Root Finding in One Dimension

Jamie Hiley

April 8, 2020

Contents

1 Introduction and definitions

We study the roots of

$$F(x) \equiv 2x - 3\sin x + 5 = 0 \tag{†}$$

and

$$F(x) \equiv x^3 - 8.5x^2 + 20x - 8 = 0\,. \tag{‡}$$

1.1 Question 1

Since

$$x > -1 \quad \Rightarrow \quad F(x) > 2\cdot(-1) - 3\cdot(1) + 5 = 0$$

and

$$x < -4 \quad \Rightarrow \quad F(x) < 2\cdot(-4) - 3\cdot(-1) + 5 = 0\,,$$

we know for any root x_* of (†), $x_* \in (-4,-1)$.

Plotting $F(x)$ on this domain in Figure 1 shows there is only one such root.

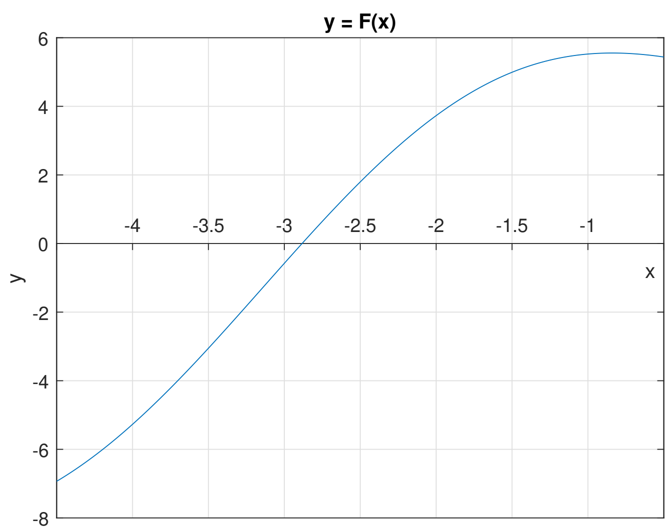

FIGURE 1.0

Now throughout this document the approximation $x_* \simeq -2.883236872558284$ returned by fzero.m, MATLAB's built in root finder, will been used when a measure of precision is necessary. It is accurate to at least 8 dp.

2 Binary Search

2.1 Programming Task

The program written is bisectionF.m ; it is a MATLAB function which takes input of 2 numbers. If x_* lies between these inputs a 2D vector $\mathbf{a}$ is output. Here a_1 is the approximation of x_* and a_2

is the number of iterations required.

To test the function, it was run for 2 random inputs x_1, $x_2 \in [-13, 7]$ several times. Inputs for which $F(x_1)F(x_2) > 0$ were excluded.

x_1 to 4 dp	x_2 to 4 dp	a_1 to 9 dp	a_2	$\lvert x_* - a_1 \rvert$ to 8 dp ($\times 10^{-7}$)
1.1209	-12.3633	-2.883236877	23	0.1
-11.0574	3.4692	-2.883236877	23	0.1
0.8966	-6.6580	-2.883235180	22	16.9
6.0044	-12.3111	-2.883236778	23	0.9
-4.0883	-0.0737	-2.883238623	21	17.5
-7.4795	0.5941	-2.883237372	22	5.0
0.1020	-9.7478	-2.883236786	22	0.8
6.1949	-6.1923	-2.883236990	23	1.1
-1.2946	-8.5238	-2.883237577	22	7.1
2.0253	-7.8981	-2.883237049	22	1.8

FIGURE 2.0

Hence we conclude bisectionF.m is working as intended, in particular to sufficient accuracy.

