SyDe312 (Winter 2005)

Unit 2 - Solutions

March 3, 2005

Chapter 3 - Root Finding for Nonlinear Equations

Problem 3.4 - 3

How many solutions are there to the equation $x = e^{-x}$? Will the iteration $x_{n+1} = e^{-x_n}$ converge for suitable choices of initial guess? Calculate the first six iterates when $x_o = 0$.

The equation $x = e^{-x}$ has only one solution. The reason is that x and e^{-x} are monotone increasing and monotone decreasing, respectively. Furthermore, when $x > 1$ then $x > e^{-x}$ and when $x < 0$ then $x < e^{-x}$. Consequently, this equation has exactly one solution.

The iteration $x_{n+1} = e^{-x_n}$ will converge for appropriate choices of initial iterate x_o. For instance, if we choose an interval of $[a, b] \equiv [0.1, 1]$then $0.1 \leq x \leq 1$ implying $0.1 \leq e^{-x} \leq 1$. If we compute the derivative of e^{-x} then:

$$\left|\left(e^{-x}\right)'\right| = |-e^{-x}| = e^{-x} < 1 \text{ for any value of } x > 0.$$

Consequently, the iteration converges when $x_o \in [a, b]$.

The first six iterates of the given iteration formula (with an initial iterate $x_o = 0.0$) are listed as follows (the true solution is $\alpha \doteq 0.56714329040978$):

n	x_n
0	0.0
1	1.0
2	0.3678794412
3	0.6922006276
4	0.5004735006
5	0.6062435351
6	0.5453957860

Problem 3.4 - 4

Repeat the previous problem with $x_{n+1} = 1 - \tan^{-1} x_n$.
Let $f(x) = x - 1 - \tan^{-1} x$.
Then $f^{'}(x) = \frac{x^2}{1+x^2} \quad > \quad 0 \qquad \text{for } x \neq 0$

This means that $f(x)$ is monotone increasing. Also, note that $f(0) = -1$ and for any value of $x > 0$ (say $x = 10$): $f(x) > 0$
For instance: $f(10) = 10 - 1 - \tan^{-1}(10) \quad \geq \quad 9 - \frac{\pi}{2} \quad > \quad 0$
Consequently, the equation $x = 1 + \tan^{-1}(x)$ has only one solution.
The iteration $x_{n+1} = 1 + \tan^{-1}(x_n)$ will converge for any choice of initial iterate x_o. The reason is that the derivative of $1 + \tan^{-1}(x)$ is $\frac{x^2}{1+x^2}$, which is always smaller than 1.
The first six iterates of the given iteration formula (with an initial iterate $x_o = 0.0$) are listed as follows (the true solution is $\alpha \overset{\circ}{=} 2.1322677252729$):

n	x_n
0	0.0
1	1.0
2	1.7853981634
3	2.0602325258
4	2.1189112720
5	2.1298472434
6	2.1318309260

Problem 3.4 - 7

What are the solutions α, if any, of the equation $x = \sqrt{1+x}$? Does the iteration $x_{n+1} = \sqrt{1+x_n}$ converge to any of these solutions (assuming x_o is chosen sufficiently close to the root α)?
The equation has one obvious solution. Squaring both sides of the equation gives: $x^2 - x - 1 = 0$ and, the two roots are: $\frac{1+\sqrt{5}}{2}$ and $\frac{1-\sqrt{5}}{2}$
The second root is negative, and since the original equation has real root only if we consider the positive root, we have: $\alpha = \frac{1+\sqrt{5}}{2}$
The iteration $x_{n+1} = \sqrt{1+x_n}$ converges to α when $x_o \in [1,2]$.
Note that $1 \leq x \leq 2$ also implies that $1 \leq \sqrt{1+x} \leq 2$, and $g^{'}(x) = \frac{1}{2\sqrt{1+x}} < \frac{1}{2}$, $x_o \in [1,2]$.

Problem 3.1 - 1 Bisection Method

We used the following code to find roots using the bisection method with the initial interval $[0, 2]$ and 20 iterations; however, in most cases, it required fewer than 20 iterations to find a root (and for the given precision, even smaller number of iterations are needed). Notice the linear convergence rate of the bisection method. A simple plot using brackplot.m may help in choosing the initial interval. The function was updated for each case; alternatively, we can use bisection.m and save the given functions in m-files.

Tables of values showing the iterations for 3.1 - 1a-g follow.

