0.1 Root Finding in One Dimension

Question 1

To show graphically that equation $2x - 3 \sin(x) + 5 = 0$ has exactly one root, plot $y = 2x - 3 \sin(x) + 5$ and $y = 0$ and show that these two lines have only one point of intersection. I will plot the graph in the range $x \in [-5, 0]$. This is because, for $x < -5, 2x + 5 < 3 \sin(x)$, so $y = 2x - 3 \sin(x) + 5 < 0$ and for $x > 0, |3 \sin(x)| < 2x + 5$, so $y = 2x - 3 \sin(x) + 5 > 0$.

Thus $y < 0$ for $x < -5$, and $y > 0$ for $x > 0$, so no root can lie outside the range $[-5, 0]$.

```
>> x=linspace(-5,0,1000);
y=2*x-3*sin(x)+5;
>> plot(x,y,'b',x,0,'g')
>> xlabel('x'); ylabel('y');legend('2x-3sin(x)+5','0');
```

![Graph of y = 2x - 3sin(x) + 5 and y = 0]

Figure 1: A graph displaying the intersection of $y = 2x - 3 \sin(x) + 5$ and $y = 0$

From the graph, the function intersects the x-axis only once in this range, outside which we have determined there are no further points of intersection. Thus $2x - 3 \sin(x) + 5 = 0$ has only one root.

Binary Search

Programming Task

The program to solve $2x - 3 \sin(x) + 5 = 0$ by binary search is shown on page 12, labelled

```
q2_binarysearch(xlow,xhigh)
```

This program requires only the input of $xlow$ and $xhigh$ with termination of the iteration as soon as the truncation error is guaranteed to be less than $\varepsilon = 0.5 \times 10^{-5}$; the program also prints out the number of iterations, $N$, as well as an estimate of the root.

Testing for functionality of the program:

(i) Initial interval $[-3,-2]$

Assessor comments in Red

This was written by a Part IB student and shows what is expected of a good project write-up. There is room for improvement, but the project would probably receive full marks.
Note, as illustrated here, the program runs for xlow and xhigh in any order.

(ii) Initial interval [-10, 10]

```
>> q2_binarysearch(-10, 10)
N= 22, x*= -2.8832388
```

(iii) Initial interval [-500, 10000]

```
>> q2_binarysearch(-500, 10000)
N= 31, x*= -2.8832417
```

Note that the program has been written to not run if \( f(x_{\text{low}}) \) and \( f(x_{\text{high}}) \) have the same sign:

```
>> q2_binarysearch(0, 5)
```

Choose xlow and xhigh with \( f(x_{\text{low}}) \cdot f(x_{\text{high}}) < 0 \)

```
+1.5C for working code and test results
```

Question 2

Suppose that the rounding error in evaluating \( F(x) \) in \( F(x) \equiv 2x - 3\sin(x) + 5 = 0 \) is at most \( \delta \) for \( |x| < \pi \).

Consider a Taylor expansion of \( F(x) \) near \( x_* \):

\[
F(x) = F(x_*) + (x - x_*)F'(x_*) + o((x - x_*)^2)
\]

The root \( | -2.88323... | < \pi \Rightarrow |F(x_*)| < \delta \)

Ignoring \( o((x - x_*)^2) \) terms, and since \( F(x_*) = 0 \),

\[
F(x) \approx (x - x_*)F'(x_*)
\]

Thus,

\[
|x - x_*| \approx \frac{|F(x)|}{|F'(x_*)|} \leq \frac{\delta}{|F'(x_*)|}
\]

And for \( -\frac{3\pi}{4} < x < -\frac{5\pi}{4} \), within which the root lies, \( |F'(x)| > 4 \)

\[
\Rightarrow |x - x_*| \leq \frac{\delta}{|F'(x_*)|} < \frac{\delta}{4}
\]

