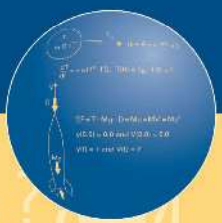
Numerical Methods for Engineers and Scientists

Second Edition
Revised and Expanded



Joe D. Hoffman

3

Nonlinear Equations

3.1 INTRODUCTION

Consider the four-bar linkage illustrated in Figure 3.1. The angle $\alpha = \theta_4 - \pi$ is the input to this mechanism, and the angle $\phi = \theta_2$ is the output. A relationship between α and ϕ can be obtained by writing the vector loop equation:

$$\vec{r}_2 + \vec{r}_3 + \vec{r}_4 - \vec{r}_1 = 0 \tag{3.1}$$

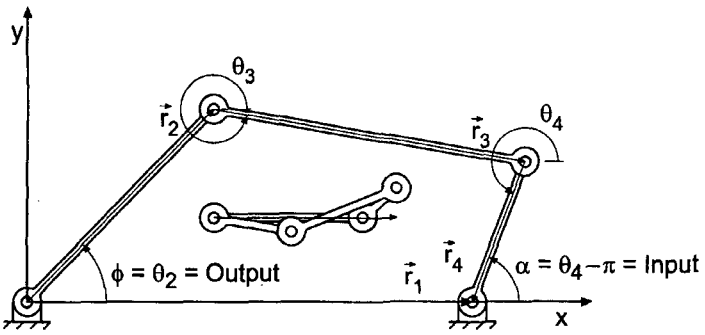


Figure 3.1 Four-bar linkage.

Let $\vec{r}_1$ lie along the x axis. Equation (3.1) can be written as two scalar equations, corresponding to the x and y components of the $\vec{r}$ vectors. Thus,

$$r_2\cos(\theta_2) + r_3\cos(\theta_3) + r_4\cos(\theta_4) - r_1 = 0 \tag{3.2a}$$

$$r_2\sin(\theta_2) + r_3\sin(\theta_3) + r_4\sin(\theta_4) = 0 \tag{3.2b}$$

Combining Eqs. (3.2a) and (3.2b), letting $\theta_2 = \phi$ and $\theta_4 = \alpha + \pi$, and simplifying yields Freudenstein's (1955) equation:

$$R_1\cos(\alpha) - R_2\cos(\phi) + R_3 - \cos(\alpha - \phi) = 0 \tag{3.3}$$

where

$$R_1 = \frac{r_1}{r_2} \qquad R_2 = \frac{r_1}{r_4} \qquad R_3 = \frac{r_1^2 + r_2^2 + r_3^2 + r_4^2}{2r_2r_4} \tag{3.4}$$

Consider the particular four-bar linkage specified by $r_1 = 10, r_2 = 6, r_3 = 8$, and $r_4 = 4$, which is illustrated in Figure 3.1. Thus, $R_1 = \frac{5}{3}$, $R_2 = \frac{5}{2}$, $R_3 = \frac{11}{6}$, and Eq. (3.3) becomes

$$\tfrac{5}{3}\cos(\alpha) - \tfrac{5}{2}\cos(\phi) + \tfrac{11}{6} - \cos(\alpha - \phi) = 0 \tag{3.5}$$

The exact solution of Eq. (3.5) is tabulated in Table 3.1 and illustrated in Figure 3.2. Table 3.1 and Figure 3.2 correspond to the case where links 2, 3, and 4 are in the upper half-plane. This problem will be used throughout Chapter 3 to illustrate methods of solving for the roots of nonlinear equations. A mirror image solution is obtained for the case where links 2, 3, and 4 are in the lower half-plane. Another solution and its mirror image about the x axis are obtained if link 4 is in the upper half plane, link 2 is in the lower half-plane, and link 3 crosses the x axis, as illustrated by the small insert in Figure 3.1.

Table 3.1. Exact Solution of the Four-Bar Linkage Problem

α, deg	ϕ, deg	α, deg	ϕ, deg	α, deg	ϕ, deg
0.0	0.000000	70.0	54.887763	130.0	90.124080
10.0	8.069345	80.0	62.059980	140.0	92.823533
20.0	16.113229	90.0	68.888734	150.0	93.822497
30.0	24.104946	100.0	75.270873	160.0	92.734963
40.0	32.015180	110.0	81.069445	170.0	89.306031
50.0	39.810401	120.0	86.101495	180.0	83.620630
60.0	47.450827				

Many problems in engineering and science require the solution of a nonlinear equation. The problem can be stated as follows:

> Given the continuous nonlinear function $f(x)$,
> find the value $x = \alpha$ such that $f(\alpha) = 0$.

Figure 3.3 illustrates the problem graphically. The nonlinear equation, $f(x) = 0$, may be an algebraic equation (i.e., an equation involving $+$, $-$, $\times$, $/$, and radicals), a transcendental equation (i.e., an equation involving trigonometric, logarithmic, exponential, etc., functions), the solution of a differential equation, or any nonlinear relationship between an input x and an output $f(x)$.

There are two phases to finding the roots of a nonlinear equation: *bounding the root* and *refining the root* to the desired accuracy. Two general types of root-finding methods exist: *closed domain (bracketing) methods* which bracket the root in an ever-shrinking closed interval, and *open domain (nonbracketing) methods*. Several classical methods of both types are presented in this chapter. Polynomial root finding is considered as a special case. There are numerous pitfalls in finding the roots of nonlinear equations, which are discussed in some detail.

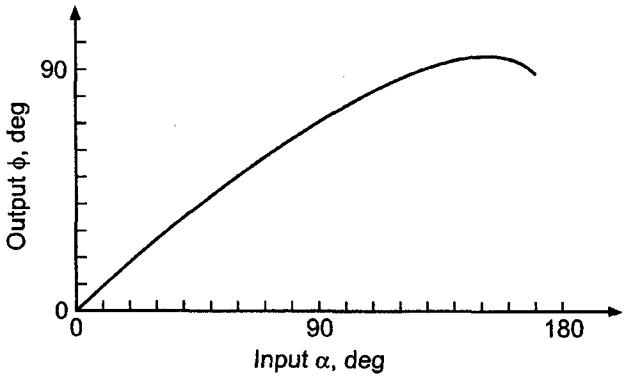


Figure 3.2 Exact solution of the four-bar linkage problem.

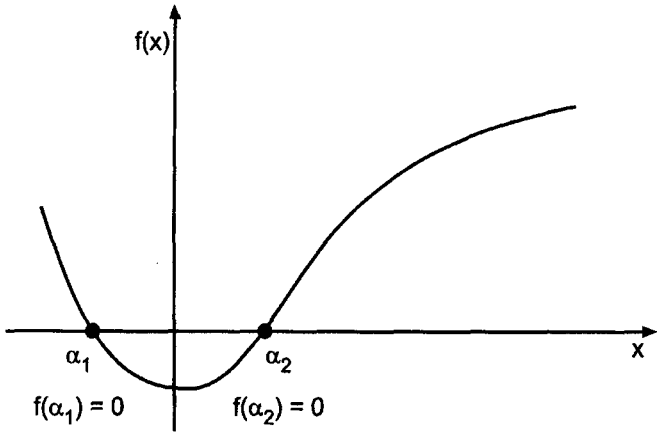


Figure 3.3 Solution of a nonlinear equation.

Figure 3.4 illustrates the organization of Chapter 3. After the introductory material presented in this section, some of the general features of root finding are discussed. The material then splits into a discussion of closed domain (bracketing) methods and open domain methods. Several special procedures applicable to polynomials are presented. After the presentation of the root finding methods, a section discussing some of the pitfalls of root finding and some other methods of root finding follows. A brief introduction to finding the roots of systems of nonlinear equations is presented. A section presenting several programs for solving nonlinear equations follows. The chapter closes with a Summary, which presents some philosophy to help you choose a specific method for a particular problem and lists the things you should be able to do after studying Chapter 3.

3.2 GENERAL FEATURES OF ROOT FINDING

Solving for the zeros of an equation, a process known as *root finding*, is one of the oldest problems in mathematics. Some general features of root finding are discussed in this section.

There are two distinct phases in finding the roots of a nonlinear equation: (1) *bounding the solution* and (2) *refining the solution*. These two phases are discussed in Sections 3.2.1 and 3.2.2, respectively. In general, nonlinear equations can behave in many different ways in the vicinity of a root. Typical behaviors are discussed in Section 3.2.3. Some general philosophy of root finding is discussed in Section 3.2.4.

3.2.1. Bounding the Solution

Bounding the solution involves finding a rough estimate of the solution that can be used as the initial approximation, or the starting point, in a systematic procedure that refines the solution to a specified tolerance in an efficient manner. If possible, the root should be bracketed between two points at which the value of the nonlinear function has opposite signs. Several possible bounding procedures are:

1. Graphing the function
2. Incremental search

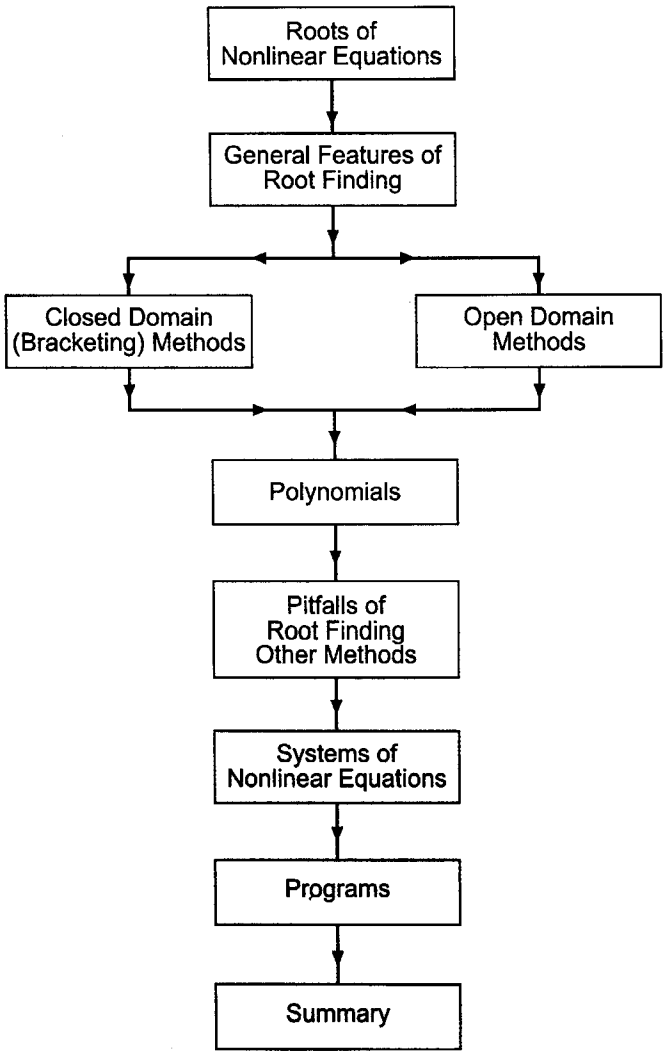


Figure 3.4 Organization of Chapter 3.

3. Past experience with the problem or a similar problem
4. Solution of a simplified approximate model
5. Previous solution in a sequence of solutions

Graphing the function involves plotting the nonlinear function over the range of interest. Many hand calculators have the capability to graph a function simply by defining the function and specifying the range of interest. Spreadsheets generally have graphing capability, as does software like Matlab and Mathcad. Very little effort is required. The resolution of the plots is generally not precise enough for an accurate result. However, the results are generally accurate enough to bound the solution. Plots of a nonlinear function display the general behavior of the nonlinear equation and permit the anticipation of problems.

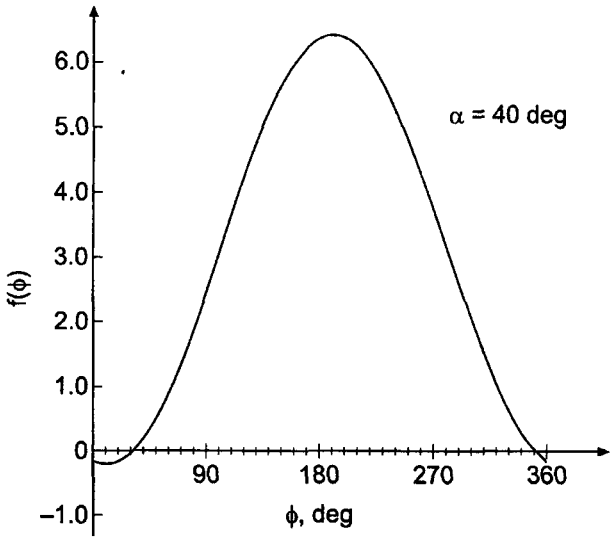


Figure 3.5 Graph of Eq. (3.5) for $\alpha = 40$ deg.

As an example of graphing a function to bound a root, consider the four-bar linkage problem presented in Section 3.1. Consider an input of $\alpha = 40$ deg. The graph of Eq. (3.5) with $\alpha = 40$ deg is presented in Figure 3.5. The graph shows that there are two roots of Eq. (3.5) when $\alpha = 40$ deg: one root between $\phi = 30$ deg and $\phi = 40$ deg, and one root between $\phi = 350$ (or -10) deg and $\phi = 360$ (or 0) deg.

An *incremental search* is conducted by starting at one end of the region of interest and evaluating the nonlinear function at small increments across the region. When the value of the function changes sign, it is assumed that a root lies in that interval. The two end points of the interval containing the root can be used as initial guesses for a refining method. If multiple roots are suspected, check for sign changes in the derivative of the function between the ends of the interval.

To illustrate an incremental search, let's evaluate Eq. (3.5) with $\alpha = 40$ deg for ϕ from 0 to 360 deg for $\Delta\phi = 10$ deg. The results are presented in Table 3.2. The same two roots identified by graphing the function are located.

Table 3.2. Incremental Search for Eq. (3.5) with $\alpha = 40$ deg

ϕ, deg	$f(\phi)$	ϕ, deg	$f(\phi)$	ϕ, deg	$f(\phi)$	ϕ, deg	$f(\phi)$
0.0	−0.155970	100.0	3.044194	190.0	6.438119	280.0	3.175954
10.0	−0.217971	110.0	3.623104	200.0	6.398988	290.0	2.597044
20.0	−0.178850	120.0	4.186426	210.0	6.259945	300.0	2.033722
30.0	−0.039797	130.0	4.717043	220.0	6.025185	310.0	1.503105
40.0	0.194963	140.0	5.198833	230.0	5.701851	320.0	1.021315
50.0	0.518297	150.0	5.617158	240.0	5.299767	330.0	0.602990
60.0	0.920381	160.0	5.959306	250.0	4.831150	340.0	0.260843
70.0	1.388998	170.0	6.214881	260.0	4.310239	350.0	0.005267
80.0	1.909909	180.0	6.376119	270.0	3.752862	360.0	−0.155970
90.0	2.467286						

Whatever procedure is used to bound the solution, the initial approximation must be sufficiently close to the exact solution to ensure (a) that the systematic refinement procedure converges, and (b) that the solution converges to the desired root of the nonlinear equation.

3.2.2. Refining the Solution

Refining the solution involves determining the solution to a specified tolerance by an efficient systematic procedure. Several methods for refining the solution are:

1. Trial and error
2. Closed domain (bracketing) methods
3. Open domain methods

Trial and error methods simply guess the root, $x = \alpha$, evaluate $f(\alpha)$, and compare to zero. If $f(\alpha)$ is close enough to zero, quit. If not, guess another α, and continue until $f(\alpha)$ is close enough to zero. This approach is totally unacceptable.

Closed domain (bracketing) methods are methods that start with two values of x which bracket the root, $x = \alpha$, and systematically reduce the interval while keeping the root trapped within the interval. Two such methods are presented in Section 3.3:

1. Interval halving (bisection)
2. False position (regula falsi)

Bracketing methods are robust in that they are guaranteed to obtain a solution since the root is trapped in the closed interval. They can be slow to converge.

Open domain methods do not restrict the root to remain trapped in a closed interval. Consequently, they are not as robust as bracketing methods and can actually diverge. However, they use information about the nonlinear function itself to refine the estimates of the root. Thus, they are considerably more efficient than bracketing methods. Four open domain methods are presented in Section 3.4:

1. The fixed-point iteration method
2. Newton's method
3. The secant method
4. Muller's method

3.2.3. Behavior of Nonlinear Equations

Nonlinear equations can behave in various ways in the vicinity of a root. Algebraic and transcendental equations may have distinct (i.e., simple) real roots, repeated (i.e., multiple) real roots, or complex roots. Polynomials may have real or complex roots. If the polynomial coefficients are all real, complex roots occur in conjugate pairs. If the polynomial coefficients are complex, single complex roots can occur.

Figure 3.6 illustrates several distinct types of behavior of nonlinear equations in the vicinity of a root. Figure 3.6a illustrates the case of a single real root, which is called a *simple root*. Figure 3.6b illustrates a case where no real roots exist. Complex roots may

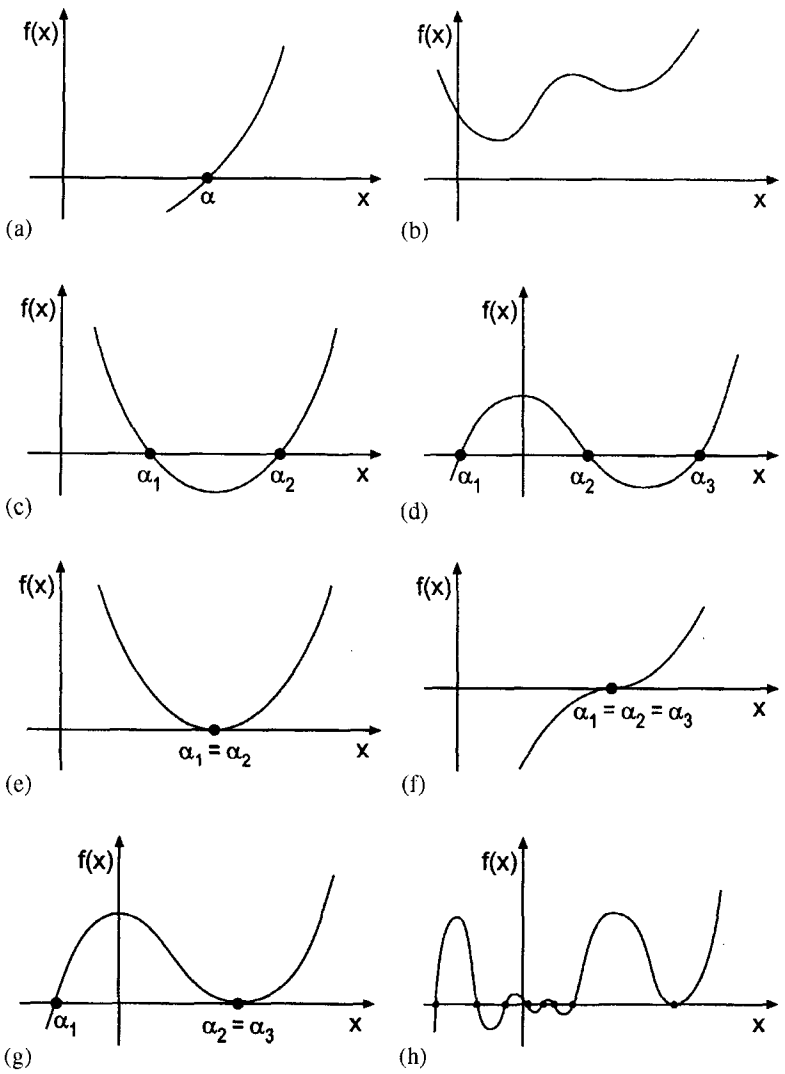


Figure 3.6 Solution behavior. (a) Simple root. (b) No real roots. (c) Two simple roots. (d) Three simple roots. (e) Two multiple roots. (f) Three multiple roots. (g) One simple and two multiple roots. (h) General case.

exist in such a case. Situations with two and three simple roots are illustrated in Figure 3.6c and d, respectively. Situations with two and three multiple roots are illustrated in Figure 3.6e and f, respectively. A situation with one simple root and two multiple roots is illustrated in Figure 3.6g. Lastly, Figure 3.6h illustrates the general case where any number of simple or multiple roots can exist.

Many problems in engineering and science involve a simple root, as illustrated in Figure 3.6a. Almost any root-finding method can find such a root if a reasonable initial approximation is furnished. In the other situations illustrated in Figure 3.6, extreme care may be required to find the desired roots.

