Set 2 — Root-Finding

Week 9

* — tutorial question; A — advanced; S — supplementary

1

(a) Attached to this exercise set is a listing of a file $mc3nm/newraph.f90 which implements the Newton-Raphson method. Copy the file to your directory using the command,

```
cpi $mc3nm/newraph.f90 set2q1.f90
```

then compile and run it for the test case, $f(x) = 2x^3 + 3x - 3$, with the initial approximation $x_0 = 1$ to the root and tolerance $5 \times 10^{-6}$ (i.e. $5.E-6$).

(b) It is good practice to limit the number of iterations which a program can perform in case of slow or non-convergence. Modify `newraph.f90` so that it stops and prints out a message if convergence is not achieved within 24 iterations. Also modify your program so that it terminates using the relative change test,

$$|x_{n+1} - x_n| < \epsilon |x_{n+1}|,$$

rather than the absolute change test.

(c) Plot the function $f(x) = x^3 - 2x - 5$ using `funplot` and hence show that $f$ has one real zero near $x_0 = 2.1$. Use your program from Part (b) to find the root to 6 significant figures. The command to run `funplot` is

`funplot`

(d) Modify your program from Part (b) to find the real root(s) of the function

$$f(x) = \exp 2x - 2 \exp x + 1.$$ 

What does the rate of convergence appear to be? Can you explain your result? Can you use the bisection method to find this root?

Solution.

See `set2q1.ans`.
2. Complex roots. Attached is a listing of the file $mc3nm/cnewraph.f90$, which implements the Newton-Raphson method for the possibly complex roots of a complex analytic function. Copy $mc3nm/cnewraph.f90$, then compile and run it for the function $f(x) = x^3 - 2x - 5$ with the initial approximation $x_0 = 2 + i$ and tolerance $5 \times 10^{-6}$. Enter the complex initial approximation as an ordered pair $(2.,1.)$. Hence find the complex roots of $f(x) = x^3 - 2x - 5$.

Solution.
See set2q2.ans.

3* The root-finding subroutine fzero from Numerical Methods and Software by Kahaner, Moler & Nash (KMN) implements a combination of the bisection and secant methods. The arguments of fzero are described in a prologue —$mc3nm/fzero.pro. In the file $mc3nm/fzero_main.f90$ is a main program, which calls fzero to find the real root of $f(x) = x^3 - 2x - 5$ in $[2,3]$. [Subroutine fzero is listed in the file $mc3nm/fzero.f$ but you do not need the source code of fzero, since we are treating it as a grey box.]

(a) Copy $mc3nm/fzero_main.f90$ to your directory, then compile and link it using the command

\texttt{f90o fzero\_main.f90 $kmnlib}

The subroutine fzero and all the other routines in KMN have been compiled into a library, kmnlib. The command \texttt{f90o fzero\_main.f90 $kmnlib} compiles the main program fzero\_main and then links it to the kmnlib routines.

(b) Modify your program in Part (a) to solve the freezing water main problem: During a cold snap the ground surface temperature drops to $T_s$. Before the cold snap the ground temperature is uniform at $T_i$ and $t_s$ after the snap the temperature $T$ at a distance $x$ m below the surface is given by

$$
\frac{T - T_s}{T_i - T_s} = \text{erf} \left( \frac{x}{2\sqrt{\alpha t}} \right), \quad \text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt,
$$

where $\alpha$ is the thermal diffusivity of the soil. If $T_i = 20^\circ C$, $T_s = -15^\circ C$ and $\alpha = 0.138 \times 10^{-6} m^2 s^{-1}$, how deep should a water main be buried so that it does not freeze in less than 60 days, given that water freezes at $0^\circ C$. For the error function erf$(x)$ use the function subprogram ERF in kmnlib. (KMN, Problem P7–1.)

Solution.
See set2q3.ans.

4. Show that the Newton-Raphson method for $f(x) = 1/x - R$ is

$$
x_{n+1} = x_n(2 - Rx_n).
$$

For what values of $x_0$ does this iteration scheme converge?
Solution.

\[ f'(x) = \frac{-1}{x^2}. \] Thus

\[ x_{n+1} = x_n - \frac{1/x_n - R}{-1/x_n^2} = x_n(2 - Rx_n). \]

This corresponds to a simple iteration scheme with \( g(x) = x(2 - Rx) \). For convergence \(|g'(x)| < 1\), i.e. \( |2(1 - Rx)| < 1 \) or \( 1/2R < x < 3/2R \) \((R > 0)\).

5. (Calculator.) Consider the equation

\[ x = e^{-x}. \]

(a) Sketch the function \( f(x) = x - e^{-x} \) and hence show that \( f(x) \) has a root near \( x = 0.5 \).

