0.1

So that your candidate number can be added to each project, on the first page of each project write-up you should include the project number at the top on the left hand side and should leave a gap 11 cm wide by 5 cm deep in the top right hand corner. Your name or user identifier should not appear anywhere in the write-up (including any printouts), as the scripts are marked anonymously. Do not use green or red text in your reports. Leave a margin at least 2 cm wide at the left, and number each page, table and graph.

Root Finding in One Dimension: Answer

- This project is an optional, introductory, non-examinable project. Unlike the other projects **there are no Tripos marks awarded for it**.
- Unlike the other projects, you may collaborate as much as you like, and your College can arrange a supervision on the project.
- This document is a model answer for the project. The IB Manual recommends that in general **six** sides of A4 of text, *excluding in-line graphs, tables, etc.*, should be plenty for a clear concise report. Excluding in-line graphs, tables and printouts, this report comes to about five pages.
- This Introductory Project, like many other Projects, has a raw marking scheme where the maximum mark is 20. However, for each of the Core Projects and the Additional Projects, the maximum Tripos mark is 40. The Tripos mark is obtained by doubling the raw mark. You are reminded that a possible maximum of 160 Tripos marks are available for the Part IB Computational Projects course.
- Please email comments to `catam@maths.cam.ac.uk`.

0.1 Root Finding in One Dimension

Question 1

In order to find the roots of $2x + 5 - 3\sin x = 0$, it is sufficient to plot the range $[-4, -1]$ because

(i) $2x + 5 < -3 \leqslant -|3\sin x|$ for $x < -4$,

(ii) $2x + 5 > 3 \geqslant |3\sin x|$ for $x > -1$.

Hence there can be no root outside the range $[-4, -1]$. See figure 1 for plots of $2x + 5$ and $3\sin x$ in the range $[-4, -1]$; the program can be found on page 10.

Marking Scheme

$+\frac{1}{2}$ programming mark for program and graph.

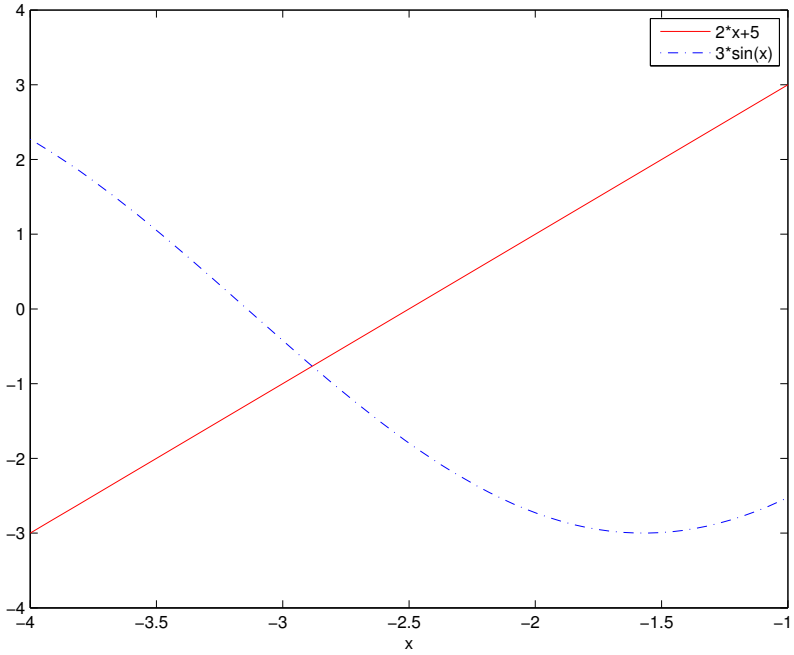

Figure 1: Plot of $2x + 5$ and $3\sin x$.

+1 theory mark for choice of bounds and other reasoning.

Moreover, there can be no root in $-4 < x < -\pi$ or $-5/2 < x < -1$ since $2x + 5$ and $3\sin x$ have opposite signs in these ranges. That leaves the interval $[-\pi, -5/2]$ where $2x + 5$ is increasing from negative to zero and $3\sin x$ is decreasing from zero to negative, giving exactly one intersection. We conclude that there is only one root of $2x + 5 - 3\sin x = 0$ and that it is in the interval $[-\pi, -5/2]$.

Binary Search: Programming Task

$+1\frac{1}{2}$ programming marks for program and output.

A program using binary search to solve $2x + 5 - 3\sin x = 0$ is listed on page 11. Results for a representative number of runs are listed in table 1.

Table 1: Binary search results

Initial Interval	Number of Iterations	Final Iterate	Bound on Truncation Error
$[-3.0, -2.0]$	18	$x_{18} = -2.8832359\ldots$	$\pm 0.0000038\ldots$
$[-\pi, -5/2]$	17	$x_{17} = -2.8832414\ldots$	$\pm 0.0000049\ldots$
$[-5\pi/4, -3\pi/4]$	19	$x_{19} = -2.8832397\ldots$	$\pm 0.0000030\ldots$

Comments

Use titles, etc. to differentiate parts of your answers.

Remember to append printouts of your programs.

Figures can be included in-line, overleaf on a separate page, or clumped at the end of your project (but in the latter case **you must include a page reference**).

Black and white figures are fine.

Make it obvious that you have done the multiple runs requested.

