CHAPTER 1

Numerical Solution Of Nonlinear Algebraic Equations

1. Introduction.

The general form of a nonlinear equation is \( f(x) = 0 \), where \( f \) is a nonlinear function of the variable \( x \) e.g.:

\[
\begin{align*}
x^4 + x^3 + 1 &= 0 \\
x e^{-x} &= 7 \text{ or } x e^{-x} - 7 = 0 \\
\log x &= x \text{ or } \log x - x = 0
\end{align*}
\]

Solutions of the equation \( f(x) = 0 \)—i.e. any \( \alpha \) such that \( f(\alpha) = 0 \)—are called roots of the equation or zeroes of the function. Some nonlinear equations can be solved analytically; there is a well-known formula for the roots of a quadratic equation, and formulae exist—though are seldom used—for cubic and even quartic equations. However, in general no such formulae exist and the roots must be found using some numerical method.

**Definition.** \( \alpha \) is a root of multiplicity \( m \) of the equation \( f(x) = 0 \) if \( f(x) = (x - \alpha)^m g(x) \), where \( g(x) \) is continuous in a neighbourhood of \( \alpha \) and \( g(\alpha) \neq 0 \). If \( m \) is odd the root is said to be odd; if \( m \) is even, the root is even.

**Example.** \( f(x) = x^3 - 2x^2 + x = 0 \). In this case \( f(x) \) can easily be factorised as \( f(x) = (x - 1)^2 x; \) so, \( f(x) \) has a zero of multiplicity 2 at \( x = 1 \), and a zero of multiplicity 1—also called a simple root—at \( x = 0 \). (Note that if we write \( f(x) = (x - 1)^3 \frac{x}{x - 1} \) then \( g(x) = \frac{x}{x - 1} \) has a discontinuity at \( x = 1 \); so the root is not of multiplicity 3).

In general, the numerical methods, which must be used to approximate the roots fall into two types: Starting methods; Bisection method; Regula Falsi; methods which are guaranteed to work; but which are either slow or inaccurate—or both—and Higher order methods: Newton’s method; Secant method.

If possible, it is worthwhile simply to plot or tabulate the function \( f(x) \); this often gives an indication of the location of the root(s).

**Example.** \( f(x) = (\frac{x}{2})^2 - \sin x = 0 \). If we tabulate some values of \( f(x) \), we have

<table>
<thead>
<tr>
<th>( x )</th>
<th>((\frac{x}{2})^2)</th>
<th>( \sin x )</th>
<th>( f(x) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.6</td>
<td>0.64</td>
<td>0.996</td>
<td>&lt; 0</td>
</tr>
<tr>
<td>1.8</td>
<td>0.81</td>
<td>0.976</td>
<td>&lt; 0</td>
</tr>
<tr>
<td>2.0</td>
<td>1.00</td>
<td>0.909</td>
<td>&gt; 0</td>
</tr>
</tbody>
</table>

The sign change indicates a root between 1.8 and 2.0—since the function is continuous.

**Bisection method.** This only works for roots of odd multiplicity; i.e. where the graph of \( f(x) \) cuts the \( x \)-axis.

A sequence of decreasing subintervals \( I_k = (a_k, b_k) \) each containing the root \( \alpha \) is generated as follows: Assume \( f(a_0) \) and \( f(b_0) \) are of opposite sign, i.e.

\[
f(a_0) f(b_0) < 0 \quad \alpha \in I_0 = (a_0, b_0)
\]

For \( k = 1, 2, 3, \ldots \) \( I_k \) is obtained from \( I_{k-1} \) as follows:

\[
m_k = \frac{a_k + b_k}{2} \quad \text{(midpoint of } I_{k-1})
\]

if \( f(m_k) = 0 \) then \( m_k = \alpha \)

else if \( f(m_k) f(a_{k-1}) > 0 \) \( I_k = (m_k, b_{k-1}) \)

else \( I_k = (a_{k-1}, m_k) \)

