TOPIC 2: SOLUTION OF NONLINEAR EQUATIONS

We wish to find values of $x$ which satisfy $f(x) = 0$. We certainly do not need to use a computer to find the solution if $f(x)$ is a first degree polynomial and maybe not if $f(x)$ is a second degree polynomial. But if $f(x)$ is a polynomial of degree 3 or higher, or contains trig functions, logs, exponentials, or other transcendental functions, we usually need a computer to implement a root-finding procedure. We consider a number of procedures here, and ask which of the available procedures can achieve the desired level of
accuracy most quickly, with greatest certainty, and with least trouble starting it. We must first consider what we mean by the statement that an approximate solution \( x = \hat{x} \) to an actual root \( x = r \) is close enough. “Closeness” can be measured by a small residual \( |f(\hat{x})| \approx 0 \) or by the closeness between the approximation and the root (the latter is not known, of course) \( |\hat{x} - r| \approx 0 \). These two criteria are not necessarily small simultaneously. This feature is illustrated in the diagram below, where two functions have about the same
uncertainty in their values, but quite different uncertainties in the location of their roots.

The Bisection Method
In finite-precision arithmetic, there may not be a floating-point number \( x \) such that \( f(x) \) is exactly zero. What we can do, however, is to look for a small interval \([a,b]\) in which \( f \) has a change of sign, since a continuous function \( f \) must then be zero somewhere within such an interval.

The strategy in the bisection method is to begin with two values
of \( x, x_1 \) and \( x_2 \), that enclose a root of \( f(x) = 0 \). It is a requirement of the bisection method that \( f(x) \) be continuous in the interval \([x_1, x_2] \). The bisection method confirms that the values \( x = x_1 \) and \( x = x_2 \) do enclose a root by showing that \( f(x_1) \cdot f(x_2) < 0 \). The method then successively divides the interval \([x_1, x_2] \) in half and replaces one endpoint with the midpoint \( x_3 \) so that again the root is enclosed. The error in the estimate of the root will be less than \( |(x_1 - x_2) \cdot \frac{1}{2^n}| \), where \( n \) is the number of iterations performed.
To get a first approximation to a root, use the graphing capabilities of MATLAB or a graphics calculator. A graphical representation of the bisection method is as follows:

![Bisection Method Diagram]

Pseudocode for bisection method:

```
REPEAT
  SET $x_3 = (x_1 + x_2)/2$
  IF $f(x_3) * f(x_1) < 0$ SET $x_2 = x_3$
  ELSE SET $x_1 = x_3$
UNTIL $|x_1 - x_2| / 2 < $ tolerance
```

Notice that the basis for ending the computations here is a comparison.
between successive approximations for a root. The implication is that this difference can be made as small as desired by performing more iterations. This may not be the case; it may be found that this difference cannot be consistently reduced no matter how many iterations are carried out. This phenomenon is typically caused by the build up of roundoff errors. It is good programming practice to set a limit for the number of times the iteration loop is performed to avoid an unacceptable buildup of roundoff errors.
The table below shows the application of the bisection method to find the root of

\[ x^3 + x^2 - 3x - 3 = 0 \]

between \( x = 1 \) and \( x = 2 \) using a tolerance of \( 10^{-4} \):