Question 2

Suppose the final interval is $[a, b]$. If the computed values of $F(a)$ and $F(b)$ are both greater in magnitude than $\delta > 0$, then we can be sure that the exact values of $F(a)$ and $F(b)$ have opposite sign, and the root x_* lies in the interval (a, b). If the computed value of $F(a)$, say, is less in magnitude than δ, then the exact value may have the other sign, but will also be less in magnitude than δ. Further,

$$F(a) \approx F'(x_*)\,(a - x_*)\ ,$$

and hence in this case

$$|a - x_*| \approx \left|\frac{F(a)}{F'(x_*)}\right| \leqslant \frac{\delta}{|F'(x_*)|}\ .$$

Give a mathematical justification of your reasoning, and use the hint in the question.

+1 theory mark for estimate of accuracy.

So the approximation $\frac{1}{2}(a+b)$ for the root has an error bounded, more-or-less, by

$$\tfrac{1}{2}(b-a) + \frac{\delta}{|F'(x_*)|}\ .$$

Given that $|F'(x)| > 4$ for $-5\pi/4 < x < -3\pi/4$, and since $x_* \in (-5\pi/4, -3\pi/4)$, it follows that if the iteration is terminated when $\frac{1}{2}(b-a) < 0.5 \times 10^{-5}$, the error is bounded by

$$0.5 \times 10^{-5} + \tfrac{1}{4}\delta\ .$$

Fixed-Point Iteration: Programming Task

A program using this method is listed on page 12.

Question 3

For $F(x) = 2x + 5 - 3\sin x$ and $k \neq -2$, we seek a solution to $F(x) = 0$ by means of the fixed-point iteration scheme

$$x_N = f(x_{N-1}) = \frac{3\sin x_{N-1} + k x_{N-1} - 5}{2 + k}\ . \tag{1}$$

Use paragraph headings to identify answers to different parts of a multi-part question.

Divergence. The first few iterations with $k = 0$ and $x_0 = -2$, are illustrated in figure 2. The iteration rapidly settles down to the two-point oscillation shown in figure 3. The program for figure 2 can be found on page 13; the program for figure 3 is almost identical and is omitted. For x_N close to x_*,

+1 programming mark for graphs.

$$\begin{aligned} x_N = f(x_{N-1}) &= f(x_*) + f'(x_*)\,(x_{N-1} - x_*) + \ldots \\ &= x_* + f'(x_*)\,(x_{N-1} - x_*) + \ldots\ . \end{aligned} \tag{2}$$

Thus, writing $\epsilon_N = x_N - x_*$,

$$\epsilon_N \approx f'(x_*)\ \epsilon_{N-1}\ . \tag{3}$$

+1 theory mark for explaining divergence.

It follows that the iteration will diverge if

$$|f'(x_*)| \equiv \left|\frac{3\cos x_* + k}{2 + k}\right| > 1 \tag{4}$$

This is the case for $x_* = -2.88323687\ldots$ and $k = 0$, which explains why the iteration in figures 2 and 3 does not converge. Conversely the iteration can converge if $|f'(x^*)| < 1$.

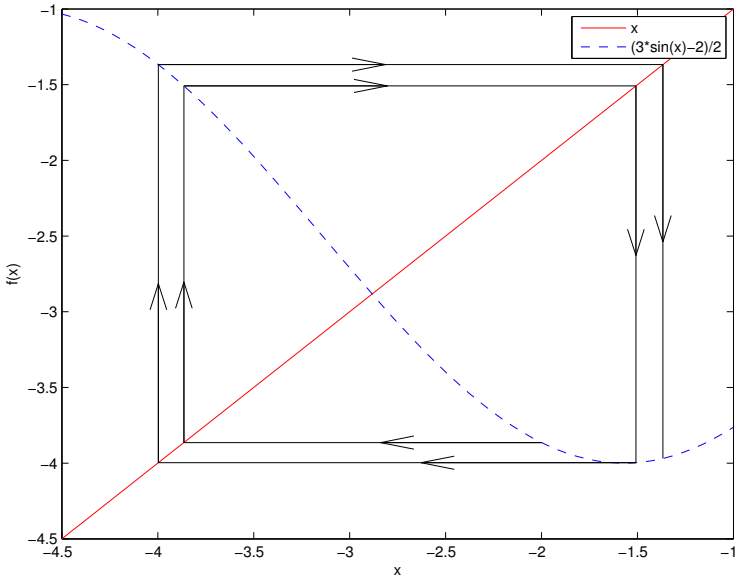

Figure 2: The first few iterations based on (1) with $k = 0$ and $x_0 = -2$.

Convergence. From the mean-value theorem we know that if a function f is continuous on the closed interval $[a, b]$, and differentiable on the open interval (a, b), there exists some $\xi \in (a, b)$ such that

+1 theory mark for explaining convergence and identifying $k > \frac{1}{2}$.

$$f(b) - f(a) = f'(\xi)(b - a)\,.$$

It follows that

$$x_N - x_* = f(x_{N-1}) - f(x_*) = f'(\xi)(x_{N-1} - x_*)\,,$$

for some ξ in (x_{N-1}, x_*). Hence if $|f'(\xi)| < 1$ for all $\xi \in [-\pi, -\pi/2]$, and if x_{N-1} is in this interval, then, since f is continuous and the interval closed, it follows that the iteration is a *contraction mapping*, the iterates will remain in this interval, and the iteration will converge. Further,

$$|f'(\xi)| = \left|\frac{(3\cos\xi + k)}{(2 + k)}\right| < 1 \quad \text{for all } \xi \in [-\pi, -\pi/2] \text{ if } k > \tfrac{1}{2}.$$

Thus convergence is guaranteed if $x_0 \in [-\pi, -\pi/2]$ and $k > \frac{1}{2}$.