3.2.4. Some General Philosophy of Root Finding

There are numerous methods for finding the roots of a nonlinear equation. The roots have specific values, and the method used to find the roots does not affect the values of the roots. However, the method can determine whether or not the roots can be found and the amount of work required to find them. Some general philosophy of root finding is presented below.

1. Bounding methods should bracket a root, if possible.
2. Good initial approximations are extremely important.
3. Closed domain methods are more robust than open domain methods because they keep the root bracketed in a closed interval.
4. Open domain methods, when they converge, generally converge faster than closed domain methods.
5. For smoothly varying functions, most algorithms will always converge if the initial approximation is close enough. The rate of convergence of most algorithms can be determined in advance.
6. Many, if not most, problems in engineering and science are well behaved and straightforward. In such cases, a straightforward open domain method, such as Newton's method presented in Section 3.4.2 or the secant method presented in Section 3.4.3, can be applied without worrying about special cases and peculiar behavior. If problems arise during the solution, then the peculiarities of the nonlinear equation and the choice of solution method can be reevaluated.
7. When a problem is to be solved only once or a few times, the efficiency of the method is not of major concern. However, when a problem is to be solved many times, efficiency of the method is of major concern.
8. Polynomials can be solved by any of the methods for solving nonlinear equations. However, the special techniques applicable to polynomials should be considered.
9. If a nonlinear equation has complex roots, that must be anticipated when choosing a method.
10. Analyst's time versus computer time must be considered when selecting a method.
11. Blanket generalizations about root-finding methods are generally not possible.

Root-finding algorithms should contain the following features:

1. An upper limit on the number of iterations.
2. If the method uses the derivative $f'(x)$, it should be monitored to ensure that it does not approach zero.
3. A convergence test for the change in the magnitude of the solution, $|x_{i+1} - x_i|$, or the magnitude of the nonlinear function, $|f(x_{i+1})|$, must be included.
4. When convergence is indicated, the final root estimate should be inserted into the nonlinear function $f(x)$ to guarantee that $f(x) = 0$ within the desired tolerance.

3.3 CLOSED DOMAIN (BRACKETING) METHODS

Two of the simplest methods for finding the roots of a nonlinear equation are:

1. Interval halving (bisection)
2. False position (regula falsi)

In these two methods, two estimates of the root which bracket the root must first be found by the bounding process. The root, $x = \alpha$, is bracketed by the two estimates. The objective is to locate the root to within a specified tolerance by a systematic procedure while keeping the root bracketed. Methods which keep the root bracketed during the refinement process are called *closed domain*, or *bracketing*, *methods*.

3.3.1. Interval Halving (Bisection)

One of the simplest methods for finding a root of a nonlinear equation is *interval halving* (also known as *bisection*). In this method, two estimates of the root, $x = a$ to the left of the root and $x = b$ to the right of the root, which bracket the root, must first be obtained, as illustrated in Figure 3.7, which illustrates the two possibilities with $f'(x) > 0$ and $f'(x) < 0$. The root, $x = \alpha$, obviously lies between a and b, that is, in the interval (a, b). The interval between a and b can be halved by averaging a and b. Thus, $c = (a + b)/2$. There are now two intervals: (a, c) and (c, b). The interval containing the root, $x = \alpha$, depends on the value of $f(c)$. If $f(a)f(c) < 0$, which is the case in Figure 3.7a, the root is in the interval (a, c). Thus, set $b = c$ and continue. If $f(a)f(c) > 0$, which is the case in Figure 3.7b, the root is in the interval (c, b). Thus, set $a = c$ and continue. If $f(a)f(c) = 0, c$ is the root. Terminate the iteration. The algorithm is as follows:

$$c = \frac{a+b}{2} \tag{3.6}$$

If $f(a)f(c) < 0$: $a = a$ and $b = c$ (3.7a)

If $f(a)f(c) > 0$: $a = c$ and $b = b$ (3.7b)

Interval halving is an iterative procedure. The solution is not obtained directly by a single calculation. Each application of Eqs. (3.6) and (3.7) is an iteration. The iterations are continued until the size of the interval decreases below a prespecified tolerance ε_1, that is, $|b_i - a_i| \leq \varepsilon_1$, or the value of $f(x)$ decreases below a prespecified tolerance ε_2, that is, $|f(c_i)| \leq \varepsilon_2$, or both.

If a nonlinear equation, such as $f(x) = 1/(x - d)$ which has a singularity at $x = d$, is bracketed between a and b, interval halving will locate the discontinuity, $x = d$. A check on $|f(x)|$ as $x \to d$ would indicate that a discontinuity, not a root, is being found.

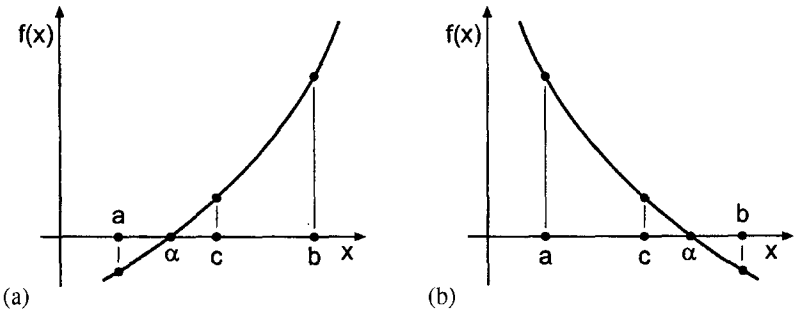


Figure 3.7 Interval halving (bisection).

Example 3.1. Interval halving (bisection).

Let's solve the four-bar linkage problem presented in Section 3.1 for an input of $\alpha = 40$ deg by interval halving. In calculations involving trigonometric functions, the angles must be expressed in radians. However, degrees (i.e., deg) are a more common unit of angular measure. Consequently, in all of the examples in this chapter, angles are expressed in degrees in the equations and in the tabular results, but the calculations are performed in radians. Recall Eq. (3.5) with $\alpha = 40.0$ deg:

$$f(\phi) = \tfrac{5}{3}\cos(40.0) - \tfrac{5}{2}\cos(\phi) + \tfrac{11}{6} - \cos(40.0 - \phi) = 0.0 \tag{3.8}$$

From the bounding procedure presented in Section 3.2, let $\phi_a = 30.0$ deg and $\phi_b = 40.0$ deg. From Eq. (3.8),

$$f(\phi_a) = f(30.0) = \tfrac{5}{3}\cos(40.0) - \tfrac{5}{2}\cos(30.0) + \tfrac{11}{6} - \cos(40.0 - 30.0)$$

$$= -0.03979719 \tag{3.9a}$$

$$f(\phi_b) = f(40.0) = \tfrac{5}{3}\cos(40.0) - \tfrac{5}{2}\cos(40.0) + \tfrac{11}{6} - \cos(40.0 - 40.0)$$

$$= 0.19496296 \tag{3.9b}$$

Thus, $\phi_a = 30.0$ deg and $\phi_b = 40.0$ deg bracket the solution. From Eq. (3.6),

$$\phi_c = \frac{\phi_a + \phi_b}{2} = \frac{30.0 + 40.0}{2} = 35.0 \text{ deg} \tag{3.10}$$

Substituting $\phi_c = 35.0$ deg into Eq. (3.8) yields

$$f(\phi_c) = f(35.0) = \tfrac{5}{3}\cos(40.0) - \tfrac{5}{2}\cos(35.0) + \tfrac{11}{6} - \cos(40.0 - 35.0) = 0.06599926 \tag{3.11}$$

Since $f(\phi_a)f(\phi_c) < 0$, $\phi_b = \phi_c$ for the next iteration and ϕ_a remains the same.

The solution is presented in Table 3.3. The convergence criterion is $|\phi_a - \phi_b| \leq 0.000001$ deg, which requires 24 iterations. Clearly, convergence is rather slow. The results presented in Table 3.3 were obtained on a 13-digit precision computer.

Table 3.3. Interval Halving (Bisection)

i	ϕ_a, deg	$f(\phi_a)$	ϕ_b, deg	$f(\phi_b)$	ϕ_c, deg	$f(\phi_c)$
1	30.0	−0.03979719	40.0	0.19496296	35.0	0.06599926
2	30.0	−0.03979719	35.0	0.06599926	32.50	0.01015060
3	30.0	−0.03979719	32.50	0.01015060	31.250	−0.01556712
4	31.250	−0.01556712	32.50	0.01015060	31.8750	−0.00289347
5	31.8750	−0.00289347	32.50	0.01015060	32.18750	0.00358236
6	31.8750	−0.00289347	32.18750	0.00358236	32.031250	0.00033288
7	31.8750	−0.00289347	32.031250	0.00033288	31.953125	−0.00128318
...	.....	.....	.....	.....	.....	.....
22	32.015176	−0.00000009	32.015181	0.00000000	32.015178	−0.00000004
23	32.015178	−0.00000004	32.015181	0.00000000	32.015179	−0.00000002
24	32.015179	−0.00000002	32.015181	0.00000000	32.015180	−0.00000001
	32.015180	−0.00000001				

The results in the table are rounded in the sixth digit after the decimal place. The final solution agrees with the exact solution presented in Table 3.1 to six digits after the decimal place.

The interval halving (bisection) method has several advantages:

1. The root is bracketed (i.e., trapped) within the bounds of the interval, so the method is guaranteed to converge.
2. The maximum error in the root is $|b_n - a_n|$.
3. The number of iterations n, and thus the number of function evaluations, required to reduce the initial interval, $(b_0 - a_0)$, to a specified interval, $(b_n - a_n)$, is given by

$$(b_n - a_n) = \frac{1}{2^n}(b_0 - a_0) \tag{3.12}$$

since each iteration reduces the interval size by a factor of 2. Thus, n is given by

$$n = \frac{1}{\log(2)} \log\left(\frac{b_0 - a_0}{b_n - a_n}\right) \tag{3.13}$$

The major disadvantage of the interval halving (bisection) method is that the solution converges slowly. That is, it can take a large number of iterations, and thus function evaluations, to reach the convergence criterion.

3.3.2. False Position (Regula Falsi)

The interval-halving (bisection) method brackets a root in the interval (a, b) and approximates the root as the midpoint of the interval. In the *false position (regula falsi) method*, the nonlinear function $f(x)$ is assumed to be a linear function $g(x)$ in the interval (a, b), and the root of the linear function $g(x), x = c$, is taken as the next approximation of the root of the nonlinear function $f(x), x = \alpha$. The process is illustrated graphically in Figure 3.8. This method is also called the *linear interpolation method*. The root of the linear function $g(x)$, that is, $x = c$, is not the root of the nonlinear function $f(x)$. It is a false position (in Latin, regula falsi), which gives the method its name. We now have two intervals, (a, c) and (c, b). As in the interval-halving (bisection) method, the interval containing the root of the nonlinear function $f(x)$ is retained, as described in Section 3.3.1, so the root remains bracketed.

The equation of the linear function $g(x)$ is

$$\frac{f(c) - f(b)}{c - b} = g'(x) \tag{3.14}$$

where $f(c) = 0$, and the slope of the linear function $g'(x)$ is given by

$$g'(x) = \frac{f(b) - f(a)}{b - a} \tag{3.15}$$

Solving Eq. (3.14) for the value of c which gives $f(c) = 0$ yields

$$\boxed{c = b - \frac{f(b)}{g'(x)}} \tag{3.16}$$

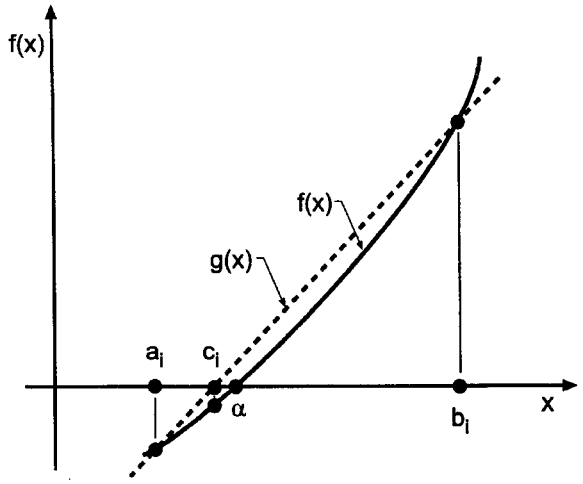


Figure 3.8 False position (regula falsi).

Note that $f(a)$ and a could have been used in Eqs. (3.14) and (3.16) instead of $f(b)$ and b. Equation (3.16) is applied repetitively until either one or both of the following two convergence criteria are satisfied:

$$|b - a| \leq \varepsilon_1 \qquad \text{and/or} \qquad |f(c)| \leq \varepsilon_2 \tag{3.17}$$

Example 3.2. False position (regula falsi).

As an example of the false position (regula falsi) method, let's solve the four-bar linkage problem presented in Section 3.1. Recall Eq. (3.5) with $\alpha = 40.0$ deg:

$$f(\phi) = \tfrac{5}{3}\cos(40.0) - \tfrac{5}{2}\cos(\phi) + \tfrac{11}{6} - \cos(40.0 - \phi) = 0.0 \tag{3.18}$$

Let $\phi_a = 30.0$ deg and $\phi_b = 40.0$ deg. From Eq. (3.18),

$$f(\phi_a) = f(30.0) = \tfrac{5}{3}\cos(40.0) - \tfrac{5}{2}\cos(30.0) + \tfrac{11}{6} - \cos(40.0 - 30.0)$$

$$= -0.03979719 \tag{3.19a}$$

$$f(\phi_b) = f(40.0) = \tfrac{5}{3}\cos(40.0) - \tfrac{5}{2}\cos(40.0) + \tfrac{11}{6} - \cos(40.0 - 40.0)$$

$$= 0.19496296 \tag{3.19b}$$

Thus, $\phi_a = 30.0$ deg and $\phi_b = 40.0$ deg bracket the solution. From Eq. (3.15),

$$g'(\phi_b) = \frac{0.19496296 - (-0.03979719)}{40.0 - 30.0} = 0.02347602 \tag{3.20}$$

Substituting these results into Eq. (3.16) yields

$$\phi_c = \phi_b - \frac{f(\phi_b)}{g'(\phi_b)} = 40.0 - \frac{0.19496296}{0.02347602} = 31.695228 \text{ deg} \tag{3.21}$$

Table 3.4. False Position (Regula Falsi)

i	ϕ_a, deg	$f(\phi_a)$	ϕ_b, deg	$f(\phi_b)$	ϕ_c, deg	$f(\phi_c)$
1	30.0	−0.03979719	40.0	0.19496296	31.695228	−0.00657688
2	31.695228	−0.00657688	40.0	0.19496296	31.966238	−0.00101233
3	31.966238	−0.00101233	40.0	0.19496296	32.007738	−0.00015410
4	32.007738	−0.00015410	40.0	0.19496296	32.014050	−0.00002342
5	32.014050	−0.00002342	40.0	0.19496296	32.015009	−0.00000356
6	32.015009	−0.00000356	40.0	0.19496296	32.015154	−0.00000054
7	32.015154	−0.00000054	40.0	0.19496296	32.015176	−0.00000008
8	32.015176	−0.00000008	40.0	0.19496296	32.015180	−0.00000001
9	32.015180	−0.00000001	40.0	0.19496296	32.015180	−0.00000000
	32.015180	0.00000000				

Substituting ϕ_c into Eq. (3.18) gives

$$f(\phi_c) = \tfrac{5}{3}\cos(40.0) - \tfrac{5}{2}\cos(31.695228) + \tfrac{11}{6} - \cos(40.0 - 31.695228) = -0.00657688 \tag{3.22}$$

Since $f(\phi_a)f(\phi_c) > 0.0$, ϕ_a is set equal to ϕ_c and ϕ_b remains the same. This choice of ϕ_a and ϕ_b keeps the root bracketed.

These results and the results of subsequent iterations are presented in Table 3.4. The convergence criterion, $|\phi_b - \phi_a| \leq 0.000001$ deg, is satisfied on the ninth iteration. Notice that ϕ_b does not change in this example. The root is approached monotonically from the left. This type of behavior is common for the false position method.

3.3.3. Summary

Two closed domain (bracketing) methods for finding the roots of a nonlinear equation are presented in this section: interval halving (bisection) and false position (regula falsi). Both of these methods are guaranteed to converge because they keep the root bracketed within a continually shrinking closed interval. The interval-halving method gives an exact upper bound on the error of the solution, the interval size. It converges rather slowly. The false position method generally converges more rapidly than the interval halving method, but it does not give a bound on the error of the solution.

Both methods are quite robust, but converge slowly. The open domain methods presented in Section 3.4 are generally preferred because they converge much more rapidly. However, they do not keep the root bracketed, and thus, they may diverge. In such cases, the more slowly converging bracketing methods may be preferred.

3.4 OPEN DOMAIN METHODS

The interval halving (bisection) method and the false position (regula falsi) method presented in Section 3.3 converge slowly. More efficient methods for finding the roots of a nonlinear equation are desirable. Four such methods are presented in this section:

1. Fixed-point iteration
2. Newton's method

3. The secant method
4. Muller's method

These methods are called *open domain methods* since they are not required to keep the root bracketed in a closed domain during the refinement process.

Fixed-point iteration is not a reliable method and is not recommended for use. It is included simply for completeness since it is a well-known method. Muller's method is similar to the secant method. However, it is slightly more complicated, so the secant method is generally preferred. Newton's method and the secant method are two of the most efficient methods for refining the roots of a nonlinear equation.

3.4.1. Fixed-Point Iteration

The procedure known as *fixed-point iteration* involves solving the problem $f(x) = 0$ by rearranging $f(x)$ into the form $x = g(x)$, then finding $x = \alpha$ such that $\alpha = g(\alpha)$, which is equivalent to $f(\alpha) = 0$. The value of x such that $x = g(x)$ is called a *fixed point* of the relationship $x = g(x)$. Fixed-point iteration essentially solves two functions simultaneously: $x(x)$ and $g(x)$. The point of intersection of these two functions is the solution to $x = g(x)$, and thus to $f(x) = 0$. This process is illustrated in Figure 3.9.

Since $g(x)$ is also a nonlinear function, the solution must be obtained iteratively. An initial approximation to the solution x_1 must be determined by a bounding method. This value is substituted into the function $g(x)$ to determine the next approximation. The algorithm is as follows:

$$\boxed{x_{i+1} = g(x_i)} \tag{3.23}$$

The procedure is repeated (iterated) until a convergence criterion is satisfied. For example,

$$|x_{i+1} - x_i| \leq \varepsilon_1 \qquad \text{and/or} \qquad |f(x_{i+1})| \leq \varepsilon_2 \tag{3.24}$$

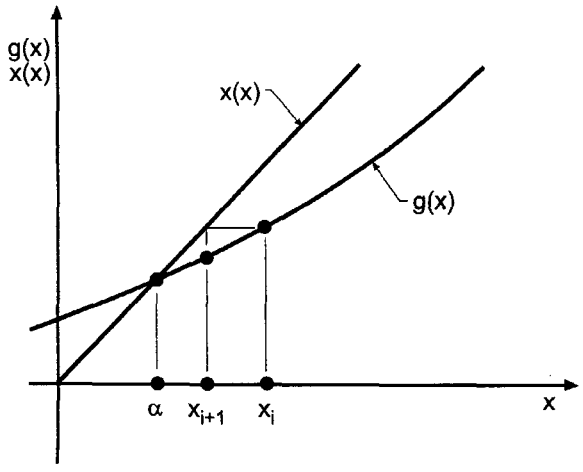


Figure 3.9 Fixed-point iteration.

Example 3.3. Fixed-point iteration.