2.2 Question 2

If $F(x)$ is calculated to be 0 for some $x \in (-\pi, \pi)$ then we have $|F(x)| < \delta$. Now since $F(x_*) = 0$,

$$\begin{aligned}\delta &> |F(x) - F(x_*)| \\ &= |(x - x_*)F'(x_*)| + o\left(|x - x_*|\right)\end{aligned}$$

Now since F is continuous $|x - x_*|$ is small for sufficiently small δ. Therefore $o\left(|x - x_*|\right)$ terms are negligible. Furthermore as $x_* \in (-\frac{5\pi}{4}, -\frac{3\pi}{4})$, $|F'(x_*)| > 4$. So we conclude that for sufficiently small δ,

$$\delta > 4|x - x_*| \quad \Rightarrow \quad |x - x_*| < \frac{\delta}{4}$$

Hence we expect accuracy of at least $\dfrac{\delta}{4}$.

3 Fixed-Point Iteration

3.1 Programming Task

The program written is FPiteration.m; it is a MATLAB function which takes input of 3 numbers. It runs for an externally defined f. The inputs are ϵ, N_{max} and x_0, in that order. It has a single output: the final approximation to x_*.

3.2 Question 3

(i) Running FPiteration$(10^{-5}\,,\,10\,,\,-2)$ for $k=0$ yields the data in Figure 3.0.

Plots $y = f(x)$ and $y = x$ are included in Figure 3.1, as well as the relevant segments of $x = x_n$ and $y = f(x_n)$ for $n = 0\,,\,1\,,\,\dots\,10\,.$

n	x_n to 8 dp
0	-2.00000000
1	-3.86394614
2	-1.50827169
3	-3.99706896
4	-1.36767492
5	-3.96916251
6	-1.39556627
7	-3.97702969
8	-1.38761535
9	-3.97490384
10	-1.38975706

FIGURE 3.0

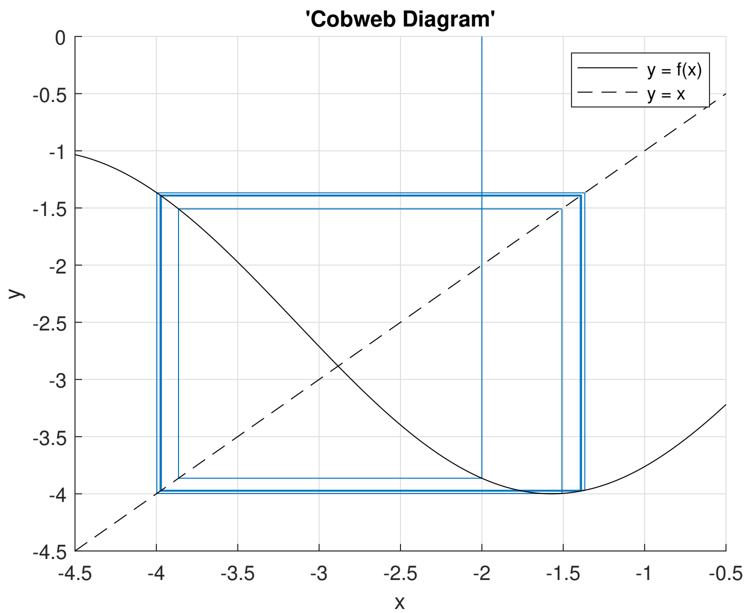

FIGURE 3.1

Figures 3.0 and 3.1 demonstrate that for $k = 0\,,\,x_n$ orbits indefinitely around $(x_*\,,\,x_*)$, approaching the period 2 solution to $x_n = f(x_n)\,.$

Let $\varepsilon_n = x_n - x_*$, then

$$\begin{aligned} \varepsilon_{n+1} &= x_{n+1} - x_* \\ &= f(x_n) - f(x_*) \\ &= \varepsilon_n f'(x_*) + o(\varepsilon_n) \\ \Rightarrow \qquad |\varepsilon_{n+1}| &= |\varepsilon_n||f'(x_*)| + o(\varepsilon_n) \end{aligned}$$

Hence the criterion for convergence is $|f'(x_*)| < 1$ for x_0 sufficiently close x_*, while if $|f'(x_*)| > 1\,,\,x_n$ will diverge [1, Thm 3.6].