+$\frac{1}{2}$ theory mark for explanation of type of convergence.

Monotonic/oscillatory convergence. Calculations also show that $f'(x_*)$ changes sign from negative to positive as k increases through $k_c \approx 2.9$. Given a sufficiently good initial guess, x_0, iterations using a value of k slightly greater/less than k_c should therefore yield rapid *monotonic/oscillatory* convergence since $f'(x_*)$ will be small and positive/negative. This is illustrated in tables 2 and 3.

Table 2: Iterates with $k = 3.5$ and $x_0 = -2$.

N	x_N	ϵ_N	$\epsilon_N/\epsilon_{N-1}$	$f'(x_N)$
0	-2.0000000	8.832369e-01		
1	-2.6777986	2.054383e-01	0.2325970	0.1485300
2	-2.8571507	2.608618e-02	0.1269782	0.1128263
3	-2.8803442	2.892679e-03	0.1108894	0.1094173
4	-2.8829210	3.159219e-04	0.1092143	0.1090560
5	-2.8832024	3.444623e-05	0.1090340	0.1090168
6	-2.8832331	3.755134e-06	0.1090144	0.1090125
7	-2.8832365	4.093556e-07	0.1090122	0.1090120

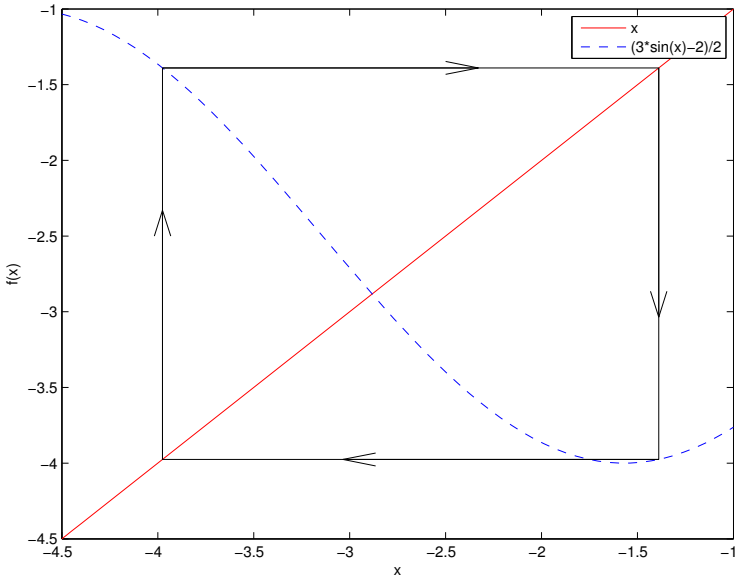

Figure 3: Later iterations based on (1) with $k = 0$ and $x_0 = -2$ illustrating a two-point oscillation.

Table 3: Iterates with $k = 2.5$ and $x_0 = -2$.

N	x_N	ϵ_N	$\epsilon_N/\epsilon_{N-1}$	$f'(x_N)$
0	-2.0000000	8.832369e-01		
1	-2.8284205	5.481637e-02	0.0620630	-0.0786852
2	-2.8878412	-4.604324e-03	-0.0839954	-0.0897628
3	-2.8828254	4.115123e-04	-0.0893752	-0.0889152
4	-2.8832735	-3.660414e-05	-0.0889503	-0.0889916
5	-2.8832336	3.257347e-06	-0.0889885	-0.0889848
6	-2.8832372	-2.898553e-07	-0.0889851	-0.0889854

Slower convergence and magnitude of truncation error. Alternatively taking $k = 16$ and $x_0 = -2$ results in the slower convergence illustrated in table 4. Further, we note

Table 4: Iterates with $k = 16$ and $x_0 = -2$.

N	x_N	ϵ_N	$\epsilon_N/\epsilon_{N-1}$	$f'(x_N)$
0	-2.0000000	8.832369e-01		
1	-2.2071051	6.761317e-01	0.7655158	0.7898504
2	-2.3736981	5.095388e-01	0.7536087	0.7689931
3	-2.5035020	3.797349e-01	0.7452521	0.7550165
4	-2.6023900	2.808468e-01	0.7395866	0.7458692
5	-2.6765887	2.066482e-01	0.7358039	0.7399189
⋮	⋮	⋮	⋮	⋮
30	-2.8831617	7.516634e-05	0.7277559	0.7277569
31	-2.8831822	5.470270e-05	0.7277553	0.7277560
32	-2.8831971	3.981016e-05	0.7277548	0.7277554
33	-2.8832079	2.897202e-05	0.7277545	0.7277549
34	-2.8832158	2.108451e-05	0.7277543	0.7277546

from (2) that

$$x_N - x_{N-1} \approx \left[f'\left(x_{N-1}\right) - 1\right]\left(x_{N-1} - x_*\right) \approx \left[f'\left(x_{N-1}\right) - 1\right]\frac{\left(x_N - x_*\right)}{f'\left(x_*\right)} \ ,$$

+1½ programming marks for programs and tables showing different types of convergence via $\epsilon_N/\epsilon_{N-1}$.

Pending *bonus* mark for including $f'(x_N)$ in tables.

Although not requested, include $f'(x_N)$ in tables 2, 3 and 4 as support for theory (with the aim of accruing a *bonus*).

Tables can be included in-line, overleaf on a separate page, or clumped at the end of your project (but in the latter case **you must include a page reference**).

i.e. that

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

+1 theory mark for explanation of the size of truncation error.

