within $\varepsilon_s = 2.5\%$. Use initial guesses of 2.0 and 5.0.

5.12 Given

$$f(x) = -2x^6 - 1.5x^4 + 10x + 2$$

Use bisection to determine the *maximum* of this function. Employ initial guesses of $x_l = 0$ and $x_u = 1$, and perform iterations until the approximate relative error falls below 5%.

5.13 The velocity v of a falling parachutist is given by

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

where $g = 9.8\ \text{m/s}^2$. For a parachutist with a drag coefficient $c = 15$ kg/s, compute the mass m so that the velocity is $v = 35$ m/s at $t = 9$ s. Use the false-position method to determine m to a level of $\varepsilon_s = 0.1\%$.

5.14 A beam is loaded as shown in Fig. P5.14. Use the bisection method to solve for the position inside the beam where there is no moment.

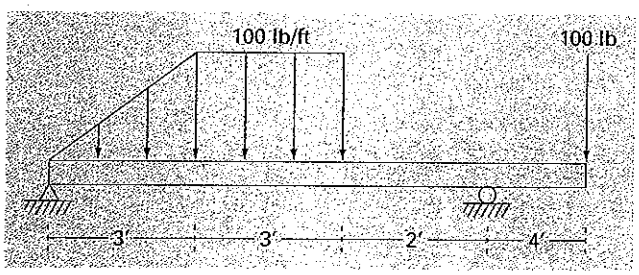


Figure P5.14

5.15 Water is flowing in a trapezoidal channel at a rate of $Q = 20\ \text{m}^3/\text{s}$. The critical depth y for such a channel must satisfy the equation

$$0 = 1 - \frac{Q^2}{gA_c^3}B$$

where $g = 9.81$ m/s^2, A_c = the cross-sectional area (m^2), and B = the width of the channel at the surface (m). For this case, the width and the cross-sectional area can be related to depth y by

$$B = 3 + y \qquad \text{and} \qquad A_c = 3y + \frac{y^2}{2}$$

Solve for the critical depth using **(a)** the graphical method, **(b)** bisection, and **(c)** false position. For **(b)** and **(c)** use initial guesses of $x_l = 0.5$ and $x_u = 2.5$, and iterate until the approximate error falls below 1% or the number of iterations exceeds 10. Discuss your results.

5.16 You are designing a spherical tank (Fig. P5.16) to hold water for a small village in a developing country. The volume of liquid it can hold can be computed as

$$V = \pi h^2 \frac{[3R - h]}{3} 2$$

where V = volume [m^3], h = depth of water in tank [m], and R = the tank radius [m].

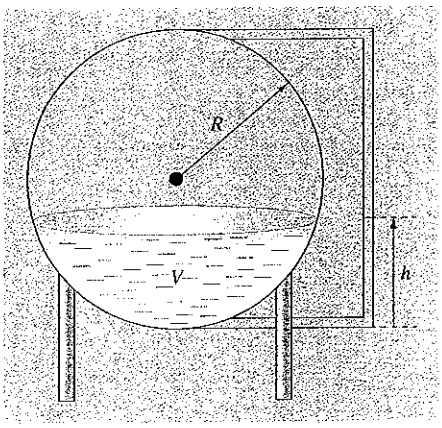


Figure P5.16

If $R = 3$ m, to what depth must the tank be filled so that it holds 30 m^3? Use three iterations of the false-position method to determine your answer. Determine the approximate relative error after each iteration.

5.17 The saturation concentration of dissolved oxygen in freshwater can be calculated with the equation (APHA, 1992)

$$\ln o_{sf} = -139.34411 + \frac{1.575701 \times 10^5}{T_a} - \frac{6.642308 \times 10^7}{T_a^2} + \frac{1.243800 \times 10^{10}}{T_a^3} - \frac{8.621949 \times 10^{11}}{T_a^4}$$

where o_{sf} = the saturation concentration of dissolved oxygen in freshwater at 1 atm (mg/L) and T_a = absolute temperature (K). Remember that $T_a = T + 273.15$, where T = temperature (°C). According to this equation, saturation decreases with increasing temperature. For typical natural waters in temperate climates, the equation can be used to determine that oxygen concentration ranges from 14.621 mg/L at 0°C to 6.413 mg/L at 40°C. Given a value of oxygen concentration, this formula and the bisection method can be used to solve for temperature in °C.

(a) If the initial guesses are set as 0 and 40°C, how many bisection iterations would be required to determine temperature to an absolute error of 0.05°C?