3.1 - 1a

Final $[a, b] = [1.8392, 1.8394]$
Root $= 1.83929$

k	a	x_{mid}	b	$f(x_{\text{mid}})$
1	0.00000	1.00000	2.00000	$-2.0E00$
2	1.00000	1.50000	2.00000	.
3	1.50000	1.75000	2.00000	.
4	1.75000	1.87500	2.00000	.
5	1.75000	1.81250	1.87500	.
6	1.81250	1.84375	1.87500	.
7	1.81250	1.82813	1.84375	.
8	1.82813	1.83594	1.84375	.
9	1.83594	1.83984	1.84375	.
10	1.83594	1.83789	1.83984	.
11	1.83789	1.83887	1.83984	.
12	1.83887	1.83936	1.83984	.
13	1.83887	1.83911	1.83936	.
14	1.83911	1.83923	1.83936	.
15	**1.83923**	**1.83929**	**1.83936**	$\mathbf{4.2E-05}$
16	1.83923	1.83926	1.83929	.
17	1.83926	1.83928	1.83929	.
18	1.83928	1.83929	1.83929	.
19	1.83928	1.83928	1.83929	.
20	1.83928	1.83928	1.83929	$-1.017E-05$

3.1 - 1b

Final $[a, b] = [1.12842, 1.1285]$
Root $= 1.12842$

k	a	x_{mid}	b	$f(x_{\text{mid}})$
1	0.00000	1.00000	2.00000	$-1.621E01$
2	1.00000	1.50000	2.00000	.
3	1.00000	1.25000	1.50000	.
4	1.00000	1.12500	1.25000	.
5	1.12500	1.18750	1.25000	.
6	1.12500	1.15625	1.18750	.
7	1.12500	1.14063	1.15625	.
8	1.12500	1.13281	1.14063	.
9	1.12500	1.12891	1.13281	.
10	1.12500	1.12695	1.12891	.
11	1.12695	1.12793	1.12891	.
12	1.12793	1.12842	1.12891	.
13	1.12842	1.12866	1.12891	.
14	1.12842	1.12854	1.12866	.
15	**1.12842**	**1.12848**	**1.12854**	$\mathbf{6.85E-05}$
16	1.12842	1.12845	1.12848	.
17	1.12842	1.12843	1.12844	.
18	1.12842	1.12843	1.12843	.
19	1.12842	1.12842	1.12843	.
20	1.12842	1.12842	1.12843	$-1.782E-06$

3.1 - 1c

Final $[a, b] = [0.42395, 0.42407]$
Root $= 0.42401$

k	a	x_{mid}	b	$f(x_{\text{mid}})$
1	0.00000	1.00000	2.00000	$-8.012E-01$
2	0.00000	0.50000	1.00000	.
3	0.00000	0.25000	0.50000	.
4	0.25000	0.27500	0.50000	.
5	0.37500	0.43750	0.50000	.
6	0.37500	0.40625	0.43750	.
7	0.40625	0.42188	0.43750	.
8	0.42188	0.42969	0.43750	.
9	0.42188	0.42578	0.42959	.
10	0.42188	0.42383	0.42578	.
11	0.42383	0.42480	0.42578	.
12	0.42383	0.42432	0.42480	.
13	0.42383	0.42407	0.42432	.
14	0.42383	0.42395	0.42407	.
15	**0.42395**	**0.42401**	**0.42407**	**2.62E − 05**
16	0.42401	0.42404	0.42407	.
17	0.42401	0.42403	0.42404	.
18	0.42403	0.42403	0.42404	.
19	0.42403	0.42403	0.42403	.
20	0.42403	0.42403	0.42403	$-1.550E-06$

3.1 - 1d

Final $[a, b] = [0.56714, 0.56726]$
Root $= 0.56720$

k	a	x_{mid}	b	$f(x_{\text{mid}})$
1	0.00000	1.00000	2.00000	$6.321E-01$
2	0.00000	0.50000	1.00000	.
3	0.50000	0.75000	1.00000	.
4	0.50000	0.625000	0.75000	.
5	0.50000	0.56250	0.62500	.
6	0.56250	0.59375	0.62500	.
7	0.56250	0.57813	0.59375	.
8	0.56250	0.57031	0.57813	.
9	0.56250	0.56641	0.57031	.
10	0.56641	0.56836	0.57031	.
11	0.56641	0.56738	0.56836	.
12	0.56641	0.56689	0.56738	.
13	0.56689	0.56714	0.56738	.
14	0.56714	0.56726	0.56738	.
15	**0.56714**	**0.56720**	**0.56726**	**8.841E − 05**
16	0.56714	0.56717	0.56720	.
17	0.56714	0.56715	0.56717	.
18	0.56714	0.56715	0.56715	.
19	0.56714	0.56714	0.56715	.
20	0.56714	0.56714	0.56715	$1.729E-06$