<table>
<thead>
<tr>
<th>Iteration</th>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( x_3 )</th>
<th>( F(x_3) )</th>
<th>Maximum error</th>
<th>Actual error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.000000</td>
<td>2.000000</td>
<td>1.500000</td>
<td>-1.875000</td>
<td>0.500000</td>
<td>-0.232051</td>
</tr>
<tr>
<td>2</td>
<td>1.500000</td>
<td>2.000000</td>
<td>1.750000</td>
<td>0.171875</td>
<td>0.250000</td>
<td>0.017949</td>
</tr>
<tr>
<td>3</td>
<td>1.500000</td>
<td>1.750000</td>
<td>1.625000</td>
<td>-0.943359</td>
<td>0.125000</td>
<td>-0.107051</td>
</tr>
<tr>
<td>4</td>
<td>1.625000</td>
<td>1.750000</td>
<td>1.687500</td>
<td>-0.409424</td>
<td>0.062500</td>
<td>-0.044551</td>
</tr>
<tr>
<td>5</td>
<td>1.687500</td>
<td>1.750000</td>
<td>1.718750</td>
<td>-0.124786</td>
<td>0.031250</td>
<td>-0.013301</td>
</tr>
<tr>
<td>6</td>
<td>1.718750</td>
<td>1.750000</td>
<td>1.734375</td>
<td>0.022030</td>
<td>0.015625</td>
<td>0.002324</td>
</tr>
<tr>
<td>7</td>
<td>1.718750</td>
<td>1.734375</td>
<td>1.726563</td>
<td>-0.051756</td>
<td>0.007813</td>
<td>-0.005488</td>
</tr>
<tr>
<td>8</td>
<td>1.726563</td>
<td>1.734375</td>
<td>1.730469</td>
<td>-0.014957</td>
<td>0.003906</td>
<td>-0.001582</td>
</tr>
<tr>
<td>9</td>
<td>1.730469</td>
<td>1.734375</td>
<td>1.732422</td>
<td>0.003512</td>
<td>0.001953</td>
<td>0.000371</td>
</tr>
<tr>
<td>10</td>
<td>1.730469</td>
<td>1.732422</td>
<td>1.731445</td>
<td>-0.005728</td>
<td>0.000977</td>
<td>-0.000605</td>
</tr>
<tr>
<td>11</td>
<td>1.731445</td>
<td>1.732422</td>
<td>1.731934</td>
<td>-0.001109</td>
<td>0.000488</td>
<td>-0.000117</td>
</tr>
<tr>
<td>12</td>
<td>1.731934</td>
<td>1.732422</td>
<td>1.732178</td>
<td>0.001202</td>
<td>0.000244</td>
<td>0.000127</td>
</tr>
<tr>
<td>13</td>
<td>1.731934</td>
<td>1.732178</td>
<td>1.732056</td>
<td>0.000046</td>
<td>0.000122</td>
<td>0.000005</td>
</tr>
</tbody>
</table>

Tolerance met

Note that, even though the error estimates are getting smaller, the magnitudes of the actual errors do not show a continual decrease.
The main advantages of the bisection method are that it is guaranteed to work and that the number of iterations to achieve a specified accuracy is known in advance. The main objections to the bisection method are that it is slow to converge and that it may not be applicable when there are multiple roots since the function may not change sign at points on either side of the roots.

The Secant Method
Suppose that $f(x)$ is continuous, and that (by graphical means, for instance) an approximation $x = x_0$ to a root $x = r$ has been obtained.
We assume that $f(x)$ is linear in the vicinity of the root $x = r$. We choose another point $x = x_1$ which is near to $x = x_0$ and also near $x = r$, and draw a straight line through the points $(x_0, f(x_0))$, $(x_1, f(x_1))$. If $f(x)$ is not linear, this straight line will meet the x-axis at a point $x = x_2$ close to $x = r$.

By similar triangles,

$$\frac{x_1 - x_2}{f(x_1)} = \frac{x_0 - x_1}{f(x_0) - f(x_1)}$$

which leads to
\[ x_2 = x_1 - f(x_1) \cdot \frac{x_0 - x_1}{f(x_0) - f(x_1)} \]

\( x = x_2 \) should be closer to \( x = r \) than either \( x = x_0 \) or \( x = x_1 \).

We can continue to get better estimates of \( x = r \) if we repeat this operation, always choosing the two \( x \)-values closest to \( x = r \) for drawing the straight line. After the second iteration, we always use the last two computed points. After the first iteration, we make sure that \( x = x_1 \) is closer to the root than \( x = x_0 \) by testing \( f(x_0) \) and \( f(x_1) \) and swapping if \( f(x_0) \) is smaller.

The pseudocode for the secant method is:
IF $|f(x_0)| < |f(x_1)|$ swap $x_0$ with $x_1$

REPEAT

SET $x_2 = x_1 - f(x_1)(x_0 - x_1)/[f(x_0) - f(x_1)]$

SET $x_0 = x_1$

SET $x_1 = x_2$

UNTIL $|f(x_2)| <$ tolerance

The table below shows the application of the secant method to find the root of $x^3 + x^2 - 3x - 3 = 0$ between $x = 1$ and $x = 2$ using a tolerance of $10^{-5}$:

<table>
<thead>
<tr>
<th>Iteration</th>
<th>$x_0$</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$F(x_2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1.571429</td>
<td>-1.364432</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1.571429</td>
<td>1.705411</td>
<td>-0.2477449</td>
</tr>
<tr>
<td>3</td>
<td>1.571429</td>
<td>1.705411</td>
<td>1.735136</td>
<td>2.925562E-02</td>
</tr>
<tr>
<td>4</td>
<td>1.705411</td>
<td>1.735136</td>
<td>1.731996</td>
<td>-5.147391E-04</td>
</tr>
<tr>
<td>5</td>
<td>1.735136</td>
<td>1.731996</td>
<td>1.732051</td>
<td>-1.422422E-06</td>
</tr>
</tbody>
</table>

At $x = 1.732051$, tolerance of .00001 met!