(b) Develop and test a bisection program to determine T as a function of a given oxygen concentration to a prespecified absolute error as in **(a)**. Given initial guesses of 0 and 40°C, test your program for an absolute error = 0.05°C and the following cases: $o_{sf} = 8$, 10 and 12 mg/L. Check your results.

5.18 Integrate the algorithm outlined in Fig. 5.10 into a complete, user-friendly bisection subprogram. Among other things:

(a) Place documentation statements throughout the subprogram to identify what each section is intended to accomplish.

(b) Label the input and output.

(c) Add an answer check that substitutes the root estimate into the original function to verify whether the final result is close to zero.

(d) Test the subprogram by duplicating the computations from Examples 5.3 and 5.4.

5.19 Develop a subprogram for the bisection method that minimizes function evaluations based on the pseudocode from Fig. 5.11. Determine the number of function evaluations (n) per total iterations. Test the program by duplicating Example 5.6.

5.20 Develop a user-friendly program for the false-position method. The structure of your program should be similar to the bisection algorithm outlined in Fig. 5.10. Test the program by duplicating Example 5.5.

5.21 Develop a subprogram for the false-position method that minimizes function evaluations in a fashion similar to Fig. 5.11. Determine the number of function evaluations (n) per total iterations. Test the program by duplicating Example 5.6.

5.22 Develop a user-friendly subprogram for the modified false-position method based on Fig. 5.15. Test the program by determining the root of the function described in Example 5.6. Perform a number of runs until the true percent relative error falls below 0.01%. Plot the true and approximate percent relative errors versus number of iterations on semilog paper. Interpret your results.

HAND OUT 8: Open methods (Chapter 2 of our syllabus). Source: Chapra, S. C., and Canale, R. P. (2006). *"Numerical methods for engineers."* McGraw-Hill, fifth edition.

CHAPTER 6

Open Methods

For the bracketing methods in the previous chapter, the root is located within an interval prescribed by a lower and an upper bound. Repeated application of these methods always results in closer estimates of the true value of the root. Such methods are said to be *convergent* because they move closer to the truth as the computation progresses (Fig. 6.1*a*).

In contrast, the *open methods* described in this chapter are based on formulas that require only a single starting value of x or two starting values that do not necessarily bracket

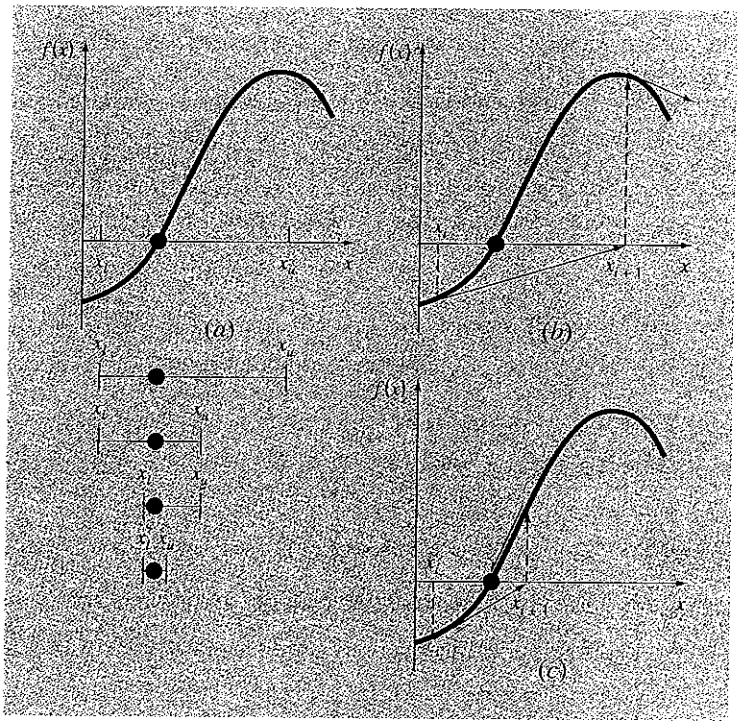


FIGURE 6.1
Graphical depiction of the fundamental difference between the (*a*) bracketing and (*b*) and (*c*) open methods for root location. In (*a*), which is the bisection method, the root is constrained within the interval prescribed by x_l and x_u. In contrast, for the open method depicted in (*b*) and (*c*), a formula is used to project from x_i to x_{i+1} in an iterative fashion. Thus, the method can either (*b*) diverge or (*c*) converge rapidly, depending on the value of the initial guess.

the root. As such, they sometimes *diverge* or move away from the true root as the computation progresses (Fig. 6.1*b*). However, when the open methods converge (Fig. 6.1*c*), they usually do so much more quickly than the bracketing methods. We will begin our discussion of open techniques with a simple version that is useful for illustrating their general form and also for demonstrating the concept of convergence.