Now for $k = 0\,,$

$$\begin{aligned} |f'(x_*)| &= |\tfrac{3}{2}\cos x_*| \\ &= 1.45\dots \\ &> 1 \end{aligned}$$

so $f(x_n)$ will not converge to $x_*\,.$

(ii) Let $J = (-\pi\,,\,-\frac{\pi}{2})\,.$ Using the mean value theorem, $\exists c \in (x_*\,,\,x_n)$ or $(x_n\,,\,x_*)$ such that

$$
\begin{aligned}
f'(c) &= \frac{f(x_*) - f(x_n)}{x_* - x_n} \\
\Rightarrow \qquad \varepsilon_{n+1} &= \varepsilon_n f'(c)
\end{aligned}
$$

If $x_n \in J$, then $c \in J$, therefore $\forall x \in J$, $|f'(x)| < 1$ is sufficient to say x_n converges.

$$
\begin{aligned}
& |f'(x)| < 1 \\
\Leftrightarrow \qquad & -1 < \frac{3\cos x + k}{2+k} < 1 \\
\Leftrightarrow \qquad & 0 < (2+k)\left[1 - \tfrac{3}{2}\cos x\right] < (2+k)^2.
\end{aligned}
$$

Now $1 - \frac{3}{2}\cos x$ has range $(1\,,\,\frac{5}{2})$ on $x \in J$, so our inequalities are satisfied by

$$
\begin{aligned}
& 2+k > 0 \quad \text{and} \quad (2+k)^2 > (2+k)\tfrac{5}{2} \\
\Leftrightarrow \qquad & 2+k > 0 \quad \text{and} \quad 2+k > \tfrac{5}{2} \\
\Leftrightarrow \qquad & k > \tfrac{1}{2}
\end{aligned}
$$

Hence we have convergence if $x_0 \in J$ and $k > \frac{1}{2}$.

[I've actually shown $|f'(x_*)| \quad \Leftrightarrow \quad k > 0.4502\ldots$ which would give me a necessary condition as well as a sufficient one, but I can't work out if its possible to guarantee convergence $\forall x_0 \in J$ for these k.]

(iii) As before

$$\varepsilon_{n+1} = \varepsilon_n f'(x_*) + o(\varepsilon_n)$$

so, for sufficently small ε_n, $f'(x_*) > 0 \Rightarrow \varepsilon_n\varepsilon_{n+1} > 0$ and $f'(x_*) < 0 \Rightarrow \varepsilon_n\varepsilon_{n+1} < 0$.

Hence, once ε_n is sufficiently small, we have monotonic convergence for $f'(x_*) > 0$ and $k > \frac{1}{2}$, e.g. $k = 4$, and oscillatory convergence for $f'(x_*) < 0$ and $k > \frac{1}{2}$, e.g. $k = 2$.

The results of running $\mathsf{FPiteration}(10^{-5}\,,\,20\,,\,-2)$ for $k = 4$ and $k = 2$ are shown in Figure 3.2 and Figure 3.3 respectively, demonstrating the expected behaviour.

n	x_n to 6 dp	$x_n - x_*$ to 6 dp	$\lvert\varepsilon_n/\varepsilon_{n-1}\rvert$ to 6 dp
0	-2.000000	0.883237	n/a
1	-2.621315	0.261921	0.296547
2	-2.829437	0.053800	0.205403
3	-2.873180	0.010057	0.186930
4	-2.881387	0.001850	0.183911
5	-2.882898	0.000339	0.183379
6	-2.883175	0.000062	0.183283
7	-2.883225	0.000011	0.183265
8	-2.883235	0.000002	0.183262

FIGURE 3.2

n	x_n to 6 dp	$x_n - x_*$ to 6 dp	$\lvert\varepsilon_n/\varepsilon_{n-1}\rvert$ to 6 dp
0	-2.000000	0.883237	n/a
1	-2.931973	-0.048736	0.055179
2	-2.872052	0.011184	0.229490
3	-2.885742	-0.002506	0.224022
4	-2.882672	0.000565	0.225348
5	-2.883364	-0.000127	0.225054
6	-2.883208	0.000029	0.225121
7	-2.883243	-0.000006	0.225106
8	-2.883235	0.000001	0.225109