Now if \( x_* \) is in the final interval \([m, n]\), that is \( x_* = \frac{1}{2}(m + n) \), then the actual value of \( x_* \) has error bounded by \( \frac{1}{2}(m - n) \). Since the iteration is terminated as soon as the truncation error is guaranteed to be less than \( 0.5 \times 10^{-5} \),

\[
\left| \frac{1}{2}(m - n) \right| < 0.5 \times 10^{-5}
\]

Thus the accuracy expected for the calculated value of the root is within

\[
0.5 \times 10^{-5} + \frac{\delta}{4}
\]

of the actual root.

```
+1T for accuracy
```

Note: incorrect use of little-o notation. Should either be big-O or square should be omitted. Probably no loss of marks though.
Fixed-Point Iteration

Programming Task

The program to solve \(2x - 3\sin(x) + 5 = 0\) by fixed point iteration, by rearranging it to

\[
f(x) = \frac{3\sin(x) + kx - 5}{2 + k}
\]

is shown on page 13, labelled

\(q_3\) \_fixedpoint1\( (\text{fun},x_0,N_{\text{max}})\)

(i) For \(k=0\), \(x_0=-2\), \(N_{\text{max}}=10\),

\[
\begin{array}{ccc}
\text{N} & \text{xN} & |eN/eN-1| \\
1 & -3.8639461 & \\
2 & -1.5082717 & 1.4020110 \\
3 & -3.9970690 & 0.8100802 \\
4 & -1.3676749 & 1.3606736 \\
5 & -3.9691625 & 0.7165168 \\
6 & -1.3955663 & 1.3699562 \\
7 & -3.9770297 & 0.7352386 \\
8 & -1.3876153 & 1.3673719 \\
9 & -3.9749038 & 0.7299086 \\
10 & -1.3897571 & 1.3680727 \\
\end{array}
\]

Plotting \(y = f(x)\) and \(y = x\) on the same graph:

![Graph showing fixed point iteration](image)

Figure 2: A graph displaying no convergence near root for \(k=0\)

This plot shows that convergence should not occur, as the iterations remain in a loop around the root and hence should never reach the root. Now,
\[ x_{N+1} = f(x_N) = f(x_*) + (x_N - x_*)f'(x_*) + \ldots \approx x_* + (x_N - x_*)f'(x_*) \]

Recall that the truncation error in the \( N^{th} \) iterate is \( e_N \).

\[ \Rightarrow e_{N+1} \approx e_N f'(x_*) \]

Thus the iteration will diverge if \( |f'(x_*)| = \frac{|3\cos(x_*)|}{2} > 1 \), which is the case for root \( x_* = -2.8832\ldots \).

Thus convergence does not occur.

(ii) Recall that the truncation error in the \( N^{th} \) iterate is \( e_N = x_N - x_* \).

\[
\begin{align*}
 e_{N+1} &= x_{N+1} - x_* \\
 &= f(x_N) - f(x_*) \\
 &= f'(c)(x_N - x_*) \text{ by the Mean Value Theorem for some } c \in (x_N, x_*) \\
 &= f'(c)e_N
\end{align*}
\]

Thus the error reduces, and hence the scheme converges, if \( |f'(x_*)| < 1 \) for all \( x \in (-\pi, -\frac{\pi}{2}) \).

For \( f(x) = \frac{3\sin(x) + kx - 5}{2 + k} \), \( f'(x) = \frac{3\cos(x) + k}{2 + 6k} \),

And for \( x_N \in (-\pi, -\frac{\pi}{2}) \), \( -1 < \cos(x_N) < 0 \),

\[ \Rightarrow \frac{-3+k}{2+k} < f'(x) < \frac{k}{2+k} \]

So for convergence we require:

\[
\begin{align*}
 (i) &\quad -1 < \frac{-3+k}{2+k} < 1 \Rightarrow k > \frac{1}{2} \\
 \text{and} \quad (ii) &\quad -1 < \frac{k}{2+k} < 1 \Rightarrow k > -1
\end{align*}
\]

Thus convergence is guaranteed for \( k > \frac{1}{2} \).