3.1 - 1e

Final $[a, b] = [0.58850, 0.58862]$
Root = 0.58856

k	a	x_{mid}	b	$f(x_{\text{mid}})$
1	0.00000	1.00000	2.00000	$4.736E-01$
2	0.00000	0.50000	1.00000	.
3	0.50000	0.75000	1.00000	.
4	0.50000	0.62500	0.75000	.
5	0.50000	0.56250	0.62500	.
6	0.56250	0.59375	0.62500	.
7	0.56250	0.57813	0.59375	.
8	0.57813	0.58594	0.59375	.
9	0.58594	0.58984	0.59375	.
10	0.58594	0.58789	0.58984	.
11	0.58789	0.58887	0.58984	.
12	0.58789	0.58838	0.58887	.
13	0.58838	0.58862	0.58887	.
14	0.58838	0.58850	0.58862	.
15	**0.58850**	**0.58856**	**0.58862**	$\mathbf{4.059E-05}$
16	0.58850	0.58853	0.58856	.
17	0.58853	0.58855	0.58856	.
18	0.58853	0.58854	0.58855	.
19	0.58853	0.58854	0.58854	.
20	0.58853	0.58853	0.58854	$9.119E-07$

3.1 - 1f

Final $[a, b] = [1.76929, 1.1.76932]$
Root $= 1.76930$

k	a	x_{mid}	b	$f(x_{\text{mid}})$
1	0.00000	1.00000	2.00000	-3.0
2	1.00000	1.50000	2.00000	.
3	1.50000	1.75000	2.00000	.
4	1.75000	1.87500	2.00000	.
5	1.75000	1.81250	1.8750	.
6	1.75000	1.78125	1.81250	.
7	1.76563	1.76563	1.78125	.
8	1.76563	1.77344	1.78125	.
9	1.76563	1.76953	1.77344	.
10	1.76758	1.76758	1.76953	.
11	1.76855	1.76855	1.76953	.
12	1.76904	1.76904	1.76953	.
13	1.76929	1.76929	1.76953	.
14	1.76929	1.76941	1.76953	.
15	1.76929	1.76935	1.76941	.
16	1.76929	1.76932	1.76935	.
17	**1.76929**	**1.76930**	**1.76932**	**7.402E − 05**
18	1.76929	**1.76929**	1.76930	.
19	1.76929	1.76929	1.76929	.
20	1.76929	1.76929	1.76929	$3.527E-06$

3.1 - 1g

Final $[a, b] = [1.22070, 1.22076]$
Root = 1.22073

k	a	x_{mid}	b	$f(x_{\text{mid}})$
1	0.00000	1.00000	2.00000	-1.00000
2	1.00000	1.50000	2.00000	.
3	1.00000	1.25000	1.50000	.
4	1.00000	1.12500	1.25000	.
5	1.12500	1.18750	1.25000	.
6	1.18750	1.21875	1.25000	.
7	1.21875	1.23438	1.25000	.
8	1.21875	1.22656	1.23438	.
9	1.21875	1.22266	1.22656	.
10	1.21875	1.22070	1.22656	.
11	1.22070	1.22168	1.22266	.
12	1.22070	1.22119	1.22168	.
13	1.22070	1.22095	1.22119	.
14	1.22070	1.22083	1.22095	.
15	1.22070	1.22076	1.22083	.
16	**1.22070**	**1.22073**	**1.22076**	$\mathbf{-6.554E-05}$
17	1.22073	1.22075	1.22076	.
18	1.22073	1.22074	1.22075	.
19	1.22074	1.22075	1.22075	.
20	1.22074	1.22074	1.22075	$-5.682E-06$

It is worth pointing that this function has another real root around -0.7245.

Problem 3.1 - 4 Proof

Show that for any real constants c and d the equation $x = c + d\cos(x)$ has at least one root.

Let $f(x) = x - c - d\cos(x)$.

Since $\cos(\frac{\pi}{2} + n\pi) = 0$,

$f(\frac{\pi}{2} + n\pi) = \frac{\pi}{2} + n\pi - c$ for any integer n.

Since c is a fixed real number, there exists an integer n_o such that:

$\frac{\pi}{2} + n_o\pi < c < \frac{\pi}{2} + (n_o + 2)\pi$

It can be seen that $f(x)$ is continuous in character and

$f(\frac{\pi}{2}+n_o\pi)=\frac{\pi}{2}+n_o\pi-c<0$
$f(\frac{\pi}{2}+(n_o+2)\pi)=\frac{\pi}{2}+(n_o+2)\pi-c>0$
which means the $f(x)$ changes sign and has a root in the following interval:
$\left(\frac{\pi}{2}+n_o\pi,\frac{\pi}{2}+(n_o+2)\pi\right)$

Problem 3.1 - 7
Bisection method with direct, reverse, and nested polynomials.

We have shown first 10 iterations for various required scenarios; however, notice that the desired accuracy was achieved in fewer iterations and is marked in boldface.