FIGURE 3.3

(iv) Running FPiteration$(10^{-5}, 50, -2)$ for $k = 16$ gives the results in Figure 3.4.

n	x_n to 6 dp	$x_n - x_*$ to 6 dp	$\lvert\varepsilon_n/\varepsilon_{n-1}\rvert$ to 6 dp
0	-2.000000	0.883237	n/a
1	-2.207105	0.676132	0.765516
2	-2.373698	0.509539	0.753609
3	-2.503502	0.379735	0.745252
⋮	⋮	⋮	⋮
32	-2.883197	0.000040	0.727755
33	-2.883208	0.000029	0.727755
34	-2.883216	0.000021	0.727754

FIGURE 3.4

Define the truncation error $\delta = |\varepsilon_M| = |x_* - x_M|$ where M denotes N_{max}. Now we use the fact that FPiteration terminates when $|x_n - x_{n+1}| < \epsilon$ and that $\varepsilon_n \simeq \varepsilon_{n-1} f'(x_*)$ for small ε_n, to see

$$\begin{aligned} \epsilon &\simeq |x_M - x_{M-1}| \\ &= |\varepsilon_M - \varepsilon_{M-1}| \\ &\simeq |\varepsilon_M| \left| 1 - \frac{1}{f'(x_*)} \right| \\ \Rightarrow \qquad |\varepsilon_M| &\simeq \epsilon \left| \frac{f'(x_*)}{1 - f'(x_*)} \right| \end{aligned}$$

So in the case of running FPiteration$(10^{-5}, 50, -2)$ for $k = 16$, we have $|\varepsilon_M| \simeq 10^{-5} \cdot 2.6731\ldots$ so should not expect the truncation error to be less than 10^{-5}.

(v) First order convergence requires

$$\lim_{n\to\infty} \left| \frac{\varepsilon_n}{\varepsilon_{n-1}} \right| = c$$

for some $c < 1$.

Figures 3.2, 3.3 and 3.4 demonstrate that this is indeed the case for values of k for which $x_n \to x_*$ so our results are consistent with first-order convergence.

In fact in these cases

$$\begin{aligned} \lim_{n\to\infty} \left| \frac{\varepsilon_n}{\varepsilon_{n-1}} \right| &= \lim_{n\to\infty} \left| \frac{f(x_*) - f(x_{n-1})}{x_* - x_{n-1}} \right| \\ &= |f'(x_*)| \end{aligned}$$

and comparing Figures 3.2, 3.3 and 3.4 with Figure 3.5 confirms this.

k	$\lvert f'(x_*)\rvert$ to 6 dp
4	0.183261
2	0.225109
16	0.727754

FIGURE 3.5

3.3 Question 4

The program used for this question is FPiteration2.m; it is a modified version of FPiteration.m to suit the new requirements of the question.

Running FPiteration2$(10^{-5}, 1000, 4.5)$ results in termination at $N = 736$ and final aproximation $x_{736} = 4.00753487$ to 8dp.

Note $f'(x_*) = 1 - F'(x_*)h'(F)$. Since at a multiple root of (‡) $F'(x_*) = 0$, we have $f'(x_*) = 1$. Hence

$$\varepsilon_{n+1} = \varepsilon_n + \frac{\varepsilon_n^2}{2} f''(x_*) + o(\varepsilon_n^2) \qquad (*)$$

so changes to the error in consecutive terms are only to second order in ε_n, explaining the slow convergence observed for FPiteration2$(10^{-5}, 1000, 4.5)$.

We also note that $\dfrac{\varepsilon_{n+1}}{\varepsilon_n} = 1 + o(\varepsilon_n) \to 1$ as $n \to \infty$ so this is not an example of first order convergence.