(iii) The error reduces monotonically, and hence the scheme converges monotonically, if \( 0 \leq f'(x) < 1 \) for all \( x \in (a, b) \). Similarly, the error reduces, and hence the scheme converges, in an oscillatory manner if \( -1 < f'(x) < 0 \) for all \( x \in (a, b) \).

**Monotonic** convergence requires \( 0 < f'(x) < 1 \) for all \( x \) around the root. If \( x_N \) remains in the range \( (-\pi, -\frac{\pi}{2}) \), for monotonic convergence, we require:

\[
\begin{align*}
 (i) &\quad 0 < \frac{-3+k}{2+k} < 1 \Rightarrow k > 3 \\
 \text{and} \quad (ii) &\quad 0 < \frac{k}{2+k} \leq 1 \Rightarrow k > 0
\end{align*}
\]

Thus monotonic convergence should occur near the root for \( k > 3 \).

Choose \( k=4 \), such that \( f(x) = \frac{3\sin(x) + 4x - 5}{6} \). Verifying that \( k=4 \) gives expected monotonic convergence with \( N_{\text{max}}=20 \):

\[
\text{>> } f=\text{inline}('3*\text{sin}(x) + 4*x - 5)/6');
\]
Clearly this convergence is monotonic, as illustrated by this graph given by program on page 14:

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{cobweb_k4.png}
\caption{A graph displaying the monotonic convergence near root for \( k=4 \)}
\end{figure}

(iii) **Oscillatory** convergence requires \(-1 < f'(x) < 0\) for all \( x \) around the root. If \( x_N \) remains in the range \((-\pi, -\frac{3\pi}{4})\) (note: \(-\pi < -2.8832 < -\frac{3\pi}{4}\), \(-1 < \cos x_N < -\frac{\sqrt{2}}{2}\). So for oscillatory convergence, we require:

\[(i) -1 < \frac{-3 + k}{2 + k} < 0 \Rightarrow \frac{1}{2} < k < 3\]

and

\[(ii) -1 < \frac{-\frac{3\sqrt{2}}{2} + k}{2 + k} < 0 \Rightarrow \frac{-4 + 3\sqrt{2}}{4} < k < \frac{3\sqrt{2}}{2}\]

Thus oscillatory convergence should occur near the root for \( \frac{1}{2} < k < \frac{3\sqrt{2}}{2} \).

Choose \( k = \frac{3}{2} \), such that \( f(x) = \frac{6\sin(x) + 3x - 10}{7} \). Verifying that \( k = \frac{3}{2} \) gives expected oscillatory convergence with \( N_{\text{max}}=20 \):

\[
\frac{\text{q3_fixedpoint1}(f,-2,20 )}{
\begin{array}{c|c|c}
N & xN & |eN/eN-1| \\
1 & -3.0651121 & \\
2 & -2.8076818 & 0.4154227 \\
3 & -2.9127840 & 0.3910668 \\
4 & -2.8713224 & 0.4032384 \\
5 & -2.8831747 & 0.3988001 \\
6 & -2.8831332 & 0.4006404 \\
\end{array}
\]
Clearly this convergence is oscillatory, as illustrated by this graph given on page 14 by program:

```
cobweb_koneandhalf
```

![Graph showing oscillatory convergence](image)

Figure 4: A graph displaying the oscillatory convergence near root for $k = \frac{3}{2}$

(iv) For $k = 16$, $f(x) = \frac{3\sin(x) + 16x - 5}{18}$, and for $N_{\text{max}} = 50$ due to slower convergence,

```
>> f=inline('(3*sin(x) + 16*x -5)/18');
q3_fixedpoint1(f,-2,50);
```

