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.

Assessor comments in Red

0.1 Root Finding in One Dimension

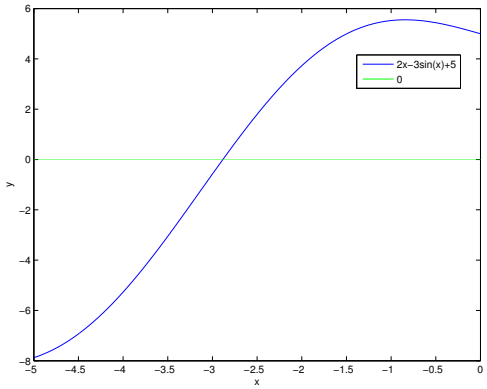
Question 1

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].

+1T for reasoning

```
>> 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');
```



+0.5C for graph

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 show 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 trunction error is guaranteed to be less than $\epsilon = 0.5$ x 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]

```
>> q2_binarysearch(-3,-2)
N= 18, x*= -2.8832359
>>
>> q2_binarysearch(-2,-3)
N= 18, x*= -2.8832359
```

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 $f(xlow)$ and $f(xhigh)$ have the same sign:

```
>> q2_binarysearch(0,5)
Choose xlow and xhigh with f(xlow)*f(xhigh)<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 δ for $|x| < \pi$.

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

$$F(x) = F(x_*) + (x - x_*)F'(x_*) + o((x - x_*)^2)$$

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

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 $\frac{-5\pi}{4} < x < \frac{-3\pi}{4}$, 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 x 10^{-5},

$$|\frac{1}{2}(m - n)| < 0.5 \text{ x } 10^{-5}$$

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

$$0.5 \text{ x } 10^{-5} + \frac{\delta}{4}$$

+1T for accuracy

of the actual root.

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 show on page 13, labelled

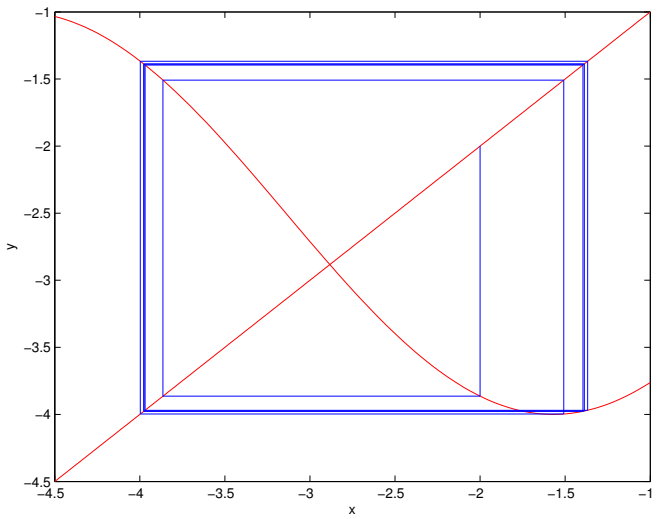
```
q3_fixedpoint1(fun,x0,Nmax)
```

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

```
>> f=inline('(3*sin(x)-5)/2');

q3_fixedpoint1( f,-2,10 );
   N      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
```

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



+1C for graph, although presentation could be improved by somehow indicating direction, or somehow calling attention to initial point

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,

$$\begin{aligned} x_{N+1} = f(x_N) &= f(x_*) + (x_N - x_*)f'(x_*) + ... \\ &\approx x_* + (x_N - x_*)f'(x_*) \end{aligned}$$

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 $x_* = -2.8832...$ Thus convergence does not occur.

+1T for explanation of divergence

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

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

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

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

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:

$$(i) -1 < \frac{-3+k}{2+k} < 1 \Rightarrow k > \frac{1}{2}$$

and

$$(ii) -1 < \frac{k}{2+k} < 1 \Rightarrow k > -1$$

+1T for convergence conditions

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

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

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

and

$$(ii) 0 < \frac{k}{2+k} \leq 1 \Rightarrow k > 0$$

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_{max}=20:

```
>> f=inline('(3*sin(x) + 4*x - 5)/6');
```

```
q3_fixedpoint1(f,-2,20 );
   N      xN     |eN/eN-1|
   1 -2.6213154
   2 -2.8294373   0.2054035
   3 -2.8731801   0.1869296
   4 -2.8813873   0.1839113
   5 -2.8828977   0.1833783
   6 -2.8831747   0.1832765
   7 -2.8832255   0.1832313
   8 -2.8832348   0.1830783
```

Clearly this convergence is monotonic, as illustrated by this graph given by program on page 14:

```
cobweb_k4
```

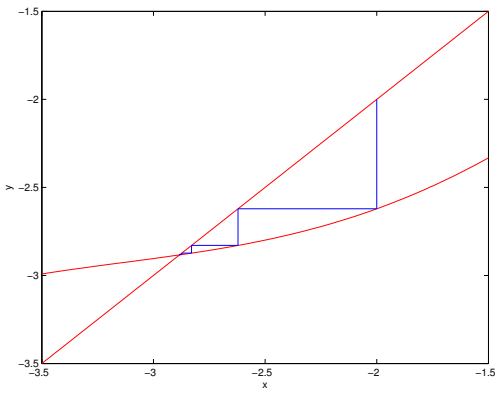


Figure 3: *A graph displaying the monotonic convergence near root for k=4*

(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_{max}=20:

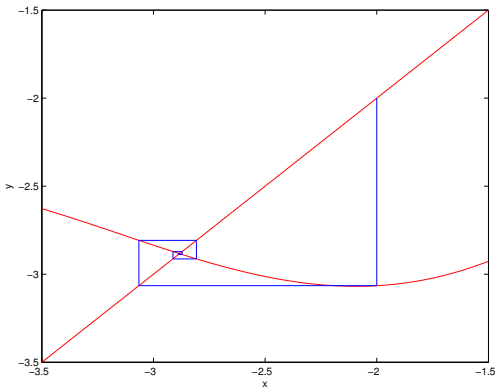
```
>>  f=inline('(6*sin(x) + 3*x - 10)/7');

q3_fixedpoint1(f,-2,20 );
   N      xN     |eN/eN-1|
   1 -3.0651121
   2 -2.8076818   0.4154227
   3 -2.9127840   0.3910668
   4 -2.8713224   0.4032384
   5 -2.8879884   0.3988001
   6 -2.8813332   0.4006404
```

```
   7 -2.8839982  0.3999170
   8 -2.8829322  0.4002026
   9 -2.8833588  0.4001024
  10 -2.8831881  0.4001080
  11 -2.8832564  0.4001921
  12 -2.8832291  0.3999427
  13 -2.8832400  0.4005820
  14 -2.8832356  0.3989789
```

Clearly this convergence is oscillatory, as illustrated by this graph given on page 14 by program:

```
cobweb_koneandhalf
```



+0.5T for correct discussion of monotonic/oscillatory convergence

+1.5C for example tables/plots

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 $N_{max} = 50$ due to slower convergence,

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

q3_fixedpoint1(f,-2,50);
   N      xN     |eN/eN-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  0.7332821
   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
```

```
21 -2.8819244  0.7277918
22 -2.8822816  0.7277811
23 -2.8825417  0.7277733
24 -2.8827309  0.7277675
25 -2.8828687  0.7277631
26 -2.8829689  0.7277596
27 -2.8830419  0.7277568
28 -2.8830949  0.7277542
29 -2.8831336  0.7277518
30 -2.8831617  0.7277491
31 -2.8831822  0.7277460
32 -2.8831971  0.7277421
33 -2.8832079  0.7277370
34 -2.8832158  0.7277302
```

Now

$$\begin{aligned} x_N = f(x_{N-1}) &= f(x_*) + (x_{N-1} - x_*)f'(x_*) + ... \\ &\approx x_* + (x_{N-1} - x_*)f'(x_*) \end{aligned}$$