3.1 - 7(i)
$f(x)=x^3-3x^2+3x-1=0$ **(Direct)**

The bisection method with an accuracy of $\epsilon=10E-06$ for different initial intervals:

$[\mathbf{0},\mathbf{1.5}]$

k	a	x_{mid}	b	$f(x_{\text{mid}})$
1	0.00000000	0.75000000	1.5000000	$-1.562E-02$
2	0.7500000	1.1250000	1.5000000	.
3	0.7500000	0.9375000	1.1250000	.
4	0.9375000	1.0312500	1.1250000	.
5	0.9375000	0.9843750	1.0312500	.
6	0.9843750	1.0078125	1.0312500	.
7	**0.9843750**	**0.9960938**	**1.0078125**	$\mathbf{-5.960E-08}$
8	0.9960938	1.0019531	1.0078125	.
9	0.9960938	0.9990234	1.0019531	.
10	0.9990234	1.0004883	1.0019531	$1.164E-10$

$[\mathbf{0.5, 2}]$

k	a	x_{mid}	b	$f(x_{\text{mid}})$
1	0.50000000	1.2500000	2.0000000	$1.5625E-02$
2	0.5000000	0.8750000	1.2500000	.
3	0.8750000	1.0625000	1.2500000	.
4	0.8750000	0.9687500	1.0625000	.
5	0.9687500	1.0156250	1.0625000	.
6	0.9687500	0.9921875	1.0156250	.
7	**0.9921875**	**1.0039063**	**1.0156250**	$\mathbf{5.9605E-08}$
8	0.9921875	0.9980469	1.0039063	.
9	0.9980469	1.0009766	1.0039063	.
10	0.9980469	0.9995117	1.0009766	$-1.1642E-10$

$[\mathbf{0.5, 1.1}]$

k	a	x_{mid}	b	$f(x_{\text{mid}})$
1	0.5000000	0.8000000	1.1000000	$-8.000E-03$
2	0.8000000	0.9500000	1.1000000	.
3	0.9500000	1.0250000	1.1000000	.
4	0.9500000	0.9875000	1.0250000	.
5	0.9875000	1.0062500	1.0250000	.
6	**0.9875000**	**0.9968750**	**1.0062500**	$\mathbf{-3.052E-08}$
7	0.9968750	1.0015625	1.0062500	.
8	0.9968750	0.9992188	1.0015625	.
9	0.99992188	1.0003906	1.0015625	$5.961E-11$
10	0.9992188	0.9998047	1.0003096	$-7.450E-12$

3.1 - 7(ii)
$f(x) = -1 + 3x - 3x^2 + x^3 = 0$ **(Reverse)**

$[\mathbf{0}, \mathbf{1.5}]$

k	a	x_{mid}	b	$f(x_{\text{mid}})$
1	0.0000000	0.7500000	1.5000000	$-1.562E-02$
2	0.7500000	1.1250000	1.5000000	.
3	0.7500000	0.9375000	1.1250000	.
4	0.9375000	1.0312500	1.1250000	.
5	0.9375000	50.9843750	1.0312500	.
6	0.9843750	1.007812	1.0312500	.
7	**0.9843750**	**0.9660938**	**1.0078125**	$\mathbf{-5.960E-08}$
8	0.9660938	1.0019531	1.0078125	.
9	0.9660938	0.9990234	1.0019531	.
10	0.9990234	1.0004883	1.0019531	$1.164E-10$

$[\mathbf{0.5}, \mathbf{2}]$

k	a	x_{mid}	b	$f(x_{\text{mid}})$
1	0.5000000	1.2500000	2.000000	$1.563E-02$
2	0.5000000	0.8750000	1.2500000	.
3	0.8750000	1.0625000	1.2500000	.
4	0.8750000	0.9687500	1.0625000	.
5	0.9687500	1.0156250	1.0625000	.
6	0.9687500	0.9921875	1.0156250	.
7	**0.9921875**	**1.0039063**	**1.0156250**	$\mathbf{5.960E-08}$
8	0.9921875	0.9980469	1.0039063	.
9	0.9980469	1.0009766	1.0039063	.
10	0.9980469	0.9995117	1.0009766	$-1.164E-10$

[0.5, 1.1]

k	a	x_{mid}	b	$f(x_{\text{mid}})$
1	0.5000000	0.8000000	1.1000000	$-8.0E-03$
2	0.8000000	0.9500000	1.1000000	.
3	0.9500000	1.0250000	1.1000000	.
4	0.9500000	0.9875000	1.025000	.
5	0.9875000	1.0062500	1.0250000	.
6	**0.9875000**	**0.9968750**	**1.0062500**	$\mathbf{-3.052E-08}$
7	0.9968750	1.0015625	1.0062500	.
8	0.9968750	0.9992188	1.0015625	.
9	0.9992188	1.0003906	1.0015625	.
10	0.9992188	0.9998047	1.0003906	$-7.45E-12$