| $N$ | $x_N$ | $|e_N/e_{N-1}|$ |
|-----|-------|----------------|
| 1   | -2.2071051 |               |
| 2   | -2.3736981 | 0.7536087     |
| 3   | -2.5035020 | 0.7452521     |
| 4   | -2.6023900 | 0.7395866     |
| 5   | -2.6765887 | 0.7358039     |
| 6   | -2.7317054 | 0.732821      |
| 7   | -2.7723780 | 0.7315897     |
| 8   | -2.8022610 | 0.7304414     |
| 9   | -2.8241526 | 0.7296528     |
| 10  | -2.8401582 | 0.7291050     |
| 11  | -2.8518446 | 0.7287205     |
| 12  | -2.8603692 | 0.7284484     |
| 13  | -2.8665834 | 0.7282545     |
| 14  | -2.8711112 | 0.7281156     |
| 15  | -2.8744092 | 0.7280157     |
| 16  | -2.8768108 | 0.7279436     |
| 17  | -2.8785594 | 0.7278915     |
| 18  | -2.8798324 | 0.7278537     |
| 19  | -2.8807590 | 0.7278263     |
| 20  | -2.8814334 | 0.7278063     |

+0.5T for correct discussion of monotonic/oscillatory convergence
+1.5C for example tables/plots
Now
\[ x_N = f(x_{N-1}) = f(x_*) + (x_{N-1} - x_*) f'(x_*) + ... \]
\[ \approx x_* + (x_{N-1} - x_*) f'(x_*) \]
Thus
\[ x_N - x_{N-1} \approx (x_* - x_{N-1}) + (x_{N-1} - x_*) f'(x_*) \]
\[ = (x_{N-1} - x_*) (f'(x_*) - 1) \]
\[ \approx (f'(x_*) - 1) \left\{ \frac{x_N - x_*}{f'(x_*)} \right\} \]
\[ \Rightarrow (x_N - x_*) \approx (x_N - x_{N-1}) \left\{ \frac{f'(x_*)}{f'(x_*) - 1} \right\} \]

Since \( |x_N - x_{N-1}| < \epsilon \) when iteration is terminated, \( |x_N - x_*| \) is greater than \( \epsilon = 10^{-5} \) by a factor of approx
\[ \frac{|f'(x_*)|}{|f'(x_*) - 1|} \]

For \( k=16 \), \( f'(x) = \frac{3 \cos(x) + 16}{18} \), so for \( x_* \approx -2.88323687 \),
\[ \frac{|f'(x_*)|}{|f'(x_*) - 1|} \approx 2.6731... \]
Thus for the case \( k=16 \), the truncation error is expected to be greater than \( 10^{-5} \).

(v) If the truncation error in the \( N^{th} \) iterate is \( e_N = x_N - x_* \), the method is said to have \( p^{th} \) order convergence if:
\[ |e_{N-1}| < \eta \Rightarrow |e_N| \leq C |e_{N-1}|^p \]
Thus this method has first-order convergence if \( |e_N|/|e_{N-1}| \) tends to some constant, \( C < 1 \). The final column displaying \( |e_N|/|e_{N-1}| \) in the tables for \( k=4 \) and \( k = \frac{3}{2} \) are consistent with this condition for first-order convergence, with \( C \approx 0.4 \) for \( k = 4 \) and \( C \approx 0.727 \) for \( k = \frac{3}{2} \).