The speed of convergence using the secant method is clearly
superior to that obtained with the bisection method. However, if \( f(x) \) is far from linear near the root the secant method may not converge, as the following diagram shows:

![Diagram](image)

The secant method can be adapted for finding complex roots as well as real roots; the initial guesses must be complex numbers.
The Method of False Position
A way to avoid the problem of lack of convergence that may occur using the secant method is to ensure that the root is enclosed between two starting values $x = x_0$ and $x = x_1$ and remains between the succeeding pairs; this gives rise to the method of false position for determining a root of $f(x) = 0$, where $f(x)$ is continuous. The technique is similar to the bisection method except that the next iterate is taken at the intersection of a line between the pair of $x$-values and the $x$-axis rather than at the midpoint. Doing so gives faster convergence than does the
bisection method, but at the expense of a more complicated algorithm. Moreover, the speed of convergence for the method of false position is not as good as for the secant method; note that the method of false position converges to the root from only one side, slowing it down, especially if that end of the interval is further from the root.

The pseudocode for the method of false position is:

```
REPEAT
  SET \( x_2 = x_1 - \frac{f(x_1)(x_0 - x_1)}{f(x_0) - f(x_1)} \)
  IF \( f(x_2)f(x_0) < 0 \) SET \( x_1 = x_2 \)
  ELSE SET \( x_0 = x_2 \)
```
UNTIL $|f(x_2)| < \text{tolerance}$

The table below gives a comparison of three methods for finding the root of

$$f(x) = 3x + \sin x - e^x = 0$$

which lies between $x = 0$ and $x = 1$:

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Interval halving</th>
<th>False position</th>
<th>Secant method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$x$</td>
<td>$f(x)$</td>
<td>$x$</td>
</tr>
<tr>
<td>1</td>
<td>0.5</td>
<td>0.330704</td>
<td>0.470990</td>
</tr>
<tr>
<td>2</td>
<td>0.25</td>
<td>-0.286621</td>
<td>0.372277</td>
</tr>
<tr>
<td>3</td>
<td>0.375</td>
<td>0.036281</td>
<td>0.361598</td>
</tr>
<tr>
<td>4</td>
<td>0.3125</td>
<td>-0.121899</td>
<td>0.360538</td>
</tr>
<tr>
<td>5</td>
<td>0.34375</td>
<td>0.041956</td>
<td>0.360433</td>
</tr>
</tbody>
</table>

Error after 5 iterations: 0.01667 \quad -1.17 * $10^{-5}$ \quad < -1 * $10^{-7}$

(Exact value of root is 0.360421703.)

**Newton’s Method**

Like the secant method and the method of false position, Newton’s
Method is based on a linear approximation of the continuous function $f(x)$, but does so using a tangent to a curve. A single estimate $x = x_0$ to the root is taken, and a tangent to the curve is drawn at the point $(x_0, f(x_0))$ which intersects the x-axis at the point $x = x_1$. This point is taken as the next approximation to the root, a tangent is drawn to the curve at the point $(x_1, f(x_1))$, which intersects the x-axis at $x = x_2$, and this is taken as the next iterate, and so on.
From the diagram
\[ \tan \theta = f'(x_0) = \frac{f(x_0)}{x_0 - x_1} \]
which leads to \( x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \)

Similarly \( x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} \)

Generally \( x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \), \( n = 0,1,2,... \)