3.1 - 7(iii)
$f(x) \equiv -1 + x(3 + x(-3 + x) = 0$ **(Nested)**

[0, 1.5]

k	a	x_{mid}	b	$f(x_{\text{mid}})$
1	0.0000000	0.7500000	1.5000000	$-1.562E-02$
2	0.7500000	1.1250000	1.5000000	
3	0.7500000	0.9375000	1.1250000	
4	0.9375000	1.0312500	1.1250000	
5	0.9375000	0.9843750	1.0312500	
6	0.9843750	1.0078125	1.0312500	
7	**0.9843750**	**0.9960938**	**1.0078125**	
8	**0.9960938**	**1.0019531**	**1.0078125**	$\mathbf{7.451E-09}$
9	0.9960938	0.9990234	1.0019531	
10	0.9990234	1.0004883	1.0019531	$1.164E-10$

[0.5, 2.0]

k	a	x_{mid}	b	$f(x_{\text{mid}})$
1	0.5000000	1.2500000	2.0000000	$1.562E-02$
2	0.5000000	0.8750000	1.2500000	.
3	0.8750000	1.0625000	1.2500000	.
4	0.8750000	0.9687500	1.0625000	.
5	0.9687500	1.0156250	1.0625000	.
6	0.9687500	0.9980469	1.0156250	.
7	**0.9921875**	**1.0039063**	**1.0156250**	$\mathbf{5.960E-08}$
8	0.9921875	0.9980469	1.0039063	.
9	0.9980469	1.0009766	1.0039063	.
10	0.9980469	0.9995117	1.0009766	$-1.164E-10$

[0.5, 1.1]

k	a	x_{mid}	b	$f(x_{\text{mid}})$
1	0.5000000	0.8000000	1.1000000	$-8.0E-03$
2	0.8000000	0.9500000	1.1000000	.
3	0.9500000	1.0250000	1.1000000	.
4	0.9500000	0.9875000	1.0250000	.
5	0.9875000	1.0062500	1.0250000	.
6	0.9875000	0.9968750	1.0062500	
7	**0.9968750**	**1.0015625**	**1.0062500**	$\mathbf{3.814E-09}$
8	0.9968750	0.9992188	1.0015625	.
9	0.9992188	1.0003906	1.0015625	.
10	0.9992188	0.9998407	1.0003906	$-7.450E-12$

The following is a summary of our experimental results:

Initial_Interval	Direct	Reverse	Nested
$[0, 1.5]$	0.9960938	0.9960938	1.0019531
$[0.5, 2]$	1.0039063	1.0039063	1.0039063
$[0.5, 1.1]$	0.9968750	0.9968750	1.0015625

It can be observed from the above summary that the nested form of function evaluation tends to give slightly better results. Indeed, it is an established fact that the nested form frequently gives slightly better results that improves the probability that the sign$(f(x))$ is correctly checked.

3.1 - 7
Using MATLAB function *fzero*

We may also use MATLAB function *fzero* for finding roots for the given polynomial with different initial iterates. Use of *fzero* for this problem with an initial iterate of 0.0: `fzero(inline('x^3-3* x^2+3* x-1'), 0.0)` results in a root of 1.0.

3.1 - 7
Using eigenvalues of the companion matrix

The companion matrix for the given polynomial is:

$$\begin{bmatrix} 3 & -3 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Using *eig* routine in MATLAB results in the following eigenvalues of the companion matrix (which are also the roots of the given polynomial): 1.0, 1.0, 1.0 (Notice the repeated roots)

3.1 - 7
Using Laguerre's Method
$f(x) = x^3 - 3x^2 + 3x - 1 = 0$

We will only use the original polynomial for demonstrating the power of Laguerre's Method. We employed the following code for Laguerre's method (of course, the function as well as the degree of polynomial need to be updated to reflect the problem at hand):

```
%Use Laguerre's method
%Input: x0 = initial guess of x
function Laguerre(x0)
   x = x0;    %initial guess—
   k=0;   %current iteration
   a = 1;   %just to start the while loop
   n=3;   % degree of the polynomial
while abs(a) > 0.000001   %stop when a is small
   p = x^3 - 3*x^2 + 3*x -1;   %calc p(x)
   dp = 3*x^2 - 6*x + 3 ;   % calc p'(x)
   d2p = 6*x - 6 ;    %calc p''(x)
   g = dp/p;    % calc G
```

```
h = (dp/p)^2 - d2p/p;    %calc H
if g > 0.0
   a = n/(g+((n-1)*(n*h-g^2))^(1/2));   % Estimate of a if G +ve
elseif g < 0.0
   a = n/(g-((n-1)*(n*h-g^2))^(1/2));    % Estimate of a if G -ve
end
x = x - a;   %new guess of root
fprintf('%10.6f \\n',x); % Real Roots Only + for succinct display
end
```

The following is a summary of results obtained (only 10 iterations) using various starting points. Notice the extremely fast (cubic) convergence of Laguerre's Method.