Question 4

The modified program to find the double root of equation (5a) by fixed-point iteration by taking
\[ h(F) = \frac{F}{20} \]
in (6), so that
\[ f(x) = \frac{-x^3 + 8.5x^2 + 8}{20} \]
is given on page 15, labelled
\[
\text{q4\textunderscore fixedpoint2}(x_0, N_{\text{max}})
\]
For \( N_{\text{max}} = 1000, x_0 = 5 \),
\[
\gg \text{q4\textunderscore fixedpoint2}(5, 1000)
\]
\[
N = 740, x_N = 4.0075342
\]
Since \( f(x) = x - h(F(x)), \) then \( f'(x) = 1 - h'(F(x))F'(x). \) Since \( F'(x_\ast) = 0, \) as it is a double root, \( f'(x_\ast) = 1. \)
Thus \( x_\ast + f'(x_\ast)(x_{N-1} - x_\ast) = x_{N-1}. \) So in the Taylor expansion of \( x_N = f(x_{N-1}), \) we need to consider higher orders,
\[
x_N = f(x_{N-1}) = f(x_\ast) + (x_{N-1} - x_\ast)f'(x_\ast) + \frac{1}{2}f''(x_\ast)(x_{N-1} - x_\ast)^2 + \ldots
\approx x_{N-1} + \frac{1}{2}f''(x_\ast)(x_{N-1} - x_\ast)^2
= x_{N-1} + \frac{1}{2}f''(x_\ast)(e_{N-1})^2
\]
Thus convergence will be slow at a multiple root for any choice of differentiable function \( h. \) Thus,
\[
e_{N-1} \approx \pm \frac{\sqrt{2(e_{N_0} - e_{N-1})}}{|f''(x_\ast)|}
\]
If the iterations are terminated when \( |x_N - x_{N-1}| < \epsilon, \)
\[
|e_N| \approx \left| \frac{2\epsilon}{f''(x_\ast)} \right| = \left| \frac{40\epsilon}{7} \right| \approx 0.007559
\]
for \( \epsilon = 10^{-5} \) and \( f''(x_\ast) = -\frac{7}{20} \) for double root \( x_\ast = 4. \) Thus the termination criterion does not ensure a truncation error of less than \( 10^{-5}. \)

For first-order convergence, must show that \( \frac{|e_N|}{|e_{N-1}|} \leq C \) for some constant \( C < 1. \)

Now it can be shown that the truncation error,
\[
e_N \approx \frac{40}{7N} \text{ as } N \to \infty
\]
Thus \( \frac{|e_N|}{|e_{N-1}|} \to 1 \) as \( N \to \infty, \) indicating slower than first-order convergence.

Newton-Raphson Iteration

Programming Task

The program to solve \( 2x - 3\sin(x) + 5 = 0 \) by Newton-Raphson iteration is shown on page 16, labelled
\[
\text{q5\textunderscore newtonraphson1}(x_0, \text{Nmax}, \text{tol})
\]
and the program to solve \( x^3 - 8.5x^2 + 20x - 8 = 0 \) by Newton-Raphson iteration is shown on page 17, labelled

\[ q5\_newtonraphson2(x0,Nmax,tol). \]

**Question 5**

For \( 2x - 3\sin(x) + 5 = 0 \), starting with \( x_0 = -4.7 \), \( \epsilon = 10^{-5} \), it does converge to the root, but only after 85 iterations.

![Figure 5: A graph displaying the first couple of Newton Raphson iterations for \( F(x) = 2x - 3\sin(x) + 5 \) starting at \( x_0 = -4.7 \)](image)

For \( 2x - 3\sin(x) + 5 = 0 \), starting with \( x_0 = -4 \), \( \epsilon = 10^{-5} \), it converges after 4 iterations,

\[
\begin{array}{c|c|c|c}
N & x_N & |e_N/e_{N-1}| & |e_N/(e_{N-1})^2| \\
\hline
1 & -2.6694018 & & \\
2 & -2.8889594 & 0.0267613 & 0.1251491 \\
3 & -2.8832394 & 0.0004411 & 0.0770849 \\
4 & -2.8832369 & 0.0010137 & 401.5613254 \\
\end{array}
\]

For \( \epsilon = 10^{-10} \), it converges after 5 iterations,

\[
\begin{array}{c|c|c|c}
N & x_N & |e_N/e_{N-1}| & |e_N/(e_{N-1})^2| \\
\hline
1 & -2.6694018 & & \\
2 & -2.8889594 & 0.0267613 & 0.1251491 \\
3 & -2.8832394 & 0.0004411 & 0.0770849 \\
\end{array}
\]