Let's solve the four-bar linkage problem presented in Section 3.1 by fixed-point iteration. Recall Eq. (3.3):

$$f(\phi) = R_1 \cos(\alpha) - R_2 \cos(\phi) + R_3 - \cos(\alpha - \phi) = 0 \tag{3.25}$$

Equation (3.25) can be rearranged into the form $\phi = g(\phi)$ by separating the term $\cos(\alpha - \phi)$ and solving for ϕ. Thus,

$$\phi = \alpha - \cos^{-1}[R_1 \cos(\alpha) - R_2 \cos(\phi) + R_3] = \alpha - \cos^{-1}[u(\phi)] = g(\phi) \tag{3.26}$$

where

$$u(\phi) = R_1 \cos(\alpha) - R_2 \cos(\phi) + R_3 \tag{3.27}$$

The derivative of $g(\phi)$, that is, $g'(\phi)$, is of interest in the analysis of convergence presented at the end of this section. Recall

$$d(\cos^{-1}(u)) = -\frac{1}{\sqrt{1-u^2}}du \tag{3.28}$$

Differentiating Eq. (3.26) gives

$$g'(\phi) = -\frac{1}{\sqrt{1-u^2}}\frac{du}{d\phi} \tag{3.29}$$

which yields

$$g'(\phi) = \frac{R_2 \sin(\phi)}{\sqrt{1-u^2}} \tag{3.30}$$

For the four-bar linkage problem presented in Section 3.1, $R_1 = \frac{5}{3}, R_2 = \frac{5}{2}$, and $R_3 = \frac{11}{6}$. Let's find the output ϕ for an input $\alpha = 40$ deg. Equations (3.27), (3.26), and (3.30) become

$$u(\phi_i) = \tfrac{5}{3}\cos(40.0) - \tfrac{5}{2}\cos(\phi_i) + \tfrac{11}{6} \tag{3.31}$$

$$\phi_{i+1} = g(\phi_i) = 40.0 - \cos^{-1}[u(\phi_i)] \tag{3.32}$$

$$g'(\phi_i) = \frac{5}{2}\frac{\sin(\phi_i)}{\sqrt{1-[u(\phi_i)]^2}} \tag{3.33}$$

Let $\phi_1 = 30.0$ deg. Substituting $\phi_1 = 30.0$ deg into Eq. (3.25) gives $f(\phi_1) = -0.03979719$. Equations (3.31) to (3.33) give

$$u(30.0) = \tfrac{5}{3}\cos(40.0) - \tfrac{5}{2}\cos(30.0) + \tfrac{11}{6} = 0.945011 \tag{3.34}$$

$$\phi_2 = 40.0 - \cos^{-1}(0.945011) = 20.910798 \text{ deg} \tag{3.35}$$

$$g'(20.910798) = \frac{5}{2}\frac{\sin(20.910798)}{\sqrt{1-(0.945011)^2}} = 2.728369 \tag{3.36}$$

Substituting $\phi_2 = 20.910798$ deg into Eq. (3.25) gives $f(\phi_2) = -0.17027956$. The entire procedure is now repeated with $\phi_2 = 20.910798$ deg.

Table 3.5. Fixed-Point Iteration for the First Form of $g(\phi)$

i	ϕ_i, deg	$f(\phi_i)$	$u(\phi_i)$	$g'(\phi_i)$	ϕ_{i+1}, deg	$f(\phi_{i+1})$
1	30.000000	−0.03979719	0.94501056	2.548110	20.910798	−0.17027956
2	20.910798	−0.17027956	0.77473100	0.940796	0.780651	−0.16442489
3	0.780651	−0.16442489	0.61030612	0.028665	−12.388360	0.05797824
4	−12.388360	0.05797824	0.66828436	−0.480654	−8.065211	−0.03348286
5	−8.065211	−0.03348286	0.63480150	−0.302628	−10.594736	0.01789195
...	.....					
29	−9.747104	0.00000002	0.64616253	−0.369715	−9.747106	0.00000001
30	−9.747106	−0.00000001	0.64616254	−0.369715	−9.747106	−0.00000001
	−9.747105	−0.00000001				

These results and the results of 29 more iterations are summarized in Table 3.5. The solution has converged to the undesired root illustrated by the small insert inside the linkage illustrated in Figure 3.1. This configuration cannot be reached if the linkage is connected as illustrated in the large diagram. However, this configuration could be reached by reconnecting the linkage as illustrated in the small insert.

The results presented in Table 3.5 illustrate the major problem of open (nonbracketing) methods. Even though the desired root is bracketed by the two initial guesses, 30 deg and 40 deg, the method converged to a root outside that interval.

Let's rework the problem by rearranging Eq. (3.25) into another form of $\phi = g(\phi)$ by separating the term $R_2 \cos(\phi)$ and solving for ϕ. Thus,

$$\phi = \cos^{-1}\left\{\frac{1}{R_2}[R_1 \cos(\alpha) + R_3 - \cos(\alpha - \phi)]\right\} = \cos^{-1}[u(\phi)] = g(\phi) \tag{3.37}$$

where

$$u(\phi) = \frac{1}{R_2}[R_1 \cos(\alpha) + R_3 - \cos(\alpha - \phi)] \tag{3.38}$$

The derivative of $g(\phi)$ is

$$g'(\phi) = \frac{\sin(\alpha - \phi)}{R_2\sqrt{1 - u^2}} \tag{3.39}$$

For the four-bar linkage problem presented in Section 3.1, $R_1 = \frac{5}{3}$, $R_2 = \frac{5}{2}$, and $R_3 = \frac{11}{6}$. Let's find the output ϕ for the input $\alpha = 40$ deg. Equations (3.38), (3.37), and (3.39) become

$$u(\phi) = \tfrac{2}{5}[\tfrac{5}{3}\cos(40.0) + \tfrac{11}{6} - \cos(40.0 - \phi)] \tag{3.40}$$

$$\phi_{i+1} = g(\phi_i) = \cos^{-1}[u(\phi_i)] \tag{3.41}$$

$$g'(\phi_i) = \frac{2}{5}\frac{\sin(40.0 - \phi_i)}{\sqrt{1 - [u(\phi_i)]^2}} \tag{3.42}$$

Table 3.6. Fixed-Point Iteration for the Second Form of $g(\phi)$

i	ϕ_i, deg	$f(\phi_i)$	$u(\phi_i)$	$g'(\phi)$	ϕ_{i+1} deg	$f(\phi_{i+1})$
1	30.000000	−0.03979719	0.85010653	0.131899	31.776742	−0.00491050
2	31.776742	−0.00491050	0.84814233	0.107995	31.989810	−0.00052505
3	31.989810	−0.00052505	0.84793231	0.105148	32.012517	−0.00005515
4	32.012517	−0.00005515	0.84791025	0.104845	32.014901	−0.00000578
5	32.014901	−0.00000578	0.84790794	0.104814	32.015151	−0.00000061
6	32.015151	−0.00000061	0.84790769	0.104810	32.015177	−0.00000006
7	32.015177	−0.00000006	0.84790767	0.104810	32.015180	−0.00000001
8	32.015180	−0.00000001	0.84790767	0.104810	32.015180	−0.00000000
	32.015180					

Let $\phi_1 = 30.0$ deg. Substituting $\phi_1 = 30.0$ deg into Eq. (3.25) gives $f(\phi_1) = -0.03979719$. Equations (3.40) to (3.42) give

$$u(30.0) = \tfrac{2}{5}[\tfrac{5}{3}\cos(40.0) + \tfrac{11}{6} - \cos(40.0 - 30.0)] = 0.850107 \tag{3.43}$$

$$\phi_2 = g(\phi_1) = g(30.0) = \cos^{-1}[u(30.0)] = \cos^{-1}(0.850107) = 31.776742 \text{ deg} \tag{3.44}$$

$$g'(30.0) = \frac{2}{5}\frac{\sin(40.0 - 30.0)}{\sqrt{1 - (0.850107)^2}} = 0.131899 \tag{3.45}$$

Substituting $\phi_2 = 31.776742$ into Eq. (3.25) gives $f(\phi_2) = -0.00491050$. The entire procedure is now repeated with $\phi_2 = 31.776742$ deg.

These results and the results of the subsequent iterations are presented in Table 3.6. The convergence criterion is $|\phi_{i+1} - \phi_i| \leq 0.000001$ deg, which requires eight iterations. This is a considerable improvement over the interval halving (bisection) method presented in Example 3.1, which requires 24 iterations. It is comparable to the false position (regula falsi) method presented in Example 3.2, which requires nine iterations.

Convergence of the fixed-point iteration method, or any iterative method of the form $x_{i+1} = g(x_i)$, is analyzed as follows. Consider the iteration formula:

$$x_{i+1} = g(x_i) \tag{3.46}$$

Let $x = \alpha$ denote the solution and let $e = (x - \alpha)$ denote the error. Subtracting $\alpha = g(\alpha)$ from Eq. (3.46) gives

$$x_{i+1} - \alpha = e_{i+1} = g(x_i) - g(\alpha) \tag{3.47}$$

Expressing $g(\alpha)$ in a Taylor series about x_i gives:

$$g(\alpha) = g(x_i) + g'(\xi)\,(\alpha - x_i) + \cdots \tag{3.48}$$

where $x_i \leq \xi \leq \alpha$. Truncating Eq. (3.48) after the first-order term, solving for $[g(x_i) - g(\alpha)]$, and substituting the result into Eq. (3.47) yields

$$\boxed{e_{i+1} = g'(\xi)e_i} \tag{3.49}$$

Equation (3.49) can be used to determine whether or not a method is convergent, and if it is convergent, its rate of convergence. For any iterative method to converge,

$$\boxed{\left|\frac{e_{i+1}}{e_i}\right| = |g'(\xi)| < 1} \tag{3.50}$$

Consequently, the fixed-point iteration method is convergent only if $|g'(\xi)| < 1$. Convergence is linear since e_{i+1} is linearly dependent on e_i. If $|g'(\xi)| > 1$, the procedure diverges. If $|g'(\xi)| < 1$ but close to 1.0, convergence is quite slow. For the example presented in Table 3.5, $g'(\alpha) = 9.541086$, which explains why that form of $\phi = g(\phi)$ diverges. For the example presented in Table 3.6, $g'(\alpha) = 0.104810$, which means that the error decreases by a factor of approximately 10 at each iteration. Such rapid convergence does not occur when $|g'(\alpha)|$ is close to 1.0. For example, for $|g'(\alpha)| = 0.9$, approximately 22 times as many iterations would be required to reach the same convergence criterion.

If the nonlinear equation, $f(\phi) = 0$, is rearranged into the form $\phi = \phi + f(\phi) = g(\phi)$, the fixed-point iteration formula becomes

$$\phi_{i+1} = \phi_i + f(\phi_i) = g(\phi_i) \tag{3.51}$$

and $g'(\phi)$ is given by

$$g'(\phi) = 1 + R_2 \sin(\phi) - \sin(\alpha - \phi) \tag{3.52}$$

Substituting the final solution value, $\phi = 32.015180$ deg, into Eq. (3.52) gives $g'(\phi) = 2.186449$, which is larger than 1.0. The iteration method would not converge to the desired solution for this rearrangement of $f(\phi) = 0$ into $\phi = g(\phi)$. In fact, the solution converges to $\phi = -9.747105$ deg, for which $g'(\phi) = -0.186449$. This is also a solution to the four-bar linkage problem, but not the desired solution.

Methods which sometimes work and sometimes fail are undesirable. Consequently, the fixed-point iteration method for solving nonlinear equations is not recommended.

Methods for accelerating the convergence of iterative methods based on a knowledge of the convergence rate can be developed. *Aitkens* Δ^2 *acceleration method* applies to linearly converging methods, such as fixed-point iteration, in which $e_{i+1} = ke_i$. The method is based on starting with an initial approximation x_i for which

$$x_i = \alpha + e_i \tag{3.53a}$$

Two more iterations are made to give

$$x_{i+1} = \alpha + e_{i+1} = \alpha + ke_i \tag{3.53b}$$

$$x_{i+2} = \alpha + e_{i+2} = \alpha + ke_{i+1} = \alpha + k^2 e_i \tag{3.53c}$$

There are three unknowns in Eqs. (3.53a) to (3.53c): e_i, α, and k. These three equations can be solved for these three unknowns. The value of α obtained by the procedure is not the exact root, since higher-order terms have been neglected. However, it is an improved approximation of the root. The procedure is then repeated using α as the initial approximation. It can be shown that the successive approximations to the root, α, converge quadratically. When applied to the fixed-point iteration method for finding the roots of a nonlinear equation, this procedure is known as *Steffensen's method*. Steffensen's method is not developed in this book since Newton's method, which is presented in Section 3.4.2, is a more straightforward procedure for achieving quadratic convergence.

3.4.2. Newton's Method

Newton's method (sometimes called the *Newton-Rhapson method*) for solving nonlinear equations is one of the most well-known and powerful procedures in all of numerical analysis. It always converges if the initial approximation is sufficiently close to the root, and it converges quadratically. Its only disadvantage is that the derivative $f'(x)$ of the nonlinear function $f(x)$ must be evaluated.

Newton's method is illustrated graphically in Figure 3.10. The function $f(x)$ is nonlinear. Let's locally approximate $f(x)$ by the linear function $g(x)$, which is tangent to $f(x)$, and find the solution for $g(x) = 0$. Newton's method is sometimes called the tangent method. That solution is then taken as the next approximation to the solution of $f(x) = 0$. The procedure is applied iteratively to convergence. Thus,

$$f'(x_i) = \text{slope of } f(x) = \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} \tag{3.54}$$

Solving Eq. (3.54) for x_{i+1} with $f(x_{i+1}) = 0$ yields

$$\boxed{x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}} \tag{3.55}$$

Equation (3.55) is applied repetitively until either one or both of the following convergence criteria are satisfied:

$$|x_{i+1} - x_i| \leq \varepsilon_1 \qquad \text{and/or} \qquad |f(x_{i+1})| \leq \varepsilon_2 \tag{3.56}$$

Newton's method also can be obtained from the Taylor series. Thus,

$$f(x_{i+1}) = f(x_i) + f'(x_i)(x_{i+1} - x_i) + \cdots \tag{3.57}$$

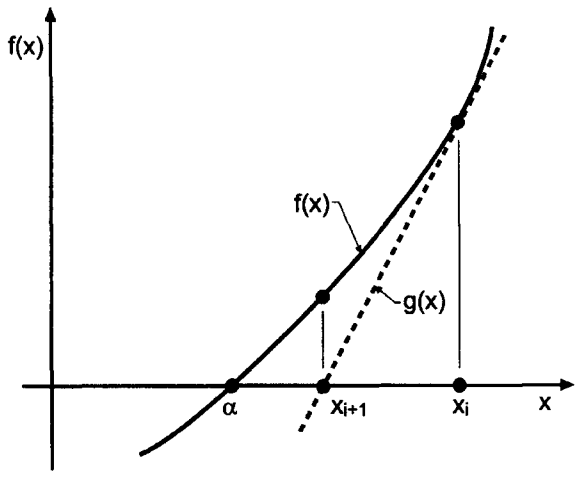


Figure 3.10 Newton's method.

Truncating Eq. (3.57) after the first derivative term, setting $f(x_{i+1}) = 0$, and solving for x_{i+1} yields

$$\boxed{x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}} \tag{3.58}$$

Equation (3.58) is the same as Eq. (3.55).

Example 3.4. Newton's method.

To illustrate Newton's method, let's solve the four-bar linkage problem presented in Section 3.1. Recall Eq. (3.3):

$$f(\phi) = R_1 \cos(\alpha) - R_2 \cos(\phi) + R_3 - \cos(\alpha - \phi) = 0 \tag{3.59}$$

The derivative of $f(\phi), f'(\phi)$ is

$$f'(\phi) = R_2 \sin(\phi) - \sin(\alpha - \phi) \tag{3.60}$$

Thus, Eq. (3.55) becomes

$$\boxed{\phi_{i+1} = \phi_i - \frac{f(\phi_i)}{f'(\phi_i)}} \tag{3.61}$$

For $R_1 = \frac{5}{3}, R_2 = \frac{5}{2}, R_3 = \frac{11}{6}$, and $\alpha = 40.0$ deg, Eqs. (3.59) and (3.60) yield

$$\boxed{f(\phi) = \tfrac{5}{3}\cos(40.0) - \tfrac{5}{2}\cos(\phi) + \tfrac{11}{6} - \cos(40.0 - \phi)} \tag{3.62}$$

$$\boxed{f'(\phi) = \tfrac{5}{2}\sin(\phi) - \sin(40.0 - \phi)} \tag{3.63}$$

For the first iteration let $\phi_1 = 30.0$ deg. Equations (3.62) and (3.63) give

$$f(\phi_1) = \tfrac{5}{3}\cos(40.0) - \tfrac{5}{2}\cos(30.0) + \tfrac{11}{6} - \cos(40.0 - 30.0) = -0.03979719 \tag{3.64}$$

$$f'(\phi_1) = \tfrac{5}{2}\sin(30.0) - \sin(40.0 - 30.0) = 1.07635182 \tag{3.65}$$

Substituting these results into Eq. (3.61) yields

$$\phi_2 = 30.0 - \frac{(-0.03979719)(180/\pi)}{1.07635182} = 32.118463 \text{ deg} \tag{3.66}$$

Substituting $\phi_2 = 32.118463$ deg into Eq. (3.62) gives $f(\phi_2) = 0.00214376$.

These results and the results of subsequent iterations are presented in Table 3.7. The convergence criterion, $|\phi_{i+1} - \phi_i| \leq 0.000001$ deg, is satisfied on the fourth iteration. This is a considerable improvement over the interval-halving method, the false position method, and the fixed-point iteration method.

Convergence of Newton's method is determined as follows. Recall Eq. (3.55):

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \tag{3.67}$$

Table 3.7. Newton's Method

i	ϕ_i, deg	$f(\phi_i)$	$f'(\phi_i)$	ϕ_{i+1}, deg	$f(\phi_{i+1})$
1	30.000000	−0.03979719	1.07635164	32.118463	0.00214376
2	32.118463	0.00214376	1.19205359	32.015423	0.00000503
3	32.015423	0.00000503	1.18646209	32.015180	0.00000000
4	32.015180	0.00000000	1.18644892	32.015180	0.00000000
	32.015180	0.00000000			

Equation (3.67) is of the form

$$x_{i+1} = g(x_i) \tag{3.68}$$

where $g(x)$ is given by

$$g(x) = x - \frac{f(x)}{f'(x)} \tag{3.69}$$

As shown by Eq. (3.50), for convergence of any iterative method in the form of Eq. (3.68),

$$|g'(\xi)| \leq 1 \tag{3.70}$$

where ξ lies between x_i and α. From Eq. (3.69),

$$g'(x) = 1 - \frac{f'(x)f'(x) - f(x)f''(x)}{[f'(x)]^2} = \frac{f(x)f''(x)}{[f'(x)]^2} \tag{3.71}$$

At the root, $x = \alpha$ and $f(\alpha) = 0$. Thus, $g'(\alpha) = 0$. Consequently, Eq.(3.70) is satisfied, and Newton's method is convergent.

The convergence rate of Newton's method is determined as follows. Subtract α from both sides of Eq. (3.67), and let $e = x - \alpha$ denote the error. Thus,

$$x_{i+1} - \alpha = e_{i+1} = x_i - \alpha - \frac{f(x_i)}{f'(x_i)} = e_i - \frac{f(x_i)}{f'(x_i)} \tag{3.72}$$

Expressing $f(x)$ in a Taylor series about x_i, truncating after the second-order term, and evaluating at $x = \alpha$ yields

$$f(\alpha) = f(x_i) + f'(x_i)(\alpha - x_i) + \tfrac{1}{2}f''(\xi)(\alpha - x_i)^2 = 0 \qquad x_i \leq \xi \leq \alpha \tag{3.73}$$

Letting $e_i = x_i - \alpha$ and solving Eq. (3.73) for $f(x_i)$ gives

$$f(x_i) = f'(x_i)e_i - \tfrac{1}{2}f''(\xi)e_i^2 \tag{3.74}$$

Substituting Eq. (3.74) into Eq. (3.72) gives

$$e_{i+1} = e_i - \frac{f'(x_i)e_i - \frac{1}{2}f''(\xi)e_i^2}{f'(x_i)} = \frac{1}{2}\frac{f''(\xi)}{f'(x_i)}e_i^2 \tag{3.75}$$

In the limit as $i \to \infty, x_i \to \alpha, f'(x_i) \to f'(\alpha), f''(\xi) \to f''(\alpha)$, and Eq. (3.75) becomes

$$\boxed{e_{i+1} = \frac{1}{2}\frac{f''(\alpha)}{f'(\alpha)}e_i^2} \tag{3.76}$$

Equation (3.76) shows that convergence is second-order, or quadratic. The number of significant figures essentially doubles each iteration.

As the solution is approached, $f(x_i) \rightarrow 0$, and Eq. (3.70) is satisfied. For a poor initial estimate, however, Eq. (3.70) may not be satisfied. In that case, the procedure may converge to an alternate solution, or the solution may jump around wildly for a while and then converge to the desired solution or an alternate solution. The procedure will not diverge disastrously like the fixed-point iteration method when $|g'(x)| > 1$. Newton's method has excellent local convergence properties. However, its global convergence properties can be very poor, due to the neglect of the higher-order terms in the Taylor series presented in Eq. (3.57).

