



**HACETTEPE UNIVERSITY  
ENGINEERING FACULTY  
ELECTRICAL AND ELECTRONICS  
ENGINEERING PROGRAM**

2023-2024  
SPRING SEMESTER

ELE708  
NUMERICAL METHODS IN ELECTRICAL ENGINEERING

HW5

N23239410 – Ali Bölücü

# 1) Computer Problems

## 1.a) 5.2

5.2. For the equation

$$f(x) = x^2 - 3x + 2 = 0,$$

each of the following functions yields an equivalent fixed-point problem:

$$g_1(x) = (x^2 + 2)/3,$$

$$g_2(x) = \sqrt{3x-2},$$

$$g_3(x) = 3 - 2/x,$$

$$g_4(x) = (x^2 - 2)/(2x - 3).$$

(a) Analyze the convergence properties of each of the corresponding fixed-point iteration schemes for the root  $x = 2$  by considering  $|g'_i(2)|$ .

(b) Confirm your analysis by implementing each of the schemes and verifying its convergence (or lack thereof) and approximate convergence rate.

a.

- The roots of this equation are  $x=1$  and  $x=2$ , we can find the value the equation going to converge by taking derivative.
- First equation is divergence for  $x=2$ , but if we calculate  $g'(1) = 2/3$ . We can see that it is going to converge to root  $x=1$
- The other 3 equations are going to converge root  $x=2$

a)

1-) $g(x) = (x^2 + 2)/3 \rightarrow g'(2) = 2x/3 = 4/3$	1-) $4/3 > 1$ , divergence
2-) $g(x) = \sqrt{3x-2} \rightarrow g'(2) = 3/(2\sqrt{3x-2}) = 3/4$	2-) $3/4 < 1$ , linear convergence
3-) $g(x) = 3 - 2/x \rightarrow g'(2) = 2/x^2 = 1/2$	3-) $1/2 < 1$ , linear convergence
4-) $g(x) = (x^2 - 2)/(2x - 3) \rightarrow g'(2) = \frac{-2 \cdot (x^2 - 2)}{(2x - 3)^2} + \frac{2x}{2x - 3} = 0$	4-) $0$ , quadratic convergence

b.

$g1(x) = (x^2 + 2)/3$		$g2(x) = \sqrt{3x-2}$	
k	x	k	x
0	2.500000000000e+00	0	2.500000000000e+00
1	2.750000000000e+00	1	2.345207879912e+00
2	3.187500000000e+00	2	2.244019527485e+00
3	4.053385416667e+00	3	2.175329534221e+00
4	6.143311112015e+00	4	2.127437097229e+00
5	1.324675713967e+01	5	2.093397069761e+00
6	5.915885823914e+01	6	2.068862298290e+00
7	1.167256836053e+03	7	2.050996561399e+00
8	4.541635071039e+05	8	2.037888535763e+00
9	6.875483039565e+10	9	2.028217347152e+00
10	1.575742234245e+21	10	2.021052211462e+00

$g3(x) = 3 - 2/x$		$g4(x) = (x^2 - 2)/(2x - 3)$	
k	x	k	x
0	2.500000000000e+00	0	2.500000000000e+00
1	2.200000000000e+00	1	2.125000000000e+00
2	2.090909090909e+00	2	2.012500000000e+00
3	2.043478260870e+00	3	2.000152439024e+00
4	2.021276595745e+00	4	2.000000023231e+00
5	2.010526315789e+00	5	2.000000000000e+00
6	2.005235602094e+00	6	2.000000000000e+00
7	2.002610966057e+00		
8	2.001303780965e+00		
9	2.000651465798e+00		
10	2.000325626832e+00		

- For the 1<sup>st</sup> equation, we can see that X does not converge to 2 and we found that it was divergence.
- For the 2<sup>nd</sup> and 3<sup>rd</sup> equations, we found that they are linear converge and by looking the image we can see that they converged to 2.
- For the 4<sup>th</sup> equation, it quadratically converge to 2. So it takes less iteration.

### 1.b) 5.3

Implement the bisection, Newton, and secant methods for solving nonlinear equations in one dimension, and test your implementations by finding at least one root for each of the following equations. What termination criterion should you use? What convergence rate is achieved in each case? Compare your results (solutions and convergence rates) with those for a library routine for solving nonlinear equations.

(a)  $x^3 - 2x - 5 = 0$ .

(b)  $e^{-x} = x$ .

(c)  $x \sin(x) = 1$ .