For \( \epsilon = 10^{-10} \), it converges after 5 iterations,
Consider the convergence of this root,

\[ e_N = x_N - x^* = x_{N-1} - x^* - \frac{F(x_{N-1})}{F'(x_{N-1})} \]

\[ = e_{N-1} - \frac{\left( F(x_N) + (x_{N-1} - x_N)F'(x_N) + \frac{1}{2} (x_{N-1} - x_N)^2 F''(x_N) + \ldots \right)}{F'(x_N)} \]

\[ = e_{N-1} - \frac{e_{N-1} F'(x_N) + \frac{1}{2} (e_{N-1})^2 F''(x_N)}{F'(x_N)} \]

\[ = e_{N-1} - e_{N-1} - \frac{1}{2} (e_{N-1})^2 F''(x_N) + o(e_{N-1}^3) \]

\[ \approx -\frac{1}{2} (e_{N-1})^2 \frac{F''(x_N)}{F'(x_N)} \]

Thus for the single root of \( 2x - 3\sin(x) + 5 = 0 \), we have quadratic convergence. The result of quadratic convergence is consistent with the results in the column \( \frac{|e_N|}{|e_{N-1}|^2} \) for \( \epsilon = 10^{-5} \) and \( \epsilon = 10^{-10} \), except for large differences in the final two values, due to rounding error.

For the double root of \( x^3 - 8.5x^2 + 20x - 8 = 0 \), starting with \( x_0 = 5 \), \( \epsilon = 10^{-5} \), it converges after 17 iterations.

\[
\text{Starting with } x_0 = 5, \quad \epsilon = 10^{-10}, \text{ it converges after 26 iterations,}
\]

\[
\text{q5_newtonraphson2}(5, 50, 10^{-10});
\]

\begin{tabular}{|c|c|c|c|}
\hline
N & xN & \mid eN/eN-1\mid & \mid eN/(eN-1)^2\mid \\
\hline
1 & 4.5500000 & 0.5317919 & 0.9668944 \\
2 & 4.2924855 & 0.5185647 & 1.7729584 \\
3 & 4.0773792 & 0.5017253 & 3.3636414 \\
4 & 4.0391036 & 0.5034925 & 6.5308175 \\
5 & 4.0098570 & 0.5019255 & 12.5042078 \\
6 & 4.0049354 & 0.5007011 & 25.5042078 \\
7 & 4.0024694 & 0.5003518 & 50.5042078 \\
8 & 4.0012352 & 0.5001762 & 101.5042078 \\
9 & 4.0006177 & 0.5000882 & 202.5042078 \\
10 & 4.0003089 & 0.5000441 & 404.5042078 \\
11 & 4.0001544 & 0.5000221 & 809.5042078 \\
12 & 4.0000772 & 0.5000110 & 1618.5042078 \\
13 & 4.0000386 & 0.5000058 & 3237.5359322 \\
14 & 4.0000193 & 0.5000023 & 6474.8629634 \\
15 & 4.0000097 & 0.4999960 & 12949.485625 \\
16 & 4.0000000 & 0.4999960 & 25898.5250833 \\
\hline
\end{tabular}
For a double root, \( F(x_\ast) = F'(x_\ast) = 0 \). Consider the convergence of this root,

\[
e_N = e_{N-1} - \frac{1}{2} \left( x_{N-1} - x_\ast \right)^2 \frac{F''(x_\ast)}{\left( x_{N-1} - x_\ast \right)} + \ldots
\]

\[
\approx e_{N-1} - \frac{1}{2} \left( x_{N-1} - x_\ast \right)
\]

\[
\approx \frac{1}{2} e_{N-1}
\]

Thus for the double root of \( x^3 - 8.5x^2 + 20x - 8 = 0 \) we have first-order convergence. Thus the results in the column \( |e_N| \) should converge to \( \frac{1}{2} \), which is consistent with the results in the table above for \( \epsilon = 10^{-5} \) and \( \epsilon = 10^{-10} \).