If the termination condition is that $|x_N - x_{N-1}| < \epsilon$, then the above implies that $\epsilon_N \equiv x_N - x_*$ may be larger than ϵ by a factor of approximately

$$\left| \frac{f'(x_*)}{f'(x_*) - 1} \right| .$$

+$\frac{1}{2}$ theory mark for explanation of first-order convergence using $\epsilon_N/\epsilon_{N-1}$.

For $k = 16$

$$\left| \frac{f'(x_*)}{f'(x_*) - 1} \right| \approx 2.673 .$$

First-order convergence. The result (3) implies that if $|f'(x_*)| < 1$ and $f'(x_*) \neq 0$ then fixed-point iteration should yield first-order convergence. Further

(i) the smaller $|f'(x_*)|$, the quicker should be the convergence;

(ii) if $|f'(x_*)| < \frac{1}{2}$ fixed-point iteration has a faster rate of convergence than the bisection method.

+$\frac{1}{2}$ *bonus* mark for comment about $|f'(x_*)|$.

The final two columns of the tables 2, 3 and 4 are consistent with these predictions.

Double Roots: Question 4

The fixed-point iteration program was modified to solve $x^3 - 8.5x^2 + 20x - 8 = 0$ by taking $f(x) = \frac{1}{20}(-x^3 + 8.5x^2 + 8)$ (see the program on page 14). The iterates starting with $x_0 = 4.5$ are given in table 5.

Include $7N\epsilon_N/40$, which should tend to 1, in table for a possible *bonus*.

Table 5: Fixed-point iteration to double root with $x_0 = 4.5$.

N	x_N	ϵ_N	$\epsilon_N/\epsilon_{N-1}$	$7N\epsilon_N/40$
0	4.5000000	5.000000e-01		
1	4.4500000	4.500000e-01	0.9000000	0.0787500
2	4.4100063	4.100063e-01	0.9111250	0.1435022
3	4.3771416	3.771416e-01	0.9198436	0.1979994
4	4.3495682	3.495682e-01	0.9268884	0.2446978
5	4.3260478	3.260478e-01	0.9327157	0.2852918
⋮	⋮	⋮	⋮	⋮
730	4.0075952	7.595174e-03	0.9986662	0.9702834
731	4.0075851	7.585057e-03	0.9986680	0.9703184
732	4.0075750	7.574966e-03	0.9986697	0.9703532
733	4.0075649	7.564903e-03	0.9986715	0.9703880
734	4.0075549	7.554867e-03	0.9986733	0.9704226
735	4.0075449	7.544857e-03	0.9986750	0.9704572
736	4.0075349	7.534874e-03	0.9986768	0.9704917

+1 programming mark for program and table.

Convergence. At a double root $F'(x_*) = 0$. Since

$$f'(x_*) = 1 - h'(F) F'(x_*) ,$$

+$\frac{1}{2}$ theory mark for explaining slow convergence.

it follows that if h is differentiable, then at a double root $f'(x_*) = 1$. From (3) this means that the iteration is on the boundary between convergence and divergence; indeed convergence in table 5 is clearly slower than before.

To understand this we take the Taylor expansion (2) to higher-order:

$$x_N = x_{N-1} + \tfrac{1}{2} f''(x_*) (x_{N-1} - x_*)^2 + \dots . \qquad (5)$$

It follows that

$$\epsilon_N = \epsilon_{N-1} + \tfrac{1}{2} f''(x_*)\, \epsilon_{N-1}^2 + \dots .$$

By seeking a solution of the form $\epsilon_N \sim \kappa/N$, we conclude that as $N \to \infty$,

$$\epsilon_N \sim -\frac{2}{f''(x_*)\, N} = \frac{40}{7N},$$

Identify expression that leads to $40/7$, and evaluate asymptote of ϵ_N for a possible *bonus*.

Hence convergence is slower than first-order since as $N \to \infty$

$$\frac{\epsilon_N}{\epsilon_{N-1}} \to 1 .$$

These two results are consistent with the final two columns of table 5.

Magnitude of error. It follows from (5) that

$$\epsilon_{N-1} \approx \pm\sqrt{\frac{2\,(x_N - x_{N-1})}{f''(x_*)}} .$$

+1 theory mark for identifying truncation error.

Thus, if the iteration is terminated once $|x_N - x_{N-1}| < \epsilon$, the truncation error in the root is given approximately by

$$|\epsilon_N| \approx \left|\sqrt{\frac{2\epsilon}{f''(x_*)}}\right| = \left|\sqrt{\frac{40\epsilon}{7}}\right| \approx 0.00756 \quad \text{if } \epsilon = 10^{-5} .$$

Pending *bonus* for including $7N\epsilon_N/40$ in table, in expanding hint and for evaluating asymptote of ϵ_N.

This is consistent with column two of table 5, and we conclude that the termination criterion does not ensure a truncation error in the root of less than $\epsilon = 10^{-5}$.

Newton-Raphson Iteration: Programming Task

Programs using Newton-Raphson iteration to calculate the root of $2x + 5 - 3\sin x = 0$ and the double root of $x^3 - 8.5x^2 + 20x - 8 = 0$ can be found on pages 15 and 16.

Question 5

Solution for the single root of $F(x) = 2x - 3\sin x + 5 = 0$. In the case of this equation, the first few Newton-Raphson iterations starting with $x_0 = -4.8$ are illustrated in figure 4; the program for this figure is listed on page 17. While the process does converge, if $\epsilon = 10^{-5}$ it does so only after 66 iterations.