Newton's algorithm is popular, because if it converges, it converges more rapidly than any of the other methods we deal with. Newton's method is quadratically convergent, which means that the
error at each step approaches proportionality to the square of the error at the previous step as the number of steps increases, so that effectively the number of decimal places of accuracy almost doubles at each step. For example, if Newton’s method is applied to 
\[ f(x) = 3x + \sin x - e^x = 0 \] with \( x_0 = 0 \), then 
\[ x_1 = 0.33333 \]
\[ x_2 = 0.36017 \]
\[ x_3 = 0.3604217 \]
which is correct to 7D. Note that the method of false position is linearly convergent, and that the secant method exhibits better than linear convergence but
poorer than quadratic convergence.
Symbolically, if we denote the error at iteration $k$ by $e_k$, then
\[
\frac{|e_{k+1}|}{|e_k|^r} = c, \quad \text{for large } k
\]
where $c$ is a constant,
r = 1, for linear convergence
  (False Position)
r = 1+, for superlinear convergence
  (Secant)
r = 2, for quadratic convergence
  (Newton)
Newton’s method does have the disadvantage that $f'(x)$ needs to be known (and may be difficult to
determine), and that at each step, \( f(x) \) and \( f'(x) \) must both be evaluated. Compare this with the previous methods, where (apart from the first step, which requires two function evaluations) each step requires just one function evaluation.
The pseudocode for Newton’s method is:

\[
\text{REPEAT}
\]

\[
\begin{align*}
\text{SET } x_1 &= x_0 \\
\text{SET } x_0 &= x_0 - \frac{f(x_0)}{f'(x_0)}
\end{align*}
\]

\[
\text{UNTIL } |f(x_0)| < \text{tolerance}
\]

In some cases, Newton’s method will not converge. The diagram below illustrates one such possible situation, where starting with \( x = x_0 \)
one never reaches the root $r$ because $x_6 = x_1$ and our calculation produces an endless loop.

Note also that if we should ever have an $x$-value corresponding to the minimum or maximum of the curve $f(x)$, the tangent will fly off to infinity. In fact, a sufficient condition for Newton's method to converge, for
any initial value $x_0$ in an interval about the root $r$, is that
\[
\frac{f(x) \cdot f''(x)}{[f'(x)]^2} < 1
\]
Newton’s method can be adapted for finding complex roots as well as real roots; the initial estimates of the roots must be complex numbers.
Note, finally, that Newton’s Method converges only linearly to a double root. Consider the equation
\[
(x - 1) \cdot (e^{x-1} - 1) = 0
\]
which has a double root at $x = 1$. Using a starting value of $x = 2$, the table below shows the linear convergence – each error is about
one-half of the preceding error, especially as we get near the root.

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Error</th>
<th>Iteration</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.0</td>
<td>6</td>
<td>0.0199</td>
</tr>
<tr>
<td>1</td>
<td>0.5</td>
<td>7</td>
<td>0.0100</td>
</tr>
<tr>
<td>2</td>
<td>0.2798</td>
<td>8</td>
<td>0.0050</td>
</tr>
<tr>
<td>3</td>
<td>0.1494</td>
<td>9</td>
<td>0.0025</td>
</tr>
<tr>
<td>4</td>
<td>0.0775</td>
<td>10</td>
<td>0.00125</td>
</tr>
<tr>
<td>5</td>
<td>0.0395</td>
<td>11</td>
<td>0.000625</td>
</tr>
</tbody>
</table>

Successive errors with Newton's method, for \( f(x) = (x - 1) \cdot (e^{x-1} - 1) \)
The Fixed-Point Iteration Method
Given a function $g: \mathbb{R} \rightarrow \mathbb{R}$, a value $x$ such that $x = g(x)$ is called a fixed point of the function $g$, since $x$ is unchanged when $g$ is applied to it. In the fixed-point iteration method, the equation $f(x) = 0$ is rearranged into an equivalent form $x = g(x)$, which can usually be done in a number of ways. If $r$ is a root, then $f(r) = 0$ and $r = g(r)$; $r$ is said to be a fixed point for the function $g$. Under suitable conditions, the form $x_{n+1} = g(x_n)$, $n = 0, 1, 2, \ldots$ converges to the fixed point $r$, that is, to a root of $f(x) = 0$.
Consider the simple example $f(x) = x^2 - 2x - 3 = 0$
(for which the roots are obviously \( x = -1, \ x = 3 \))

Let us rearrange this to the form
\[
x = g_1(x) = \sqrt{2x + 3}
\]

Starting with \( x_0 = 4 \) leads to
\[
x_1 = \sqrt{11} = 3.31662
\]
\[
x_2 = \sqrt{9.63325} = 3.10375
\]
\[
x_3 = \sqrt{9.20750} = 3.03439
\]
\[
x_4 = \sqrt{9.06877} = 3.01144
\]
\[
x_5 = \sqrt{9.02288} = 3.00381
\]

and clearly there is convergence to the root \( x = 3 \).