If the rounding error is changed from \( 10^{-5} \) to \( 10^{-10} \) the iterations remain the same until the last two iterations. There is a larger error in the final two iterations for \( \epsilon = 10^{-10} \), as \( F'(x_{N-1}) \) tends to 0 as \( N \to \infty \) as it is converging to a double root; thus the division by the small \( F'(x_{N-1}) \) for large \( N \) increases the rounding error in the calculation of \( x_N \).

\[+0.5T \text{ for rounding error}\]

\[+1E \text{ for clear presentation}\]

\[+1E \text{ for several observations}\]

\[\text{made without explicit prompting}\]

\textbf{Totals: 8.0 computation; 10.0 theory; 2.0 excellence = 20 project marks}

\[= 40 \text{ tripos marks}\]
Programs

Binary Search

(i) q2_subbinarysearch.m

function [ xroot ] = q2_subbinarysearch(xlow,xhigh,n)

%The root lies between xhigh and xlow, i.e. f(xhigh)f(xlow)<0
%Iterate until have calculated root to tolerance 0.5*(10)^(-5)
%Start with n=1; the value of n gives the number of iterations until the
%root is found to the required tolerance
%f=inline('2*x-3*sin(x)+5');
tol=0.5*(10)^(-5);

xmid=(xlow+xhigh)/2;

if feval(f,xlow)*feval(f,xhigh)>0
    disp('Choose xlow and xhigh with f(xlow)*f(xhigh)<0')
else
    if (abs(xhigh-xlow)<2*tol)
        xroot=xmid;
        N=n;
        fprintf('N=%3g, x*=%10.7f
', N,xroot)
        return
    elseif (feval(f,xmid)*feval(f,xhigh)>0);
        xroot=q2_subbinarysearch(xlow,xmid,n+1);
    else
        xroot=q2_subbinarysearch(xmid,xhigh,n+1);
    end

(ii) q2_binarysearch.m

function [ xroot ] = q2_binarysearch(xlow,xhigh)

%subbinarysearch must start with n=1 for N to give the number of iterates.
%n=1;
q2_subbinarysearch(xlow,xhigh,1); end
Fixed point iteration

(i) q3_fixedpoint1.m

function [N, xroot ] = q3_fixedpoint1( fun,x0,Nmax )

%Fixed-point iteration to solve F(x)=2x-3sinx+5=0 by finding the root of
%x=fun(x) where fun(x)=(3sinx + kx -5)/(2+k) for some constant k
%x0 is a suitable initial guess
%Find root using iteration scheme xN=f(xN-1)
%Iterate until either abs(xN-xN-1)<tol, or N=Nmax, whichever occurs first

%To investigate convergence, calculate E=abs(e(N))/abs(e(N-1)) where
%eN is truncation error for xN. Use xexact=-2.88323687 for purpose of this
%calculation.

N=1;
tol=10^-(-5);
xexact=-2.88323687;

while N<=Nmax
    x=feval(fun,x0);
    if N==1
        E='';
    else
        e(N)= x - xexact;
        e(N-1)=x0 - xexact;
        E= abs(e(N))/abs(e(N-1));
    end
    fprintf('%4d %10.7f %10.7f
',N, x, E)
    if abs(x-x0)<tol
        xroot=x;
        return
    end
    N=N+1; x0=x;
end
end
(ii) cobweb_k4.m

% Generate the cobweb plot associated with the orbits \( x_{n+1} = f(x_n) \) for \( k=4 \).

\[
x = \text{linspace}(-3.5,-1.5);
y = (3*\sin(x)+4*x-5)/6;
\]

clf
plot(x,y,'r',x,x,'r');
hold on

\[
x(1)=-2;
\]
for i=1:8
    \[
x(i+1)=(3*\sin(x(i))+4*(x(i))-5)/6;
    \]
    line([x(i),x(i)],[x(i),x(i+1)]);
    line([x(i),x(i+1)],[x(i+1),x(i+1)]);
end

(ii) cobweb_koneandhalf.m

% Generate the cobweb plot associated with the orbits \( x_{n+1} = f(x_n) \) for \( k=3/2 \).