+1 programming mark for graph.

If instead $x_0 = -4$, then as illustrated in the table 6, the process converges in 4 iterations so that $|x_N - x_{N-1}| < \epsilon = 10^{-5}$, and in 5 iterations so that $|x_N - x_{N-1}| < \epsilon = 10^{-10}$. Note that in this and earlier calculations we have used the value of x_5 as the *de facto* 'exact' solution to within rounding error.

Convergence. For x_{N-1} close to x_*,

+1 programming mark for program and table.

$$\begin{aligned}
x_N &= x_{N-1} - \frac{F(x_{N-1})}{F'(x_{N-1})} \\
&= x_{N-1} - \frac{F'(x_*)\,(x_{N-1} - x_*) + \frac{1}{2}F''(x_*)\,(x_{N-1} - x_*)^2 + \dots}{F'(x_*) + F''(x_*)\,(x_{N-1} - x_*) + \dots} \\
&= x_* + (x_{N-1} - x_*) - (x_{N-1} - x_*)\left[1 - \frac{F''(x_*)}{2F'(x_*)}\,(x_{N-1} - x_*) + \dots\right] \\
&= x_* + \frac{F''(x_*)}{2F'(x_*)}\,(x_{N-1} - x_*)^2 + \dots .
\end{aligned} \tag{6}$$

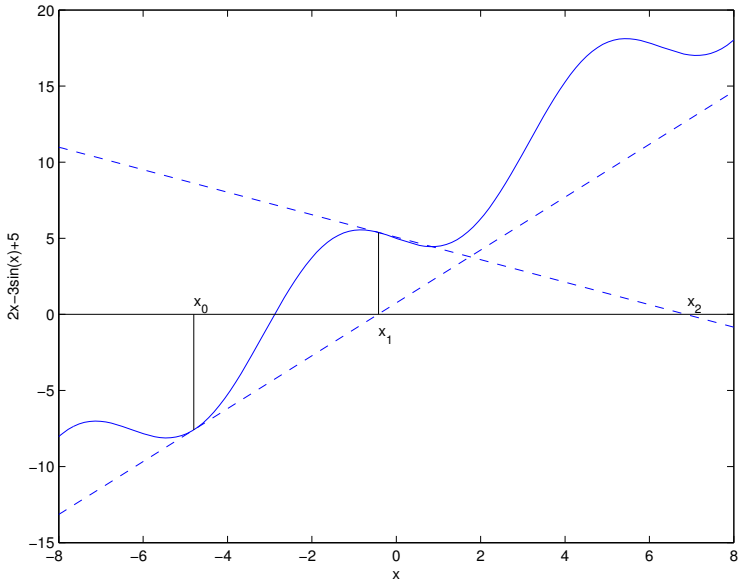

Figure 4: First few Newton-Raphson iterations for $F(x) = 2x - 3\sin x + 5 = 0$ starting with $x_0 = -4.8$.

Table 6: Newton-Raphson iteration to single root with $x_0 = -4$.

N	x_N	ϵ_N	$\epsilon_N / (\epsilon_{N-1})^2$	$\log\lvert\epsilon_N\rvert / \log\lvert\epsilon_{N-1}\rvert$	$F''(x_N)/2F'(x_n)$
0	-4.000000000000000	-1.116763e+00			
1	-2.669401797516753	2.138351e-01	0.1714576	-13.9680184	-0.1460399
2	-2.888959367133085	-5.722495e-03	-0.1251490	3.3472819	-0.0764422
3	-2.883239394297850	-2.521740e-06	-0.0770069	2.4965498	-0.0782039
4	-2.883236872558781	-4.969358e-13	-0.0781448	2.1977565	-0.0782047
5	-2.883236872558284	4.440892e-16	1.798e+09	1.2477978	-0.0782047

+1 theory mark for explaining convergence.

It follows that if the iterates converge,

$$\epsilon_N \sim K\,(\epsilon_{N-1})^2 \quad \text{as } N \to \infty\,, \quad \text{where } K = \frac{F''\,(x_*)}{2F'\,(x_*)} \approx -0.0782\,.$$

Evaluate K and include $\frac{F''(x_N)}{2F'(x_n)}$ in table 6 for a possible *bonus*.

Pending *bonus* for evaluating K, estimating size of error, and for comparison.

The method is thus second-order since

$$\frac{\log|\epsilon_N|}{\log|\epsilon_{N-1}|} \to 2 \quad \text{as } N \to \infty\,.$$

These two results are, on the whole, consistent with the first four entries in the final two columns of table 6. However, the fifth entry in the penultimate column shows a variation because of rounding error.

Compare with fixed-point iteration and the bisection method for a possible *bonus*.

We observe that convergence for Newton-Raphson iteration is, in this case, much faster than fixed-point iteration or the bisection method.

Magnitude of error and rounding error. From (6),

$$\epsilon_{N-1} \sim -(x_N - x_{N-1})\,.$$

It follows that once the iteration has converged, and assuming no rounding error,

$$|\epsilon_N| \sim |K\,(\epsilon_{N-1})^2\,| \sim |K(x_N - x_{N-1})^2| \leqslant |K|\epsilon^2\,.$$

Table 7: Newton-Raphson iteration to double root with $x_0 = 5$ and $\epsilon = 10^{-5}$.