(b) Use the following methods to find the root near \( x_0 = 0.5 \) accurate to four significant figures:

(i) the bisection method with \( x_1 = 0.6 \);
(ii) the simple iteration scheme \( x_{n+1} = e^{-x_n} \);
(iii) the Newton-Raphson method;
(iv) the secant method with \( x_1 = 0.6 \).

Solution.

(a) By hand or use funplot.

(b)

<table>
<thead>
<tr>
<th>( n )</th>
<th>(i)</th>
<th>(ii)</th>
<th>(iii)</th>
<th>(iv)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.55000</td>
<td>0.60000</td>
<td>0.60653</td>
<td>0.56631</td>
</tr>
<tr>
<td>2</td>
<td>0.55000</td>
<td>0.57500</td>
<td>0.54524</td>
<td>0.56714</td>
</tr>
<tr>
<td>3</td>
<td>0.56250</td>
<td>0.57500</td>
<td>0.57970</td>
<td>0.56714</td>
</tr>
<tr>
<td>4</td>
<td>0.56250</td>
<td>0.56875</td>
<td>0.56006</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.56563</td>
<td>0.56875</td>
<td>0.57117</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.56563</td>
<td>0.56719</td>
<td>0.56486</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.56641</td>
<td>0.56719</td>
<td>0.56844</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0.56680</td>
<td>0.56719</td>
<td>0.56641</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0.56699</td>
<td>0.56719</td>
<td>0.56756</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0.56709</td>
<td>0.56719</td>
<td>0.56691</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>0.56714</td>
<td>0.56719</td>
<td>0.56728</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>0.56714</td>
<td>0.56716</td>
<td>0.56707</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>0.56714</td>
<td>0.56715</td>
<td>0.56719</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>0.56714</td>
<td>0.56714</td>
<td>0.56712</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td>0.56716</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td>0.56714</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td>0.56715</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td>0.56714</td>
<td></td>
</tr>
</tbody>
</table>
6. (Calculator.) Consider the equation

\[ f(x) = \tan x - x = 0. \]

(a) Sketch the function \( f(x) = \tan x - x \) in the interval \( 4 \leq x \leq 4.7 \) and hence show that \( f(x) \) has a root near \( x = 4.5 \).

(b) The above root is to be found by writing \( f(x) = 0 \) in fixed-point form \( x = g(x) \) and then applying the iteration scheme \( x_{n+1} = g(x_n) \) with \( x_0 = 4.5 \). Possible choices for \( g \) are

\[ (i) \quad g(x) = \tan x, \]
\[ (ii) \quad g(x) = (51x - \tan x)/50, \]
\[ (iii) \quad g(x) = \pi + \tan^{-1} x. \]

Note that \( y = \tan^{-1} x \) if \( x = \tan y \), where \(-\pi/2 < y < \pi/2\), and that

\[ \frac{dy}{dx} = \frac{1}{1 + x^2}. \]

Use some test (other than actually performing the iteration) to decide which of these forms will give the iteration scheme with the best convergence. Use this scheme to find the root accurate to five significant figures.

**Solution.**

(a) Construct a table of values and then draw a neat sketch.

<table>
<thead>
<tr>
<th>( x )</th>
<th>4.0</th>
<th>4.1</th>
<th>4.2</th>
<th>4.3</th>
<th>4.4</th>
<th>4.5</th>
<th>4.6</th>
<th>4.7</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f(x) )</td>
<td>-2.8</td>
<td>-2.7</td>
<td>-2.4</td>
<td>-2.0</td>
<td>-1.3</td>
<td>0.1</td>
<td>4.3</td>
<td>76.0</td>
</tr>
</tbody>
</table>

(b) To choose the iteration scheme with the best convergence, compare the derivatives, \( dg/dx \), of the three schemes at the zeroth approximation \( x_0 = 4.5 \). The values are

\[ (i) \quad 22.50485, \quad (ii) \quad 0.5699030, \quad (iii) \quad 4.7058824 \times 10^{-2}. \]

Expect (iii) to converge the most rapidly since its derivative is the smallest, (i) to probably not converge and (ii) to converge but not as quickly as (iii). Note that this test is *not* foolproof. The iterates for the three schemes are as follows:
\[ x_{n+1} = x_n - \frac{f(x_n)}{D(x_n)}, \quad D(x_n) := \frac{f(x_n + f(x_n)) - f(x_n)}{f(x_n)} , \]

converges quadratically to a simple root of \( f(x) = 0 \).