6.1 SIMPLE FIXED-POINT ITERATION

As mentioned above, open methods employ a formula to predict the root. Such a formula can be developed for simple *fixed-point iteration* (or, as it is also called, one-point iteration or successive substitution) by rearranging the function $f(x) = 0$ so that x is on the left-hand side of the equation:

$$x = g(x) \tag{6.1}$$

This transformation can be accomplished either by algebraic manipulation or by simply adding x to both sides of the original equation. For example,

$$x^2 - 2x + 3 = 0$$

can be simply manipulated to yield

$$x = \frac{x^2 + 3}{2}$$

whereas $\sin x = 0$ could be put into the form of Eq. (6.1) by adding x to both sides to yield

$$x = \sin x + x$$

The utility of Eq. (6.1) is that it provides a formula to predict a new value of x as a function of an old value of x. Thus, given an initial guess at the root x_i, Eq. (6.1) can be used to compute a new estimate x_{i+1} as expressed by the iterative formula

$$x_{i+1} = g(x_i) \tag{6.2}$$

As with other iterative formulas in this book, the approximate error for this equation can be determined using the error estimator [Eq. (3.5)]:

$$\varepsilon_a = \left| \frac{x_{i+1} - x_i}{x_{i+1}} \right| 100\%$$

EXAMPLE 6.1 Simple Fixed-Point Iteration

Problem Statement. Use simple fixed-point iteration to locate the root of $f(x) = e^{-x} - x$.

Solution. The function can be separated directly and expressed in the form of Eq. (6.2) as

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

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

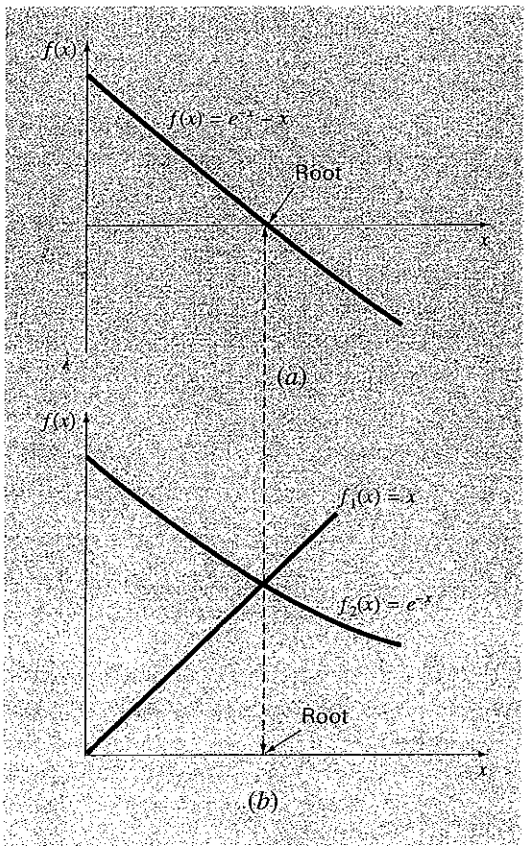
Problem Statement. Separate the equation $e^{-x} - x = 0$ into two parts and determine its root graphically.

Solution. Reformulate the equation as $y_1 = x$ and $y_2 = e^{-x}$. The following values can be computed:

x	y_1	y_2
0.0	0.0	1.000
0.2	0.2	0.819
0.4	0.4	0.670
0.6	0.6	0.549
0.8	0.8	0.449
1.0	1.0	0.368

These points are plotted in Fig. 6.2*b*. The intersection of the two curves indicates a root estimate of approximately $x = 0.57$, which corresponds to the point where the single curve in Fig. 6.2*a* crosses the x axis.

FIGURE 6.2
Two alternative graphical methods for determining the root of $f(x) = e^{-x} - x$. (*a*) Root at the point where it crosses the x axis; (*b*) root at the intersection of the component functions.



The two-curve method can now be used to illustrate the convergence and divergence of fixed-point iteration. First, Eq. (6.1) can be re-expressed as a pair of equations $y_1 = x$ and $y_2 = g(x)$. These two equations can then be plotted separately. As was the case with Eqs. (6.3) and (6.4), the roots of $f(x) = 0$ correspond to the abscissa value at the intersection of the two curves. The function $y_1 = x$ and four different shapes for $y_2 = g(x)$ are plotted in Fig. 6.3.