Rewritting $(*)$ gives

$$\begin{aligned} x_{n+1} - x_n &\simeq \frac{\varepsilon_n^2}{2} f''(x_*) \\ \Rightarrow \qquad \varepsilon_n &\simeq \pm\sqrt{\frac{2(x_{n+1}-x_n)}{f''(x_*)}} \end{aligned}$$

So if the iteration terminates at $n = N_{fin}$, using the fact that for large n, $|\varepsilon_{n+1}| \simeq |\varepsilon_n|$ and that the termination criterion is $|x_{n+1} - x_n| < \epsilon$ we have

$$|\varepsilon_{N_{fin}}| \simeq \sqrt{\frac{2\epsilon}{|f''(x_*)|}}$$

Hence, noting $f''(x_*) = -\frac{7}{20}$, we have

$$|\varepsilon_{N_{fin}}| \simeq \sqrt{\frac{2 \cdot 10^{-5}}{7/20}} = 0.007559\ldots$$

in the case of $\epsilon = 10^{-5}$. So we should not expect the truncation error to be less than 10^{-5}, which is reflected in our final approximation of x_* as $x_{736} = 4.007534\ldots = x_* + 0.007534\ldots$.

[not sure how to use, or even show, the fact $\varepsilon_n \sim \frac{40}{7n}$]

4 Newton-Raphson Iteration

4.1 Programming Task

The programs used are FPiteration3.m and FPiteration4.m. They are minor modifications of FPiteration.m which estimate the root(s) to (†) and (‡) respectively, using the Newton-Raphson Iteration method.

4.2 Question 5

Calculating x_* for (†) by running FPiteration3(10^{-5}, 100, -4) and FPiteration3(10^{-5}, 100, -1.3). yields the results in Figures 4.0 and 4.1 respectively.

n	x_n to 14 dp	ε_n to 14 dp	$\varepsilon_{n+1}/\varepsilon_n^2$ to 6 dp
0	-4.00000000000000	-1.11676312744172	n/a
1	-2.66940179751675	0.21383507504153	0.171458
2	-2.88895936713309	-0.00572249457480	0.125149
3	-2.88323939429785	-0.00000252173957	0.077007
4	-2.88323687255878	-0.00000000000050	0.078215

FIGURE 4.0

n	x_n to 14 dp	$x_n - x_*$ to 14 dp	$\lvert\varepsilon_n/\varepsilon_{n-1}^2\rvert$ to 6 dp
0	-1.30000000000000	1.58323687255828	n/a
1	-5.71808687309944	-2.83485000054115	1.130937
2	-20.79034108581136	-17.90710421325308	2.228257
3	-9.83873225028047	-6.95549537772219	0.021691
4	-6.49231132526740	-3.60907445270911	0.074600
5	-14.36895020044105	-11.48571332788277	0.881792
6	-6.62739618976328	-3.74415931720500	0.028382
7	-15.41647512560112	-12.53323825304284	0.894035
8	-10.29262915701106	-7.40939228445277	0.047169
9	-5.75537161715424	-2.87213474459596	0.052317
10	-19.31167257761758	-16.42843570505929	1.991529
$\vdots$	$\vdots$	$\vdots$	$\vdots$
46	-2.88323687255828	-0.00000000000000	0.078741

FIGURE 4.1

So we see that x_n converges very rapidly for $x_0 = -4$ while it initially appears to diverge for $x_0 = -1.3$, converging only after substantially more terms.

In the latter case we show graphically what is occurring for the first few iterations by plotting $y = F(x)$ and its tangents at $x = x_0, x_1, x_2$ in Figure 4.2.