Note that in either case \( f(a_k) f(b_k) < 0 \); so \( \alpha \in I_k \).
**Bisection subroutine.**

```fortran
subroutine bisect(f,a0,b0,tol,status,root,resid,a,b)
    integer status
    real f,fa,fb,fm,a0,b0,a,b,resid,root,tol,m
    external f
    c-----------------------------------------------------------------------
c On input a0 and b0 are the endpoints of the initial interval; f is the
c function defined elsewhere; tol is the accuracy required; status is 0
c on entry and 0 at end if root is found, otherwise it is 1;
c resid =f(root); and (a,b) is the final interval.
c------------------------------------------------------------------------
a=a0
b=b0
fa=f(a)
fb=f(b)

if(fa*fb.gt.0.) then
    status=1
    c
    c The interval does not appear to bracket a root.
c
    else
    c
    c BISECTION METHOD
    c
    do while (abs(b-a).gt.tol)
        m=(a+b)/2.
        fm=f(m)
        c
        c If the root has been located exactly
        c
        if(abs(fm).lt.1.E-8) then
            a=m
            b=m
        else if(fa*fm.lt.0.) then
            b=m
        else
            a=m
            fa=fm
        endif
    endif
end do

root=(a+b)/2.
resid=f(root)
endif
end
```
Example. \( f(x) = \left( \frac{x}{2} \right)^2 - \sin x = 0 \).

Here, if we take \( a_0 = 1.5, \ b_0 = 2 \), we have \( f(1.5) < 0 \) and \( f(2) > 0 \); so the root is between 1.5 and 2.

<table>
<thead>
<tr>
<th>( k )</th>
<th>( a_{k-1} )</th>
<th>( b_{k-1} )</th>
<th>( b_{k-1} - a_{k-1} )</th>
<th>( m_k )</th>
<th>( f(m_k) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.5</td>
<td>2</td>
<td>0.5</td>
<td>1.75</td>
<td>&lt; 0</td>
</tr>
<tr>
<td>2</td>
<td>1.75</td>
<td>2</td>
<td>0.25</td>
<td>1.875</td>
<td>&lt; 0</td>
</tr>
<tr>
<td>3</td>
<td>1.875</td>
<td>2</td>
<td>0.125</td>
<td>1.9375</td>
<td>&gt; 0</td>
</tr>
<tr>
<td>4</td>
<td>1.875</td>
<td>1.9375</td>
<td>0.0625</td>
<td>1.90625</td>
<td>&lt; 0</td>
</tr>
<tr>
<td>5</td>
<td>1.90625</td>
<td>1.9375</td>
<td>0.03125</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

So, \( \alpha = 1.90625 \pm 0.03125 \). Note that the convergence is very slow. After \( n \) steps, the root is contained in an interval \( (a_n, b_n) \) of length \( b_n - a_n = \frac{1}{2} (b_{n-1} - a_{n-1}) = \ldots = \frac{1}{2^n} (b_0 - a_0) \). Taking \( m_{n+1} \) as an estimate of \( \alpha \),

\[
\alpha = m_{n+1} \pm \frac{1}{2^{n+1}} (b_0 - a_0)
\]

So the error at step \( n + 1 \) is half the error at step \( n \).

**Advantages:**
1. It always converges—to odd roots.
2. Error bounds are easily obtained.
3. Convergence is unaffected by the multiplicity of the root.

**Disadvantages:**
1. Slow to converge.
2. Skips even roots.

We will consider the Regula Falsi method later.

2. **Newton's Method.**

If the point \( x_0 \) is close to the root \( \alpha \), then the tangent line to the graph of \( f(x) \) at \( x_0 \) is a good approximation to \( f(x) \) near \( \alpha \). So, the root of the tangent line—where the line cuts the \( x \)-axis—\( x_1 \) is a better approximation to \( \alpha \) than \( x_0 \) is.

The slope of the tangent \( = \frac{f(x_0)}{x_0 - x_1} = f'(x_0) \). So

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

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

Repeating the process, we obtain a better approximation

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

and, in general, for \( n = 0, 1, 2, \ldots \),

\[
x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}
\]

If \( x_0 \) is sufficiently close to \( \alpha \), \( x_n \to \alpha \) as \( n \to \infty \).