Newton's method requires the value of the derivative $f'(x)$ in addition to the value of the function $f(x)$. When the function $f(x)$ is an algebraic function or a transcendental function, $f'(x)$ can be determined analytically. However, when the function $f(x)$ is a general nonlinear relationship between an input x and an output $f(x)$, $f'(x)$ cannot be determined analytically. In that case, $f'(x)$ can be estimated numerically by evaluating $f(x)$ at x_i and $x_i + \varepsilon$, and approximating $f'(x_i)$ as

$$f'(x_i) = \frac{f(x_i + \varepsilon) - f(x_i)}{\varepsilon} \tag{3.77}$$

This procedure doubles the number of function evaluations at each iteration. However, it eliminates the evaluation of $f'(x)$ at each iteration. If ε is small, round-off errors are introduced, and if ε is too large, the convergence rate is decreased. This process is called the *approximate Newton method.*

In some cases, the efficiency of Newton's method can be increased by using the same value of $f'(x)$ for several iterations. As long as the sign of $f'(x)$ does not change, the iterates x_i move toward the root, $x = \alpha$. However, the second-order convergence is lost, so the overall procedure converges more slowly. However, in problems where evaluation of $f'(x)$ is more costly than evaluation of $f(x)$, this procedure may be less work. This is especially true in the solution of systems of nonlinear equations, which is discussed in Section 3.7. This procedure is called the *lagged Newton's method.*

A higher-order version of Newton's method can be obtained by retaining the second derivative term in the Taylor series presented in Eq. (3.57). This procedure requires the evaluation of $f''(x)$ and the solution of a quadratic equation for $\Delta x = x_{i+1} - x_i$. This procedure is not used very often.

Newton's method can be used to determine complex roots of real equations or complex roots of complex equations simply by using complex arithmetic. Newton's method also can be used to find multiple roots of nonlinear equations. Both of these applications of Newton's method, complex roots and multiple roots, are discussed in Section 3.5, which is concerned with polynomials, which can have both complex roots and multiple roots. Newton's method is also an excellent method for *polishing* roots obtained by other methods which yield results polluted by round-off errors, such as roots of deflated functions (see Section 3.5.2.2).

Newton's method has several disadvantages. Some functions are difficult to differentiate analytically, and some functions cannot be differentiated analytically at all. In such cases, the approximate Newton method defined by Eq. (3.77) or the secant method presented in Section 3.4.3 is recommended. When multiple roots occur, convergence drops to first order. This problem is discussed in Section 3.5.2 for polynomials. The presence of a

local extremum (i.e., maximum or minimum) in $f(x)$ in the neighborhood of a root may cause oscillations in the solution. The presence of inflection points in $f(x)$ in the neighborhood of a root can cause problems. These last two situations are discussed in Section 3.6.1, which is concerned with pitfalls in root finding.

When Newton's method misbehaves, it may be necessary to bracket the solution within a closed interval and ensure that successive approximations remain within the interval. In extremely difficult cases, it may be necessary to make several iterations with the interval halving method to reduce the size of the interval before continuing with Newton's method.

3.4.3. The Secant Method

When the derivative function, $f'(x)$, is unavailable or prohibitively costly to evaluate, an alternative to Newton's method is required. The preferred alternative is the *secant method.*

The secant method is illustrated graphically in Figure 3.11. The nonlinear function $f(x)$ is approximated locally by the linear function $g(x)$, which is the secant to $f(x)$, and the root of $g(x)$ is taken as an improved approximation to the root of the nonlinear function $f(x)$. A *secant* to a curve is the straight line which passes through two points on the curve. The procedure is applied repetitively to convergence. Two initial approximations, x_0 and x_1, which are not required to bracket the root, are required to initiate the secant method. The slope of the secant passing through two points, x_{i-1} and x_i, is given by

$$g'(x_i) = \frac{f(x_i) - f(x_{i-1})}{x_i - x_{i-1}} \tag{3.78}$$

The equation of the secant line is given by

$$\frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i} = g'(x_i) \tag{3.79}$$

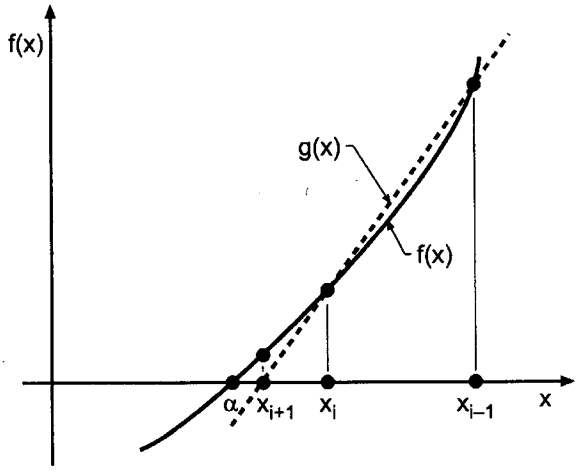


Figure 3.11 The secant method.

where $f(x_{i+1}) = 0$. Solving Eq. (3.79) for x_{i+1} yields

$$\boxed{x_{i+1} = x_i - \frac{f(x_i)}{g'(x_i)}} \tag{3.80}$$

Equation (3.80) is applied repetitively until either one or both of the following two convergence criteria are satisfied:

$$|x_{i+1} - x_i| \leq \varepsilon_1 \qquad \text{and/or} \qquad |f(x_{i+1})| \leq \varepsilon_2 \tag{3.81}$$

Example 3.5. The secant method.

Let's solve the four-bar linkage problem presented in Section 3.1 by the secant method. Recall Eq. (3.3):

$$f(\phi) = R_1 \cos(\alpha) - R_2 \cos(\phi) + R_3 - \cos(\alpha - \phi) = 0 \tag{3.82}$$

Thus, Eq. (3.80) becomes

$$\boxed{\phi_{i+1} = \phi_i - \frac{f(\phi_i)}{g'(\phi_i)}} \tag{3.83}$$

where $g'(\phi_i)$ is given by

$$g'(\phi_i) = \frac{f(\phi_i) - f(\phi_{i-1})}{\phi_i - \phi_{i-1}} \tag{3.84}$$

For $R_1 = \frac{5}{3}, R_2 = \frac{5}{2}, R_3 = \frac{11}{6}$, and $\alpha = 40.0$ deg, Eq. (3.82) yields

$$\boxed{f(\phi) = \tfrac{5}{3}\cos(40.0) - \tfrac{5}{2}\cos(\phi) + \tfrac{11}{6} - \cos(40.0 - \phi)} \tag{3.85}$$

For the first iteration, let $\phi_0 = 30.0$ deg and $\phi_1 = 40.0$ deg. Equation (3.85) gives

$$f(\phi_0) = \tfrac{5}{3}\cos(40.0) - \tfrac{5}{2}\cos(30.0) + \tfrac{11}{6} - \cos(40.0 - 30.0) = -0.03979719 \tag{3.86a}$$

$$f(\phi_1) = \tfrac{5}{3}\cos(40.0) - \tfrac{5}{2}\cos(40.0) + \tfrac{11}{6} - \cos(40.0 - 40.0) = 0.19496296 \tag{3.86b}$$

Substituting these results into Eq. (3.84) gives

$$g'(\phi_1) = \frac{(0.19496296) - (-0.03979719)}{40.0 - 30.0} = 0.02347602 \tag{3.87}$$

Substituting $g'(\phi_1)$ into Eq. (3.83) yields

$$\phi_2 = 40.0 - \frac{0.19496296}{0.02347602} = 31.695228 \text{ deg} \tag{3.88}$$

Substituting $\phi_2 = 31.695228$ deg into Eq. (3.85) gives $f(\phi_2) = -0.00657688$.

These results and the results of subsequent iterations are presented in Table 3.8. The convergence criterion, $|\phi_{i+1} - \phi_i| \leq 0.000001$ deg, is satisfied on the fifth iteration, which is one iteration more than Newton's method requires.

Table 3.8. The Secant Method

i	ϕ_i, deg	$f(\phi_i)$	$g'(\phi_i)$	ϕ_{i+1}, deg	$f(\phi_{i+1})$
0	30.000000	−0.03979719			
1	40.000000	0.19496296	0.02347602	31.695228	−0.00657688
2	31.695228	−0.00657688	0.02426795	31.966238	−0.00101233
3	31.966238	−0.00101233	0.02053257	32.015542	0.00000749
4	32.015542	0.00000749	0.02068443	32.015180	−0.00000001
5	32.015180	−0.00000001	0.02070761	32.015180	0.00000000
	32.015180	0.00000000			

The convergence rate of the secant method was analyzed by Jeeves (1958), who showed that

$$\boxed{e_{i+1} = \left[\frac{1}{2}\frac{f''(\alpha)}{f'(\alpha)}\right]^{0.62\ldots} e_i^{1.62\ldots}} \tag{3.89}$$

Convergence occurs at the rate 1.62..., which is considerably faster than the linear convergence rate of the fixed-point iteration method but somewhat slower than the quadratic convergence rate of Newton's method.

The question of which method is more efficient, Newton's method or the secant method, was also answered by Jeeves. He showed that if the effort required to evaluate $f'(x)$ is less than 43 percent of the effort required to evaluate $f(x)$, then Newton's method is more efficient. Otherwise, the secant method is more efficient.

The problems with Newton's method discussed at the end of Section 3.4.2 also apply to the secant method.

3.4.4. Muller's Method

Muller's method (1956) is based on locally approximating the nonlinear function $f(x)$ by a quadratic function $g(x)$, and the root of the quadratic function $g(x)$ is taken as an improved approximation to the root of the nonlinear function $f(x)$. The procedure is applied repetitively to convergence. Three initial approximations x_1, x_2, and x_3, which are not required to bracket the root, are required to start the algorithm. The only difference between Muller's method and the secant method is that $g(x)$ is a quadratic function in Muller's method and a linear function in the secant method.

Muller's method is illustrated graphically in Figure 3.12. The quadratic function $g(x)$ is specified as follows:

$$g(x) = a(x - x_i)^2 + b(x - x_i) + c \tag{3.90}$$

The coefficients (i.e., a, b, and c) are determined by requiring $g(x)$ to pass through the three known points (x_i, f_i), (x_{i-1}, f_{i-1}), and (x_{i-2}, f_{i-2}). Thus

$$g(x_i) = f_i = a(x_i - x_i)^2 + b(x_i - x_i) + c = c \tag{3.91a}$$

$$g(x_{i-1}) = f_{i-1} = a(x_{i-1} - x_i)^2 + b(x_{i-1} - x_i) + c \tag{3.91b}$$

$$g(x_{i-2}) = f_{i-2} = a(x_{i-2} - x_i)^2 + b(x_{i-2} - x_i) + c \tag{3.91c}$$

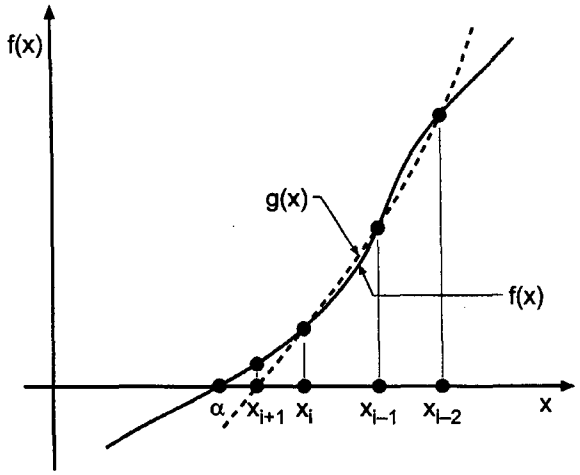


Figure 3.12 Muller's method.

Equation (3.91a) shows that $c = f_i$. Define the following parameters:

$$h_1 = (x_{i-1} - x_i) \quad \text{and} \quad h_2 = (x_{i-2} - x_i) \tag{3.92a}$$

$$\delta_1 = (f_{i-1} - f_i) \quad \text{and} \quad \delta_2 = (f_{i-2} - f_i) \tag{3.92b}$$

Then Eqs. (3.91b) and (3.91c) become

$$h_1^2 a + h_1 b = \delta_1 \tag{3.93a}$$

$$h_2^2 a + h_2 b = \delta_2 \tag{3.93b}$$

Solving Eq. (3.93) by Cramers rule yields

$$a = \frac{\delta_1 h_2 - \delta_2 h_1}{h_1 h_2 (h_1 - h_2)} \quad \text{and} \quad b = \frac{\delta_2 h_1^2 - \delta_1 h_2^2}{h_1 h_2 (h_1 - h_2)} \tag{3.94}$$

Now that the coefficients (i.e., a, b, and c) have been evaluated, Eq. (3.90) can be solved for the value of $(x_{i+1} - x_i)$ which gives $g(x_{i+1}) = 0$. Thus,

$$g(x_{i+1}) = a(x_{i+1} - x_i)^2 + b(x_{i+1} - x_i) + c = 0 \tag{3.95}$$

Solving Eq. (3.95) for $(x_{i+1} - x_i)$ by the rationalized quadratic formula, Eq. (3.112), gives

$$(x_{i+1} - x_i) = -\frac{2c}{b \pm \sqrt{b^2 - 4ac}} \tag{3.96}$$

Solving Eq. (3.96) for x_{i+1} yields

$$\boxed{x_{i+1} = x_i - \frac{2c}{b \pm \sqrt{b^2 - 4ac}}} \tag{3.97}$$

The sign, + or −, of the square root term in the denominator of Eq. (3.97) is chosen to be the same as the sign of b to keep x_{i+1} close to x_i. Equation (3.97) is applied repetitively until either one or both of the following two convergence criteria are satisfied:

$$|x_{i+1} - x_i| \leq \varepsilon_1 \quad \text{and/or} \quad |f(x_{i+1})| \leq \varepsilon_2 \tag{3.98}$$

Example 3.6. Muller's method.

Let's illustrate Muller's method by solving the four-bar linkage problem presented in Section 3.1. Recall Eq. (3.3):

$$f(\phi) = R_1 \cos(\alpha) - R_2 \cos(\phi) + R_3 - \cos(\alpha - \phi) = 0 \tag{3.99}$$

Thus, Eq. (3.97) becomes

$$\boxed{\phi_{i+1} = \phi_i - \frac{2c}{b \pm \sqrt{b^2 - 4ac}}} \tag{3.100}$$

where $c = f(\phi_i)$ and a and b are given by Eq. (3.94). For $R_1 = \frac{5}{3}, R_2 = \frac{5}{2}, R_3 = \frac{11}{6}$, and $\alpha = 40.0$ deg. Eq. (3.99) becomes

$$\boxed{f(\phi) = \tfrac{5}{3}\cos(40.0) - \tfrac{5}{2}\cos(\phi) + \tfrac{11}{6} - \cos(40.0 - \phi)} \tag{3.101}$$

For the first iteration, let $\phi_1 = 30.0$ deg, $\phi_2 = 30.5$ deg, and $\phi_3 = 31.0$ deg. Equation (3.101) gives $f(\phi_1) = f_1 = -0.03979719, f(\phi_2) = f_2 = -0.03028443$, and $f(\phi_3) = f_3 = -0.02053252$. Thus, $c = f_3 = -0.02053252$. Substituting these values of ϕ_1, ϕ_2, f_1, and f_2 into Eq. (3.92) gives

$$h_1 = (\phi_2 - \phi_3) = -0.50 \quad \text{and} \quad h_2 = (\phi_1 - \phi_3) = -1.00 \tag{3.102a}$$

$$\delta_1 = (f_2 - f_3) = -0.97519103 \quad \text{and} \quad \delta_2 = (f_1 - f_3) = -0.19264670 \tag{3.102b}$$

Substituting these results into Eq. (3.94) yields

$$a = \frac{\delta_1 h_2 - \delta_2 h_1}{h_1 h_2 (h_1 - h_2)} = -0.00047830092 \tag{3.103}$$

$$b = \frac{\delta_2 h_1^2 - \delta_1 h_2^2}{h_1 h_2 (h_1 - h_2)} = 0.019742971 \tag{3.104}$$

Substituting these results into Eq. (3.100) yields

$$\phi_{i+1} = 31.0 - \frac{2.0(-0.02053252)}{0.019742971 + \sqrt{(0.019742971)^2 - 4.0(0.00047830092)(-0.020532520)}} \tag{3.105}$$

which yields $\phi_{i+1} = 32.015031$ deg.

These results and the results of subsequent iterations are presented in Table 3.9. The convergence criterion, $|\phi_{i+1} - \phi_i| \leq 0.000001$ deg, is satisfied on the third iteration.

Table 3.9. Muller's Method

i	ϕ_{i-2}, deg	ϕ_{i-1}, deg	ϕ_i, deg	ϕ_{i+1}, deg	$f(\phi_{i+1})$
1	30.000000				−0.03979717
2	30.500000				−0.03028443
3	31.000000				−0.02053252
4	30.000000	30.500000	31.000000	32.015031	−0.00000309
5	30.500000	31.000000	32.015031	32.015180	0.00000000
6	31.000000	32.015031	32.015180	32.015180	0.00000000
	32.015180				0.00000000

The convergence rate of Muller's method is 1.84, which is faster than the 1.62 rate of the secant method and slower than the 2.0 rate of Newton's method. Generally speaking, the secant method is preferred because of its simplicity, even though its convergence rate, 1.62, is slightly smaller than the convergence rate of Muller's method, 1.84.

3.4.5. Summary

Four open methods for finding the roots of a nonlinear equation are presented in this section: the fixed-point iteration method, Newton's method, the secant method, and Muller's; method. The fixed-point iteration method has a linear convergence rate and converges slowly, or not at all if $|g'(\alpha)| > 1.0$. Consequently, this method is not recommended.

Newton's method, the secant method, and Muller's method all have a higher-order convergence rate (2.0 for Newton's method, 1.62 for the secant method, and 1.84 for Muller's method). All three methods converge rapidly in the vicinity of a root. When the derivative $f'(x)$ is difficult to determine or time consuming to evaluate, the secant method is more efficient. In extremely sensitive problems, all three methods may misbehave and require some bracketing technique. All three of the methods can find complex roots simply by using complex arithmetic. The secant method and Newton's method are highly recommended for finding the roots of nonlinear equations.

3.5 POLYNOMIALS

The methods of solving for the roots of nonlinear equations presented in Sections 3.3 and 3.4 apply to any form of nonlinear equation. One very common form of nonlinear equation is a polynomial. Polynomials arise as the characteristic equation in eigenproblems, in curve-fitting tabular data, as the characteristic equation of higher-order ordinary differential equations, as the characteristic equation in systems of first-order-ordinary differential equations, etc. In all these cases, the roots of the polynomials must be determined. Several special features of solving for the roots of polynomials are discussed in this section.

3.5.1. Introduction

The basic properties of polynomials are presented in Section 4.2. The general form of an nth-degree polynomial is

$$\boxed{P_n(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n} \tag{3.106}$$

where n denotes the degree of the polynomial and a_0 to a_n are constant coefficients. The coefficients a_0 to a_n may be real or complex. The evaluation of a polynomial with real coefficients and its derivatives is straightforward, using nested multiplication and synthetic division as discussed in Section 4.2. The evaluation of a polynomial with complex coefficients requires complex arithmetic.

The *fundamental theorem of algebra* states that an nth-degree polynomial has exactly n zeros, or *roots*. The roots may be real or complex. If the coefficients are all real, complex roots always occur in conjugate pairs. The roots may be single (i.e., simple) or repeated (i.e., multiple). The single roots of a linear polynomial can be determined directly. Thus,

$$P_1(x) = ax + b \tag{3.107}$$

has the single root, $x = \alpha$, given by

$$\alpha = -\frac{b}{a} \tag{3.108}$$

The two roots of a second-degree polynomial can also be determined directly. Thus,

$$P_2(x) = ax^2 + bx + c = 0 \tag{3.109}$$

has two roots, α_1 and α_2, given by the *quadratic formula*:

$$\boxed{\alpha_1, \alpha_2 = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}} \tag{3.110}$$

Equation (3.110) yields two distinct real roots when $b^2 > 4ac$, two repeated real roots when $b^2 = 4ac$, and a pair of complex conjugate roots when $b^2 < 4ac$. When $b^2 \gg 4ac$, Eq. (3.110) yields two distinct real roots which are the sum and difference of two nearly identical numbers. In that case, a more accurate result can be obtained by rationalizing Eq. (3.110). Thus,

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}\left(\frac{-b \mp \sqrt{b^2 - 4ac}}{-b \mp \sqrt{b^2 - 4ac}}\right) \tag{3.111}$$

which yields the *rationalized quadratic formula*:

$$\boxed{x = -\frac{2c}{b \pm \sqrt{b^2 - 4ac}}} \tag{3.112}$$

Exact formulas also exist for the roots of third-degree and fourth-degree polynomials, but they are quite complicated and rarely used. Iterative methods are used to find the roots of higher-degree polynomials.