For the first case (Fig. 6.3*a*), the initial guess of x_0 is used to determine the corresponding point on the y_2 curve $[x_0, g(x_0)]$. The point (x_1, x_1) is located by moving left horizontally to the y_1 curve. These movements are equivalent to the first iteration in the fixed-point method:

$$x_1 = g(x_0)$$

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

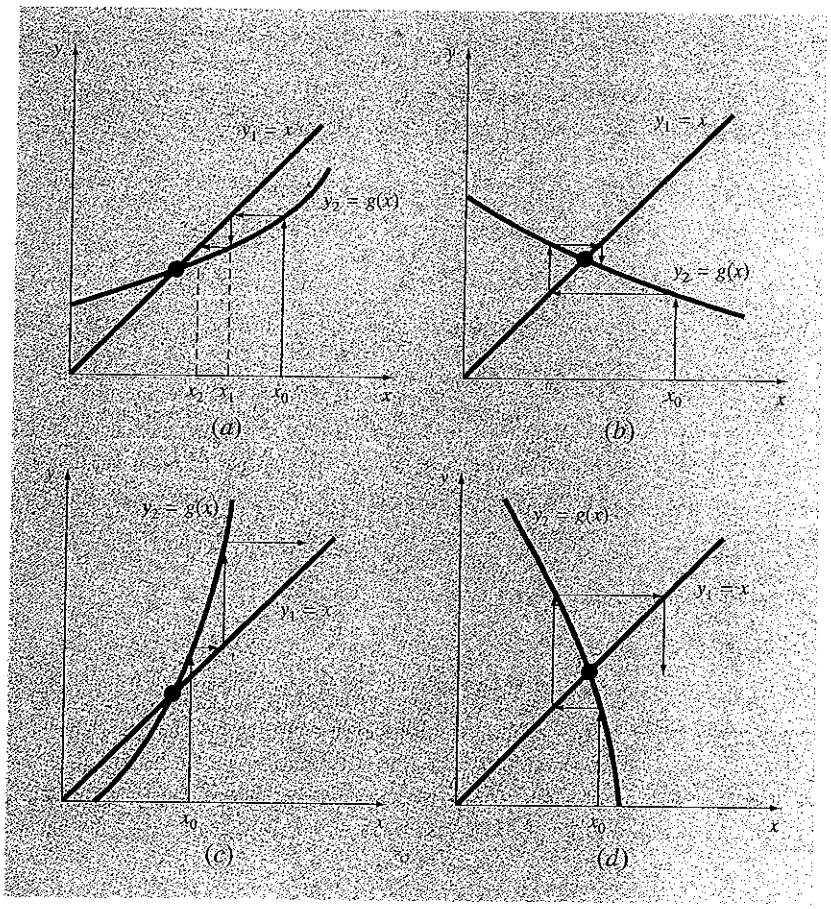


FIGURE 6.3
Graphical depiction of (*a*) and (*b*) convergence and (*c*) and (*d*) divergence of simple fixed-point iteration. Graphs (*a*) and (*c*) are called monotone patterns, whereas (*b*) and (*d*) are called oscillating or spiral patterns. Note that convergence occurs when $|g'(x)| < 1$.

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*.