**Newton's method subroutine**

```fortran
subroutine newton(f, df, x0, tol, itmax, status, root, resid, noits, xlast)
    integer count, status, itmax, noits
    real epsilon, x, fx, dfx, dx, tol, root, resid, xlast
    parameter(epsilon=1.E-10)
    external f, df
    x=x0
    status=-1
    count=0
    do while (status.eq.-1.and.count.le.itmax)
        dfx=df(x)
        if(abs(dfx).le.epsilon) then
            print *,'Derivative is almost zero, so graph is flat'
        else
            x=x - f(x)/df(x)
        endif
        count = count + 1
        if(count.le.itmax) then
            status = 0
            resid = f(x)
        endif
    enddo
end
```

```fortran
if status .ne. 0
    root = x
    resid = f(x)
    noits = itmax
    xlast = x
end if
```
2. NEWTON’S METHOD

```fortran
status=1
else

    \text{NEWTON’S METHOD}

    \begin{verbatim}
    fx=f(x)
dx=-fx/dfx
x=x+dx
    \end{verbatim}

    if (abs(dx).le.abs(x)*tol) status=0

    root has been found
    count=count+1
end do

if(status.eq.0) then
    noits=count
    root=x
    resid=f(root)
else
    xlast=x
endif
end
```

2.1. Order of convergence. Let \( x_0, x_1, x_2, \ldots, x_n, \ldots \) be a sequence of approximations to a root \( \alpha \) produced by a numerical method, where \( \lim_{n \to \infty} x_n = \alpha \). Let \( \varepsilon_n = \alpha - x_n \). If \( \lim_{n \to \infty} \frac{\lvert \varepsilon_{n+1} \rvert}{\lvert \varepsilon_n \rvert^p} = C \), for some \( p \) and some non-zero constant \( C \), then the method has order of convergence \( p \), and \( C \) is the asymptotic error constant. (The bigger \( p \) is the faster the convergence).

If we now consider Newton’s method, where \( f(x) \) is assumed to be twice differentiable, and \( \alpha \) is a simple root, then:

\[
f(\alpha) = f(x_n) + (\alpha - x_n)f'(x_n) + \frac{(\alpha - x_n)^2}{2} f''(\xi) = 0
\]

where \( \xi \) is an unknown point between \( x_n \) and \( \alpha \); i.e.

\[
\begin{align*}
\alpha - x_n + \frac{f(x_n)}{f'(x_n)} &= -\frac{(\alpha - x_n)^2 f''(\xi)}{2f'(x_n)} \\
\Rightarrow \alpha - x_{n+1} &= -\frac{(\alpha - x_n)^2 f''(\xi)}{2f'(x_n)} \\
\Rightarrow |\varepsilon_{n+1}| &= \frac{|f''(\xi)|}{2|f'(x_n)|} \\
\Rightarrow \lim_{n \to \infty} \frac{|\varepsilon_{n+1}|}{|\varepsilon_n|^2} &= \frac{|f''(\alpha)|}{2|f'(\alpha)|}
\end{align*}
\]

if \( x_n \to \alpha \) as \( n \to \infty \).

Thus, in the case of a simple root \( \alpha \), Newton’s method has second order—quadratic—convergence, with asymptotic error constant \( \frac{|f''(\alpha)|}{2|f'(\alpha)|} \).

**Note.** If \( \alpha \) is a root of multiplicity \( m \), then

\[
\begin{align*}
f(x) &= (x - \alpha)^m g(x) \quad g(\alpha) \neq 0 \\
\Rightarrow f'(x) &= (x - \alpha)^m g'(x) + m(x - \alpha)^{m-1} g(x) \\
&= (x - \alpha)^{m-1} [g'(x) + mg(x)]
\end{align*}
\]

So, \( f'(\alpha) = 0 \) unless \( m = 1 \), in which case \( f'(\alpha) = g(\alpha) \neq 0 \). Hence the requirement that \( \alpha \) be simple for the method to be quadratically convergent.