N	x_N	ϵ_N	$\epsilon_N/\epsilon_{N-1}$	$\log\lvert\epsilon_N\rvert/\log\lvert\epsilon_{N-1}\rvert$
0	5.000000000	1.000000e+00		
1	4.550000000	5.500000e-01	0.5500000	
2	4.292485549	2.924855e-01	0.5317919	2.0563130
3	4.151672687	1.516727e-01	0.5185647	1.5341813
4	4.077379237	7.737924e-02	0.5101725	1.3568375
5	4.039103573	3.910357e-02	0.5053497	1.2667037
6	4.019659207	1.965921e-02	0.5027471	1.2121423
7	4.009856979	9.856979e-03	0.5013925	1.1757010
8	4.004935400	4.935400e-03	0.5007011	1.1497423
9	4.002469436	2.469436e-03	0.5003518	1.1303713
10	4.001235153	1.235153e-03	0.5001762	1.1153934
11	4.000617686	6.176855e-04	0.5000882	1.1034816
12	4.000308870	3.088700e-04	0.5000441	1.0937893
13	4.000154442	1.544418e-04	0.5000221	1.0857526
14	4.000077223	7.722263e-05	0.5000111	1.0789824
15	4.000038612	3.861176e-05	0.5000057	1.0732019
16	4.000019306	1.930602e-05	0.5000038	1.0682093
17	4.000009653	9.652994e-06	0.4999991	1.0638547

If $\epsilon = 10^{-5}$ then $|\epsilon_N|$ is bounded by $|K|\epsilon^2 \approx 7.8 \times 10^{-12}$, which is consistent with the results in column 2 of table 6. If $\epsilon = 10^{-10}$ then, after the fifth iteration, rounding error dominates the above estimate since MATLAB's double precision has an accuracy of about 10^{-16} (again see table 6).

+½ programming mark for program and table.

Solution for the double root of $F(x) = x^3 - 8.5x^2 + 20x - 8 = 0$. In the case of this equation, the first few Newton-Raphson iterations starting with $x_0 = 5$ are given in table 7.

Convergence. For $x_* = 4$, $F'(x_*) = 0$ and $F''(x_*) \neq 0$, implying from result (6) above that

Compare with the bisection method for a possible *bonus*.

$$\epsilon_N \sim \tfrac{1}{2}\epsilon_{N-1}\,.$$

+1 theory mark for explaining convergence.

Hence the last two columns of table 7 should tend to $\frac{1}{2}$ and 1 (as is indeed the case). We note that in this case Newton-Raphson iteration has the same rate of convergence as the bisection method, and that from (6)

Pending *bonus* for bisection comparison.

$$x_N - x_{N-1} \sim -\tfrac{1}{2}\epsilon_{N-1}\,.$$

Include calculation of rounding error for a possible *bonus*.

Rounding error. If ϵ is reduced from 10^{-5} to 10^{-10}, the iteration continues as in table 8. The larger-than-predicted error in the final iterate arises because division by the small $F'(x_{N-1})$ amplifies the rounding error in $F(x_{N-1})$. More precisely, suppose that the rounding error in evaluating a function is δ. Then it follows from the definition of $F(x)$ that close to the double root $x_* = 4$ (where $F(x_*) = F'(x_*) = 0$ and $F''(x_*) = 7$), that in a computation

+½ theory mark for general explanation.

$$F(x_{N-1}) = F(x_* + \epsilon_{N-1}) \sim \tfrac{7}{2}\epsilon_{N-1}^2 + \delta \quad \text{and} \quad F'(x_{N-1}) \sim 7\epsilon_{N-1} + \delta\,.$$

+½ *bonus* mark for pending *bonuses* and rounding-error calculation.

Hence, while in exact arithmetic

$$\epsilon_N = \epsilon_{N-1} - \frac{F(x_{N-1})}{F'(x_{N-1})}\,,$$

Table 8: Newton-Raphson iteration to double root with $x_0 = 5$ and $\epsilon = 10^{-10}$.

N	x_N	ϵ_N	$\epsilon_N/\epsilon_{N-1}$	$\log\lvert\epsilon_N\rvert/\log\lvert\epsilon_{N-1}\rvert$
18	4.000004826	4.826398e-06	0.4999897	1.0600237
19	4.000002413	2.413255e-06	0.5000116	1.0566212
20	4.000001206	1.206080e-06	0.4997730	1.0536240
21	4.000000602	6.017966e-07	0.4989692	1.0510129
22	4.000000302	3.015609e-07	0.5011010	1.0482393
23	4.000000147	1.467238e-07	0.4865479	1.0479823
24	4.000000064	6.370571e-08	0.4341880	1.0530215
25	4.000000032	3.183852e-08	0.4997750	1.0418612
26	3.999999968	-3.192455e-08	-1.0027018	0.9998437
27	3.999999968	-3.192455e-08	1.0000000	1.0000000

on a computer (and on the assumption that $|\delta| \ll |\epsilon_{N-1}|$)

$$\begin{aligned}\epsilon_N &\sim \epsilon_{N-1} - \frac{\frac{7}{2}\epsilon_{N-1}^2 + \delta}{7\epsilon_{N-1} + \delta} \\ &\sim \tfrac{1}{2}\epsilon_{N-1} - \frac{\delta}{7\epsilon_{N-1}} + \dots .\end{aligned}$$

+1 *bonus* mark for quality of write-up.

Rounding error will be comparable with the update when $\epsilon_{N-1} = O(\delta^{\frac{1}{2}})$. Table 8 is consistent with this result since, as noted above, MATLAB's double precision has an accuracy of about 10^{-16}.