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 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.

```
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.

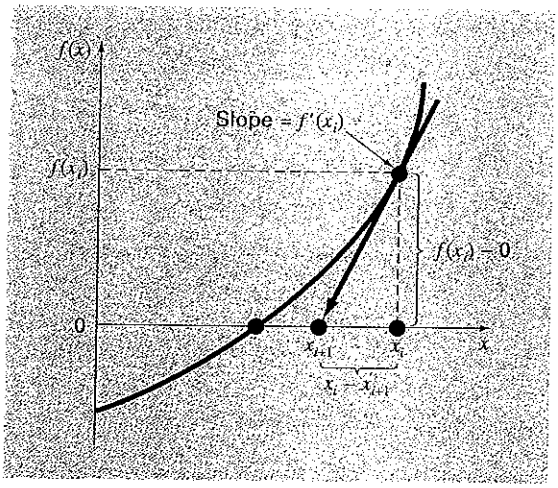


FIGURE 6.5
Graphical depiction of the Newton-Raphson method. A tangent to the function of x_i [that is, $f'(x_i)$] is extrapolated down to the x axis to provide an estimate of the root at x_{i+1}.

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.

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 (%)
0	0	100
1	0.500000000	11.8
2	0.566311003	0.147
3	0.567143165	0.0000220
4	0.567143290	$< 10^{-8}$

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, the

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

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

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.

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

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

which can be evaluated at $x_r = 0.56714329$ as $f'(0.56714329) = -1.56714329$. The second derivative is

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

which can be evaluated as $f''(0.56714329) = 0.56714329$. These results can be substituted into Eq. (E6.4.1) to yield

$$E_{t,i+1} \cong -\frac{0.56714329}{2(-1.56714329)} E_{t,i}^2 = 0.18095 E_{t,i}^2$$

From Example 6.3, the initial error was $E_{t,0} = 0.56714329$, which can be substituted into the error equation to predict

$$E_{t,1} \cong 0.18095(0.56714329)^2 = 0.0582$$

which is close to the true error of 0.06714329. For the next iteration,

$$E_{t,2} \cong 0.18095(0.06714329)^2 = 0.0008158$$

which also compares favorably with the true error of 0.0008323. For the third iteration,

$$E_{t,3} \cong 0.18095(0.0008323)^2 = 0.000000125$$

which is the error obtained in Example 6.3. The error estimate improves in this manner because, as we come closer to the root, x and ξ are better approximated by x_r [recall our assumption in going from Eq. (B6.2.5) to Eq. (B6.2.6) in Box 6.2]. Finally,