Otherwise, if \( \alpha \) is not simple,
This method does not require calculation of which will be quadratically convergent to a root of any multiplicity; it is more unwieldy than Newton’s method and

\[ \lim_{n \to \infty} x_n = \alpha. \]

So, the order of convergence for multiple roots is only 1—linear—and as \( m \), the multiplicity of the root, gets bigger, the convergence slows down since the asymptotic error constant \( \frac{m-1}{m} \to 1. \)

However, the function \( q(x) = \frac{f(x)}{f'(x)} \) always has simple roots:

\[ q(x) = \frac{(x - \alpha)g(x)}{(x - \alpha)g'(x) + mg(x)} \]

So \( q(\alpha) = 0 \), and also:

\[ q'(x) = \frac{[(x - \alpha)g'(x) + mg(x)][(x - \alpha)g'(x) + g(x)] - (x - \alpha)g(x)[(x - \alpha)g''(x) + (m + 1)g'(x)]}{[(x - \alpha)g'(x) + mg(x)]^2} \]

Therefore, \( q'(\alpha) = \frac{mg(\alpha)^2}{m^2g(\alpha)^2} = \frac{1}{m} \neq 0. \) Hence, Newton’s method applied to \( q(x) \) will be quadratically convergent:

\[
\begin{align*}
x_{n+1} &= x_n - \frac{q(x_n)}{q'(x_n)} \\
q'(x) &= \frac{f''(x) - f(x) f'''(x)}{f'(x)^2} \\
\Rightarrow q(x) &= \frac{f(x) f'(x)}{f'(x)^2 - f(x) f'''(x)}
\end{align*}
\]

Thus, we have the modified Newton method:

\[ x_{n+1} = x_n - \frac{f(x_n)f'(x_n)}{f'(x_n)^2 - f(x_n) f'''(x_n)} \]

which will be quadratically convergent to a root of any multiplicity; it is more unwieldy than Newton’s method and requires calculation of \( f'''(x_n) \).

Alternatively, note that if \( f(x) = k(x - \alpha)^m \) for some constant \( k \), then \( f'(x) = km(x - \alpha)^{m-1} \); and so:

\[ \frac{f(x)}{f'(x)} = \frac{(x - \alpha)}{m} \]

So, \( \alpha = x - m \frac{f(x)}{f'(x)} \) for any \( x. \)

In the general case of a multiple root, this becomes an approximation rather than an equality, and we have another modified Newton method:

\[ x_{n+1} = x_n - m \frac{f(x_n)}{f'(x_n)} \]

This method does not require calculation of \( f'''(x_n) \); but it does require calculation of the multiplicity \( m \) from

\[ \frac{\left| \varepsilon_{n+1} \right|}{\varepsilon_n} \approx \frac{m - 1}{m} \]

where \( \varepsilon_n \) and \( \varepsilon_{n+1} \) are the errors in Newton’s method.

**Example.** Newton’s method.
\[ f(x) = x - \cos x \quad (\alpha = 0.739085) \]
\[ f'(x) = 1 + \sin x \]
\[ x_{n+1} = x_n - \frac{x_n - \cos x_n}{1 + \sin x_n} \]
\[ = \frac{x_n \sin x_n + \cos x_n}{1 + \sin x_n} \]

<table>
<thead>
<tr>
<th>(n)</th>
<th>(x_n)</th>
<th>(\varepsilon_n = \alpha - x_n)</th>
<th>(x_{n+1} - x_n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0.739085</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>-0.260915</td>
<td>-0.249636</td>
</tr>
<tr>
<td>2</td>
<td>0.750364</td>
<td>-0.011279</td>
<td>-0.011251</td>
</tr>
<tr>
<td>3</td>
<td>0.739113</td>
<td>-0.000028</td>
<td>-0.000028</td>
</tr>
<tr>
<td>4</td>
<td>0.739085</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0.739085</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

In this case
\[ \frac{|\varepsilon_1|}{|\varepsilon_0|^2} = 0.47765 \]
\[ \frac{|\varepsilon_2|}{|\varepsilon_1|^2} = 0.165681 \]
\[ \frac{|\varepsilon_3|}{|\varepsilon_2|^2} = 0.220098 \]

Note that \(x_{n+1} - x_n\) is a good approximation for \(\alpha - x_n\) as \(x_n\) approaches \(\alpha\).