k	InitialGuess(x)				
0	**0.0**	**0.50**	**1.1**	**1.5**	**2.0**
1	0.632993	0.816497	1.036701	1.183503	1.367007
2	0.865306	0.932653	1.013469	1.067347	1.134694
3	0.950566	0.975283	1.004943	1.024717	1.049434
4	0.981858	0.990929	1.001814	1.009071	1.018142
5	0.993342	0.996671	1.000666	1.003329	1.006658
6	0.997556	0.998778	1.000244	1.001222	1.002444
7	0.999103	0.999552	1.000090	1.000448	1.000897
8	0.999671	0.999835	1.000033	1.000165	1.000329
9	0.999879	0.999940	1.000012	1.000060	1.000121
10	0.999956	0.999978	1.000004	1.000022	1.000044

Problem 3.1 - 8
Bisection method with direct, reverse, and nested polynomials.
$x^4 - 5.4x^3 + 10.56x^2 - 8.954 + 2.7951$

We followed an approach similar to that of Q. 3.1-7 for various required scenarios using the bisection method with an accuracy of $\epsilon = 10E-06$ for different initial intervals. The following is the summary of our experimental results:

Initial_Interval	Direct	Reverse	Nested
$[1, 1.2]$	1.0971108	1.1063600	1.0955323
$[1.06, 1.16]$	1.0965791	1.0966697	1.0945702
$[0, 2]$	1.1040297	1.1040344	1.0947266

It can be seen that the nested form of $f(x)$ gives slightly better results than the other two (the actual root = 1.1). With nesting, $f(x)$ is calculated more accurately, which improves the chance that sign($f(x)$) is detected correctly.

3.1 - 8
Using MATLAB function *fzero*

We may also use MATLAB function *fzero* for finding roots for the given polynomial with different initial iterates. Use of *fzero* for the same problem with an initial iterate of 0.0: `fzero(inline('x^4-5.4*x^3+10.56* x^2-8.954* x+2.7951'), 0.0)` results in a root of 1.10000.

Similarly, the use of **fzero** MATLAB function for the same problem with a different initial iterate of 2.0: `fzero(inline('x^4-5.4*x^3+10.56* x^2-8.954* x+2.7951'), 2.0))` results in a root of 2.10000.

3.1 - 8
Using eigenvalues of the companion matrix

The companion matrix for the given polynomial is:

$$\begin{bmatrix} 5.4 & -10.56 & 8.954 & -2.7951 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Using *eig* routine in MATLAB results in the following eigenvalues of the companion matrix (which are also the roots of the given polynomial): 1.1, 1.1, 1.1, 2.1 (notice repeated roots at 1.1)

3.1 - 8
Using Laguerre's Method

Applying Laguerre's method to the same problem we get the following results for various starting points. Once again, notice the fast convergence of Laguerre.

k	InitialGuess(x)				
0	**0.0**	**1.06**	**1.16**	**1.2**	**2.0**
1	0.660510	1.097636	1.112887	1.126195	2.100804
2	1.155674	1.100977	1.101461	1.104020	2.100000
3	1.047032	1.100033	1.100061	1.100270	
4	1.171069	1.100005	1.100000	1.100006	
5	1.115564				
6	1.101702				
7	1.100051				
8	1.099999				
9	1.100005				

Notice that the initial iterate (or guess) affects both the convergence rate as well as root it converges to.

3.2 - 1
Newton's Method: $f(x) = x^6 - x - 1 = 0$

Initial iterate of 1.5 gives:

k	x_k	$f(x_k)$	$x_k - x_{k-1}$
0	1.5	$8.89E+01$	
1	1.30049088	$2.54E+01$	$-2.00E-01$
2	1.18148042	$5.38E-01$	$-1.19E-01$
3	1.13945559	$4.92E-02$	$-4.2E-02$
4	1.13477763	$5.50E-04$	$-4.68E-03$
5	1.13472415	$7.11E-08$	$-5.35E-05$
6	1.13472414	$1.55E-15$	$-6.91E-09$

Initial iterate of 1.0 results in the following:

k	x_k	$f(x_k)$	$x_k - x_{k-1}$
0	1.0000000000	-1.00	
1	1.2000000477	$7.86E-01$	$2.00E-01$
2	1.1435759068	$9.30E-02$	$-5.64E-02$
3	1.1349095106	$1.91E03$	$-8.67E-03$
4	1.1347242594	$1.07E-06$	$-1.85E-04$
5	1.1347241402	0.0	$-1.19E-07$
6	1.1347241402	0.0	0.0

Initial iterate of 2.0 results in the following:

k	x_k	$f(x_k)$	$x_k - x_{k-1}$
0	2.0000000000	$6.10E+01$	
1	1.6806282997	$1.99E+01$	$-3.19E-01$
2	1.4307390451	$6.15E+00$	$-2.50E-01$
3	1.2549710274	$1.65E+00$	$-1.76E-01$
4	1.1615384817	$2.94E-01$	$-9.34E-02$
5	1.1363532543	$1.68E-02$	$-2.52E-02$
6	1.1347305775	$6.65E-05$	$-1.62E-03$
7	1.1347241402	0.0	$-6.44E-06$
8	1.1347241402	0.0	0.0

It should be noted that the choice of initial iterate (or guess) affects the number of iterations Newton's method takes to converge. The choice of initial iterate is very important in determining whether Newton's method will converge at all. Furthermore, Newton's method converges slowly in the beginning when the iterate is not close to a root. However, as the iterate comes closer to the root, the rate of convergence increases rapidly.