Thus

$$\begin{aligned} x_N - x_{N-1} &\approx (x_* - x_{N-1}) + (x_{N-1} - x_*)f'(x_*) \\ &= (x_{N-1} - x_*)(f'(x_*) - 1) \\ &\approx (f'(x_*) - 1)\{\frac{x_N - x_*}{f'(x_*)}\} \end{aligned}$$

$$\Rightarrow (x_N - x_*) \approx (x_N - x_{N-1})\{\frac{f'(x_*)}{f'(x_*) - 1}\}$$

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 $x_* \approx -2.88323687$,

$$\frac{|f'(x_*)|}{|f'(x_*) - 1|} \approx 2.6731...$$

+1T for discussion of truncation error in case of slow convergence

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 $k = \frac{3}{2}$.

+0.5T for order of convergence

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

```
q4_fixedpoint2(x0, Nmax)
```

For N_{max}=1000, x_0=5,

```
>> q4_fixedpoint2(5,1000)
N= 740, xN= 4.0075342
```

+1C for correct code

Since $f(x) = x - h(F(x))$, then $f'(x) = 1 - h'(F(x))F'(x)$. Since $F'(x_*) = 0$, as it is a double root, $f'(x_*) = 1$.

Thus $x_* + f'(x_*)(x_{N-1} - x_*) = x_{N-1}$. So in the Taylor expansion of $x_N = f(x_{N-1})$, we need to consider higher orders,

$$\begin{aligned} x_N = f(x_{N-1}) &= f(x_*) + (x_{N-1} - x_*)f'(x_*) + \frac{1}{2}f''(x_*)(x_{N-1} - x_*)^2 + ... \\ &\approx x_{N-1} + \frac{1}{2}f''(x_*)(x_{N-1} - x_*)^2 \\ &= x_{N-1} + \frac{1}{2}f''(x_*)(e_{N-1})^2 \end{aligned}$$

Thus convergence will be slow at a multiple root for any choice of differentiable function h. Thus,

$$e_{N-1} \approx \pm \frac{|\sqrt{2(x_N - x_{N-1})}|}{|\sqrt{f''(x_*)}|}$$

If the iterations are terminated when $|x_N - x_{N-1}| < \epsilon$,

$$|e_N| \approx |\sqrt{\frac{2\epsilon}{f''(x_*)}}| = |\sqrt{\frac{40\epsilon}{7}}| \approx 0.007559$$

for $\epsilon = 10^{-5}$ and $f''(x_*) = -\frac{7}{20}$ for double root $x_* = 4$. 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 $C < 1$.