Max raw mark: 20

Max Tripos mark: 40=20x2

Program `0-1/question1.m` to produce figure 1

```
% Plot 2*x+5 and 3*sin(x)
%
fplot(@(x)[2*x+5],[-4 -1 -4 4],'r');
hold on;
fplot(@(x)[3*sin(x)],[-4 -1 -4 4],'b-.');
xlabel('x'); legend('2*x+5','3*sin(x)');
hold off;
%
% Output plot first in colour postscript and then as a pdf
% to leave options open for inclusion in write-up.
%
print -depsc 'question1';
print -dpdf  'question1';
```


Program 0-1/question2.m for question 2

```
% Bisection method to solve 2.0*x-3.0*sin(x)+5.0=0
%
% Define an anonymous function
% Set the tolerance for convergence
%
f=@(x)2.0*x-3.0*sin(x)+5.0;
tol=0.5e-5;
%
% Read in lower guess and higher guess
% Check that interval includes a zero and is increasing
%
yl=1.0; yu=1.0;
while yl*yu > 0
  xl=input('Please enter the lower end of initial interval [-3]: ');
  if isempty(xl)
    xl=-3.0;
  end
  xu=input('Please enter the upper end of initial interval [-2]: ');
  if isempty(xu)
    xu=-2.0;
  end
  yl=f(xl);
  yu=f(xu);
  if yl*yu >= 0
    disp('Chosen interval does not include a zero')
  elseif yl >= yu
    yl=1.0; yu=1.0;
    disp('Lower guess greater than the upper guess')
  end
end
%
% Start iteration & while xu-xl>2*tol keep interval halving
%
fprintf('\nStarting iteration\n\n')
n=0;
while( (xu-xl) > 2.0*tol )
  n=n+1;
  xm=0.5*(xl+xu);
  fprintf('%2d %11.7f %13.6e %13.6e %13.6e\n',n,xm,(xu-xl)/2,yl,yu)
  ym=f(xm);
  if yl*ym < 0
    xu=xm;
    yu=ym;
  else
    xl=xm;
    yl=ym;
  end
end
%
% xu-xl<2*tol; calculate xm (xu-xm<tol and xm-xl<tol); output results
%
n=n+1;
xm=0.5*(xl+xu);
fprintf('%2d %11.7f %13.6e %13.6e %13.6e\n',n,xm,(xu-xl)/2,yl,yu)
fprintf('\nRoot is within %10.3e of %11.7f\n',tol,xm)
```


Program `0-1/picard1.m` for question 3

```
% Fixed-point interation method to solve 2*x-3*sin(x)+5=0
%
% Define an anonymous function
% Hard wire the solution to 15 decimal places (obtained by NR iteration)
% Set the tolerance for convergence
% Set the maximum number of iterations
%
f=@(x,k)(3*sin(x)+k*x-5)/(2+k);
fd=@(x,k)(3*cos(x)+k)/(2+k);
xexact = -2.8832368725582835;
tol = 1e-5;
nmax = 50;
%
% Input k and x
%
k=input('Please enter k [3.5]: ');
if isempty(k)
  k=3.5;
end
x=input('Please enter initial guess [-2]: ');
if isempty(x)
  x=-2;
end
%
% Initialise the iteration
% Output initial guess and error
%
n = 0;
err = 1;
epnew=x-xexact;
fprintf('%2d %10.7f %13.6e\n',n,x,epnew)
%
% Start the iteration
% Terminate when difference between successive guesses < tol
%
while(err>=tol & n<nmax)
   n = n+1;
   y = f(x,k);
   epold=epnew;
   epnew=y-xexact;
   fprintf('%2d %10.7f %13.6e %10.7f %10.7f\n',n,y,epnew,epnew/epold,fd(y,k));
   err = abs(y-x);
   x = y;
end
```


Program `0-1/question3_1plot.m` **to produce figure 2**

```
% Matlab plot for Figure 2, illustrating first few iterations:

k=0;
f=@(x)(3*sin(x)+k*x-5)/(2+k);
fplot(@(x)[x],[-4.5 -1 -4.5 -1],'r');
hold on;
fplot(f,[-4.5 -1 -4.5 -1],'b--');
legend('x','(3*sin(x)-2)/2');

% Find first few iterations explicitly: (Alternatively could simply take the
% first few values output directly by picard1.m)

niter=10;
x0=-2.0; xnew=x0;
for i=1:niter
  x(i)=xnew;
  x(i+1)=f(x(i));
  xnew=x(i+1);
end

% 'Join the dots', to illustrate behaviour of iterates

xiter = [x(1) x(2) x(2) x(3) x(3) x(4) x(4) x(5) x(5)] ;
fiter = [x(2) x(2) x(3) x(3) x(4) x(4) x(5) x(5) x(6)];
plot(xiter,fiter,'black')

% Label graph and distinguish curves:

xlabel('x');
ylabel('f(x)')

% If we want to put arrows to indicate direction of iteration
% one way is the following:
dd1=diff(xiter);
dd2=diff(fiter);
xx1=xiter(1:end-1);
ff1=fiter(1:end-1);
quiver(xx1,ff1,dd1,dd2,'black');

hold off;
%
% Output plot first in colour postscript and then as a pdf
% to leave options open for inclusion in write-up.
%
print -depsc 'question3-1';
print -dpdf  'question3-1';
```