Descartes' rule of signs, which applies to polynomials having real coefficients, states that the number of positive roots of $P_n(x)$ is equal to the number of sign changes in the nonzero coefficients of $P_n(x)$, or is smaller by an even integer. The number of negative roots is found in a similar manner by considering $P_n(-x)$. For example, the fourth-degree polynomial

$$P_4(x) = -4 + 2x + 3x^2 - 2x^3 + x^4 \tag{3.113}$$

has three sign changes in the coefficients of $P_n(x)$ and one sign change in the coefficients of $P_n(-x) = -4 - 2x + 3x^2 + 2x^3 + x^2$. Thus, the polynomial must have either three positive real roots and one negative real root, or one positive real root, one negative real root, and two complex conjugate roots. The actual roots are $-1, 1, 1 + \mathrm{I}1$, and $1 - \mathrm{I}1$, where $\mathrm{I} = \sqrt{-1.0}$.

The roots of high-degree polynomials can be quite sensitive to small changes in the values of the coefficients. In other words, high-degree polynomials can be *ill-conditioned*. Consider the factored fifth-degree polynomial:

$$P_5(x) = (x - 1)(x - 2)(x - 3)(x - 4)(x - 5) \tag{3.114}$$

which has five positive real roots, 1, 2, 3, 4, and 5. Expanding Eq. (3.114) yields the standard polynomial form:

$$P_5(x) = -120 + 274x - 225x^2 + 85x^3 - 15x^4 + x^5 \tag{3.115}$$

Descartes' rule of signs shows that there are either five positive real roots, or three positive real roots and two complex conjugate roots, or one positive real root and two pairs of complex conjugate roots. To illustrate the sensitivity of the roots to the values of the coefficients, let's change the coefficient of x^2, which is 225, to 226, which is a change of only 0.44 percent. The five roots are now $1.0514\ldots$, $1.6191\ldots$, $5.5075\ldots$, $3.4110\ldots + \mathrm{I}1.0793\ldots$, and $3.4110\ldots - \mathrm{I}1.0793\ldots$. Thus, a change of only 0.44 percent in one coefficient has made a major change in the roots, including the introduction of two complex conjugate roots. This simple example illustrates the difficulty associated with finding the roots of high-degree polynomials.

One procedure for finding the roots of high-degree polynomials is to find one root by any method, then deflate the polynomial one degree by factoring out the known root using synthetic division, as discussed in Section 4.2. The deflated $(n - 1)$st-degree polynomial is then solved for the next root. This procedure can be repeated until all the roots are determined. The last two roots should be determined by applying the quadratic formula to the $P_2(x)$ determined after all the deflations. This procedure reduces the work as the subsequent deflated polynomials are of lower and lower degree. It also avoids converging to an already converged root. The major limitation of this approach is that the coefficients of the deflated polynomials are not exact, so the roots of the deflated polynomials are not the precise roots of the original polynomial. Each deflation propagates the errors more and more, so the subsequent roots become less and less accurate. This problem is less serious if the roots are found in order from the smallest to the largest. In general, the roots of the deflated polynomials should be used as first approximations for those roots, which are then refined by solving the original polynomial using the roots of the deflated polynomials as the initial approximations for the refined roots. This process is known as *root polishing*.

The bracketing methods presented in Section 3.3, interval halving and false position, cannot be used to find repeated roots with an even multiplicity, since the nonlinear function $f(x)$ does not change sign at such roots. The first derivative $f'(x)$ does change sign at such roots, but using $f'(x)$ to keep the root bracketed increases the amount of work. Repeated roots with an odd multiplicity can be bracketed by monitoring the sign of $f(x)$, but even in this case the open methods presented in Section 3.4 are more efficient.

Three of the methods presented in Section 3.4 can be used to find the roots of polynomials: Newton's method, the secant method, and Muller's method. Newton's method for polynomials is presented in Section 3.5.2, where it is applied to find a simple root, a multiple root, and a pair of complex conjugate roots.

These three methods also can be used for finding the complex roots of polynomials, provided that complex arithmetic is used and reasonably good complex initial approximations are specified. Complex arithmetic is straightforward on digital computers. However, complex arithmetic is tedious when performed by hand calculation. Several methods exist for extracting complex roots of polynomials that have real coefficients which do not require complex arithmetic. Among these are Bairstow's method, the QD (quotient-difference) method [see Henrici (1964)], and Graeffe's method [see Hildebrand (1956)]. The QD method and Graeffe's method can find all the roots of a polynomial, whereas Bairstow's method extracts quadratic factors which can then be solved by the quadratic formula. Bairstow's method is presented in Section 3.5.3. These three methods use only real arithmetic.

When a polynomial has complex coefficients, Newton's method or the secant method using complex arithmetic and complex initial approximations are the methods of choice.

3.5.2. Newton's method

Newton's method for solving for the root, $x = \alpha$, of a nonlinear equation, $f(x) = 0$, is presented in Section 3.4. Recall Eq. (3.55):

$$\boxed{x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}} \tag{3.116}$$

Equation (3.116) will be called Newton's basic method in this section to differentiate it from two variations of Newton's method which are presented in this section for finding multiple roots. Newton's basic method can be used to find simple roots of polynomials, multiple roots of polynomials (where the rate of convergence drops to first order), complex conjugate roots of polynomials with real coefficients, and complex roots of polynomials with complex coefficients. The first three of these applications are illustrated in this section.

3.5.2.1. Newton's Method for Simple Roots.

Newton's basic method can be applied directly to find simple roots of polynomials. Generally speaking, $f(x)$ and $f'(x)$ should be evaluated by the nested multiplication algorithm presented in Section 4.2 for maximum efficiency. No special problems arise. Accurate initial approximations are desirable, and in some cases they are necessary to achieve convergence.

Example 3.7. Newton's method for simple roots.

Let's apply Newton's basic method to find the simple root of the following cubic polynomial in the neighborhood of $x = 1.5$:

$$f(x) = P_3(x) = x^3 - 3x^2 + 4x - 2 = 0 \tag{3.117}$$

Newton's basic method is given by Eq. (3.116). In this example, $f(x_i)$ and $f'(x_i)$ will be evaluated directly for the sake of clarity. The derivative of $f(x), f'(x)$, is given by the second-degree polynomial:

$$f'(x) = P_2(x) = 3x^2 - 6x + 4 \tag{3.118}$$

Table 3.10. Newton's Method for Simple Roots

i	x_i	$f(x_i)$	$f'(x_i)$	x_{i+1}	$f(x_{i+1})$
1	1.50	0.6250	1.750	1.142857	0.14577259
2	1.142857	0.14577259	1.06122449	1.005495	0.00549467
3	1.005495	0.00549476	1.00009057	1.000000	0.00000033
4	1.000000	0.00000033	1.00000000	1.000000	0.00000000
	1.000000	0.00000000			

Let $x_1 = 1.5$. Substituting this value into Eqs. (3.117) and (3.118) gives $f(1.5) = 0.6250$ and $f'(1.5) = 1.750$. Substituting these values into Eq. (3.116) yields

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} = 1.5 - \frac{0.6250}{1.750} = 1.142857 \tag{3.119}$$

These results and the results of subsequent iterations are presented in Table 3.10. Four iterations are required to satisfy the convergence tolerance, $|x_{i+1} - x_i| \leq 0.000001$.

Newton's method is an extremely rapid procedure for finding the roots of a polynomial if a reasonable initial approximation is available.

3.5.2.2. Polynomial Deflation

The remaining roots of Eq. (3.117) can be found in a similar manner by choosing different initial approximations. An alternate approach for finding the remaining roots is to *deflate* the original polynomial by factoring out the linear factor corresponding to the known root and solving for the roots of the deflated polynomial.

Example 3.8. Polynomial deflation

Let's illustrate polynomial deflation by factoring out the linear factor, $(x - 1.0)$, from Eq. (3.117). Thus, Eq. (3.117) becomes

$$P_3(x) = (x - 1.0)Q_2(x) \tag{3.120}$$

The coefficients of the deflated polynomial $Q_2(x)$ can be determined by applying the synthetic division algorithm presented in Eq. (4.26). Recall Eq. (3.117):

$$P_3(x) = x^3 - 3x^2 + 4x - 2 \tag{3.121}$$

Applying Eq. (4.26) gives

$$b_3 = a_3 = 1.0 \tag{3.122.3}$$
$$b_2 = a_2 + xb_3 = -3.0 + (1.0)(1.0) = -2.0 \tag{3.122.2}$$
$$b_1 = a_1 + xb_2 = 4.0 + (1.0)(-2.0) = 2.0 \tag{3.122.1}$$

Thus, $Q_2(x)$ is given by

$$x^2 - 2.0x + 2.0 = 0 \tag{3.123}$$

Equation (3.123) is the desired deflated polynomial. Since Eq. (3.123) is a second-degree polynomial, its roots can be determined by the quadratic formula. Thus,

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = \frac{-(-2.0) \pm \sqrt{(-2.0)^2 - 4.0(1.0)(2.0)}}{2(1.0)} \tag{3.124}$$

which yields the complex conjugate roots, $\alpha_{1,2} = 1 \pm \mathrm{I}1$.

3.5.2.3. Newton's Method for Multiple Roots

Newton's method, in various forms, can be used to calculate multiple real roots. Ralston and Rabinowitz (1978) show that a nonlinear function $f(x)$ approaches zero faster than its derivative $f'(x)$ approaches zero. Thus, Newton's basic method can be used, but care must be exercised to discontinue the iterations as $f'(x)$ approaches zero. However, the rate of convergence drops to first-order for a multiple root. Two variations of Newton's method restore the second-order convergence of the basic method:

1. Including the multiplicity m in Eq. (3.116)
2. Solving for the root of the modified function, $u(x) = f(x)/f'(x)$

These two variations are presented in the following discussion.

First consider the variation which includes the multiplicity m in Eq. (3.116):

$$\boxed{x_{i+1} = x_i - m\frac{f(x_i)}{f'(x_{i+1})}} \tag{3.125}$$

Equation (3.125) is in the general iteration form, $x_{i+1} = g(x_i)$. Differentiating $g(x)$ and evaluating the result at $x = \alpha$ yields $g'(\alpha) = 0$. Substituting this result into Eq. (3.50) shows that Eq. (3.125) is convergent. Further analysis yields

$$e_{i+1} = \frac{g''(\xi)}{2} e_i^2 \tag{3.126}$$

where ξ is between x_i and α, which shows that Eq. (3.125) converges quadratically.

Next consider the variation where Newton's basic method is applied to the function $u(x)$:

$$u(x) = \frac{f(x)}{f'(x)} \tag{3.127}$$

If $f(x)$ has m repeated roots, $f(x)$ can be expressed as

$$f(x) = (x - \alpha)^m h(x) \tag{3.128}$$

where the deflated function $h(x)$ does not have a root at $x = \alpha$, that is, $h(\alpha) \neq 0$. Substituting Eq. (3.128) into Eq. (3.127) gives

$$u(x) = \frac{(x - r)^m h(x)}{m(x - \alpha)^{m-1} h(x) + (x - \alpha)^m g'(x)} \tag{3.129}$$

which yields

$$u(x) = \frac{(x - \alpha)h(x)}{mh(x) + (x - \alpha)g'(x)} \tag{3.130}$$

Equation (3.130) shows that $u(x)$ has a single root at $x = \alpha$. Thus, Newton's basic method, with second-order convergence, can be applied to $u(x)$ to give

$$\boxed{x_{i+1} = x_i - \frac{u(x_i)}{u'(x_i)}} \tag{3.131}$$

Differentiating Eq. (3.127) gives

$$u'(x) = \frac{f'(x)f'(x) - f(x)f(x)}{[f'(x)]^2} \tag{3.132}$$

Substituting Eqs. (3.127) and (3.132) into Eq. (3.131) yields an alternate form of Eq. (3.131):

$$\boxed{x_{i+1} = x_i - \frac{f(x_i)f'(x_i)}{[f'(x_i)]^2 - f(x_i)f''(x_i)}} \tag{3.133}$$

The advantage of Eq. (3.133) over Newton's basic method for repeated roots is that Eq. (3.133) has second-order convergence. There are several disadvantages. There is an additional calculation for $f''(x_i)$. Equation (3.133) requires additional effort to evaluate. Round-off errors may be introduced due to the difference appearing in the denominator of Eq. (3.133). This method can also be used for simple roots, but it is less efficient than Newton's basic method in that case.

In summary, three methods are presented for evaluating repeated roots: Newton's basic method (which reduces to first-order convergence), Newton's basic method with the multiplicity m included, and Newton's basic method applied to the modified function, $u(x) = f(x)/f'(x)$. These three methods can be applied to any nonlinear equation. They are presented in this section devoted to polynomials simply because the problem of multiple roots generally occurs more frequently for polynomials than for other nonlinear functions. The three techniques presented here can also be applied with the secant method, although the evaluation of $f''(x)$ is more complicated in that case. These three methods are illustrated in Example 3.9.

Example 3.9. Newton's method for multiple roots.

Three versions of Newton's method for multiple roots are illustrated in this section:

1. Newton's basic method.
2. Newton's basic method including the multiplicity m.
3. Newton's basic method applied to the modified function, $u(x) = f(x)/f'(x)$.

These three methods are specified in Eqs. (3.116), (3.125), and (3.133), respectively, which are repeated below:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \tag{3.134}$$

$$x_{i+1} = x_i - m\frac{f(x_i)}{f'(x_i)} \tag{3.135}$$

where m is the multiplicity of the root, and

$$x_{i+1} = x_i - \frac{u(x_i)}{u'(x_i)} = x_i - \frac{f(x_i)f'(x_i)}{[f'(x_i)]^2 - f(x_i)f''(x_i)} \tag{3.136}$$

where $u(x) = f(x)/f'(x)$ has the same roots as $f(x)$. Let's solve for the repeated root, $r = 1, 1$, of the following third-degree polynomial:

$$f(x) = P_3(x) = (x+1)(x-1)(x-1) = 0 \tag{3.137}$$

$$f(x) = x^3 - x^2 - x + 1 = 0 \tag{3.138}$$

From Eq. (3.138),

$$f'(x) = 3x^2 - 2x - 1 \tag{3.139}$$

$$f''(x) = 6x - 2 \tag{3.140}$$

Let the initial approximation be $x_1 = 1.50$. From Eqs. (3.138) to (3.140), $f(1.50) = 0.6250, f'(1.50) = 2.750$, and $f''(1.5) = 7.0$. Substituting these values into Eqs. (3.134) to (3.136) gives

$$x_2 = 1.5 - \frac{0.6250}{2.750} = 2.272727 \tag{3.141}$$

$$x_2 = 1.5 - 2.0\frac{0.6250}{2.750} = 1.045455 \tag{3.142}$$

$$x_2 = 1.5 - \frac{(0.6250)(2.750)}{(2.750)^2 - (0.6250)(7.0)} = 0.960784 \tag{3.143}$$

These results and the results of subsequent iterations required to achieve the convergence tolerance, $|\Delta x_{i+1}| \leq 0.000001$, are summarized in Table 3.11.

Newton's basic method required 20 iterations, while the two other methods required only four iterations each. The advantage of these two methods over the basic method for repeated roots is obvious.

3.5.2.4. Newton's Method for Complex Roots

Newton's method, the secant method, and Muller's method can be used to calculate complex roots simply by using complex arithmetic and choosing complex initial approximations.

Bracketing methods, such as interval halving and false position, cannot be used to find complex roots, since the sign of $f(x)$ generally does not change sign at a complex root. Newton's method is applied in this section to find the complex conjugate roots of a polynomial with real coefficients.

Example 3.10. Newton's method for complex roots.

The basic Newton method can find complex roots by using complex arithmetic and choosing a complex initial approximation. Consider the third-degree polynomial:

$$f(x) = P_3(x) = (x-1)(x-1-\mathrm{I}1)(x-1+\mathrm{I}1) \tag{3.144}$$

$$f(x) = x^3 - 3x^2 + 4x - 2 = 0 \tag{3.145}$$

Table 3.11. Newton's Method for Multiple Real Roots

Newton's basic method, Eq. (3.134)

i	x_i	$f(x_i)$	x_{i+1}	$f(x_{i+1})$
1	1.50	0.6250	1.272727	0.16904583
2	1.272727	0.16904583	1.144082	0.04451055
3	1.144082	0.04451055	1.074383	0.01147723
...	.....			
19	1.000002	0.00000000	1.000001	0.00000000
20	1.000000	0.00000000	1.000001	0.00000000
	1.000001	0.00000000		

Newton's multiplicity method, Eq. (3.135), with $m = 2$

i	x_i	$f(x_i)$	x_{i+1}	$f(x_{i+1})$
1	1.50	0.6250	1.045455	0.00422615
2	1.045455	0.00422615	1.00500	0.00000050
3	1.005000	0.00000050	1.000000	0.00000000
4	1.000000	0.00000000	1.000000	0.00000000
	1.000000	0.00000000		

Newton's modified method, Eq. (3.136)

i	x_i	$f(x_i)$	x_{i+1}	$f(x_{i+1})$
1	1.50	0.6250	0.960784	0.00301543
2	0.960784	0.00301543	0.999600	0.00000032
3	0.999600	0.00000032	1.000000	0.00000000
4	1.000000	0.00000000	1.000000	0.00000000
	1.000000	0.00000000		

Table 3.12. Newton's Method for Complex Roots

i	x_i	$f(x_i)$	$f'(x_i)$
1	0.500000 + I0.500000	1.75000000 − I0.25000000	−1.00000000 + I0.50000000
2	2.000000 + I1.000000	1.00000000 + I7.00000000	5.00000000 + I10.00000000
3	1.400000 + I0.800000	0.73600000 + I1.95200000	1.16000000 + I5.12000000
4	1.006386 + I0.854572	0.53189072 + I0.25241794	−1.16521249 − I3.45103149
5	0.987442 + I1.015093	−0.03425358 − I0.08138309	−2.14100172 − I3.98388821
6	0.999707 + I0.999904	0.00097047 − I0.00097901	−2.00059447 − I3.99785801
7	1.000000 + I1.000000	−0.00000002 + I0.00000034	−1.99999953 − I4.00000030
	1.000000 + I1.000000	0.00000000 + I0.00000000	

The roots of Eq. (3.144) are $r = 1, 1 + \mathrm{I}1$, and $1 - \mathrm{I}1$. Let's find the complex root $r = 1 + \mathrm{I}1$ starting with $x_1 = 0.5 + \mathrm{I}0.5$. The complex arithmetic was performed by a FORTRAN program for Newton's method. The results are presented in Table 3.12.

3.5.3. Bairstow's Method

A special problem associated with polynomials $P_n(x)$ is the possibility of complex roots. Newton's method, the secant method, and Muller's method all can find complex roots if complex arithmetic is used and complex initial approximations are specified. Fortunately, complex arithmetic is available in several programming languages, such as FORTRAN. However, hand calculation using complex arithmetic is tedious and time consuming. When polynomials with real coefficients have complex roots, they occur in conjugate pairs, which corresponds to a quadratic factor of the polynomial $P_n(x)$. Bairstow's method extracts quadratic factors from a polynomial using only real arithmetic. The quadratic formula can then be used to determine the corresponding pair of real roots or complex conjugate roots.

Consider the general nth-degree polynomial, $P_n(x)$:

$$P_n(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_0 \tag{3.146}$$

Let's factor out a quadratic factor from $P_n(x)$. Thus,

$$P_n(x) = (x^2 - rx - s)Q_{n-2}(x) + \text{remainder} \tag{3.147}$$

This form of the quadratic factor (i.e., $x^2 - rx - s$) is generally specified. Performing the division of $P_n(x)$ by the quadratic factor yields

$$P_n(x) = (x^2 - rx - s)(b_n x^{n-2} + b_{n-1}x^{n-3} + \cdots + b_3 x + b_2) + \text{remainder} \tag{3.148}$$

where the remainder is given by

$$\text{Remainder} = b_1(x - r) + b_0 \tag{3.149}$$

When the remainder is zero, $(x^2 - rx - s)$ is an exact factor of $P_n(x)$. The roots of the quadratic factor, real or complex, can be determined by the quadratic formula.