3.2 - 2 Newton's Method

It is to be noted that convergence of Newton's method in all the following problems depends on the selection of initial iterate. Furthermore, when Newton's method converges it converges faster that bisection. In addition, Newton's method generally provides more accurate results than bisection method.

3.2 - 2a
$x^3 - x^2 - x - 1 = 0$

Since the root of the equation lies in the range $[0, 2]$, we first try an initial iterate $x_o = 1.0$. However, at $x_o = 1.0$ the first derivative of the given function fails to exist giving NaN as a result. It points towards the fact that the choice of initial iterate is very important in determining whether Newton's method will converge.

Next we try an initial iterate $x_o = 2.0$. The application of Newton's method results in the following iterations:

k	$\mathbf{x}_k$	$\mathbf{f}(\mathbf{x}_k)$	$\mathbf{x}_{k+1}$
0	2.0000000000	$1.00E+00$	–
1	1.857142	$9.913E-02$	1.839544
2	1.839544	$1.410E-03$	1.839287
3	1.839287	$3.001E-07$	1.839287
4	1.839287	$1.377E-14$	1.839287

It can be seen that the new initial iterate has resulted in convergence to 1.839287 within 4 iterations. It is worth noting that the same problem when solved using bisection method took more than 15 iterations. Furthermore, Newton's method has provided more accurate results. It shows that when Newton's method converges, it converges faster than bisection method.

3.2 - 2b
$x - 1 - 0.3 * \cos(x) = 0$

Initial iterate = $x_o = 1.0$
Iterations to converge = 3
Root = 1.1284251

3.2 - 2c
$\cos(x) - \sin(x) - \frac{1}{2} = 0$

Initial iterate = $x_o = 1.0$
Iterations to converge = 4
Root = 0.4240310

3.2 - 2d
$x - e^x = 0$

Intial iterate = $x_o = 1.0$
Iterations to converge = 4
Root = 0.5671433

3.2 - 2e
$e^{-x} - \sin(x) = 0$

Initial iterate = $x_o = 1.0$
Iterations to converge = 4
Root = 0.5885327

3.2 - 2f
$x^3 - 2x - 2 = 0$

Initial iterate = $x_o = 1.0$
Iterations to converge = 5
Root = 1.7692923

3.2 - 2g
$x^4 - x - 1 = 0$

Initial iterate = $x_o = 1.0$
Iterations to converge = 3
Root = 1.22074408

Problem 3.2 - 10 Newton's Method:
$f(x) = x^3 - 3x^2 + 3x - 1 = 0$

We tried various initial iterates for solving the given problem using Newton's method. The final results of the experimentation are summarized in the following:

$\mathbf{x}_0$	Iterationstoconverge	RootFound
-100	39	1.00244546
0.0	16	0.99898504
0.9	12	0.99668354
0.99	04	0.99716872
1.0	$*$	NaN
1.001	10	1.00406170
1.5	14	1.00029719
10.0	21	1.00123203
1000000	48	1.00476933

Observations:

- The convergence rate of Newton's method is highly dependent on the initial iterate (or initial guess).
- Newton's method fails to converge when the first derivative does not exist for a given iterate (as is the case with $x_0 = 1$).
- Since this problem has multiple roots at x = 1.0 (as would be seen in the solution using companion matrix), the convergence becomes very slow near the root. For elaboration purposes we only provide full list of iteration for $x_o = 0.0$ in the following table. It can be seen that the Newton iterations start converging quickly towards the root; however, the fact that there were multiple roots at $x = 1.0$ results in slower convergence near the root(s).

k	$\mathbf{x}_k$	$\mathbf{f}(\mathbf{x}_k)$	$\mathbf{x}_{k+1}$
0	0.0	$-1.000E+00$	
1	0.333333333333333	$-2.963E-01$	0.555555555555556
2	0.555555555555556	$5.926E-01$	0.703703703703704
3	0.703703703703704	$2.634E-01$	0.802469135802470
4	0.802469135802470	$1.171E-01$	0.868312757201647
5	0.868312757201647	$5.202E-02$	0.912208504801105
6	0.912208504801105	$2.312E-02$	0.941472336534071
7	0.941472336534071	$1.028E-02$	0.960981557689391
8	0.960981557689391	$4.567E-03$	0.973987705126321
9	0.973987705126321	$2.030E-03$	0.982658470084071
10	0.982658470084071	$9.022E-04$	0.988438980056271
11	0.988438980056271	$4.010E-04$	0.992292653372078
12	0.992292653372078	$1.782E-04$	0.994861768914193
13	0.994861768914193	$7.920E-05$	0.996574512606094
14	0.996574512606094	$3.520E-05$	0.997716341740185
15	0.997716341740185	$1.565E-05$	0.998477561150052
16	0.998477561150052	$6.953Ee-06$	0.998985040735747

3.2 - 10
Companion matrix method: $f(x) = x^3 - 3x^2 + 3x - 1 = 0$

The companion matrix for the given polynomial is given by:

$$\begin{bmatrix} 3 & -3 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

Using *eig* routine in MATLAB results in the following eigenvalues of the companion matrix (which are also the roots of the given polynomial): 1.0, 1.0, 1.0 (notice repeated roots at 1.0).