+1T for trunc error

Now it can be shown that the truncation error,

$$e_N \sim \frac{40}{7N} \text{ as } N \to \infty$$

+0.5T for slow convergence

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

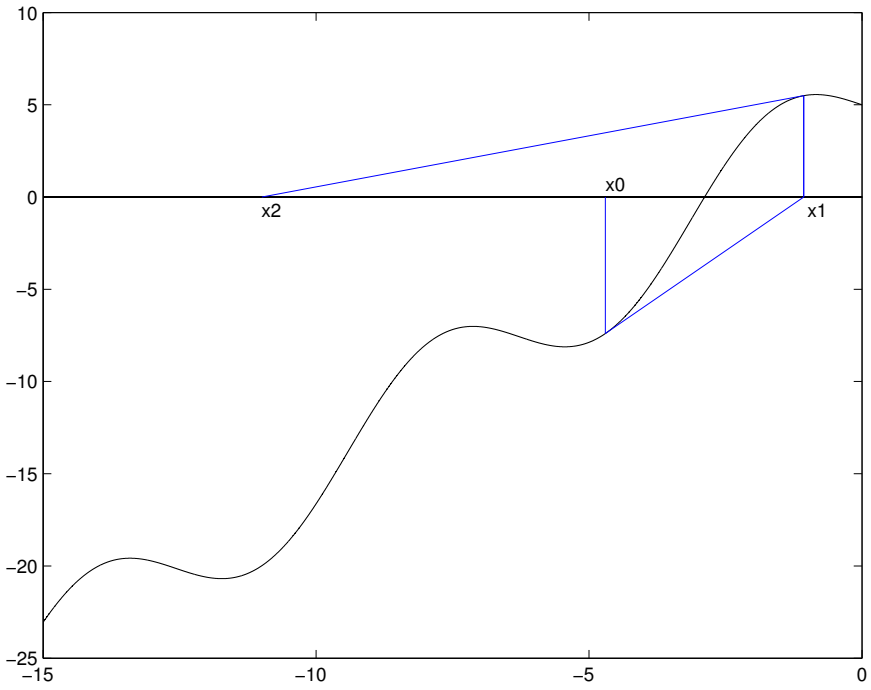
```
q5_newtonraphson1(x0,Nmax,tol)
```

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.



+1C for graph

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*

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

```
>> q5_newtonraphson1(-4,10,10^(-5));
    N       xN     |eN/eN-1|    |eN/(eN-1)^2|
   1 -2.6694018
   2 -2.8889594   0.0267613   0.1251491
   3 -2.8832394   0.0004411   0.0770849
   4 -2.8832369   0.0010137 401.5613254
```

For ϵ=10^{-10}, it converges after 5 iterations,

```
>>  q5_newtonraphson1(-4,10,10^(-10));
   N       xN     |eN/eN-1|    |eN/(eN-1)^2|
   1 -2.6694018
   2 -2.8889594   0.0267613   0.1251491
   3 -2.8832394   0.0004411   0.0770849
```

```
   4 -2.8832369   0.0010137 401.5613254
   5 -2.8832369   0.9998056 390735166.0308452
```

+1C for program and tables

Consider the convergence of this root,

$$
\begin{aligned}
e_N &= x_N - x_* \\
&= x_{N-1} - x_* - \frac{F(x_{N-1})}{F'(x_{N-1})} \\
&= e_{N-1} - \{\frac{F(x_*) + (x_{N-1}-x_*)F'(x_*) + \frac{1}{2}(x_{N-1}-x_*)^2F''(x_*) + ...}{F'(x_*) + (x_{N-1}-x_*)F''(x_*) + ...}\} \\
&\approx e_{N-1} - \{\frac{e_{N-1}F'(x_*) + \frac{1}{2}(e_{N-1})^2F''(x_*)}{F'(x_*) + e_{N-1}F''(x_*)}\} \\
&= e_{N-1} - \{e_{N-1}F'(x_*) + \frac{1}{2}(e_{N-1})^2F''(x_*)\}\frac{1}{F'(x_*)}\{1 - e_{N-1}\frac{f''(x_*)}{f'(x_*)} + ...\} \\
&= e_{N-1} - e_{N-1} - \frac{1}{2}(e_{N-1})^2\frac{F''(x_*)}{F'(x_*)} + o((e_{N-1})^3) \\
&\approx -\frac{1}{2}(e_{N-1})^2\frac{F''(x_*)}{F'(x_*)}
\end{aligned}
$$

+1T for convergence

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 ϵ=10^{-5} and ϵ=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, ϵ= 10^{-5}, it converges after 17 iterations,

```
>> q5_newtonraphson2(5,20,10^(-5));
   N      xN     |eN/eN-1|    |eN/(eN-1)^2|
   1  4.5500000
   2  4.2924855  0.5317919   0.9668944
   3  4.1516727  0.5185647   1.7729584
   4  4.0773792  0.5101725   3.3636414
   5  4.0391036  0.5053497   6.5308175
   6  4.0196592  0.5027471  12.8568067
   7  4.0098570  0.5013925  25.5042078
   8  4.0049354  0.5007011  50.7966100
   9  4.0024694  0.5003518 101.3801847
  10  4.0012352  0.5001762 202.5467114
  11  4.0006177  0.5000882 404.8794511
  12  4.0003089  0.5000441 809.5447759
  13  4.0001544  0.5000221 1618.8754020
  14  4.0000772  0.5000110 3237.5359322
  15  4.0000386  0.5000058 6474.8629634
  16  4.0000193  0.5000023 12949.4856425
  17  4.0000097  0.4999960 25898.5250833
```