For the remainder to be zero, both b_1 and b_0 must be zero. Both b_1 and b_0 depend on both r and s. thus,

$$b_1 = b_1(r, s) \qquad \text{and} \qquad b_0 = b_0(r, s) \tag{3.150}$$

Thus, we have a two-variable root-finding problem. This problem can be solved by Newton's method for a system of nonlinear equations, which is presented in Section 3.7.

Expressing Eq. (3.150) in the form of two two-variable Taylor series in terms of $\Delta r = (r^* - r)$ as $\Delta s = (s^* - s)$, where r^* and s^* are the values of r and s which yield $b_1 = b_0 = 0$, gives

$$b_1(r^*, s^*) = b_1 + \frac{\partial b_1}{\partial r}\Delta r + \frac{\partial b_1}{\partial s}\Delta s + \cdots = 0 \tag{3.151a}$$

$$b_0(r^*, s^*) = b_0 \frac{\partial b_0}{\partial r}\Delta r + \frac{\partial b_0}{\partial s}\Delta s + \cdots = 0 \tag{3.151b}$$

where b_1, b_0, and the four partial derivatives are evaluated at point (r, s). Truncating Eq. (3.151) after the first-order terms and solving for Δr and Δs gives

$$\frac{\partial b_1}{\partial r}\Delta r + \frac{\partial b_1}{\partial s}\Delta s = -b_1 \tag{3.152}$$

$$\frac{\partial b_0}{\partial r}\Delta r + \frac{\partial b_0}{\partial s}\Delta s = -b_0 \tag{3.153}$$

Equations (3.152) and (3.153) can be solved for Δr and Δs by Cramer's rule or Gauss elimination. All that remains is to relate b_1, b_0, and the four partial derivatives to the coefficients of the polynomial $P_n(x)$, that is, a_i $(i = 0, 1, 2, \ldots, n)$.

Expanding the right-hand side of Eq. (3.148), including the remainder term, and comparing the two sides term by term, yields

$$b_n = a_n \tag{3.154.n}$$

$$b_{n-1} = a_{n-1} + rb_n \tag{3.154n-1}$$

$$b_{n-2} = a_{n-2} + rb_{n-1} + sb_n \tag{3.154n-2}$$

$$\cdots\cdots\cdots\cdots\cdots$$

$$b_1 = a_1 + rb_2 + sb_3 \tag{3.154.1}$$

$$b_0 = a_0 + rb_1 + sb_2 \tag{3.154.0}$$

Equation (3.154) is simply the synthetic division algorithm presented in Section 4.2 applied for a quadratic factor.

The four partial derivatives required in Eqs. (3.152) and (3.153) can be obtained by differentiating the coefficients b_i $(i = n, n-1, \ldots, b_1, b_0)$, with respect to r and s, respectively. Since each coefficient b_i contains b_{i+1} and b_{i+2}, we must start with the partial derivatives of b_n and work our way down to the partial derivatives of b_1 and b_0. Bairstow showed that the results are identical to dividing $Q_{n-2}(x)$ by the quadratic factor, $(x^2 - rx - s)$, using the synthetic division algorithm. The details are presented by Gerald and Wheatley (1999). The results are presented below.

$$c_n = b_n \tag{3.155.n}$$

$$c_{n-1} = b_{n-1} + rc_n \tag{3.155.n-1}$$

$$c_{n-2} = b_{n-2} + rc_{n-1} + sc_n \tag{3.155n-2}$$

$$\cdots\cdots\cdots\cdots\cdots$$

$$c_2 = b_2 + rc_3 + sc_4 \tag{3.155.2}$$

$$c_1 = b_1 + rc_2 + sc_3 \tag{3.155.1}$$

The required partial derivatives are given by

$$\frac{\partial b_1}{\partial r} = c_2 \quad \text{and} \quad \frac{\partial b_1}{\partial s} = c_3 \tag{3.156a}$$

$$\frac{\partial b_0}{\partial r} = c_1 \quad \text{and} \quad \frac{\partial b_0}{\partial s} c_2 \tag{3.156b}$$

Thus, Eqs. (3.152) and (3.153) become

$$\boxed{c_2\,\Delta r + c_3\,\Delta s = -b_1} \tag{3.157a}$$

$$\boxed{c_1\,\Delta r + c_2\,\Delta s = -b_0} \tag{3.157b}$$

where $\Delta r = (r^* - r)$ and $\Delta s = (s^* - s)$. Thus,

$$\boxed{r_{i+1} = r_i + \Delta r_i} \tag{3.158a}$$

$$\boxed{s_{i+1} = s_i + \Delta s_i} \tag{3.158b}$$

Equations (3.157) and (3.158) are applied repetitively until either one or both of the following convergence criteria are satisfied:

$$|\Delta r_i| \leq \varepsilon_1 \quad \text{and} \quad |\Delta s_i| \leq \varepsilon_1 \tag{3.159a}$$

$$|(b_1)_{i+1} - (b_1)_i| \leq \varepsilon_2 \quad \text{and} \quad |(b_0)_{i+1} - (b_0)_i| \leq \varepsilon_2 \tag{3.159b}$$

Example 3.11. Bairstow's method for quadratic factors.

Let's illustrate Bairstow's method by solving for a quadratic factor of Eq. (3.145):

$$f(x) = x^3 - 3x^2 + 4x - 2 = 0 \tag{3.160}$$

The roots of Eq. (3.160) are $r = 1, 1 + \mathrm{I}1$, and $1 - \mathrm{I}1$.

To initiate the calculations, let $r_1 = 1.5$ and $s_1 = -2.5$. Substituting these values into Eq. (3.154) gives

$$b_3 = a_3 = 1.0 \tag{3.161.3}$$

$$b_2 = a_2 + rb_3 = (-3.0) + (1.5)(1) = -1.50 \tag{3.161.2}$$

$$b_1 = a_1 + rb_2 + sb_3 = 4.0 + (1.5)(-1.5) + (-2.5)(1.0) = -0.750 \tag{3.161.1}$$

$$b_0 = a_0 + rb_1 + sb_2 = -2.0 + (1.5)(-0.75) + (-2.5)(-1.5) = 0.6250 \tag{3.161.0}$$

Substituting these results into Eq. (3.155) gives

$$c_3 = b_3 = 1.0 \tag{3.162.8}$$

$$c_2 = b_2 + rc_3 = -(1.5) + (1.5)(1.0) = 0.0 \tag{3.162.2}$$

$$c_1 = b_1 + rc_2 + sc_3 = (-0.750) + (1.5)(0.0) + (-2.5)(1.0) = -3.250 \tag{3.162.1}$$

Substituting the values of b_1, b_0, c_3, c_2, and c_1 into Eq. (3.157) gives

$$(0.0)\Delta r + (1.0)\Delta s = -(-0.75) = 0.750 \tag{3.163a}$$

$$-3.250\Delta r + (0.0)\Delta s = -0.6250 \tag{3.163b}$$

Table 3.13. Bairstow's Method for Quadratic Factors

i	r	s	Δr	Δs
1	1.50	−2.50	0.192308	0.750000
2	1.692308	−1.750	0.278352	−0.144041
3	1.970660	−1.894041	0.034644	−0.110091
4	2.005304	−2.004132	−0.005317	0.004173
5	1.999988	−1.999959	0.000012	−0.000041
6	2.000000	−2.000000	0.000000	0.000000

Solving Eq. (3.163) gives

$$\Delta r = 0.192308 \quad \text{and} \quad \Delta s = 0.750 \tag{3.164}$$

Substituting Δr and Δs into Eq. (3.158) gives

$$r_2 = r_1 + \Delta r = 1.50 + 0.192308 = 1.692308 \tag{3.165a}$$

$$s_2 = s_1 + \Delta s = -2.50 + 0.750 = -1.750 \tag{3.165b}$$

These results and the results of subsequent iterations are presented in Table 3.13. The convergence criteria, $|\Delta r_i| \leq 0.000001$ and $|\Delta s_i| \leq 0.000001$, are satisfied on the sixth iteration, where $r = 2.0$ and $s = -2.0$. Thus, the desired quadratic factor is

$$x^2 - rx - s = x^2 - 2.0x + 2.0 = 0 \tag{3.166}$$

Solving for the roots of Eq. (3.166) by the quadratic formula yields the pair of complex conjugate roots:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} = \frac{-(-2.0) \pm \sqrt{(-2.0)^2 - 4.0(1.0)(2.0)}}{2(1.0)} = 1 + \mathrm{I}1, 1 - \mathrm{I}1 \tag{3.167}$$

3.5.4. Summary

Polynomials are a special case of nonlinear equation. Any of the methods presented in Sections 3.3 and 3.4 can be used to find the roots of polynomials. Newton's method is especially well suited for this purpose. It can find simple roots and multiple roots directly. However, it drops to first-order for multiple roots. Two variations of Newton's method for multiple roots restore the second-order convergence. Newton's method, like the secant method and Muller's method, can be used to find complex roots simply by using complex arithmetic with complex initial approximations. Bairstow's method can find quadratic factors using real arithmetic, and the quadratic formula can be used to find the two roots of the quadratic factor. Good initial guesses are desirable and may be necessary to find the roots of high-degree polynomials.

3.6 PITFALLS OF ROOT FINDING METHODS AND OTHER METHODS OF ROOT FINDING

The root-finding methods presented in Sections 3.3 to 3.5 generally perform as described. However, there are several pitfalls, or problems, which can arise in their application. Most of these pitfalls are discussed in Sections 3.3 to 3.5. They are summarized and discussed in Section 3.6.1.

The collection of root-finding methods presented in Sections 3.3 to 3.5 includes the more popular methods and the most well-known methods. Several less well-known root-finding methods are listed in Section 3.6.2.

3.6.1. Pitfalls of Root Finding Methods

Numerous pitfalls, or problems, associated with root finding are noted in Sections 3.3 to 3.5. These include:

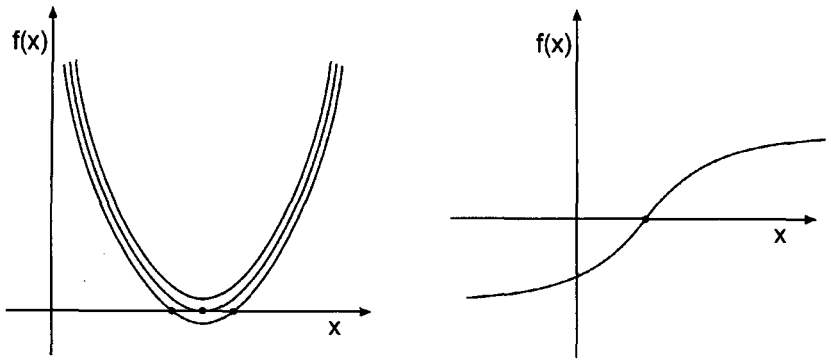


Figure 3.13 Pitfalls of root finding. (a) Closely spaced roots. (b) Inflection point.

1. Lack of a good initial approximation
2. Convergence to the wrong root
3. Closely spaced roots
4. Multiple roots
5. Inflection points
6. Complex roots
7. Ill-conditioning of the nonlinear equation
8. Slow convergence

These problems, and some strategies to avoid the problems, are discussed in this section.

Probably the most serious pitfall associated with root finding is the lack of a good initial approximation. Lack of a good initial approximation can lead to convergence to the wrong root, slow convergence, or divergence. The obvious way to avoid this problem is to obtain a better initial approximation. This can be accomplished by either graphing the function or a fine incremental search.

Closely spaced roots can be difficult to evaluate. Consider the situation illustrated in Figure 3.13. It can be difficult to determine where there are no roots, a double root, or two closely spaced distinct roots. This dilemma can be resolved by an enlargement of a graph of the function or the use of a smaller increment near the root in an incremental search.

Multiple roots, when known to exist, can be evaluated as described for Newton's method in Section 3.5.2. The major problem concerning multiple roots is not knowing they exist. Graphing the function or an incremental search can help identify the possibility of multiple roots.

Roots at an inflection point can send the root-finding procedure far away from the root. A better initial approximation can eliminate this problem.

Complex roots do not present a problem if they are expected. Newton's method or the secant method using complex arithmetic and complex initial approximations can find complex roots in a straightforward manner. However, if complex roots are not expected, and the root-finding method is using real arithmetic, complex roots cannot be evaluated. One solution to this problem is to use Bairstow's method for quadratic factors.

Ill-conditioning of the nonlinear function can cause serious difficulties in root finding. The problems are similar to those discussed in section 1.6.2 for solving ill-

conditioned systems of linear algebraic equations. In root-finding problems, the best approach for finding the roots of ill-conditioned nonlinear equations is to use a computing device with more precision (i.e., a larger number of significant digits).

The problem of slow convergence can be addressed by obtaining a better initial approximation or by a different root-finding method.

Most root-finding problems in engineering and science are well behaved and can be solved in a straightforward manner by one or more of the methods presented in this chapter. Consequently, each problem should be approached with the expectation of success. However, one must always be open to the possibility of unexpected difficulties and be ready and willing to pursue other approaches.

3.6.2. Other Methods of Root Finding

Most of the straightforward popular methods for root finding are presented in Sections 3.3 to 3.5. Several additional methods are identified, but not developed in this section.

Brent's (1978) method uses a superlinear method (i.e., inverse quadratic interpolation) and monitors its behavior to ensure that it is behaving properly. If not, some interval halving steps are used to ensure at least linear behavior until the root is approached more closely, at which time the procedure reverts to the superlinear method. Brent's method does not require evaluation of the derivative. This approach combines the efficiency of open methods with the robustness of closed methods.

Muller's method (1956), mentioned in Section 3.4.4, is an extension of the secant method which approximates the nonlinear function $f(x)$ with a quadratic function $g(x)$, and uses the root of $g(x)$ as the next approximation to the root of $f(x)$. A higher-order version of Newton's method, mentioned in Section 3.4.2, retains the second-order term in the Taylor series for $f(x)$. This method is not used very often because the increased complexity, compared to the secant method and Newton's method, respectively, is not justified by the slightly increased efficiency.

Several additional methods have been proposed for finding the roots of polynomials. Graeff's root squaring method (see Hildebrand, 1956), the Lehmer-Schur method (see Acton, 1970), and the QD (quotient-difference) method (see Henrici, 1964) are three such methods. Two of the more important additional methods for polynomials are Laguerre's method (Householder, 1970) and the Jenkins-Traub method. Ralston and Rabinowitz (1979) present a discussion of these methods. An algorithm for Laguerre's method is presented by Press et al. (1989). The Jenkins-Traub method is implemented in the IMSL library.

3.7 SYSTEMS OF NONLINEAR EQUATIONS

Many problems in engineering and science require the solution of a system of nonlinear equations. Consider a system of two nonlinear equations:

$$f(x, y) = 0 \tag{3.168a}$$

$$g(x, y) = 0 \tag{3.168b}$$

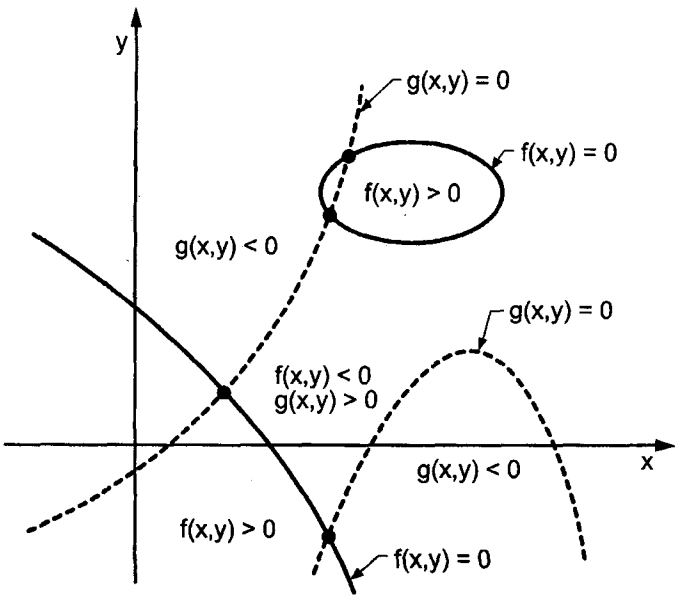


Figure 3.14 Solution of two nonlinear equations.

The problem can be stated as follows:

> Given the continuous functions $f(x, y)$ and $g(x, y)$, find the values $x = x^*$ and $y = y^*$ such that $f(x^*, y^*) = 0$ and $g(x^*, y^*) = 0$.

The problem is illustrated graphically in Figure 3.14. The functions $f(x, y)$ and $g(x, y)$ may be algebraic equations, transcendental equations, the solution of differential equations, or any nonlinear relationships between the inputs x and y and the outputs $f(x, y)$ and $g(x, y)$. The $f(x, y) = 0$ and $g(x, y) = 0$ contours divide the xy plane into regions where $f(x, y)$ and $g(x, y)$ are positive or negative. The solutions to Eq. (3.168) are the intersections of the $f(x, y) = g(x, y) = 0$ contours, if any. The number of solutions is not known a priori. Four such intersections are illustrated in Figure 3.14. This problem is considerably more complicated than the solution of a single nonlinear equation.

Interval halving and fixed-point iteration are not readily extendable to systems of nonlinear equations. Newton's method, however, can be extended to solve systems of nonlinear equations. In this section, Newton's method is extended to solve the system of two nonlinear equations specified by Eq. (3.168).

Assume that an approximate solution to Eq. (3.168) is known: (x_i, y_i). Express $f(x, y)$ and $g(x, y)$ in two-variable Taylor series about (x_i, y_i), and evaluate the Taylor series at (x^*, y^*). Thus,

$$f(x^*, y^*) = f_i + f_x|_i(x^* - x_i) + f_y|_i(y^* - y_i) + \cdots = 0 \tag{3.169a}$$

$$g(x^*, y^*) = g_i + g_x|_i(x^* - x_i) + g_y|_i(y^* - y_i) + \cdots = 0 \tag{3.169b}$$

Truncating Eq. (3.169) after the first derivative terms and rearranging yields

$$\boxed{f_x|_i\,\Delta x_i + f_y|_i\,\Delta y_i = -f_i} \tag{3.170a}$$

$$\boxed{g_x|_i\,\Delta x_i + g_y|_i\,\Delta y_i = -g_i} \tag{3.170b}$$

where Δx_i and Δy_i denote $(x^* - x_i)$ and $(y^* - y_i)$, respectively. Thus,

$$\boxed{x_{i+1} = x_i + \Delta x_i} \tag{3.171a}$$

$$\boxed{y_{i+1} = y_i + \Delta y_i} \tag{3.171b}$$

Equations (3.170) and (3.171) are applied repetitively until either one or both of the following convergence criteria are satisfied:

$$|\Delta x_i| \le \varepsilon_x \quad \text{and} \quad |\Delta y_i| \le \varepsilon_y \tag{3.172a}$$

$$|f(x_{i+1}, y_{i+1}) \le \varepsilon_f \quad \text{and} \quad |g(x_{i+1}, y_{i+1})| \le \varepsilon_g \tag{3.172b}$$

Example 3.12. Newton's method for two coupled nonlinear equations.

As an example of Newton's method for solving two nonlinear equations, let's solve the four-bar linkage problem presented in Section 3.1. Recall the two scalar components of the vector loop equation, Eq. (3.2):

$$\boxed{f(\theta_2, \theta_3) = r_2\cos(\theta_2) + r_3\cos(\theta_3) + r_4\cos(\theta_4) - r_1 = 0} \tag{3.173a}$$

$$\boxed{g(\theta_2, \theta_3) = r_2\sin(\theta_2) + r_3\sin(\theta_3) + r_4\sin(\theta_4) = 0} \tag{3.173b}$$

where r_1 to r_4 are specified, θ_4 is the input angle, and θ_2 and θ_3 are the two output angles.