**Solution.**

For root-finding schemes of simple form, \( x_{i+1} = g(x_i) \), where \( x = g(x) \) is a fixed point form of \( f(x) = 0 \), that the error \( e_{i+1} \) in \( x_{i+1} \) is related to the error \( e_i \) in \( x_i \) by

\[ e_{i+1} = x_{i+1} - x^* = g(x_i) - g(x^*) = g(x^* + e_i) - g(x^*) . \]

Expanding \( g(x^* + e_i) \) using Taylor’s Theorem gives

\[ e_{i+1} = g'(x^*)e_i + \frac{1}{2} g''(x^*)e_i^2 + O(e_i^3) . \]

The error is defined by \( e_i = x_i - x^* \), where \( x^* \) is the root of \( f(x) = 0 \). Expand \( f(x + z) \) in a Taylor series about \( z = 0 \), \( f(x+z) = f(x) + f'(x)z + \frac{1}{2} f''(x)z^2 + O(z^3) \), and set \( z = f(x) \). This gives

\[ f(x + f(x)) = f(x) + f'(x)f(x) + \frac{1}{2} f''(x)\{f(x)\}^2 + O\{f(x)\}^3 , \]

and hence

\[ D(x) = \frac{f(x + f(x)) - f(x)}{f(x)} = f'(x) + \frac{1}{2} f(x) f''(x) + O\{f(x)\}^2 . \]
Thus \( D(x^*) = f'(x^*) \neq 0 \), since the root is simple. Steffenson’s method is of simple form with \( g(x) = x - f(x)/D(x) \). Since

\[
g'(x^*) = 1 - \frac{f'(x^*)}{D(x^*)} + \frac{f(x^*) D'(x^*)}{(D(x^*))^2} = 1 - 1 + 0 = 0,
\]
the method is at least second order. To find \( g''(x^*) \) we need

\[
D'(x) = f''(x) + \frac{1}{2} f'(x) f''(x) + O\{f(x)\},
\]

which gives \( D'(x^*) = f''(x^*) + \frac{1}{2} f'(x^*) f''(x^*) \). Thus

\[
g''(x^*) = \frac{-f''(x^*)}{D(x^*)} + \frac{2 f'(x^*) D'(x^*)}{(D(x^*))^2} + \frac{f(x^*) D''(x^*)}{(D(x^*))^2} - \frac{2 f(x^*) D'(x^*)}{(D(x^*))^3}
\]

\[
= \frac{f''(x^*)}{f'(x^*)} + f''(x^*),
\]

which is not generally zero. The method is thus usually second order for simple roots with asymptotic error constant

\[
C = \frac{f''(x^*)}{2 f'(x^*)} + \frac{1}{2} f''(x^*).
\]

Steffenson’s method does not use \( f'(x) \), which is an improvement on the Newton-Raphson method. But since it requires two function evaluations per iteration it must be compared with two steps of the secant method, which are better than quadratic.

8. Show that the rate of convergence of

\[
x_{n+1} = \frac{x_n(x_n^2 + 3a)}{3x_n^2 + a}
\]

to \( \sqrt{a} \) is three and find the asymptotic error constant.

**Solution.**

Here \( g(x) = x(x^2 + 3a)/(3x^2 + a) \). Since

\[
g'(x) = \frac{3(x^2 - a)^2}{(3x^2 + a)^2}, \quad g''(x) = \frac{48ax(x^2 - a)}{(3x^2 + a)^3},
\]

\( g'(\sqrt{a}) = g''(\sqrt{a}) = 0 \) and \( g'''(\sqrt{a}) = 3/2a \). The only term in differentiating \( g'' \) that will contribute to \( g''' \) is the one which differentiates \( x^2 - a \) in the numerator, as the other terms will contain \( x^2 - a \) as a factor. The asymptotic error constant is \( C = \frac{1}{6} g'''(x^*) = 1/4a \).
Some third-order methods. Show that the rate of convergence of the following methods to a simple root of \( f(x) = 0 \) is at least three.

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

(b) \[
x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} - \frac{[f(x_n)]^2 f''(x_n))}{2[f'(x_n)]^3}.
\]

Solution.

