number of trial values for $x$ to be considered. Fortunately, most problems
do not require an exact value for the root; instead, an estimate of the root
(to within some specified degree of precision) is often sufficient.

The direct-search method is perhaps the clearest, but it also can be the
least efficient technique for estimating the roots of functions. The steps in
this method can be outlined as follows:

1. Specify a range or interval for $x$ within which the root is
   assumed to occur. Some knowledge of the behavior of the func-
tion is required to specify this interval. Smaller initial intervals
will require fewer calculations to obtain the root to the desired
accuracy.

2. Divide the interval into smaller, uniformly spaced subintervals.
   The size of these subintervals will be dictated by the required
   precision for the estimate of the root. For example, if the root
   must be estimated to within $\pm 0.001$, then the subintervals for the
   root will be 0.001 units long.

3. Search through all subintervals until the subinterval containing
   the root is located; this occurs when the equality of Equation 4.1
   exists within the interval.

The last step is the key to the method. A simple test for determining
if the root occurs within a given subinterval is illustrated in Figure 4.2.
The function $f(x)$ is evaluated at the beginning and end points, $A$ and
$B$, respectively, of the subinterval. If a root does not lie within the sub-
interval, then the values of $f(A)$ and $f(B)$ will have the same sign—that
is, either both negative or both positive (Figures 4.2a and b). However,
if a root does lie within the subinterval, then the values of $f(A)$ and $f(B)$
will have different signs (Figures 4.2c and d). Alternatively stated, if the
product of $f(A)$ and $f(B)$ is positive, then the function does not likely cross
the $x$ axis and a root does not lie within the subinterval; if the product is
negative, then the function crosses the $x$ axis and a root lies within the
subinterval.

A FORTRAN program listing for the direct-search method for finding
roots of functions is given in Figure 4.3. This program will find all roots
of $f(x)$ within the interval $A \leq x \leq B$ as long as the input precision value
(that is, tolerance) TOL is sufficiently small that multiple roots will not
occur within a single subinterval. Note that $f(x)$ is evaluated in a function
subprogram; thus the program can be used to find the roots of any func-
tion simply be substituting the appropriate function subprogram for $f(x)$. In
Figure 4.3, the function $f(x) = (x/2)^2 - \sin(x)$ is used in the FUNCTION
F(X) subprogram for the purpose of illustration.

The direct-search method will find the roots of any function as long as
all the roots are real and within the specified interval. However, it should
be obvious that this method is very cumbersome. For a high degree of
precision, the subinterval size must be very small and a large number of
calculations must be performed. To minimize the number of calcula-
tions, the subinterval size must be increased, but this obviously reduces
the precision of the estimate roots. In addition, some roots might be
missed entirely if the subinterval size is so large that more than one root
occurs in a single subinterval. Nevertheless, the direct-search method is
the most straightforward technique for determining all roots within a
given interval.

The direct-search method assumes that there is one and only one root
within each subinterval. If there is an even number of roots within the
search interval, then \( f(A) \) and \( f(B) \) will have the same sign, and the search
process will miss the roots within the interval. For example, given the
function

\[
(4.13) \quad f(x) = (x - 1.1)(x - 1.2)(x + 0.8) = x^3 - 1.52x - 0.52x + 1.056
\]

the two positive roots will not be found if the search interval is 0.25,
with \( A = 1.0 \) and \( B = 1.25 \). In this case, \( f(A) = 0.036 \) and \( f(B) = 0.0154 \),

![Figure 4.2](image-url)
and so the search will proceed to the next search interval and miss the roots \( X = 1.1 \) and \( X = 1.2 \). This is especially critical where the function has two equal roots (called multiple roots), because the direct-search method will not identify any roots regardless of the size of the search subinterval. The multiple-root case is illustrated graphically in Figure 4.4 as the tangent point to the \( x \) axis. This root cannot be determined by the direct-search method. An example function that has multiple roots (double roots) is

\[
(4.14) \quad f(x) = (x - 1)^2 = x^2 - 2x + 1
\]

Another case that might result in the failure of the direct-search method in finding the roots is shown in Figure 4.5. In this case, the function has a discontinuity at a point within the interval \([A, B]\) of interest.