(d)  $x^3 - 3x^2 + 3x - 1 = 0$ .

a.  $x^3 - 2x - 5 = 0$

- Roots of this function are shown in the image, so we can converge the real value.

```
>> roots([1 0 -2 -5])

ans =

    2.0946 + 0.0000i
   -1.0473 + 1.1359i
   -1.0473 - 1.1359i
```

#### 1. Bisection Method

- With this method, we can find an accurate answer but the converge rate is too small. It linearly converged.

f(x) = x^3-2*x-5 = 0				
k	a	f(a)	b	f(b)
0	-5.0000000000e+00	-1.20000e+02	5.00000e+00	1.10000e+02
1	0.0000000000e+00	-5.00000e+00	5.00000e+00	1.10000e+02
2	0.0000000000e+00	-5.00000e+00	2.50000e+00	5.62500e+00
3	1.2500000000e+00	-5.54688e+00	2.50000e+00	5.62500e+00
4	1.8750000000e+00	-2.15820e+00	2.50000e+00	5.62500e+00
5	1.8750000000e+00	-2.15820e+00	2.18750e+00	1.09253e+00
6	2.0312500000e+00	-6.81610e-01	2.18750e+00	1.09253e+00
7	2.0312500000e+00	-6.81610e-01	2.10938e+00	1.66836e-01
8	2.0703125000e+00	-2.66864e-01	2.10938e+00	1.66836e-01
9	2.0898437500e+00	-5.24059e-02	2.10938e+00	1.66836e-01
10	2.0898437500e+00	-5.24059e-02	2.09961e+00	5.66142e-02
11	2.0898437500e+00	-5.24059e-02	2.09473e+00	1.95435e-03
12	2.0922851562e+00	-2.52632e-02	2.09473e+00	1.95435e-03
13	2.0935058594e+00	-1.16638e-02	2.09473e+00	1.95435e-03
14	2.0941162109e+00	-4.85706e-03	2.09473e+00	1.95435e-03
15	2.0944213867e+00	-1.45194e-03	2.09473e+00	1.95435e-03
16	2.0944213867e+00	-1.45194e-03	2.09457e+00	2.51058e-04
17	2.0944976807e+00	-6.00477e-04	2.09457e+00	2.51058e-04
18	2.0945358276e+00	-1.74719e-04	2.09457e+00	2.51058e-04
19	2.0945358276e+00	-1.74719e-04	2.09455e+00	3.81675e-05
20	2.0945453644e+00	-6.82761e-05	2.09455e+00	3.81675e-05

## 2. Newton Method

- We converged the same value in less iteration yet since we calculated result of derivative in each iteration, this method is costly.

```
Newton's method:

f(x) = x^3-2*x-5 = 0
k    x                                f(x)
0    5.0000000000e+00                1.10000e+02
1    3.4931506849e+00                3.06375e+01
2    2.6078357558e+00                7.51972e+00
3    2.1992092709e+00                1.23810e+00
4    2.1002366630e+00                6.36581e-02
5    2.0945695946e+00                2.02169e-04
6    2.0945514817e+00                2.06152e-09
7    2.0945514815e+00                -8.88178e-16
8    2.0945514815e+00                -8.88178e-16
```

## 3. Secant Method

- This method is more process efficient compared to the others but since we guess the result of derivative, it iterates little bit slower compared to the Newton's method.

```
Secant method:

f(x) = x^3-2*x-5 = 0
k    x                                f(x)
0    0.0000000000e+00                -5.0000000000e+00
1    -5.0000000000e+00                -1.2000000000e+02
2    2.1739130435e-01                -5.4245089176e+00
3    4.6440561246e-01                -5.8286516716e+00
4    -3.0980988379e+00                -2.8540025428e+01
5    1.3786874935e+00                -5.1367944694e+00
6    2.3613010185e+00                3.4434044074e+00
7    1.9669586739e+00                -1.3238989581e+00
8    2.0764690992e+00                -1.9977671442e-01
9    2.0959310688e+00                1.5410139616e-02
10   2.0945373421e+00                -1.5781522990e-04
11   2.0945514706e+00                -1.2250513048e-07
12   2.0945514815e+00                9.7521990483e-13
13   2.0945514815e+00                -8.8817841970e-16
```

## 4. Library Routine

x	f(x)
2.09455e+00	-1.19904e-13

b.  $e^{-x} - x = 0$

### 1. Bisection Method

- With this method, we can find an accurate answer but the converge rate is too small. It linearly converged.