(a) Both \( x_n \) and \( z_n \) are approximations to a root of \( f(x) = 0 \). Let the root be \( x^* \) and let the errors in \( x_n \) and \( z_n \) be \( e_n = x_n - x^* \) and \( E_n = z_n - x^* \). Then, since \( z_{n+1} \) is calculated from \( x_n \) by one step of the Newton-Raphson method,
\[
E_{n+1} = \frac{f''(x^*) e_n^2}{2f'(x^*)} + O(e_n^3).
\]

We want \( e_{n+1} \) as a power series in \( e_n \). Now
\[
e_{n+1} = x_{n+1} - x^* = z_{n+1} - x^* - \frac{f(z_{n+1})}{f'(x_n)} = E_{n+1} - f(x^*) + E_{n+1} = \frac{f(x^* + E_{n+1})}{f'(x^* + e_n)}.
\]

Expanding in Taylor series,
\[
f(x^* + E_{n+1}) = f(x^*) + f'(x^*) E_{n+1} + O(E_{n+1}^2)
\]
\[
f'(x^* + e_n) = f'(x^*) + f''(x^*) e_n + O(e_n^2).
\]

By the binomial expansion, \((a + y)^{-1} = a^{-1} - ya^{-2} + O(y^2)\), with \( a = f'(x^*) \) and \( y = f''(x^*) e_n + O(e_n^2)\),
\[
\frac{1}{f'(x^* + e_n)} = \frac{f''(x^*) e_n}{f'(x^*)} + O(e_n^2).
\]

Thus
\[
e_{n+1} = E_{n+1} - \left\{ f'(x^*) E_{n+1} + O(E_{n+1}^2) \right\} \left( \frac{1}{f'(x^* )} - \frac{f''(x^*) e_n}{f'(x^*)} + O(e_n^2) \right)
\]
\[
= \frac{f''(x^*)}{f'(x^*)} e_n E_{n+1} + O(e_n^4)
\]
\[
= \frac{1}{2} \left( \frac{f''(x^*)}{f'(x^*)} \right)^2 e_n^3 + O(e_n^4).
\]

The method is third order with asymptotic error constant,
\[
C = \frac{1}{2} \left( \frac{f''(x^*)}{f'(x^*)} \right)^2 .
\]

This method does not improve on the Newton-Raphson method (why?).
(b) Use the method of Question 7 with

\[ g(x) = x - \frac{f(x)}{f'(x)} - \frac{[f(x)]^2 f''(x))}{2[f'(x)]^3}. \]

Then

\[ g'(x) = 1 - \frac{f'(x)}{f'(x)} + \frac{f(x)f''(x)}{[f'(x)]^2} - \frac{[f(x)]f''(x))}{[f'(x)]^2} + \mathcal{O}(f^2) = \mathcal{O}(f^2) \]

and \( g''(x) = \mathcal{O}(f) \). Hence \( g'(x^*) = 0 \) and \( g''(x^*) = 0 \), so the method is third order. This method is the second in a series of methods of increasing order, the first being the Newton-Raphson. To derive them expand the inverse function \( x = F(y) \) of \( y = f(x) \) in a Taylor series about the \( i \)-th iterate \( y_i = f(x_i) \):

\[ x = \sum_{k=0}^{\infty} \frac{F^{(k)}(y_i)}{k!}(y - y_i)^k, \]

where \( F^{(k)}(y) \) is the \( k \)-th derivative of \( F(y) \). The zero \( x^* \) of \( f(x) \) near \( y_i \) is given by \( y = 0, \)

\[ x^* = \sum_{k=0}^{\infty} \frac{F^{(k)}(y_i)}{k!}(-y_i)^k. \]

It remains to evaluate \( F^{(k)}(y_i) \) in terms of \( f \) and its derivatives at \( x_i, F^{(0)}(y) = F(y) = x, F^{(1)}(y) = dF(y)/dy = dx/dy = 1/(dy/dx) = 1/f'(x) \) and iteratively,

\[ F^{(k)}(y) = \frac{dF^{(k-1)}(y)}{dy} = \frac{dx}{dy} \frac{dF^{(k-1)}(y)}{dx} = \frac{1}{f'(x)} \frac{dF^{(k-1)}(y)}{dx}. \]

Thus, for example,

\[ F^{(2)}(y) = \frac{1}{f'(x)} \frac{dF^{(1)}(y)}{dx} = \frac{1}{f'(x)} \frac{d[1/f'(x)]}{dx} = -\frac{f''(x)}{[f'(x)]^3}. \]

Hence truncating the series for \( x^* \) after 3 terms,

\[ x^* \approx F^{(0)}(y_i) + F^{(1)}(y_i)(-y_i) + \frac{1}{2!}F^{(2)}(y_i)(y_i)^2 \]

\[ = x_i + \frac{1}{f'(x_i)}(-y_i) + \frac{1}{2!} \left\{ -\frac{f''(x_i)}{[f'(x_i)]^3} \right\} (y_i)^2. \]

Since \( y_i = f(x_i) \) the given method is obtained. To proceed further computer algebra is easiest. Successively higher-order derivatives are required — a major drawback.