An alternative rearrangement of \( f(x) = 0 \) is
\[
x = g_2(x) = \frac{3}{x - 2}
\]

Again starting with \( x_0 = 4 \) leads to
\[ x_1 = 1.5 \]
\[ x_2 = -6 \]
\[ x_3 = -0.375 \]
\[ x_4 = -1.26316 \]
\[ x_5 = -0.919355 \]
\[ x_6 = -1.02762 \]
\[ x_7 = -0.990876 \]
\[ x_8 = -1.00305 \]

and clearly there is convergence to the root \( x = -1 \). Note that the convergence is oscillatory here; it was monotonic in the first case.

A third rearrangement of \( f(x) = 0 \) is

\[ x = g_3(x) = \frac{x^2 - 3}{2} \]

Again starting with \( x_0 = 4 \) leads to
\[ x_1 = 6.5, x_2 = 19.625, x_3 = 191.070 \]

and clearly there is divergence.
The difference in behaviour of these rearrangements can be viewed graphically. The fixed point of $x = g(x)$ is the intersection of the line $y = x$ and the curve $y = g(x)$. To obtain the successive iterates, we use the following construction: Start on the $x$-axis at the initial guess $x = x_0$, then go vertically to the curve, then horizontally to the line $y = x$, then vertically to the curve, then horizontally to the line $y = x$, and so on. It is clear that the different behaviours depend on whether the slope of the curve is greater than, less than, or of opposite sign to the slope of the line $y = x$, which of course is 1.
\( g_1(x) = \sqrt{2x+3} \)

\( g_2(x) = \frac{3}{x-2} \)

\( g_3(x) = \frac{x^2-3}{2} \)

(c)
It can be shown that, if $g(x)$ and $g'(x)$ are continuous on an interval about a root $r$ of the equation $x = g(x)$ and if $|g'(x)| < 1$ for all $x$ in the interval, then $x_{n+1} = g(x_n)$, $n = 0, 1, 2, \ldots$ will converge to the root $r$. (Note that this is a sufficient condition; convergence is secured in some equations even though not all of these conditions hold.) Referring to the three graphs above: in the first case, the slope of the curve, $g'(x)$, is positive and less than 1, and there is monotonic convergence. In the second case, the slope of the curve is negative and less than 1 in magnitude, and there is oscillatory convergence. In the third case, the
slope of the curve is greater than 1, and the iterates diverge. The fixed-point iteration method exhibits linear convergence. It is possible to accelerate convergence as follows (the Aitken acceleration technique). Consider the starting value $x_0$ and the first two iterates $x_1$ and $x_2$. Set

$D1 = x_0 - x_1$

$D2 = x_2 - 2x_1 + x_0$

and compute

$X = x_0 - \frac{D1^2}{D2} = \frac{x_0 x_2 - x_1^2}{x_2 - 2x_1 + x_0}$

Then $X$ will be closer to the root than if we were to apply fixed-point iteration a third time to obtain $x_3$. 
Applying the fixed-point iteration method with $X$, and then iterating two more times, we can use these last three values to accelerate again. And so on.

To illustrate, consider the first of our previous examples, where

$x_0 = 4,$
$x_1 = 3.31662,$
$x_2 = 3.10375.$

Then:

$D1 = 4 - 3.31662$
$ = 0.68338$

$D2 = 3.10375 - 2*3.31662 + 4$
$ = 0.47051$

$X = 4 - \frac{0.68338^2}{0.47051} = 3.00744$
And this is closer to the root than is $x_3$ (and, indeed, $x_4$) i.e. we have jumped two iterations.

Pseudocode for the fixed-point iteration method without acceleration:

```
REPEAT
    SET $x_2 = x_1$
    SET $x_1 = g(x_1)$
UNTIL $|x_1 - x_2| < $ tolerance
```

Hybrid Methods

Rapidly converging methods for solving non-linear equations, such as Newton’s method and the secant method, are unsafe in that they may not converge unless they are started close enough to the solution. Safe methods, such as the bisection method, are slow. Which should one choose?
A solution to this dilemma is provided by hybrid methods that combine the best features of both types of methods. For example, one could use a rapidly convergent method but maintain a bracket around the solution. If a particular iterate given by the rapid algorithm falls outside the bracketing interval, one would fall back on a safe method for one iteration. Then one can try the rapid method again on a smaller interval with a greater chance of success. Ultimately, the fast method should prevail. This approach, developed by Brent, seldom does worse than the slow method and usually does much better.