Let θ_2^* and θ_3^* be the solution to Eq. (3.173), and θ_2 and θ_3 be an approximation to the solution. Writing Taylor series for $f(\theta_2, \theta_3)$ and $g(\theta_2, \theta_3)$ about (θ_2, θ_3) and evaluating at (θ_2^*, θ_3^*) gives

$$f(\theta_2^*, \theta_3^*) = f|_{\theta_2,\theta_3} + f_{\theta_2}|_{\theta_2,\theta_3}\,\Delta\theta_2 + f_{\theta_3}|_{\theta_2,\theta_3}\,\Delta\theta_3 + \cdots = 0 \tag{3.174a}$$

$$g(\theta_2^*, \theta_3^*) = g|_{\theta_2,\theta_3} + g_{\theta_2}|_{\theta_2,\theta_3}\,\Delta\theta_2 + g_{\theta_3}|_{\theta_2,\theta_3}\,\Delta\theta_3 + \cdots = 0 \tag{3.174b}$$

where $\Delta\theta_2 = (\theta_2^* - \theta_2)$ and $\Delta\theta_3 = (\theta_3^* - \theta_3)$. From Eq. (3.173),

$$f_{\theta_2} = -r_2\sin(\theta_2) \quad \text{and} \quad f_{\theta_3} = -r_3\sin(\theta_3) \tag{3.175a}$$

$$g_{\theta_2} = r_2\cos(\theta_2) \quad \text{and} \quad g_{\theta_3} = r_3\cos(\theta_3) \tag{3.174b}$$

Solving Eqs. (3.174) for $\Delta\theta_2$ and $\Delta\theta_3$ yields the following equations:

$$\boxed{(f_{\theta_2}|_{\theta_2,\theta_3})\,\Delta\theta_2 + (f_{\theta_3}|_{\theta_2,\theta_3})\,\Delta\theta_3 = -f(\theta_2, \theta_3)} \tag{3.176a}$$

$$\boxed{(g_{\theta_2}|_{\theta_2,\theta_3})\,\Delta\theta_2 + (g_{\theta_3}|_{\theta_2,\theta_3})\,\Delta\theta_3 = -g(\theta_2, \theta_3)} \tag{3.176b}$$

Equations (3.176a) and (3.176b) can be solved by Cramer's rule or Gauss elimination.

Table 3.14. Newton's Method for Two Coupled Nonlinear Equations

i	θ_2, deg	θ_3, deg	$f(\theta_2, \theta_3)$	$g(\theta_2, \theta_3)$	$\Delta\theta_2$, deg	$\Delta\theta_3$, deg
1	30.000000	0.000000	0.131975E + 00	0.428850E + 00	2.520530	−4.708541
2	32.520530	−4.708541	−0.319833E − 01	−0.223639E − 02	−0.500219	0.333480
3	32.020311	−4.375061	−0.328234E − 03	−0.111507E − 03	−0.005130	0.004073
4	32.015181	−4.370988	−0.405454E − 07	−0.112109E − 07	−0.000001	0.000000
	32.015180	−4.370987				

For the problem presented in Section 3.1, $r_1 = 10, r_2 = 6, r_3 = 8$, and $r_4 = 4$. Consider the case where $\theta_4 = 220.0$ deg. Let $\theta_2^{(1)} = 30.0$ deg and $\theta_3^{(1)} = 0.0$ deg. From Eq. (3.173):

$$f(30.0, 0.0) = 6.0\cos(30.0) + 8.0\cos(0.0) + 4.0\cos(220.0) - 10.0 = 0.131975 \tag{3.177a}$$

$$g(30.0, 0.0) = 6.0\sin(30.0) + 8.0\sin(0.0) + 4.0\sin(220.0) = 0.428850 \tag{3.177b}$$

Equations (3.175a) and (3.175b) give

$$f_{\theta_2} = -6.0\sin(30.0) = -3.000000 \quad \text{and} \quad f_{\theta_3} = -8.0\sin(0.0) = 0.0 \tag{3.178a}$$

$$g_{\theta_2} = 6.0\cos(30.0) = 5.196152 \quad \text{and} \quad g_{\theta_3} = 8.0\cos(0.0) = 8.0 \tag{3.178b}$$

Substituting these results into Eq. (3.176) gives

$$-3.000000\,\Delta\theta_2 + 0.0\,\Delta\theta_3 = -0.131975 \tag{3.179a}$$

$$5.196152\,\Delta\theta_2 + 8.0\,\Delta\theta_3 = -0.428850 \tag{3.179b}$$

Solving Eq. (3.179) gives

$$\Delta\theta_2 = 0.043992(180/\pi) = 2.520530 \text{ deg} \tag{3.180a}$$

$$\Delta\theta_3 = -0.082180(180/\pi) = -4.708541 \text{ deg} \tag{3.180b}$$

where the factor $(180/\pi)$ is needed to convert radians to degrees. Thus,

$$\theta_2 = 32.520530 \text{ deg} \quad \text{and} \quad \theta_3 = -4.708541 \text{ deg} \tag{3.181}$$

These results and the results of subsequent iterations are presented in Table 3.14. Four iterations are required to satisfy the convergence criteria $|\Delta\theta_2| \leq 0.000001$ and $|\theta_3| \leq 0.000001$. These results were obtained on a 13-digit precision computer. As illustrated in Figure 3.1, $\theta_2 = \phi$. From Table 3.1, $\phi = 32.015180$ deg, which is the same as θ_2.

In the general case,

$$\boxed{\mathbf{f}(\mathbf{x}) = 0} \tag{3.182}$$

where $\mathbf{f}(\mathbf{x})^T = [f_1(\mathbf{x}) \quad f_2(\mathbf{x}) \quad \cdots \quad f_n(\mathbf{x})]$ and $\mathbf{x}^T = [x_1 \quad x_2 \quad \cdots \quad x_n]$. In this case, Eqs. (3.170) and (3.171) become

$$\boxed{\mathbf{A}\Delta = \mathbf{f}} \tag{3.183}$$

where **A** is the $n \times n$ matrix of partial derivatives,

$$\mathbf{A} = \begin{bmatrix} (f_1)_{x_1} & (f_1)_{x_2} & \cdots & (f_1)_{x_n} \\ (f_2)_{x_1} & (f_2)_{x_2} & \cdots & (f_2)_{x_n} \\ \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots \\ (f_n)_{x_1} & (f_n)_{x_2} & \cdots & (f_n)_{x_n} \end{bmatrix} \tag{3.184}$$

Δ is the column vector of corrections,

$$\Delta^T = [\Delta x_1 \quad \Delta x_2 \quad \cdots \quad \Delta x_n] \tag{3.185}$$

and **f** is the column vector of function values

$$\mathbf{f}^T = [f_1 \quad f_2 \quad \cdots \quad f_n] \tag{3.186}$$

The most costly part of solving systems of nonlinear equations is the evaluation of the matrix of partial derivatives, **A**. Letting **A** be constant may yield a much less costly solution. However, **A** must be reasonably accurate for this procedure to work. A strategy based on making several corrections using constant **A**, then reevaluating **A**, may yield the most economical solution.

In situations where the partial derivatives of $\mathbf{f}(\mathbf{x})$ cannot be evaluated analytically, the above procedure cannot be applied. One alternate approach is to estimate the partial derivatives in Eq. (3.184) numerically. Thus,

$$\frac{\partial f_i}{\partial x_j} = \frac{f_i(\mathbf{x} + \Delta x_j) - f_i(\mathbf{x})}{\Delta x_j} \qquad (i, j = 1, 2, \ldots, n) \tag{3.187}$$

This procedure has two disadvantages. First, the number of calculations is increased. Second, if Δx_j is too small, round-off errors pollute the solution, and if Δx_j is too large, the convergence rate can decrease to first order. Nevertheless, this is one procedure for solving systems of nonlinear equations where the partial derivatives of $\mathbf{f}(\mathbf{x})$ cannot be determined analytically.

An alternate approach involves constructing a single nonlinear function $F(\mathbf{x})$ by adding together the sums of the squares of the individual functions $f_i(\mathbf{x})$. The nonlinear function $F(\mathbf{x})$ has a global minimum of zero when all of the individual functions are zero. Multidimensional minimization techniques can be applied to minimize $F(\mathbf{x})$, which yields the solution to the system of nonlinear equations, $\mathbf{f}(\mathbf{x}) = 0$. Dennis et al. (1983) discuss such procedures.

3.8 PROGRAMS

Three FORTRAN subroutines for solving nonlinear equations are presented in this section:

1. Newton's method
2. The secant method
3. Newton's method for two simultaneous equations

The basic computational algorithms are presented as completely self-contained subroutines suitable for use in other programs. Input data and output statements are contained in a main (or driver) program written specifically to illustrate the use of each subroutine.

3.8.1. Newton's Method

The general algorithm for Newton's method is given by Eq. (3.55):

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)} \tag{3.188}$$

A FORTRAN subroutine, *subroutine newton*, for implementing Newton's method is presented in Program 3.1. *Subroutine newton* requires a function subprogram, *function funct*, which evaluates the nonlinear equation of interest. *Function funct* must be completely self-contained, including all numerical values and conversion factors. The value of x is passed to *function funct*, and the values of $f(x)$ and $f'(x)$ are returned as f and fp, respectively. Every nonlinear equation requires its own individual *function funct*. *Subroutine newton* calls *function funct* to evaluate $f(x)$ and $f'(x)$ for a specified value of x, applies Eq. (3.188), checks for convergence, and either continues or returns. After *iter* iterations, an error message is printed and the solution is terminated. Program 3.1 defines the data set and prints it, calls *subroutine newton* to implement the solution, and prints the solution.

Program 3.1. Newton's method program.

```
      program main
c     main program to illustrate nonlinear equation solvers
c     x1    first guess for the root
c     iter  number of iterations allowed
c     tol   convergence tolerance
c     iw    intermediate results output flag:  0 no,  1 yes
      data x1,iter,tol,iw / 30.0, 10, 0.000001, 1 /
      write (6,1000)
      write (6,1010)
      call newton (x1,iter,tol,iw,i)
      call funct (x1,f1,fp1)
      write (6,1020) i,x1,f1
      stop
 1000 format (' Newtons method')
 1010 format (' '/'   i',6x,'xi',10x,'fi',12x,'fpi',11x,'xi+1'/' ')
 1020 format (i4,f12.4.g,f14.8)
      end

      subroutine newton (x1,iter,tol,iw,i)
c     Newton's method
      do i=1,iter
         call funct (x1,f1,fp1)
         dx=-f1/fp1
         x2=x1+dx
         if (iw.eq.1) write (6,1000) i,x1,f1,fp1,x2
         x1=x2
         if (abs(dx).le.tol) return
      end do
      write (6,1010)
      return
 1000 format (i4,f12.4.g,2f14.8,f12.6)
```

```
1010 format (' '/' Iterations failed to converge')
     end

     function funct (x,f,fp)
c    evaluates the nonlinear function
     data r1,r2,r3,alpha / 1.66666667, 2.5, 1.83333333, 40.0 /
     rad=acos(-1.0)/180.0
     f=r1*cos(alpha*rad)-r2*cos(x*rad)+r3-cos((alpha-x)*rad)
     fp=(r2*sin(x*rad)-sin((alpha-x)*rad))*rad
     return
     end
```

The data set used to illustrate *subroutine newton* is taken from Example 3.4. The output generated by the Newton method program is presented in Output 3.1.

Output 3.1. Solution by Newton's method.

```
Newtons method

  i      xi          fi           fpi          xi+1

  1   30.000000   -0.03979719   0.01878588   32.118463
  2   32.118463    0.00214376   0.02080526   32.015423
  3   32.015423    0.00000503   0.02070767   32.015180
  4   32.015180    0.00000000   0.02070744   32.015180
  4   32.015180    0.00000000
```

Subroutine newton can be used to solve most of the nonlinear equations presented in this chapter. The values in the *data* statement must be changed accordingly, and the function subprogram, *function funct*, must evaluate the desired nonlinear equation. Complex roots can be evaluated simply by declaring all variables to be complex variables. As an additional example, *function funct* presented below evaluates a real coefficient polynomial up to fourth degree. This *function funct* is illustrated by solving Example 3.7. Simple roots can be evaluated directly. Multiple roots can be evaluated in three ways: directly (which reduces the order to first order), by including the multiplicity m as a coefficient of $f(x)$ in *function funct*, or by defining $u(x) = f(x)/f'(x)$ in *function funct*.

Polynomial *function funct.*

```
     function funct (x,f,fp)
c    evaluates a polynomial of up to fourth degree
     data a0,a1,a2,a3,a4 / -2.0, 4.0, -3.0, 1.0, 0.0 /
     f=a0+a1*x+a2*x**2+a3*x**3+a4*x**4
     fp=a1+2.0*a2*x+3.0*a3*x**2+4.0*a4*x**3
     return
     end
```

The data set used to illustrate the polynomial *function funct* is taken from Example 3.7. The results are presented below.

Solution by Newton's method.

```
Newtons method

  i       xi          fi          fpi          xi+1

  1    1.500000   0.62500000   1.75000000   1.142857
  2    1.142857   0.14577259   1.06122449   1.005495
  3    1.005495   0.00549467   1.00009057   1.000000
  4    1.000000   0.00000033   1.00000000   1.000000
  4    1.000000   0.00000000
```

3.8.2. The Secant Method

The general algorithm for the secant method is given by Eq. (3.80):

$$x_{i+1} = x_i - \frac{f(x_i)}{g'(x_i)} \tag{3.189}$$

A FORTRAN subroutine, *subroutine secant*, for implementing the secant method is presented below. *Subroutine secant* works essentially like *subroutine newton* discussed in Section 3.8.1, except two values of x, x_1 and x_2, are supplied instead of only one value. Program 3.2 defines the data set and prints it, calls *subroutine secant* to implement the secant method, and prints the solution. Program 3.2 shows only the statements which are different from the statements in Program 3.1.

Program 3.2. The secant method program.

```
      program main
c     main program to illustrate nonlinear equation solvers
c     x2    second guess for the root
      data x1,x2,iter,tol,iw / 30.0, 40.0, 10, 0.000001, 1 /
      call secant (x1,x2,iter,tol,iw,i)
      call funct (x1,f1)
 1000 format (' The secant method')
 1010 format (' '/'   i',6x,'xi',10x,'fi',12x,'gpi',11x,'xi+1'/' ')
      end

      subroutine secant (x1,x2,iter,tol,iw,i)
c     the secant method
      call funct (x1,f1)
      if (iw.eq.1) write (6,1000) i,x1,f1
      do i=1,iter
         call funct (x2,f2)
         gp2=(f2-f1)/(x2-x1)
         dx=-f2/gp2
         x3=x2+dx
```

```
         if (iw.eq.1) write (6,1000) i,x2,f2,gp2,x3
         x1=x3
         if (abs(dx).le.tol) return
         x1=x2
         f1=f2
         x2=x3
      end do
      write (6,1010)
      return
 1000 format (i4,f12.4.g,2f14.8,f12.6)
 1010 format (' '/' Iterations failed to converge')
      end

      function funct (x,f)
c     evaluates the nonlinear function
      end
```

The data set used to illustrate *subroutine secant* is taken from Example 3.5. The output generated by the secant method program is presented in Output 3.2.

Output 3.2. Solution by the secant method.

```
The secant method

  i       xi           fi             gpi          xi+1

  0    30.000000   -0.03979719
  1    40.000000    0.19496296    0.02347602   31.695228
  2    31.695228   -0.00657688    0.02426795   31.966238
  3    31.966238   -0.00101233    0.02053257   32.015542
  4    32.015542    0.00000749    0.02068443   32.015180
  5    32.015180   -0.00000001    0.02070761   32.015180
  5    32.015180    0.00000000
```

3.8.3. Newton's Method for Two Coupled Nonlinear Equations

The general algorithm for Newton's method for two simultaneous nonlinear equations is given by Eqs. (3.170) and (3.171).

$$f_x|_i \, \Delta x_i + f_y|_i \, \Delta y_i = -f_i \tag{3.190a}$$

$$g_x|_i \, \Delta x_i + g_y|_i \, \Delta y_i = -g_i \tag{3.190b}$$

$$x_{i+1} = x_i + \Delta x_i \tag{3.191a}$$

$$y_{i+1} = y_i + \Delta y_i \tag{3.191b}$$

The general approach to this problem is the same as the approach presented in Section 3.8.1 for a single nonlinear equation. A FORTRAN subroutine, *subroutine simul*, for implementing the procedure is presented below. Program 3.3 defines the data set and prints it, calls *subroutine simul* to implement the solution, and prints the solution.

Program 3.3. Newton's method for simultaneous equations program.

```
      program main
c     main program to illustrate nonlinear equation solvers
c     x1,y1 first guess for the root
c     iter  number of iterations allowed
c     tol   convergence tolerance
c     iw    intermediate results output flag:  0 no,  1 yes
      data x1,y1,iter,tol,iw / 30.0, 0.0, 10, 0.000001, 1 /
      write (6,1000)
      write (6,1010)
      call simul (x1,y1,iter,tol,iw,i)
      call funct (x1,y1,f1,g1,fx,fy,gx,gy)
      write (6,1020) i,x1,y1,f1,g1
      stop
 1000 format (' Newtons method for two coupled nonlinear equations')
 1010 format (' '/'  i',6x,'xi',10x,'yi',9x,'fi',10x,'gi',9x,'dx',
     1 6x,'dy'/' ')
 1020 format (i3,2f12.6,2e12.4)
      end

      subroutine simul (x1,y1,iter,tol,iw,i)
c     Newton's method for two coupled nonlinear equations
      do i=1,iter
         call funct (x1,y1,f1,g1,fx,fy,gx,gy)
         del=fx*gy-fy*gx
         dx=(fy*g1-f1*gy)/del
         dy=(f1*gx-fx*g1)/del
         x2=x1+dx
         y2=y1+dy
         if (iw.eq.1) write (6,1000) i,x1,y1,f1,g1,dx,dy
         x1=x2
         y1=y2
         if ((abs(dx).le.tol).and.(abs(dy).le.tol)) return
      end do
      write (6,1010)
      return
 1000 format (i3,2f12.6,2e12.4,2f8.4)
 1010 format (' '/' Iteration failed to converge')
      end

      function funct (x,y,f,g,fx,fy,gx,gy)
c     evaluates the two nonlinear functions
      data r1,r2,r3,r4,theta4 / 10.0, 6.0, 8.0, 4.0, 220.0 /
      rad=acos(-1.0)/180.0
      f=r2*cos(x*rad)+r3*cos(y*rad)+r4*cos(theta4*rad)-r1
      g=r2*sin(x*rad)+r3*sin(y*rad)+r4*sin(theta4*rad)
      fx=(-r2*sin(x*rad))*rad
      fy=(-r3*sin(y*rad))*rad
      gx=(r2*cos(x*rad))*rad
      gy=(r3*cos(y*rad))*rad
      return
      end
```

The data set used to illustrate *subroutine simul* is taken from Example 3.12. The output is presented in Output 3.3.

Output 3.3. Solution by Newton's method for simultaneous equations.

```
Newtons method for two coupled nonlinear equations

 i      xi          yi          fi          gi          dx       dy

 1   30.000000    0.000000  0.1320E+00  0.4288E+00  2.5205 -4.7085
 2   32.520530   -4.708541 -0.3198E-01 -0.2236E-02 -0.5002  0.3335
 3   32.020311   -4.375061 -0.3282E-03 -0.1115E-03 -0.0051  0.0041
 4   32.015181   -4.370988 -0.4055E-07 -0.1121E-07  0.0000  0.0000
 4   32.015180   -4.370987  0.0000E+00  0.0000E+00
```

3.8.4. Packages for Nonlinear Equations

Numerous libraries and software packages are available for solving nonlinear equations. Many workstations and mainframe computers have such libraries attached to their operating systems.

Many commercial software packages contain nonlinear equation solvers. Some of the more prominent packages are Matlab and Mathcad. More sophisticated packages, such as IMSL, Mathematica, Macsyma, and Maple, also contain nonlinear equation solvers. Finally, the book *Numerical Recipes* (Press et al., 1989) contains numerous subroutines for solving nonlinear equations.

3.9 SUMMARY

Several methods for solving nonlinear equations are presented in this chapter. The nonlinear equation may be an algebraic equation, a transcendental equation, the solution of a differential equation, or any nonlinear relationship between an input x and a response $f(x)$.

Interval halving (bisection) and false position (regula falsi) converge very slowly, but are certain to converge because the root lies in a closed interval. These methods are not recommended unless the nonlinear equation is so poorly behaved that all other methods fail.

Fixed-point iteration converges only if the derivative of the nonlinear function is less than unity in magnitude. Consequently, it is not recommended.

Newton's method and the secant method are both effective methods for solving nonlinear equations. Both methods generally require reasonable initial approximations. Newton's method converges faster than the secant method (i.e., second order compared to 1.62 order), but Newton's method requires the evaluation of the derivative of the nonlinear function. If the effort required to evaluate the derivative is less than 43 percent of the effort required to evaluate the function itself, Newton's method requires less total effort than the secant method. Otherwise, the secant method requires less total effort. For functions whose derivative cannot be evaluated, the secant method is recommended. Both methods can find complex roots if complex arithmetic is used. The secant method is recommended as the best general purpose method.