**Example.**
\[ f(x) = x^2 - 4x + 4 = 0 \quad \alpha = 2 \quad m = 2 \]
\[ f'(x) = 2x - 4 \]

Newton’s method:
\[ x_{n+1} = x_n - \frac{x_n^2 - 4x_n + 4}{2x_n - 4} = \frac{x_n^2 - 4}{2x_n - 4} = \frac{x_n + 2}{2}. \]

<table>
<thead>
<tr>
<th>(n)</th>
<th>(x_n)</th>
<th>(\varepsilon_n = \alpha - x_n)</th>
<th>(x_{n+1} - x_n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0.5</td>
</tr>
<tr>
<td>1</td>
<td>1.5</td>
<td>0.5</td>
<td>0.25</td>
</tr>
<tr>
<td>2</td>
<td>1.75</td>
<td>0.25</td>
<td>0.125</td>
</tr>
<tr>
<td>3</td>
<td>1.875</td>
<td>0.125</td>
<td>0.0625</td>
</tr>
<tr>
<td>4</td>
<td>1.9375</td>
<td>0.0625</td>
<td></td>
</tr>
</tbody>
</table>

Here,
\[ \frac{|\varepsilon_{n+1}|}{|\varepsilon_n|} = \frac{|\alpha - x_{n+1}|}{|\alpha - x_n|} = \frac{1}{2} = \frac{|x_{n+1} - x_n|}{|x_n - x_{n-1}|} \]

for all \(n\). Thus convergence is only linear:
\[ \frac{m - 1}{m} = \frac{1}{2} \Rightarrow 2m - 2 = m \Rightarrow m = 2 \]

So an alternative method should be used. Using the first modified method:
\[ f(x) = x^2 - 4x + 4 \]
\[ f'(x) = 2(x - 2) \]
\[ f''(x) = 2 \]
\[ \Rightarrow x_{n+1} = x_n - \frac{2(x_n - 2)^3}{4(x_n - 2)^2 - 2(x_n - 2)^2} = x_n - (x_n - 2) = 2 \]

And, using the second method,
\[ x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \]
This was a particularly easy problem for each of these two methods; so we consider a more difficult one:

**Example.**

\[ f(x) = x^4 - 2x^3 + 2x - 1 = 0 \quad \alpha = 1 \]

\[ f'(x) = 4x^3 - 6x^2 + 2 \]

Using Newton’s method again:

\[
\begin{align*}
x_{n+1} &= x_n - \frac{f(x_n)}{f'(x_n)} \\
&= x_n - \frac{x_n^4 - 2x_n^3 + 2x_n - 1}{4x_n^3 - 6x_n^2 + 2} \\
&= \frac{3x_n^4 - 4x_n^3 + 1}{4x_n^3 - 6x_n^2 + 2}
\end{align*}
\]

\[
\begin{array}{|c|c|c|c|c|}
\hline
n & x_n & \varepsilon_n = \alpha - x_n & x_{n+1} - x_n & \frac{|\alpha - x_{n+1}|}{|\alpha - x_n|} \\
\hline
0 & 0 & 1 & 0.5 & 0.375 \\
1 & 0.5 & 0 & 0.1875 & 0.625 \\
2 & 0.6875 & 0.3125 & 0.11102 & 0.64474 \\
3 & 0.79852 & 0.20148 & 0.06976 & 0.65376 \\
4 & 0.86828 & 0.13172 & 0.04497 & 0.65859 \\
5 & 0.91325 & 0.08675 & 0.02936 & 0.66156 \\
6 & 0.94261 & 0.05739 & & \\
\hline
\end{array}
\]

Here \( \frac{m - 1}{m} \approx \frac{2}{3} \Rightarrow 3m - 3 = 2m \Rightarrow m = 3 \). Since the root is of multiplicity 3, we use the modified method:

\[
\begin{align*}
x_{n+1} &= x_n - \frac{3f(x_n)}{f'(x_n)} \\
&= x_n - \frac{3(x_n^4 - 2x_n^3 + 2x_n - 1)}{4x_n^3 - 6x_n^2 + 2} \\
&= \frac{x_n^4 - 4x_n^3 + 3}{4x_n^3 - 6x_n^2 + 2}
\end{align*}
\]

and using this method, we have:

\[
\begin{array}{|c|c|}
\hline
n & x_n \\
\hline
0 & 0 \\
1 & 1.5 \\
2 & 1.03125 \\
3 & 1.00016 \\
4 & 0.99993 \\
5 & 0.99961 \\
6 & 1.00006 \\
7 & 1.00000 \\
\hline
\end{array}
\]