<table>
<thead>
<tr>
<th>PROGRAM ROOTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>c Computes roots of a function using direct-search method</td>
</tr>
<tr>
<td>c Read input data</td>
</tr>
<tr>
<td>PRINT *, 'ENTER STARTING POINT FOR SEARCH INTERVAL:'</td>
</tr>
<tr>
<td>READ *, A</td>
</tr>
<tr>
<td>PRINT *, 'ENTER ENDING POINT FOR SEARCH INTERVAL:'</td>
</tr>
<tr>
<td>READ *, B</td>
</tr>
<tr>
<td>PRINT *, 'ENTER REQUIRED PRECISION:'</td>
</tr>
<tr>
<td>READ *, PREC</td>
</tr>
<tr>
<td>c Compute number and size of subintervals</td>
</tr>
<tr>
<td>NINTVL = INT((B-A)/PREC) + 1</td>
</tr>
<tr>
<td>DELTAX = (B-A)/REAL(NINTVL)</td>
</tr>
<tr>
<td>c Initialize subinterval end points</td>
</tr>
<tr>
<td>XB = A</td>
</tr>
<tr>
<td>XE = XB + DELTAX</td>
</tr>
<tr>
<td>c Loop over all subintervals</td>
</tr>
<tr>
<td>DO 10 N = 1,NINTVL</td>
</tr>
<tr>
<td>FXB = F(XB)</td>
</tr>
<tr>
<td>FXE = F(XE)</td>
</tr>
<tr>
<td>IF (FXB*FXE.LE.0.0) THEN</td>
</tr>
<tr>
<td>PRINT 100, XB, XE</td>
</tr>
<tr>
<td>100 FORMAT('A ROOT LIES BETWEEN X=', E12.4,'AND', E12.4)</td>
</tr>
<tr>
<td>END IF</td>
</tr>
<tr>
<td>XB = XE</td>
</tr>
<tr>
<td>XE = XB + DELTAX</td>
</tr>
<tr>
<td>10 CONTINUE</td>
</tr>
<tr>
<td>END</td>
</tr>
</tbody>
</table>

FUNCTION F(X) |
| c Evaluates function at specified point |
| F = (X/2.0)**2 - SIN(X) |
| RETURN |

END
Example 4.2: Direct Search Solution of Eigenvalues

Consider the polynomial for Equation 4.11, which was the expansion of the determinant given in Section 4.1 as

\[ \lambda^3 - 3\lambda^2 + 2.3146\lambda - 0.504188 = 0 \]

The sum of the eigenvalues for a matrix with values of 1.0 on the principal diagonal equals the rank of the matrix, which equals the order of the characteristic polynomial; thus the three roots for this polynomial must be between 0.0 and 3.0, since all roots must be located in the range \(0 \leq \lambda \leq 3\). We will assume for the purposes of illustration that an accuracy of 0.1 is sufficient. Thus the direct-search procedure can begin with a value of \(\lambda = 0\) and continue in increments of 0.1 until estimates of all three roots are found. Table 4.1 gives the values of \(\lambda\) and \(f(\lambda)\) using the direct-search method.

The smallest root occurs in the interval \(0.3 < \lambda < 0.4\) since \(f(0.3)\) is negative and \(f(0.4)\) is positive. The second root occurs in the interval \(0.6 < \lambda < 0.7\) since \(f(0.6)\) is positive and \(f(0.7)\) is negative. The largest root has a value between 1.9 and 2.0 since \(f(1.9) < 0\) and \(f(2.0) > 0\). We could assume roots of 0.4, 0.7, and 1.9, since these represent the values of \(\lambda\) where \(f(\lambda)\) is closer to zero. Linear interpolation could be used to get better estimates, but we assumed that an accuracy of 0.1 was sufficient. If greater precision is required, the direct search method could be applied to each of the three relevant intervals.
4.4 BISECTION METHOD

The bisection method is an extension of the direct-search method for cases when it is known that only one root occurs within a given interval of $x$. For the same level of precision, the bisection method will, in general, require fewer calculations than the direct-search method. Using Figure 4.6 as a reference, the steps in the bisection method can be outlined as follows:

1. For the interval of $x$ from the starting point $x_s$ to the end point $x_e$, locate the midpoint $x_m$ at the center of the interval. These points correspond to the starting and ending points of the half-intervals.
2. At the starting point $x_s$, midpoint $x_m$, and the ending point $x_e$ of the interval, evaluate the function resulting into $f(x_s), f(x_m),$ and $f(x_e)$, respectively.
3. Compute the products of the functions evaluated at the ends of the two half-intervals, that is, $f(x_s) \cdot f(x_m)$ and $f(x_m) \cdot f(x_e)$. The root lies in the interval for which the product is negative, and the midpoint $x_m$ is used as the estimate of the root for this iteration.
4. Check for convergence as follows:
   a. If the convergence criterion (that is, tolerance) is satisfied, then use $x_m$ as the final estimate of the root.
   b. If the tolerance has not been met, specify the ends of the half-interval in which the root is located as the starting and ending points for a new interval and return to step 1.

<table>
<thead>
<tr>
<th>$\lambda$</th>
<th>$f(\lambda)$</th>
<th>$\lambda$</th>
<th>$f(\lambda)$</th>
<th>$\lambda$</th>
<th>$f(\lambda)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>-0.504</td>
<td>0.7</td>
<td>-0.011</td>
<td>1.4</td>
<td>-0.400</td>
</tr>
<tr>
<td>0.1</td>
<td>-0.302</td>
<td>0.8</td>
<td>-0.061</td>
<td>1.5</td>
<td>-0.407</td>
</tr>
<tr>
<td>0.2</td>
<td>-0.153</td>
<td>0.9</td>
<td>-0.122</td>
<td>1.6</td>
<td>-0.385</td>
</tr>
<tr>
<td>0.3</td>
<td>-0.053</td>
<td>1.0</td>
<td>-0.190</td>
<td>1.7</td>
<td>-0.326</td>
</tr>
<tr>
<td>0.4</td>
<td>0.006</td>
<td>1.1</td>
<td>-0.257</td>
<td>1.8</td>
<td>-0.226</td>
</tr>
<tr>
<td>0.5</td>
<td>0.199</td>
<td>1.2</td>
<td>-0.319</td>
<td>1.9</td>
<td>-0.077</td>
</tr>
<tr>
<td>0.6</td>
<td>0.021</td>
<td>1.3</td>
<td>-0.368</td>
<td>2.0</td>
<td>0.125</td>
</tr>
</tbody>
</table>

**FIGURE 4.6** Schematic of bisection method.
The bisection method will always converge on the root, provided that only one root lies within the starting interval for \( x \).

### 4.4.1 ERROR ANALYSIS AND CONVERGENCE CRITERION

Since numerical methods for finding the roots of functions are iterative, it is important to include a convergence criterion into the process of finding a root; otherwise, a computerized solution could continue to iterate indefinitely. Since the value of a root has significance to the engineering problem under consideration, the allowable error would depend on the tolerance permitted by the problem. Thus the absolute value between an estimate and the true value would be the ideal criterion to indicate convergence. But since the true value is not known, we need a surrogate value to represent the true value. In practice, the change in the estimate of the root from one trial to the next is used as the convergence criterion; this can be expressed as either the absolute value or as a percentage of the estimate. If the change between estimates on successive iterations is used as a convergence criterion, it is important to recognize that this criterion may not reflect the true accuracy of the estimate—that is, the absolute difference between the estimated value and the true value of the root.

To ensure closure of the iteration loop, a convergence criterion is needed to terminate the iterative procedure of a numerical method for finding the roots of a function. The convergence criterion used in step 4 of the bisection method can be expressed in terms of either the absolute value of the difference (\( \epsilon_d \)) or the percent relative error (\( \epsilon_r \)) for each iteration. The convergence criterion when the error is expressed as an absolute value is

\[
\epsilon_d = |x_{m,i+1} - x_{m,i}|
\]

For the case when the convergence criterion is expressed as a relative percent error, the criterion is

\[
\epsilon_r = \frac{|x_{m,i+1} - x_{m,i}|}{x_{m,i+1}} \times 100
\]

where \( x_{m,i} \) = the midpoint in the previous root-search iteration (\( i \)th iteration), and \( x_{m,i+1} \) = the midpoint in a new root-search iteration (\( i + 1 \) iteration). The tolerance used with the convergence criterion of Equation 4.15 can be specified depending on the accuracy that is needed for the specific problem. For some engineering problems, a tolerance of 5% according to Equation 4.16 is adequate; other problems may require a tolerance of 0.01%.