3.2 - 10
Laguerre's method: $f(x = x^3 - 3x^2 + 3x - 1 = 0$

Applying Laguerre's metho to the same problem with various initial iterates. Laguerre converges to the root(s) in only 2 iterations. Once again, notice the fast convergence of Laguerre.

Problem 3.2 - 11 Newton's Method:
$f(x) = x^4 - 5.4x^3 + 10.56x^2 - 8.954x + 2.7951 = 0$

We tried various initial iterates for solving the given problem using Newton's method. We summarize the final results of our experimentations in the followin:

$\mathbf{x}_0$	Iterationstoconverge	RootFound
-100	31	1.09353614
0.0	18	1.09555769
0.9	16	1.09426618
0.99	53	1.09545934
1.0	$*$	1.10014844
1.1	19	1.09402192
1.4	12	1.10592306
10.0	14	2.09999990
1000000	26	2.10000062

Observations:

- The convergence rate of Newton's method is highly dependent on the initial iterate (or initial guess).
- Newton's method fails to converge when the first derivative does not exist for a given iterate (as is the case with $x_0 = 1$ and an the first derivative fails to exist for an intermediate iterate $x = 1.10014844$; $f'(1.10014844) \approx 0$.
- Since this problem has multiple roots at x = 1.1 (as would be seen in the solution using companion matrix), the convergence becomes very slow near the root. For elaboration purposes we only provide full list of iteration for $x_0 = 0.0$ in the following table. It can be seen that the Newton's iterations started converging fast towards the root; however, the fact that there were multiple roots at $x = 1.1$ results in slower convergence near the root(s).

k	$\mathbf{x}_k$	$\mathbf{f}(\mathbf{x}_k)$	$\mathbf{x}_{k+1}$
0	0.0	$2.795E+00$	0.312162162162162
1	0.312162162162162	$8.743E-01$	0.541140514415998
2	0.541140514415998	$2.721E-01$	0.707541762946985
3	0.707541762946985	$8.417E-02$	0.827126376012018
4	0.827126376012018	$2.586E-02$	0.912018006441916
5	0.912018006441916	$7.891E-03$	0.971539197945021
6	0.971539197945021	$2.392E-03$	1.012794020977678
7	1.012794020977678	$7.210E-04$	1.041105710148299
8	1.041105710148299	$2.163E-04$	1.060379806903367
9	1.060379806903367	$6.466E-05$	1.073420871821820
10	1.073420871821820	$1.928E-05$	1.082204773308455
11	1.082204773308455	$5.735E-06$	1.088102145470841
12	1.088102145470841	$1.704E-06$	1.092052613829682
13	1.092052613829682	$5.060E-07$	1.094694798256185
14	1.094694798256185	$1.501E-07$	1.096460093575119
15	1.096460093575119	$4.452E-08$	1.097638676525991
16	1.097638676525991	$1.320E-08$	1.098425166641761

3.2 - 11
Companion matrix method: $f(x) = x^4 - 5.4x^3 + 10.56x^2 - 8.954x + 2.7951 = 0$

The companion matrix for the given polynomial is:

$$\begin{bmatrix} 5.4 & -10.56 & 8.954 & -2.7951 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Using *eig* routine in MATLAB results in the following eigenvalues of the companion matrix (which are also the roots of the given polynomial): 1.1, 1.1, 1.1, 2.1 (notice repeated roots at 1.1)

3.2 - 11
Laguerre's method: $f(x) = x^4 - 5.4x^3 + 10.56x^2 - 8.954x + 2.7951 = 0$

Applying Laguerre's method to the same problem we get the following results for various starting points. Once again, notice the fast convergence of Laguerre.

k	InitialGuess(x)				
0	**0.0**	**1.06**	**1.16**	**1.2**	**2.0**
1	0.660510	1.097636	1.112887	1.126195	2.100804
2	1.155674	1.100977	1.101461	1.104020	2.100000
3	1.047032	1.100033	1.100061	1.100270	
4	1.171069	1.100005	1.100000	1.100006	
5	1.115564				
6	1.101702				
7	1.100051				
8	1.099999				
9	1.100005				

Notice that the initial iterate (or guess) affects both the convergence rate as well as root it converges to.