So, there is a root of multiplicity 3 at \( x = 1 \). In practice, Newton’s method would be terminated if \( \frac{|x_{n+2} - x_{n+1}|}{|x_{n+1} - x_n|} \) approached a non-zero limit—after a preset number of steps—and whatever the most recent value of \( x_n \) was would be used as an initial guess for the modified method.

**3. Secant Method.**

The secant method may be regarded as an approximation to Newton’s method where, instead of \( \frac{f'(x_n)}{f'(x_n)} \), the quotient

\[
\frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}
\]
is used; i.e. instead of the tangent to the graph of \( f(x) \) at \( x_n \), we use the secant joining the points \( (x_{n-1}, f(x_{n-1})) \) and \( (x_n, f(x_n)) \).

If we equate the two expressions for the slope of the secant:

\[
\frac{f(x_n)}{x_n - x_{n+1}} = \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}
\]

\[
\Rightarrow x_{n+1} = x_n - \frac{(x_n - x_{n-1})f(x_n)}{f(x_n) - f(x_{n-1})}
\]

It can be shown that the convergence rate of the secant method is 1.618—i.e. almost quadratic—if \( \alpha \) is a simple root. The advantage of the method is that \( f'(x) \) need not be evaluated; so the number of function evaluations is half that of Newton’s method. However, its convergence rate is slightly less than Newton and it requires two initial guesses \( x_0 \) and \( x_1 \).

In common with Newton’s method, its convergence rate deteriorates for multiple roots, and if \( m \) can be estimated, the modification

\[
x_{n+1} = x_n - m \frac{(x_n - x_{n-1})f(x_n)}{f(x_n) - f(x_{n-1})}
\]

should be used.

Also in common with Newton’s method, convergence is not guaranteed at all unless \( x_0 \)—and in the case of the secant method \( x_1 \) also—is a sufficiently good initial guess. For this reason a starting method such as the bisection method or Regula Falsi is used—where applicable—for a few iterations to obtain a good initial guess for one of the more accurate methods.

4. Regula Falsi.

This is a variation of the bisection method, where \( m_{k+1} \), instead of being the the midpoint of \( I_k = (a_k, b_k) \) is the point of intersection with the \( x \)-axis of the secant joining \( (a_k, f(a_k)) \) and \( b_k, f(b_k) \).

Here,

\[
m_{k+1} = \frac{a_k f(b_k) - b_k f(a_k)}{f(b_k) - f(a_k)}
\]

Like the bisection method, it will only work for odd roots, and has linear convergence rate.

5. Fixed-Point Methods.

A nonlinear equation \( f(x) = 0 \) can also be expressed in the form \( x = \phi(x) \) (in an infinite number of ways). A root \( \alpha \) of \( f(x) = 0 \) is also a fixed point of \( x = \phi(x) \) i.e. \( \alpha = \phi(\alpha) \). So the iteration procedure \( x_{n+1} = \phi(x_n) \) converges to \( \alpha \) under certain conditions.

Example. \( x^3 - 2 = 0 \) can be written as:

\[
\begin{align*}
(5.1a) & \quad x &= x^3 + x - 2 \\
(5.1b) & \quad x &= \frac{2 + 5x - x^3}{5}
\end{align*}
\]

Taking \( x_0 = 1.2 \), we have:

<table>
<thead>
<tr>
<th>( n )</th>
<th>(a)</th>
<th>(b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.928</td>
<td>1.2544</td>
</tr>
<tr>
<td>2</td>
<td>-0.273</td>
<td>1.2596</td>
</tr>
<tr>
<td>3</td>
<td>-2.293</td>
<td>1.2599</td>
</tr>
<tr>
<td>4</td>
<td>-16.349</td>
<td>1.25992</td>
</tr>
</tbody>
</table>