Starting with x_0=5, ϵ= 10^{-10}, it converges after 26 iterations,

```
>> q5_newtonraphson2(5,50,10^(-10));
   N      xN     |eN/eN-1|    |eN/(eN-1)^2|
   1  4.5500000
   2  4.2924855  0.5317919   0.9668944
   3  4.1516727  0.5185647   1.7729584
   4  4.0773792  0.5101725   3.3636414
```

5	4.0391036	0.5053497	6.5308175
6	4.0196592	0.5027471	12.8568067
7	4.0098570	0.5013925	25.5042078
8	4.0049354	0.5007011	50.7966100
9	4.0024694	0.5003518	101.3801847
10	4.0012352	0.5001762	202.5467114
11	4.0006177	0.5000882	404.8794511
12	4.0003089	0.5000441	809.5447759
13	4.0001544	0.5000221	1618.8754020
14	4.0000772	0.5000110	3237.5359322
15	4.0000386	0.5000058	6474.8629634
16	4.0000193	0.5000023	12949.4856425
17	4.0000097	0.4999960	25898.5250833
18	4.0000048	0.4999806	51795.8621314
19	4.0000024	0.4998973	103578.4656453
20	4.0000012	0.4995168	207041.8028610
21	4.0000006	0.5023917	416869.6508807
22	4.0000003	0.4960420	819282.9404982
23	4.0000001	0.4823409	1606020.2896278
24	4.0000001	0.5162979	3564045.8045360
25	4.0000000	0.6370831	8518019.0793381
26	4.0000000	1.0000000	20986813.6448826

+0.5C for solution & table

For a double root, $F(x_*) = F'(x_*) = 0$. Consider the convergence of this root,

$$\begin{aligned} e_N &= e_{N-1} - \{\frac{\frac{1}{2}(x_{N-1}-x_*)^2 F''(x_*) + \ldots}{(x_{N-1}-x_*)F''(x_*) + \ldots}\} \\ &\approx e_{N-1} - \frac{1}{2}(x_{N-1}-x_*) \\ &\approx e_{N-1} - \frac{1}{2}e_{N-1} \\ &\approx \frac{1}{2}e_{N-1} \end{aligned}$$

+1T for convergence

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 $\frac{|e_N|}{(|e_{N-1}|}$ should converge to $\frac{1}{2}$, which is consistent with the results in the table above for ϵ=10^{-5} and ϵ=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 ϵ= 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 for rounding error

+1E for clear presentation

+1E for several observations made without explicit prompting

Totals: 8.0 computation; 10.0 theory; 2.0 excellence = 20 project marks

= 40 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')
%
elseif (abs(xhigh-xlow)<2*tol)
        xroot=xmid;
        N=n;
        %N=number of iterations to find root to required tol
        %x*=estimated root within required tol
        fprintf('N=%3g, x*=%10.7f\n', 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;
%
fprintf(1, '   N      xN    |eN/eN-1|\n')
%
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',N, x, E)

    if abs(x-x0)<tol
         xroot=x;
        return

    else N=N+1; x0=x;
        end
    end
end
```

(ii) cobweb_k4.m

```
% Generate the cobweb plot associated with the orbits x_n+1=f(x_n) for k=4.

x=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=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',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

    else N=N+1; x0=x;
        end
    end
end
```

(i) q5_newtonraphson2(x0, Nmax, tol)

```
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|\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

    else N=N+1; x0=x;
        end
    end
end
```