The higher-order variations of Newton's method and the secant method, that is, the second-order Taylor series method and Muller's method, respectively, while quite effective, are not used frequently. This is probably because Newton's method and the secant method

are so efficient that the slightly more complicated logic of the higher-order methods is not justified.

Polynomials can be solved by any of the methods for solving nonlinear equations. However, the special features of polynomials should be taken into account.

Multiple roots can be evaluated using Newton's basic method or its variations.

Complex roots can be evaluated by Newton's method or the secant method by using complex arithmetic and complex initial approximations. Complex roots can also be evaluated by Bairstow's method for quadratic factors.

Solving systems of nonlinear equations is a difficult task. For systems of nonlinear equations which have analytical partial derivatives, Newton's method can be used. Otherwise, multidimensional minimization techniques may be preferred. No single approach has proven to be the most effective. Solving systems of nonlinear equations remains a difficult problem.

After studying Chapter 3, you should be able to:

1. Discuss the general features of root finding for nonlinear equations
2. Explain the concept of bounding a root
3. Discuss the benefits of graphing a function
4. Explain how to conduct an incremental search
5. Explain the concept of refining a root
6. Explain the difference between closed domain (bracketing methods) and open domain methods
7. List several closed domain (bracketing) methods
8. List several open domain methods
9. Discuss the types of behavior of nonlinear equations in the neighborhood of a root
10. Discuss the general philosophy of root finding
11. List two closed domain (bracketing) methods
12. Explain how the internal halving (bisection) method works
13. Apply the interval halving (bisection) method
14. List the advantages and disadvantages of the interval halving (bisection) method
15. Explain how the false position (regula falsi) method works
16. Apply the false position (regula falsi) method
17. List the advantages and disadvantages of the false position (regula falsi) method
18. List several open domain methods
19. Explain how the fixed-point iteration method works
20. Apply the fixed-point iteration method
21. List the advantages and disadvantages of the fixed-point iteration method
22. Explain how Newton's method works
23. Apply Newton's method
24. List the advantages and disadvantages of Newton's method
25. Explain and apply the approximate Newton's method
26. Explain and apply the lagged Newton's method
27. Explain how the secant method works
28. Apply the secant method
29. List the advantages and disadvantages of the secant method
30. Explain the lagged secant method

31. Explain how Muller's method works
32. Apply Muller's method
33. List the advantages and disadvantages of Muller's method
34. Discuss the special features of polynomials
35. Apply the quadratic formula and the rationalized quadratic formula
36. Discuss the applicability (or nonapplicability) of closed domain methods and open domain methods for finding the roots of polynomials
37. Discuss the problems associated with finding multiple roots and complex roots
38. Apply Newton's basic method and its variations to find all the roots of a polynomial
39. Apply deflation to a polynomial
40. Explain the concepts underlying Bairstow's method for finding quadratic factors
41. Apply Bairstow's method to find real or complex roots
42. Discuss and give examples of the pitfalls of root finding
43. Suggest ways to get around the pitfalls
44. Explain the concepts underlying Newton's method for a system of nonlinear equations
45. Apply Newton's method to a system of nonlinear equations

EXERCISE PROBLEMS

In all of the problems in this chapter, carry at least six significant figures in all calculations, unless otherwise noted. Continue all iterative procedures until four significant figures have converged, unless otherwise noted. Consider the following four nonlinear equations:

$$f(x) = x - \cos(x) = 0 \quad \text{(A)} \qquad f(x) = e^x - \sin(\pi x/3) = 0 \quad \text{(B)}$$

$$f(x) = e^x - 2x - 2 = 0 \quad \text{(C)} \qquad f(x) = x^3 - 2x^2 - 2x + 1 = 0 \quad \text{(D)}$$

3.2 Closed Domain Methods

Interval Halving

1. Solve Eq. (A) by interval-halving starting with $x = 0.5$ and 1.0.
2. Solve Eq. (B) by interval-halving starting with $x = -3.5$ and -2.5.
3. Solve Eq. (C) by interval-halving starting with $x = 1.0$ and 2.0.
4. Solve Eq. (D) by interval-halving starting with $x = 0.0$ and 1.0.
5. Find the two points of intersection of the two curves $y = e^x$ and $y = 3x + 2$ using interval halving. Use (-1.0 and 0.0) and (2.0 and 3.0) as starting values.
6. Find the two points of intersection of the two curves $y = e^x$ and $y = x^4$ using interval halving. Use (-1.0 and 0.0) and (1.0 and 2.0) as starting values.
7. Problems 1 to 6 can be solved using any two initial values of x that bracket the root. Choose other sets of initial values of x to gain additional experience with interval halving.

False Position

8. Solve Eq. (A) by false position starting with $x = 0.5$ and 1.0.
9. Solve Eq. (B) by false position starting with $x = -3.5$ and -2.5.
10. Solve Eq. (C) by false position starting with $x = 1.0$ and 2.0.

11. Solve Eq. (D) by false position starting with $x = 0.0$ and 1.0.
12. Find the two points of intersection of the two curves $y = e^x$ and $y = 3x + 2$ using false position. Use (-1.0 and 0.0) and (2.0 and 3.0) as starting values.
13. Find the two points of intersection of the two curves $y = e^x$ and $y = x^4$ using false position. Use (-1.0 and 0.0) and (1.0 and 2.0) as starting values.
14. Problems 8 to 13 can be solved using any two initial values of x that bracket the root. Choose other sets of initial values of x to gain additional experience with false position.

3.4 Open Domain Methods

Fixed-Point Iteration

15. Solve Eq. (A) by fixed-point iteration with $x_0 = 0.5$.
16. Solve Eq. (A) by fixed-point iteration with $x_0 = 1.0$.
17. Solve Eq. (B) by fixed-point iteration with $x_0 = -3.5$.
18. Solve Eq. (B) by fixed-point iteration with $x_0 = -2.5$.
19. Solve Eq. (C) by fixed-point iteration with $x_0 = 1.0$.
20. Solve Eq. (C) by fixed-point iteration with $x_0 = 2.0$.
21. Solve Eq. (D) by fixed-point iteration with $x_0 = 0.0$.
22. Solve Eq. (D) by fixed-point iteration with $x_0 = 1.0$.
23. Problem 5 considers the function $f(x) = e^x - (3x + 2) = 0$, which can be rearranged into the following three forms: (a) $x = e^x - (2x + 2)$, (b) $x = (e^x - 2)/3$, and (c) $x = \ln(3x + 2)$. Solve for the positive root by fixed-point iteration for all three forms, with $x_0 = 1.0$.
24. Solve Problem 6 by fixed-point iteration with $x_0 = -1.0$ and $x_0 = 1.0$.
25. The function $f(x) = (x + 2)(x - 4) = x^2 - 2x - 8 = 0$ has the two roots $x = -2$ and 4. Rearrange $f(x)$ into the form $x = g(x)$ to obtain the root (a) $x = -2$ and (b) $x = 4$, starting with $x_0 = -1$ and 3, respectively. The function $f(x)$ can be rearranged in several ways, for example, (a) $x = 8/(x - 2)$, (b) $x = (2x + 8)^{1/2}$, and (c) $x = (x^2 - 8)/2$. One form always converges to $x = -2$, one form always converges to $x = 4$, and one form always diverges. Determine the behavior of the three forms.
26. For what starting values of x might the expression $x = 1/(x + 1)$ not converge?
27. The function $f(x) = e^x - 3x^2 = 0$ has three roots. The function can be rearranged into the form $x = \pm [e^x/3]^{1/2}$. Starting with $x_0 = 0.0$, find the roots corresponding to (a) the + sign (near $x = 1.0$) and (b) the − sign (near -0.5). (c) The third root is near $x = 4.0$. Show that the above form will not converge to this root, even with an initial guess close to the exact root. Develop a form of $x = g(x)$ that will converge to this root, and solve for the third root.
28. The cubic polynomial $f(x) = x^3 + 3x^2 - 2x - 4 = 0$ has a root near $x = 1.0$. Find two forms of $x = g(x)$ that will converge to this root. Solve these two forms for the root, starting with $x_0 = 1.0$.

Newton's Method

29. Solve Eq. (A) by Newton's method. Use $x_0 = 1.0$ as the starting value.
30. Solve Eq. (B) by Newton's method. Use $x_0 = -3.0$ as the starting value.

31. Solve Eq. (C) by Newton's method. Use $x_0 = 1.0$ as the starting value.
32. Solve Eq. (D) by Newton's method. Use $x_0 = 1.0$ as the starting value.
33. Find the positive root of $f(x) = x^{15} - 1 = 0$ by Newton's method, starting with $x_0 = 1.1$.
34. Solve Problem 33 using $x = 0.5$ as the initial guess. You may want to solve this problem on a computer, since a large number of iterations may be required.
35. The nth root of the number N can be found by solving the equation $x^n - N = 0$. (a) For this equation, show that Newton's method gives

$$x_{i+1} = \frac{1}{n}\left[(n-1)x_i + \frac{N}{x_1^{n+1}}\right] \tag{E}$$

Use the above result to solve the following problems: (a) $(161)^{1/3}$, (b) $(21.75)^{1/4}$, (c) $(238.56)^{1/5}$. Use $x = 6.0$, 2.0, and 3.0, respectively, as starting values.

36. Consider the function $f(x) = e^x - 2x^2 = 0$. (a) Find the two positive roots using Newton's method. (b) Find the negative root using Newton's method.

The Secant Method

37. Solve Eq. (A) by the secant method. Use $x = 0.5$ and 1.0 as starting values.
38. Solve Eq. (B) by the secant method. Use $x_0 = -3.0$ and -2.5 as starting values.
39. Solve Eq. (C) by the secant method. Use $x_0 = 1.0$ and 2.0 as starting values.
40. Solve Eq. (D) by the secant method. Use $x_0 = 0.0$ and 1.0 as starting values.
41. Find the positive root of $f(x) = x^{15} - 1 = 0$ by the secant method using $x = 1.2$ and 1.1 as starting values.
42. Solve Problem 41 by the secant method with $x = 0.5$ and 0.6 as starting values. You may want to solve this problem on a computer, since a large number of iterations may be required.
43. Solve Problems 35(a) to (c) by the secant method. Use the starting values given there for x_0 and let $x_1 = 1.1x_0$.
44. Solve Problem 36 using the secant method.

3.5. Polynomials

45. Use Newton's method to find the real roots of the following polynomials:

(a) $x^3 - 5x^2 + 7x - 3 = 0$ (b) $x^4 - 9x^3 + 24x^2 - 36x + 80 = 0$
(c) $x^3 - 2x^2 - 2x + 1 = 0$ (d) $3x^3 + 4x^2 - 8x - 2 = 0$

46. Use Newton's method to find the complex roots of the following polynomials:

(a) $x^4 - 9x^3 + 24x^2 - 36x + 80 = 0$ (b) $x^3 + 2x^2 + x + 2 = 0$
(c) $x^5 - 15x^4 + 85x^3 - 226x^2 + 274x - 120 = 0$

3.7 Systems of Nonlinear Equations

Solve the following systems of nonlinear equations using Newton's method.

47. $(x-1)^2+(y-2)^2=3$ and $x^2/4+y^2/3=1$. Solve for all roots.
48. $y=\cosh(x)$ and $x^2+y^2=2$. Solve for both roots.
49. $x^2+y^2=2x+y$ and $x^2/4+y^2=1$. Solve for all four roots.
50. $y^2(1-x)=x^3$ and $x^2+y^2=1$.
51. $x^3+y^3-3xy=0$ and $x^2+y^2=1$.
52. $(x^2+y^2)^2=2xy$ and $y=x^3$.
53. $(2x)^{2/3}+y^{2/3}=(9)^{1/3}$ and $x^2/4+y^2=1$.

3.8. Programs

54. Implement the Newton method program presented in Section 3.8.1. Check out the program using the given data set.
55. Work any of Problems 29 to 36 using the Newton method program.
56. Implement the secant method program presented in Section 3.8.2. Check out the program using the given data set.
57. Work any of Problems 37 to 44 using the secant method program.
58. Implement the Newton method program presented in Section 3.8.3 for solving simultaneous equations. Check out the program using the given data set.
59. Work any of Problems 5, 6, or 47 to 53 using the Newton method program.
60. Write a computer program to solve Freudenstein's equation, Eq. (3.3), by the secant method. Calculate ϕ for $\alpha=40$ deg to 90 deg in increments $\Delta\alpha=10$ deg. For $\alpha=40$ deg, let $\phi_0=25$ deg and $\phi_1=30$ deg. For subsequent values of α, let ϕ_0 be the solution value for the previous value of α, and $\phi_1=\phi_0+1.0$. Continue the calculations until ϕ changes by less than 0.00001 deg. Design the program output as illustrated in Example 3.5.
61. Write a computer program to solve the van der Waal equation of state, Eq. (G) in Problem 69, by the secant method. Follow the procedure described in Problem 69 to initiate the calculations. Design the program output as illustrated in Example 3.5. For $P=10{,}000$ kPa, calculate v corresponding to $T=700$, 800, 900, 1000, 1100, 1200, 1300, 1400, 1500, and 1600 K. Write the program so that all the cases can be calculated in one run by stacking input data decks.
62. Write a computer program to solve the Colebrook equation, Eq. (I) in Problem 70, by the secant method for the friction coefficient f for specified value of the roughness ratio ε/D and the Reynolds number, Re. Use the approximation proposed by Genereaux (1939), Eq. (J), and 90 percent of that value as the initial approximations. Solve Problem 70 using the program.
63. Write a computer program to solve the $M-\varepsilon$ equation, Eq. (K) in Problem 71, by Newton's method for specified values of γ and ε. (a) Solve Problem 71, using the program. (b) Construct a table of M versus ε for $1.0\le\varepsilon\le10$, for subsonic flow, in increments $\Delta\varepsilon=0.1$. For $\varepsilon=1.1$, let $M_0=0.8$. For subsequent values of ε, let M_0 be the previous solution value.
64. Write a computer program to solve the $M-\varepsilon$ equation, Eq. (K) in Problem 71, by the secant method for specified values of γ and ε. (a) Solve Problem 71 using the program. (b) Construct a table of M versus ε for $1.0\le\varepsilon\le10$, for

supersonic flow, in increments $\Delta\varepsilon = 0.1$. For $\varepsilon = 1.1$ let $M_0 = 1.2$ and $M_1 = 1.3$. For subsequent values of ε, let M_0 be the previous solution value and $M_1 = 1.1M_0$.

65. Write a computer program to solve Eq. (L) in Problem 73 by the secant method for specified values of γ, δ, and M_1. (a) Solve Problem 73 using the program. (b) Construct
a table of M versus δ for $\gamma = 1.4$ and $M_1 = 1.0$, for $0 < \delta \leq 40$ deg, in increments $\Delta\delta = 1.0$ deg. For $\delta = 1.0$ deg, let $M_0 = 1.06$ and $M_1 = 1.08$. For subsequent values of δ, let M_0 be the previous solution value and $M_1 = 1.01M_0$.

APPLIED PROBLEMS

Several applied problems from various disciplines are presented in this section. All of these problems can be solved by any of the methods presented in this chapter. An infinite variety of exercises can be constructed by changing the numerical values of the parameters of the problem, by changing the starting values, or both.

66. Consider the four-bar linkage problem presented in Section 3.1. Solve for any (or all) of the results presented in Table 3.1.
67. Consider the four-bar linkage problem presented in Section 3.1. Rearrange this problem to solve for the value of r_1 such that $\phi = 60$ deg when $\alpha = 75$ deg. Numerous variations of this problem can be obtained by specifying combinations of ϕ and α.
68. Solve the four-bar linkage problem for $\theta_4 = 210$ deg by solving the two scalar components of the vector loop equation, Eq. (3.2), by Newton's method. Let the initial guesses be $\theta_2 = 20$ deg and $\theta_3 = 0$ deg. Continue the calculations until θ_2 and θ_3 change by less than 0.00001 deg. Show all calculations for the first iteration. Summarize the first iteration and subsequent iterations in a table, as illustrated in Example 3.12.
69. The van der Waal equation of state for a vapor is

$$\left(P + \frac{a}{v^2}\right)(v - b) = RT \tag{F}$$

where P is the pressure (Pa = N/m^2), v is the specific volume (m^3/kg), T is the temperature (K), R is the gas constant (J/kg-K), and a and b are empirical constants. Consider water vapor, for which $R = 461.495$ J/kg-K, $a = 1703.28$ Pa-(m^3/kg)^3, and $b = 0.00169099$ (m^3/kg). Equation (F) can be rearranged into the form

$$Pv^3 - (Pb + RT)v^2 + av - ab = 0 \tag{G}$$

Calculate the specific volume v for $P = 10{,}000$ kPa and $T = 800$ K. Use the ideal gas law, $Pv = RT$, to obtain the initial guess (or guesses). Present the results in the format illustrated in the examples.
70. When an incompressible fluid flows steadily through a round pipe, the pressure drop due to the effects of wall friction is given by the empirical formula:

$$\Delta P = -0.5 f \rho V^2 \left(\frac{L}{D}\right) \tag{H}$$

where ΔP is the pressure drop, ρ is the density, V is the velocity, L is the pipe length, D is the pipe diameter, and f is the D'Arcy friction coefficient. Several empirical formulas exist for the friction coefficient f as a function of the dimensionless Reynolds number, $\mathrm{Re} = DV\rho/\mu$, where μ is the viscosity. For flow in the turbulent regime between completely smooth pipe surfaces and wholly rough pipe surfaces, Colebrook (1939) developed the following empirical equation for the friction coefficient f:

$$\frac{1}{f^{1/2}} = -2\log_{10}\left(\frac{\varepsilon/D}{3.7} + \frac{2.51}{\mathrm{Re}\, f^{1/2}}\right) \tag{I}$$

where ε is the pipe surface roughness. Develop a procedure to determine f for specified values of ε/D and Re. Use the approximation proposed by Genereaux (1939) to determine the initial approximation(s):

$$f = 0.16\ \mathrm{Re}^{-0.16} \tag{J}$$

Solve for f for a pipe having $\varepsilon/D = 0.001$ for $\mathrm{Re} = 10^n$, for $n = 4, 5, 6$, and 7.

71. Consider quasi-one-dimensional isentropic flow of a perfect gas through a variable-area channel. The relationship between the Mach number M and the flow area A, derived by Zucrow and Hoffman [1976, Eq. (4.29)], is given by

$$\varepsilon = \frac{A}{A^*} = \frac{1}{M}\left[\frac{2}{\gamma+1}\left(1 + \frac{\gamma-1}{2}M^2\right)\right]^{(\gamma+1)/2(\gamma-1)} \tag{K}$$

where A^* is the choking area (i.e., the area where $M = 1$) and γ is the specific heat ratio of the flowing gas. For each value of ε, two values of M exist, one less than unity (i.e., subsonic flow) and one greater than unity (i.e., supersonic flow). Calculate both values of M for $\varepsilon = 10.0$ and $\gamma = 1.4$ by Newton's method. For the subsonic root, let $M_0 = 0.2$. For the supersonic root, let $M_0 = 5.0$.

72. Solve Problem 71 by the secant method. For the subsonic root, let $M_0 = 0.4$ and $M_1 = 0.6$. For the supersonic root, let $M_0 = 3.0$ and $M_1 = 4.0$.

73. Consider isentropic supersonic flow around a sharp expansion corner. The relationship between the Mach number before the corner (i.e., M_1) and after the corner (i.e., M_2), derived by Zucrow and Hoffman [1976, Eq. (8.11)], is given by

$$\delta = b^{1/2}\left(\tan^{-1}\left(\frac{M_2^2-1}{b}\right)^{1/2} - \tan^{-1}\left(\frac{M_1^2-1}{b}\right)^{1/2}\right)$$
$$- \left((\tan^{-1}((M_2^2-1)^{1/2}) - \tan^{-1}((M_1^2-1)^{1/2})\right) \tag{L}$$

where $b = (\gamma+1)/(\gamma-1)$ and γ is the specific heat ratio of the gas. Develop a procedure to solve for M_2 for specified values of γ, δ, and M_1. For $\gamma = 1.4$, solve for M_2 for the following combinations of M_1 and δ: (a) 1.0 and 10.0 deg, (b) 1.0 and 20.0 deg, (c) 1.5 and 10.0 deg, and (d) 1.5 and 20.0 deg. Use $M_2^{(0)} = 2.0$ and $M_2^{(1)} = 1.5$.