Thus (b) gives the correct root, while (a) does not converge. We can see under what conditions the method converges:

\[
\begin{align*}
x_1 &= \phi(x_0) \\
\Rightarrow \alpha - x_1 &= \alpha - \phi(x_0) = \phi(\alpha) - \phi(x_0) \\
&= \phi'(\xi_0)(\alpha - x_0), \quad x_0 \leq \xi_0 \leq \alpha \\
\alpha - x_2 &= \phi'(\xi_1)(\alpha - x_1) \\
&= \phi'(\xi_0)\phi'(\xi_1)(\alpha - x_0), \quad x_1 \leq \xi_1 \leq \alpha \\
\Rightarrow \alpha - x_n &= \phi'(\xi_0)\phi'(\xi_1)\ldots\phi'(\xi_{n-1})(\alpha - x_0)
\end{align*}
\]

So, if \( |\phi'(\xi_k)| \leq M \) for all \( k \), then \( |x_n| \leq M^n|x_0| \) and convergence is assured if \( M < 1 \), i.e. if \( |\phi'(x)| < 1 \) in a neighbourhood containing both \( \alpha \) and \( x_0 \); this condition dictates the version of the method which is to be used. Convergence is usually only linear, however, so Newton type methods are preferable if practicable.
Fixed point methods usually have linear convergence rate, since

\[ x_{n+1} - \alpha = \phi(x_n) - \phi(\alpha) = \phi'(\xi)(x_n - \alpha) \]

\[ \Rightarrow \lim_{n \to \infty} \frac{|\varepsilon_{n+1}|}{|\varepsilon_n|} = |\phi'(\alpha)| \]

if \( x_n \to \alpha \) as \( n \to \infty \). However, if \( \phi(x) \) is chosen so that \( \phi'(\alpha) = \phi''(\alpha) = \ldots = \phi^{(p-1)}(\alpha) = 0 \)

and \( \phi^{(p)} \) is continuous with \( \phi^{(p)}(\alpha) \neq 0 \), then

\[ x_{n+1} - \alpha = \phi(x_n) - \phi(\alpha) = \phi(\alpha) + (x_n - \alpha)\phi'(\alpha) + \frac{(x_n - \alpha)^2}{2}\phi''(\alpha) + \ldots + \frac{(x_n - \alpha)^{p-1}}{(p-1)!}\phi^{(p-1)}(\alpha) + \frac{(x_n - \alpha)^p}{p!}\phi^{(p)}(\xi) - \phi(\alpha) \]

\[ \Rightarrow \lim_{n \to \infty} \frac{|\varepsilon_{n+1}|}{|\varepsilon_n|^p} = \frac{|\phi^{(p)}(\alpha)|}{p!} \]

i.e. the convergence rate is \( p \), and the asymptotic error constant is \( \frac{|\phi^{(p)}(\alpha)|}{p!} \).


A generalisation of the situation we have been considering is the solution of a system of nonlinear algebraic equations; i.e.

\[ f(x) = 0 \]

where \( x \) and \( f \) are \( n \)-vectors

\[ x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \quad \text{and} \quad f(x) = \begin{pmatrix} f_1(x_1, x_2, \ldots, x_n) \\ f_2(x_1, x_2, \ldots, x_n) \\ \vdots \\ f_n(x_1, x_2, \ldots, x_n) \end{pmatrix} \]

Newton’s method for a system of equations is analogous to that for a single equation: instead of the derivative \( f'(x) \), it uses the Jacobian matrix:

\[ J = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix} \]

Since \( J \) is a matrix instead of dividing by \( f'(x) \) we multiply by \( J^{-1} \):

\[ x^{(k+1)} = x^{(k)} - J^{-1}(x^{(k)})f(x^{(k)}) \]

i.e. for each \( k = 0, 1, \ldots \), we have to solve a linear system of equations:

\[ J(x^{(k)})\delta x^{(k)} = -f(x^{(k)}) \]

where

\[ \delta x^{(k)} = x^{(k+1)} - x^{(k)} \]

using an initial guess \( x^{(0)} \). The solution of such systems is considered in the next section.