The true accuracy of the solution at any iteration can be computed if the true solution (root \( x_t \)) is known. In such a case, the true error (\( \epsilon_t \)) in the \( i \)th iteration is

\[
\epsilon_t = \frac{|x_t - x_{m,i}|}{x_t} \times 100
\]
Equation 4.17 requires knowledge of the true solution $x_t$. In practical use of numerical methods, the true solution is not known. Therefore, an approximate solution of the error can be evaluated using the relative error $x_r$ as given in Equation 4.16.

**Example 4.3: Roots of a Polynomial Using the Bisection Method**

The following polynomial is used to illustrate the use of the bisection method for finding its roots:

\[(4.18) \quad f(x) = x^3 - x^2 - 10x - 8 = 0\]

We are interested in finding the roots within the interval $3.75 \leq x \leq 5.00$ to a relative accuracy as an absolute value between successive iterations of 0.01. Using the four steps of the bisection method, the values of $x_s$, $x_m$, and $x_e$ were computed for each iteration $i$. The results are summarized in Table 4.2. Note that the midpoint is used as the estimate of the root for each iteration. The absolute value of the error is given in the last column of Table 4.2. The error decreases with each iteration, and on the seventh iteration the error equals the tolerance, so convergence is assumed. The final estimate of the root is 3.944.

It is evident that the root is near 4. Substituting $x = 4$ into Equation 4.18 shows that the true value of the root is 4. Therefore, the true accuracy of the estimated root is 0.006 or, in relative terms according to Equation 4.17, 0.15%.

**Example 4.4: Location of Maximum Bending Moment**

In Section 3.6.2, the location of the maximum bending moment for a simply supported beam was determined based on the condition that the shear force at this location is zero. The exact location was also expressed based on the conditions of statics as the solution of the following equation.

\[(4.19) \quad f(x) = 20 - x^2 = 0\]

<table>
<thead>
<tr>
<th>Iteration $i$</th>
<th>$x_s$</th>
<th>$x_m$</th>
<th>$x_e$</th>
<th>$f(x_s)$</th>
<th>$f(x_m)$</th>
<th>$f(x_e)$</th>
<th>$f(x_s)f(x_m)$</th>
<th>$f(x_m)f(x_e)$</th>
<th>Error $\epsilon_d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.750</td>
<td>4.375</td>
<td>5.000</td>
<td>-6.830</td>
<td>12.850</td>
<td>42.000</td>
<td>-</td>
<td>+</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>3.750</td>
<td>4.062</td>
<td>4.375</td>
<td>-6.830</td>
<td>1.903</td>
<td>12.850</td>
<td>-</td>
<td>+</td>
<td>0.313</td>
</tr>
<tr>
<td>3</td>
<td>3.750</td>
<td>3.906</td>
<td>4.062</td>
<td>-6.830</td>
<td>-2.724</td>
<td>1.903</td>
<td>+</td>
<td>-</td>
<td>0.156</td>
</tr>
<tr>
<td>4</td>
<td>3.906</td>
<td>3.984</td>
<td>4.062</td>
<td>-2.724</td>
<td>-0.477</td>
<td>1.903</td>
<td>+</td>
<td>-</td>
<td>0.078</td>
</tr>
<tr>
<td>5</td>
<td>3.984</td>
<td>4.023</td>
<td>4.062</td>
<td>-0.477</td>
<td>0.696</td>
<td>1.903</td>
<td>-</td>
<td>+</td>
<td>0.039</td>
</tr>
<tr>
<td>6</td>
<td>3.984</td>
<td>4.004</td>
<td>4.023</td>
<td>-0.477</td>
<td>0.120</td>
<td>0.696</td>
<td>-</td>
<td>+</td>
<td>0.019</td>
</tr>
<tr>
<td>7</td>
<td>3.984</td>
<td>3.994</td>
<td>4.004</td>
<td>-0.477</td>
<td>-0.180</td>
<td>0.120</td>
<td>+</td>
<td>-</td>
<td>0.010</td>
</tr>
</tbody>
</table>
The location of the maximum bending moment must lie within the span of the uniform load (i.e., \(0 \leq x \leq 6\) m). Therefore, the starting interval for the bisection method is \([0, 6]\). The computations of the bisection method are shown in Table 4.3. The estimated root \((x_m)\) and errors in percent are shown in the table, as well as in Figures 4.7 and 4.8, respectively. The two types of errors—the absolute error

![Figure 4.7](image1.png)

**FIGURE 4.7** Location of maximum bending moment.

![Figure 4.8](image2.png)

**FIGURE 4.8** Errors in the location of maximum bending moment.
and relative error—are shown in Figure 4.8. The relative error in an iteration was computed using Equation 4.16. The actual error for an iteration was computed according to Equation 4.17 using the true root \( x_t = 4.4721359 \).