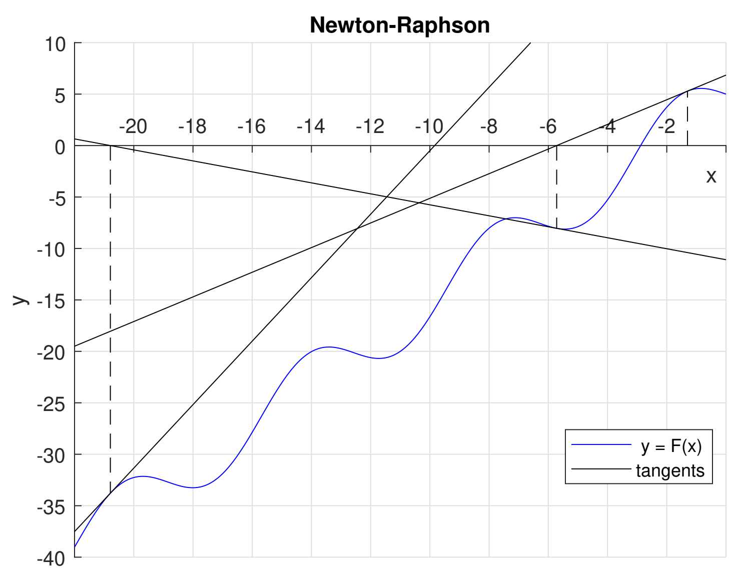

FIGURE 4.2

Calculating the double root x_* of (‡) by running FPiteration4(10^{-5}, 100, 5) yields the results in Figure 4.3.

n	x_n to 6 dp	$x_n - x_*$ to 6 dp	$\lvert\varepsilon_n/\varepsilon_{n-1}\rvert$ to 6 dp
0	5.000000	1.000000	n/a
1	4.550000	0.550000	0.550000
2	4.292486	0.292486	0.531792
3	4.151673	0.151673	0.518565
$\vdots$	$\vdots$	$\vdots$	$\vdots$
15	4.000039	0.000039	0.500006
16	4.000019	0.000019	0.500002
17	4.000010	0.000010	0.499996

FIGURE 4.3

We analysis the convergence and error of this method by using (as before)

$$\varepsilon_{n+1} = \varepsilon_n f'(x_*) + \frac{\varepsilon_n^2}{2} f''(x_*) + o(\varepsilon_n^2) \tag{$**$}$$

Now for the Newton-Raphson method $f(x) = x - \dfrac{F(x)}{F'(x)}$ so we have

$$f'(x) = \frac{FF''}{(F')^2} \quad \text{and} \quad f''(x) = \frac{FF'F''' + (F')^2F'' - 2F(F'')^2}{(F')^3} .$$

For ($\dagger$)**:**

Since $F(x_*) = 0$ and $F'(x_*) \neq 0$ we get $f'(x_*) = 0$ and $f''(x_*) = \dfrac{F''(x_*)}{F'(x_*)} \simeq -0.156409356534\ldots$.
So define $k = \dfrac{f''(x_*)}{2} \simeq -0.078204678267\ldots$ then ($**$) becomes

$$\varepsilon_{n+1} = k\varepsilon_n^2 + o(\varepsilon_n^2) \quad \Rightarrow \quad \left|\frac{\varepsilon_{n+1}}{\varepsilon_n^2}\right| = |k| + o(1) \to |k| \simeq 0.0782\ldots$$

which is sufficient for second-order convergence. The results in Figures 4.0 and 4.1 concur with this.

Now we have

$$\varepsilon_{n+1} \simeq k\varepsilon_n^2 \quad \Rightarrow \quad \varepsilon_n \simeq \frac{\varepsilon_{n+1} - \varepsilon_n}{k-1} = \frac{x_{n+1} - x_n}{k-1}.$$

So if the iteration terminates at $n = M$, using again that for large n, $\varepsilon_{n+1} \simeq k\varepsilon_n^2$ and that the termination criterion is $|x_{n+1} - x_n| < \epsilon$ we have

$$|\varepsilon_M| \simeq \left|\frac{k}{k-1}\right| \epsilon^2 \simeq \epsilon^2 \cdot 0.0725\ldots$$

We took $\epsilon = 10^{-5}$ so expect error of around $7.25 \cdot 10^{-12}$, which is consistent with Figures 4.0 and 4.1.

[I believe what I've discussed is truncation error but I'm not sure how to work in rounding error or the hint. Decreasing ϵ just seems to increase accuracy arbitrarily well.]