\[
x = \text{linspace}(-3.5,-1.5);
y = (3*\sin(x)+(1.5)*x-5)/3.5;
\]

clf
plot(x,y,'r',x,x,'r');
hold on

\[
x(1)=-2;
\]
for i=1:20
    \[
x(i+1)=(3*\sin(x(i))+1.5*(x(i))-5)/3.5;
    \]
    line([x(i),x(i)],[x(i),x(i+1)]);
    line([x(i),x(i+1)],[x(i+1),x(i+1)]);
end
(iv) q4_fixedpoint2.m

function [ N, xroot ] = q4_fixedpoint2( x0,Nmax )
%Fixed-point iteration to solve F(x)=x^3 - 8.5x^2 + 20x -8 =0 by finding
%the root of x=fun(x) where fun(x)=0.05(-x^3+8.5x^2+8)
%x0 is a suitable initial guess
%Find root using iteration scheme xN=f(xN-1)
%Iterate until either abs(xN-xN-1)<tol, or N=Nmax, whichever occurs first
%
%Suppress printing of each iterate and print only final value of N and
%xroot
%
N=1;
tol=10^(-5);
f=inline('0.05*(-x^3+8.5*x^2+8)');

while N<=Nmax
    x=feval(f,x0);
    if abs(x-x0)<tol
        xroot=x;
        fprintf('N=%4g, xN=%10.7f
',N, xroot)
        return
    else N=N+1; x0=x;
    end
end
end
Fixed point iteration

(i) q5_newtonraphson1(x0, Nmax, tol)

function [ N, xroot ] = q5_newtonraphson1(x0,Nmax,tol )
%Newton-Raphson iteration to solve F(x)=2x-3sinx+5=0 using the scheme
%xN=xN-1 - F(xN-1)/F'(xN-1)
%x0 is a suitable initial guess
%Iterate until either abs(xN-xN-1)<tol, or N=Nmax, whichever occurs first
%To investigate convergence, calculate E=abs(e(N))/e(N-1)) and
%F=abs(e(N))/e(N-1)^2 where eN is truncation error for xN. Use
%xexact=-2.88323687 for purpose of this calculation.
% syms x
N=1;
xexact=-2.88323687;
f=inline('2*x-3*sin(x)+5');
df=inline('2-3*cos(x)');
% fprintf(1, ' N xN |eN/eN-1| |eN/(eN-1)^2|
)n
% while N<=Nmax
 x=x0-[f(x0)/df(x0)];
 if N==1
 E='';F='';
 else
 e(N)= x - xexact;
e(N-1)= x0 - xexact;
E= abs(e(N))/abs(e(N-1));
F=(abs(e(N)))/(abs(e(N-1)))*(abs(e(N-1)));
 end
 fprintf('%4g %10.7f %10.7f %10.7f\n',N, x, E, F)
 if abs(x-x0)<tol
 xroot=x;
 return
 end
 else N=N+1; x0=x;
 end
end
end
function [ N, xroot ] = q5_newtonraphson2( x0,Nmax,tol )
%Newton-Raphson iteration to solve F(x)=x^3 -8.5x^2 +20x -8=0 using the
%scheme xN=xN-1 - F(xN-1)/F'(xN-1)
%x0 is a suitable initial guess
%Iterate until either abs(xN-xN-1)<tol, or N=Nmax, whichever occurs first
%
%To investigate convergence, calculate E=abs(e(N)/e(N-1)) and
%F=abs(e(N)/e(N-1)^2) where eN is truncation error for xN.
%
syms x
N=1;
xexact=4;
f=inline('x^3 - 8.5*x^2 + 20*x -8');
df=inline('3*x^2 - 17*x +20');

fprintf(1, ' N xN |eN/eN-1| |eN/(eN-1)^2|
')

while N<=Nmax
    x=x0-[f(x0)/df(x0)];
    if N==1
        E='';F='';
    else
        e(N)= x - xexact;
        e(N-1)= x0 - xexact;
        E= abs(e(N))/abs(e(N-1));
        F=(abs(e(N)))/((abs(e(N-1)))*(abs(e(N-1))));
    end
    fprintf('%4g %10.7f %10.7f %10.7f
',N, x, E, F)
    if abs(x-x0)<tol
        xroot=x;
        return
    else N=N+1; x0=x;
    end
end
end