$$E_{t,4} \cong 0.18095(0.000000125)^2 = 2.83 \times 10^{-15}$$

Thus, this example illustrates that the error of the Newton-Raphson method for this case is, in fact, roughly proportional (by a factor of 0.18095) to the square of the error of the previous iteration.

6.2.2 Pitfalls of the Newton-Raphson Method

Although the Newton-Raphson method is often very efficient, there are situations where it performs poorly. A special case—multiple roots—will be addressed later in this chapter. However, even when dealing with simple roots, difficulties can also arise, as in the following example.

EXAMPLE 6.5 Example of a Slowly Converging Function with Newton-Raphson

Problem Statement. Determine the positive root of $f(x) = x^{10} - 1$ using the Newton-Raphson method and an initial guess of $x = 0.5$.

Solution. The Newton-Raphson formula for this case is

$$x_{i+1} = x_i - \frac{x_i^{10} - 1}{10x_i^9}$$

which can be used to compute

Iteration	x
0	0.5
1	51.65
2	46.485
3	41.8365
4	37.65285
5	33.887565
.	
.	
.	
∞	1.0000000

Thus, after the first poor prediction, the technique is converging on the true root of 1, but at a very slow rate.

Aside from slow convergence due to the nature of the function, other difficulties can arise, as illustrated in Fig. 6.6. For example, Fig. 6.6*a* depicts the case where an inflection point [that is, $f''(x) = 0$] occurs in the vicinity of a root. Notice that iterations beginning at x_0 progressively diverge from the root. Figure 6.6*b* illustrates the tendency of the Newton-Raphson technique to oscillate around a local maximum or minimum. Such oscillations may persist, or as in Fig. 6.6*b*, a near-zero slope is reached, whereupon the solution is sent far from the area of interest. Figure 6.6*c* shows how an initial guess that is close to one root can jump to a location several roots away. This tendency to