For $(\ddagger)$ **:**

Since $F(x_*) = 0$ and $F'(x_*) = 0$ we instead get $f'(x_*) = \dfrac{1}{2}$ and $f''(x_*) = \dfrac{1}{7}$ (in the limit) and hence $(**)$ becomes

$$\varepsilon_{n+1} = \frac{\varepsilon_n}{2} + o(\varepsilon_n) \quad \Rightarrow \quad \left|\frac{\varepsilon_{n+1}}{\varepsilon_n}\right| = \frac{1}{2} + o(1) \to \frac{1}{2}$$

which is sufficient for first-order convergence. The results in Figure 4.3 concur with this.

Now note also

$$\varepsilon_{n+1} = \frac{\varepsilon_n}{2} + o(\varepsilon_n) \quad \Rightarrow \quad \varepsilon_{n+1} - \varepsilon_n = -\varepsilon_{n+1} + o(\varepsilon_n) \quad \Leftrightarrow \quad \varepsilon_{n+1} = -(x_{n+1} - x_n) + o(\varepsilon_n)$$

So if the iteration terminates at $n = M$, using the above and that the termination criterion is $|x_{n+1} - x_n| < \epsilon$ we have

$$|\varepsilon_M| \simeq \epsilon$$

We took $\epsilon = 10^{-5}$ so expect error of around 10^{-5} which is consistent with Figure 4.3 .

If we instead take $\epsilon = 10^{-12}$, FPiteration4$(10^{-5}\,,\,100\,,\,5)$ gives $\varepsilon_M = 4.76 \cdot 10^{-8}$ which is much larger than what would be expected by the $|\varepsilon_M| \simeq \epsilon$ approximation. This is because for small enough ϵ rounding error due to MATLAB's inability to exactly calculate expressions dominates over truncation error.

5 Appendix

All relevant MATLAB code written is included here

5.1 Introductions and definitions

5.1.1 F.m

Function $F(x)$ as defined in $(\dagger)$.

```
function [F] = F(x)
%F(x) = 2x - 3sinx +5
%   F is the first function to be studied.

F = 2*x -3*sin(x) + 5;
end
```

5.1.2 poltF.m

Used to generate Figure 1.0 .

```

```

5.2 Binary Search

5.2.1 bisectionF.m

Written for first programming task.

```
function [a] = bisectionF(x1,x2)
%Uses bisection method to find the root of F(x) outputs root and # of
%iterations used. Root must lie inbetween inputs.

if F(x1)*F(x2) < 0
    if x1 > x2
        xh=x1;
        xl=x2;
    else
        xh=x2;
        xl=x1;
    end
elseif  F(x1)*F(x2) > 0
    disp('root not between inputs, try different inputs');
    return
elseif  F(x1)*F(x2) == 0
    disp('root is (approximately) one of your inputs')
    return
end

xn = (xh + xl)/2;
n=1;

while xh - xl > .5*10^(-5)
    t = F(xn);

    if t > 0
        xh=xn;
    elseif t < 0
        xl=xn;
    elseif t == 0
        fprintf('you found the root (within MATLAB''s error), the root is %10.10f', xn)
        return
    end

    xn = (xh + xl)/2;
    n=n+1;
end

fprintf('root is approximately %10.10f after %3i iterates.', xn, n)
a = [xn, n];
end
```

5.2.2 table1.m

Used to generate data for Figure 2.0.

```
seed = 1234;
rns = RandStream('mt19937ar', 'seed', seed);
```

```

t = zeros(10,5);

for k = 1:10

    i1 = 0;
    i2 = 0;

    while F(i1)*F(i2) > 0
        i1 = -13 + rand(rns)*20;
        i2 = -13 + rand(rns)*20;
    end

    a = bisectionF(i1,i2);
    t(k,:) = [i1, i2, a(1), a(2), abs(-2.88323687-a(1))];
    vpa(t)
end
```