Program `0-1/picard2.m` for question 4

```
% Fixed-point interation method to solve x^3 - 8.5x^2 + 20x - 8 = 0
%
% Define an anonymous function
% Hard wire the solution
% Set the tolerance for convergence
% Set the maximum number of iterations
%
f=@(x)(-x^3+8.5*x^2+8)/20;
xexact = 4;
tol = 1e-5;
nmax = 1000;
%
% Input x
%
x=input('Please enter initial guess [4.5]: ');
if isempty(x)
  x=4.5;
end
%
% Initialise the iteration
% Output initial guess and error
%
n = 0;
err = 2*tol;
epnew=x-xexact;
fprintf('%3d %10.7f %13.6e\n',n,x,epnew)
%
% Start the iteration
% Terminate when difference between successive guesses < tol
%
while(err>=tol & n<nmax)
   n = n+1;
   y = f(x);
   epold=epnew;
   epnew=y-xexact;
   if n <= 5 || n >= 730
     fprintf('%3d %10.7f %13.6e %10.7f',n,y,epnew,epnew/epold);
     fprintf(' %10.7f\n',7*n*epnew/40);
   end
   err = abs(y-x);
   x = y;
end
```


Program `0-1/newton1.m` for the first part of question 5

```
% Newton-Raphson interation method to solve F(x)=2*x-3*sin(x)+5=0
%
% Define an anonymous function
% Hard wire the solution to 15 decimal places (obtained by NR iteration)
% Set the maximum number of iterations
%
f=@(x)2*x-3*sin(x)+5;
fd=@(x)2-3*cos(x);
fdd=@(x)3*sin(x);
xexact=-2.883236872558284;
nmax = 100;
%
% Input initial guess and tolerance for convergence
%
x=input('Please enter initial guess [-4]: ');
if isempty(x)
  x=-4;
end
tol=input('Please enter initial tolerance [1e-5]: ');
if isempty(tol)
  tol = 1e-5;
elseif tol < 0
  tol = -tol;
end
%
% Initialise the iteration
% Output initial guess and error
%
n = 0;
err = 2*tol;
epnew=x-xexact;
fprintf('%2d %20.15f %13.6e\n',n,x,epnew)
%
% Start the iteration
% Terminate when difference between successive guesses < tol
%
while(err>=tol & n<nmax)
   n = n+1;
   y = x - f(x)/fd(x);
   epold=epnew;
   epnew=y-xexact;
   if abs(epnew/(epold*epold)) < 1
     fprintf('%2d %20.15f %13.6e %11.7f',n,y,epnew,epnew/(epold*epold));
   else
     fprintf('%2d %20.15f %13.6e %11.4g',n,y,epnew,epnew/(epold*epold));
   end
   fprintf(' %11.7f'  ,log(abs(epnew))/log(abs(epold)))
   fprintf(' %10.7f\n',fdd(y)/(2*fd(y)))
   err = abs(y-x);
   x = y;
end
```


Program `0-1/newton2.m` for the second part of question 5

```
% Newton-Raphson interation method to solve F(x)=x^3-8.5*x^2+20*x-8=0
%
% Define an anonymous function
% Hard wire the exact solution and set the maximum number of iterations
%
f=@(x)x*x*x-8.5*x*x+20*x-8;
fd=@(x)3*x*x-17*x+20;
xexact=4;
nmax = 50;
%
% Input initial guess and tolerance for convergence
%
x=input('Please enter initial guess [5]: ');
if isempty(x)
  x=5;
end
tol=input('Please enter initial tolerance [1e-5]: ');
if isempty(tol)
  tol = 1e-5;
elseif tol < 0
  tol = -tol;
end
%
% Initialise the iteration
% Output initial guess and error
%
n = 0;
err = 2*tol;
epnew=x-xexact;
fprintf('%2d %12.9f %13.6e\n',n,x,epnew)
%
% Start the iteration
% Terminate when difference between successive guesses < tol
%
while(err>=tol & n<nmax)
   n = n+1;
   y = x - f(x)/fd(x);
   epold=epnew;
   epnew=y-xexact;
   fprintf('%2d %12.9f %13.6e %10.7f',n,y,epnew,epnew/epold);
   if n > 1
     fprintf(' %10.7f\n',log(abs(epnew))/log(abs(epold)));
   else
     fprintf('\n');
   end
   err = abs(y-x);
   x = y;
end
```


Program `0-1/question5_plot.m` **to produce figure 4**

```
% Matlab plot for Figure 4, illustrating first few Newton-Raphson iterations:

x0  =  -4.8;
f   =  @(x)(2*x-3*sin(x)+5);
f1  =  @(x)(f(x0)-(x0-x)*(2-3*cos(x0)));  % tangent at x0
fplot(f,[-8 8 -15 20]);
xlabel('x');
ylabel('2x-3sin(x)+5')
%legend('2x-3sin(x)+5');
hold on;
fplot(f1,[-8 8 -15 20],'--');
fplot(@(x)[0*x],[-8 8 -15 20],'black');

% Show the first few iterations:

x1=-0.42; % next iterate
f2= @(x)(f(x1)-(x1-x)*(2-3*cos(x1)));  % tangent at x1
fplot(f2,[-8 8 -15 20],'--');

x2=6.9;  % next iterate

plot([x0 x0],[0 f(x0)],'black');
plot([x1 x1],[0 f(x1)],'black');

% Put text labels on graph:
str0=('x_0');
str1=('x_1');
str2=('x_2');
text(x0,0.5,str0);
text(x1,-1.5,str1);
text(x2,0.5,str2);

hold off;
%
% Output plot first in colour postscript and then as a pdf
% to leave options open for inclusion in write-up.
%
print -depsc 'question5';
print -dpdf  'question5';
```