move away from the area of interest is because near-zero slopes are encountered. Obviously, a zero slope [$f'(x) = 0$] is truly a disaster because it causes division by zero in the Newton-Raphson formula [Eq. (6.6)]. Graphically (see Fig 6.6*d*), it means that the solution shoots off horizontally and never hits the x axis.

Thus, there is no general convergence criterion for Newton-Raphson. Its convergence depends on the nature of the function and on the accuracy of the initial guess. The only remedy is to have an initial guess that is "sufficiently" close to the root. And for some functions, no guess will work! Good guesses are usually predicated on knowledge of the physical problem setting or on devices such as graphs that provide insight into the behavior of the solution. The lack of a general convergence criterion also suggests that good computer software should be designed to recognize slow convergence or divergence. The next section addresses some of these issues.

6.2.3 Algorithm for Newton-Raphson

An algorithm for the Newton-Raphson method is readily obtained by substituting Eq. (6.6) for the predictive formula [Eq. (6.2)] in Fig. 6.4. Note, however, that the program must also be modified to compute the first derivative. This can be simply accomplished by the inclusion of a user-defined function.

Additionally, in light of the foregoing discussion of potential problems of the Newton-Raphson method, the program would be improved by incorporating several additional features:

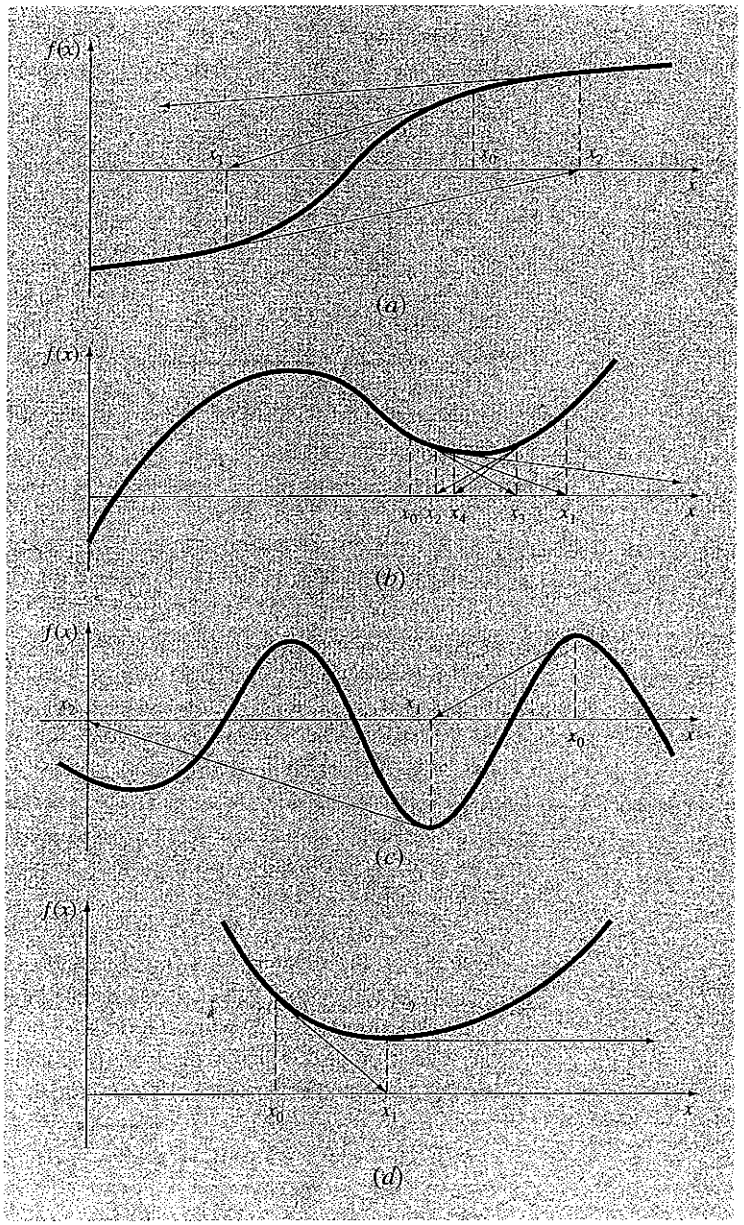


FIGURE 6.6
Four cases where the Newton-Raphson method exhibits poor convergence.

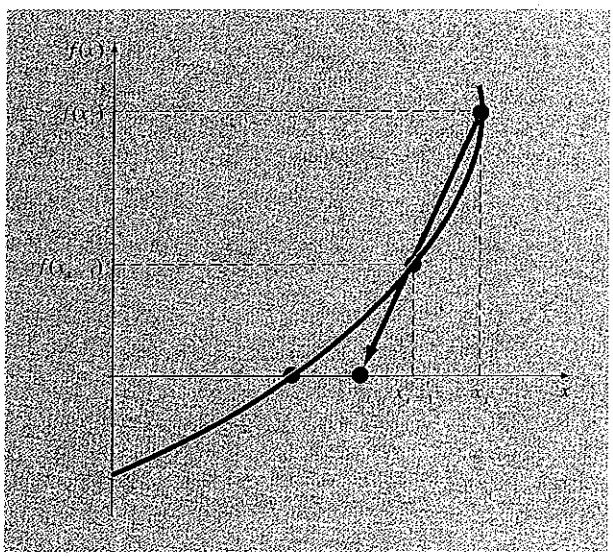
1. A plotting routine should be included in the program.
2. At the end of the computation, the final root estimate should always be substituted into the original function to compute whether the result is close to zero. This check partially guards against those cases where slow or oscillating convergence 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