### 4.5 NEWTON–RAPHSON ITERATION

Although the bisection method will always converge on the root, the rate of convergence is very slow. A faster method for converging on a single root of a function is the Newton–Raphson iteration method.

While the concept of a *Taylor series expansion* is formally introduced in Chapter 1, at this point it is sufficient to state that the equation that is used to develop the Newton–Raphson iteration method is the linear portion of a Taylor series:

\[
(4.20) \quad f(x_1) = f(x_0) + \frac{df}{dx} \Delta x
\]

where \( f(x_1) \) is the value of the function at a specific value \( x_1 \) of the independent variable \( x \), \( f(x_0) \) is the value at \( x_0 \), \( \Delta x \) equals the difference \( (x_1 - x_0) \), and \( df/dx \) is the derivative of the relationship. Since the root of the function relating \( f(x) \) and \( x \) is the value of \( x \) where \( f(x) \) equals zero, Equation 4.20 can be expressed as

\[
(4.21) \quad 0 = f(x_0) + \frac{df}{dx} (x_1 - x_0)
\]

where \( x_1 \) will now be the root of the function, since \( f(x_1) \) was assumed to be zero. Solving Equation 4.21 for the unknown \( x_1 \) yields

\[
(4.22) \quad x_1 = x_0 - \frac{f(x_0)}{df/dx}
\]

Equation 4.22 can be interpreted as follows: The new estimate of the root, \( x_1 \), is equal to the previous estimate \( x_0 \) minus the ratio of the function at \( x_0 \) to the value of the derivative at \( x_0 \). If the derivative \( df/dx \) is evaluated at \( x_0 \) and we denote this derivative as \( f'(x_0) \), and then Equation 4.22 becomes

\[
(4.23) \quad x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}
\]

Thus Equation 4.23 provides a revised approximation of the root \( x_1 \) using some initial estimate of the root \( x_0 \) and the values of both the function \( f(x_0) \) and the derivative of the function \( f'(x_0) \), both evaluated at \( x_0 \). Equation 4.23 can be iteratively evaluated as
If the function \( f(x) \) is linear, then Equation 4.24 will provide an exact solution on the first trial; however, if \( f(x) \) is nonlinear, then the linear Taylor series of Equation 4.20 is only valid over small ranges of \( \Delta x \). The importance of the nonlinear terms that were truncated for Equation 4.20 will determine the accuracy of the solution of Equations 4.23 and 4.24. In practice, Equations 4.23 and 4.24 are applied iteratively, with the first trial based on an estimate of \( x_0 \) and each subsequent trial based on the most recent estimate of the root. The solution procedure is illustrated in Figure 4.9, where the function and independent variable are denoted as \( f(x) \) and \( x \), respectively. The subscript on \( x \) refers to the trial (or iteration) number, with \( x_0 \) being the initial estimate. It can be seen from Figure 4.9 that Equations 4.23 and 4.24 can be obtained by expressing the slope \( f'(x_i) \) over the interval \([x_i, x_{i+1}]\) as

\[
(4.25) \quad f'(x_i) = \frac{f(x_i) - 0}{x_{i+1} - x_i} = \frac{f(x_i)}{x_{i+1} - x_i}
\]

Therefore, Equations 4.23 and 4.24 can be obtained by rearranging the terms of Equation 4.25.

**Example 4.5: Solution of a Third-Order Polynomial**

As an example of Newton–Raphson iteration, consider the following third-order polynomial:

\[
(4.26) \quad f(x) = x^3 - x^2 - 10x - 8
\]

The derivative of this function is

\[
(4.27) \quad f'(x) = 3x^2 - 2x - 10
\]