5.3 Fixed-Point Iteration

5.3.1 FPiteration.m

Written for second programming task.

```
function [x] = FPiteration(e, N, x0)
%Solves the problem F(x) = 0 iteratively using f1(x) = x
%   Takes inputs: e, stopping difference; N, max iterates; x0, first guess.

x1=0;
x2=x0;
n=0;

x=fzero(@F,x0);
fprintf(' $n$  & $x_n$      & $\\eps_n$   & $\\eps_{n+1}/\\eps_n$ \\\\ \\hline  \n
    ')
fprintf('%5.i & %10.6f & %10.6f & n/a        \\\\ \\hline  \n', n, x2, x2-x)

while abs(x2 - x1) > e && n < N
    err = x2 - x;
    x1=x2;
    x2=f1(x2);
    n=n+1;

    fprintf('%5.i & %10.6f & %10.6f & %10.6f \\\\ \\hline  \n', n, x2, x2-x, abs((
    x2 -x)/err))
end

x=x2;
end
```

5.3.2 f1.m

The function called in `FPiteration.m` for iteration.

```
function [f1] = f1(x)
%f(x) = (3*sin(x)+k*x - 5)/(2+k)
%   A function st f(x)=x <=> F(x)=0. k must me editied within the code

k=0;
f1 = (3*sin(x)+k*x - 5)/(2+k);
end
```

5.3.3 plotCobweb.m

Used to generate Figure 3.1 .

```

```

5.3.4 FPiteration2.m

Modified version of `FPiteration.m` for question 4.

```

```

5.3.5 f2.m

The function called in `FPiteration2.m` for iteration.

```

```

5.4 Newton-Raphson Iteration

5.4.1 FPiteration3.m

Modified version of `FPiteration.m` for solving (†) in question 5.

```
function [x] = FPiteration3(e, N, x0)
%Solves the problem F(x) = 0 iteratively using f3(x) = x
%   Takes inputs: e, stopping difference; N, max iterates; x0, first guess.

x1=0;
x2=x0;
n=0;

x=fzero(@F,x0);
fprintf(' $n$  & $x_n$      & $\\eps_n$   & $\\eps_{n+1}/\\eps_n^2$ \\\\ \\hline
    \n')
fprintf('    0 & %10.14f & %10.14f & n/a        \\\\ \\hline  \n', x2, x2-x)

while abs(x2 - x1) > e && n < N
    err = x2 - x;
```

```
    x1=x2;
    x2=f3(x2);
    n=n+1;

    fprintf('%5.i & %16.14f & %16.14f & %9.6f \\\\ \\hline  \n', n, x2, x2-x, abs((x2 -x)/err^2))
end

x=x2;
end
```

5.4.2 f3.m

The function called in `FPiteration3.m` for iteration.

```

```

5.4.3 FPiteration4.m

Modified version of `FPiteration.m` for solving (‡) in question 5.

```
function [x] = FPiteration4(e, N, x0)
%Solves the problem F(x) = 0 iteratively using f1(x) = x
%   Takes inputs: e, stopping difference; N, max iterates; x0, first guess.

x1=0;
x2=x0;
n=0;

x=4;
fprintf(' $n$  & $x_n$       & $\\eps_n$   & $\\eps_{n+1}/\\eps_n$ \\\\ \\hline  \n')
fprintf('    0 & %10.6f & %10.6f & n/a         \\\\ \\hline  \n', x2, x2-x)

while abs(x2 - x1) > e && n < N
    err = x2 - x;
    x1=x2;
    x2=f4(x2);
    n=n+1;

    fprintf('%5.i & %10.6f & %10.30f & %10.6f \\\\ \\hline  \n', n, x2, x2-x, abs((x2 -x)/err))
end

x=x2;
end
```

5.4.4 f4.m

The function called in `FPiteration4.m` for iteration.

```
function [f4] = f4(x)
%f(x) = x - (x.^3 - 8.5*x.^2 + 20*x - 8)./(3*x.^2 - 17*x + 20)
%   A function st f(x)=x <=> F(x)=0.

f4 = x - (x.^3 - 8.5*x.^2 + 20*x - 8)./(3*x.^2 - 17*x + 20);
end
```

5.4.5 plotNR.m

Used to generate Figure 4.2 .

```

```

References

[1] James F Epperson. *An introduction to numerical methods and analysis.* John Wiley & Sons, 2013.