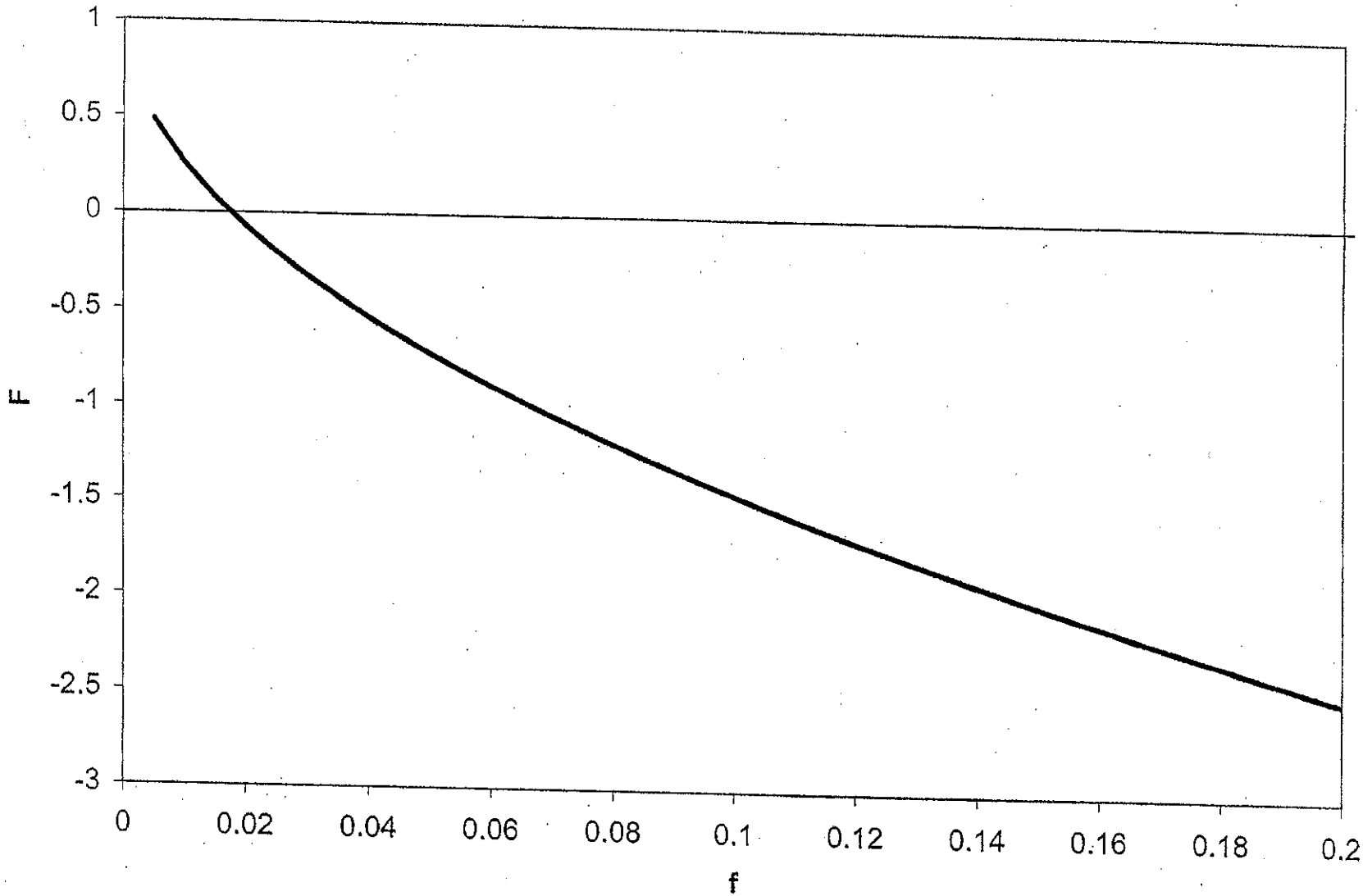
A potential problem in implementing the Newton-Raphson method is the evaluation of the derivative. Although this is not inconvenient for polynomials and many other functions, there are certain functions whose derivatives may be extremely difficult or inconvenient to evaluate. For these cases, the derivative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

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

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.



HAND OUT 9: Behavior of the Colebrook-White equation (Chapter 2 of our syllabus).


1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
F
0
0.02
0.04
0.06
0.08
0.1
0.12
0.14
0.16
0.18
0.2
f

HAND OUT 10: Example of the behavior of the Iteration of a point method (Chapter 2 of our syllabus).

Example of point iteration:

$$F(x) = x^3 - x - 5 = 0$$

x	$F(x)$
1.9	−0.041
2.0	1.0

Alternatives:

(I) $x = x^3 - 5$

(II) $x = \dfrac{5}{x^2 - 1}$

(III) $x = \sqrt[3]{x+5}$

$$x_{k+1} = \varphi(x_k)$$

$$x_0 = 1.9$$

(I)	(II)	(III)
1.859	1.9157088	1.90378
1.42448	1.87270	1.90413
−2.10951	1.99441	1.90416
−14.38738	1.87916	1.90416
−2983.1451	1.635769	

↑

HAND OUT 11: Picture of flow in the Hayden-Rhodes Aqueduct, Central Arizona Project (Chapter 2 of our syllabus). Source: Mays, L. (2006). *"Water resources engineering."* John Wiley and Sons.

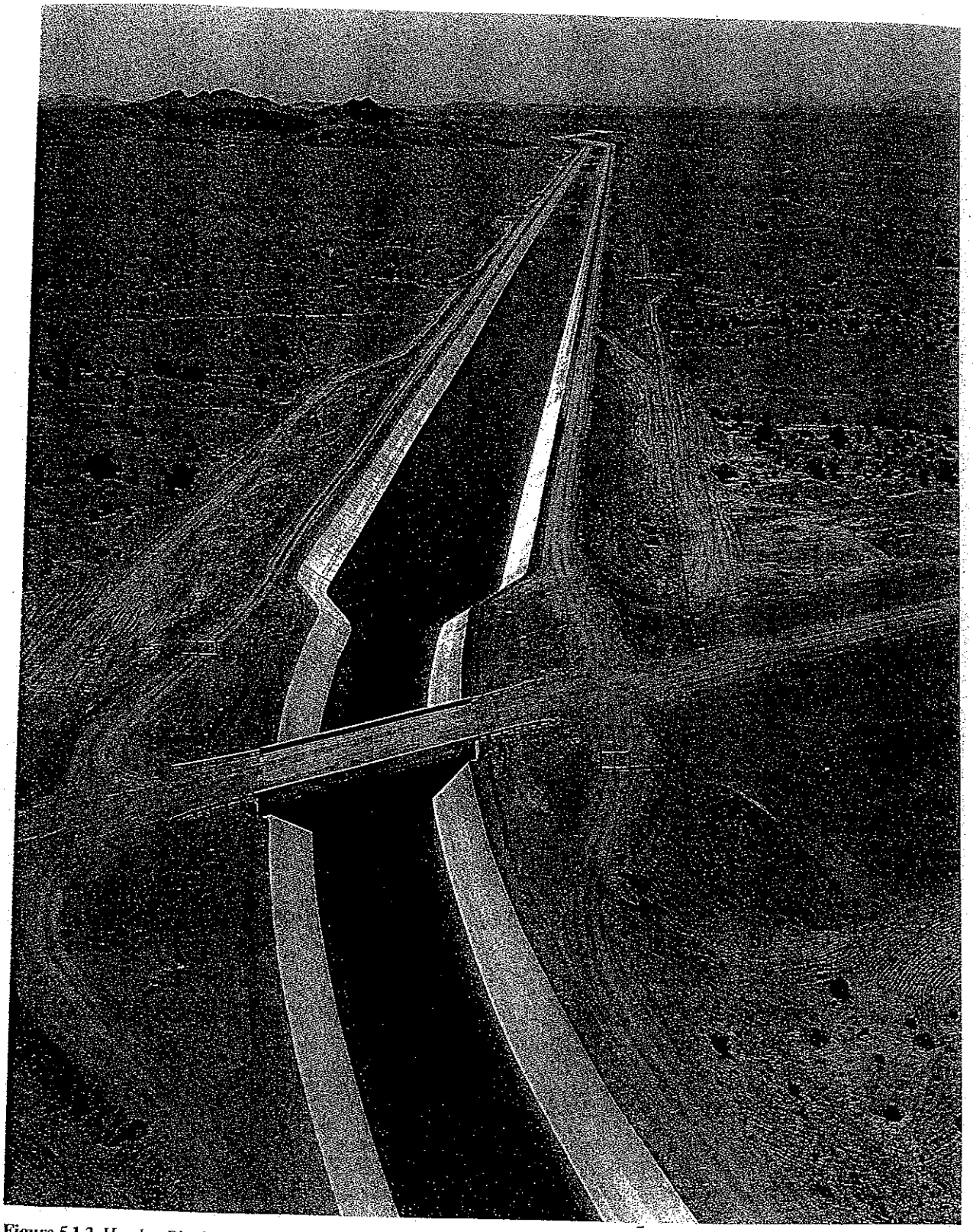


Figure 5.1.2 Hayden-Rhodes Aqueduct, Central Arizona Project. (Courtesy of the U.S. Bureau of Reclamation, (1985), photograph by Joe Madrigal Jr.)

HAND OUT 12: Picture of pollution (Chapter 4 of our syllabus). Source: Cover of the book by Altinakar, M., and Graf, W. (1998). *"Fluvial Hydraulics."* John Wiley and Sons.

HAND OUT 13: Description of the LAKE model (Chapter 4 of our syllabus).
Source: ILEC, International Lake Environment Committee.