f(x) = exp(-x)-x = 0				
k	a	f(a)	b	f(b)
0	-5.0000000000e+00	1.53413e+02	5.00000e+00	-4.99326e+00
1	0.0000000000e+00	1.00000e+00	5.00000e+00	-4.99326e+00
2	0.0000000000e+00	1.00000e+00	2.50000e+00	-2.41792e+00
3	0.0000000000e+00	1.00000e+00	1.25000e+00	-9.63495e-01
4	0.0000000000e+00	1.00000e+00	6.25000e-01	-8.97386e-02
5	3.1250000000e-01	4.19116e-01	6.25000e-01	-8.97386e-02
6	4.6875000000e-01	1.57034e-01	6.25000e-01	-8.97386e-02
7	5.4687500000e-01	3.18806e-02	6.25000e-01	-8.97386e-02
8	5.4687500000e-01	3.18806e-02	5.85938e-01	-2.93537e-02
9	5.6640625000e-01	1.15520e-03	5.85938e-01	-2.93537e-02
10	5.6640625000e-01	1.15520e-03	5.76172e-01	-1.41260e-02
11	5.6640625000e-01	1.15520e-03	5.71289e-01	-6.49215e-03
12	5.6640625000e-01	1.15520e-03	5.68848e-01	-2.67016e-03
13	5.6640625000e-01	1.15520e-03	5.67627e-01	-7.57902e-04
14	5.6701660156e-01	1.98544e-04	5.67627e-01	-7.57902e-04
15	5.6701660156e-01	1.98544e-04	5.67322e-01	-2.79706e-04
16	5.6701660156e-01	1.98544e-04	5.67169e-01	-4.05873e-05
17	5.6709289551e-01	7.89768e-05	5.67169e-01	-4.05873e-05
18	5.6713104248e-01	1.91943e-05	5.67169e-01	-4.05873e-05
19	5.6713104248e-01	1.91943e-05	5.67150e-01	-1.06966e-05
20	5.6714057922e-01	4.24882e-06	5.67150e-01	-1.06966e-05
21	5.6714057922e-01	4.24882e-06	5.67145e-01	-3.22390e-06
22	5.6714296341e-01	5.12456e-07	5.67145e-01	-3.22390e-06
23	5.6714296341e-01	5.12456e-07	5.67144e-01	-1.35572e-06
24	5.6714296341e-01	5.12456e-07	5.67144e-01	-4.21634e-07
25	5.6714326143e-01	4.54113e-08	5.67144e-01	-4.21634e-07
26	5.6714326143e-01	4.54113e-08	5.67143e-01	-1.88111e-07
27	5.6714326143e-01	4.54113e-08	5.67143e-01	-7.13499e-08
28	5.6714326143e-01	4.54113e-08	5.67143e-01	-1.29693e-08
29	5.6714328006e-01	1.62210e-08	5.67143e-01	-1.29693e-08
30	5.6714328937e-01	1.62585e-09	5.67143e-01	-1.29693e-08
31	5.6714328937e-01	1.62585e-09	5.67143e-01	-5.67173e-09
32	5.6714328937e-01	1.62585e-09	5.67143e-01	-2.02294e-09
33	5.6714328937e-01	1.62585e-09	5.67143e-01	-1.98548e-10
34	5.6714328995e-01	7.13649e-10	5.67143e-01	-1.98548e-10
35	5.6714329025e-01	2.57551e-10	5.67143e-01	-1.98548e-10
36	5.6714329039e-01	2.95013e-11	5.67143e-01	-1.98548e-10
37	5.6714329039e-01	2.95013e-11	5.67143e-01	-8.45234e-11



## 2. Newton Method

- We converged the same value in less iteration yet since we calculated result of derivative in each iteration, this method is costly.

$f(x) = \exp(-x) - x = 0$		
k	x	f(x)
0	5.0000000000e+00	-4.99326e+00
1	4.0157105546e-02	9.20481e-01
2	5.0963753116e-01	9.10757e-02
3	5.6653450904e-01	9.54153e-04
4	5.6714322334e-01	1.05115e-07
5	5.6714329041e-01	1.33227e-15
6	5.6714329041e-01	-1.11022e-16

## 3. Secant Method

- This method is more process efficient compared to the others but since we guess the result of derivative, it iterates little bit slower compared to the Newton's method.

$f(x) = \exp(-x) - x = 0$		
k	x	f(x)
0	0.0000000000e+00	1.0000000000e+00
1	-5.0000000000e+00	1.5341315910e+02
2	3.2805566327e-02	9.3492113363e-01
3	6.3664240640e-02	8.7465575573e-01
4	5.1152863712e-01	8.8049703098e-02
5	5.6166088185e-01	8.6002586375e-03
6	5.6708760666e-01	8.7265299469e-05
7	5.6714323512e-01	8.6642105157e-08
8	5.6714329041e-01	8.7296836426e-13
9	5.6714329041e-01	1.1102230246e-16

## 4. Library Routine

x	f(x)
5.67143e-01	-1.25912e-11

c.  $x^3 - 2x - 5 = 0$

### 1. Bisection Method

- With this method, we can find an accurate answer but the converge rate is too small. It linearly converged.

f(x) = x*sin(x)-1 = 0				
k	a	f(a)	b	f(b)
0	-5.0000000000e+00	-5.79462e+00	5.00000e+00	-5.79462e+00
1	0.0000000000e+00	-1.00000e+00	5.00000e+00	-5.79462e+00
2	0.0000000000e+00	-1.00000e+00	2.50000e+00	4.96180e-01
3	0.0000000000e+00	-1.00000e+00	1.25000e+00	1.86231e-01
4	6.2500000000e-01	-6.34314e-01	1.25000e+00	1.86231e-01
5	9.3750000000e-01	-2.44299e-01	1.25000e+00	1.86231e-01
6	1.0937500000e+00	-2.83617e-02	1.25000e+00	1.86231e-01
7	1.0937500000e+00	-2.83617e-02	1.17188e+00	7.98600e-02
8	1.0937500000e+00	-2.83617e-02	1.13281e+00	2.58847e-02
9	1.1132812500e+00	-1.21649e-03	1.13281e+00	2.58847e-02
10	1.1132812500e+00	-1.21649e-03	1.12305e+00	1.23411e-02
11	1.1132812500e+00	-1.21649e-03	1.11816e+00	5.56387e-03
12	1.1132812500e+00	-1.21649e-03	1.11572e+00	2.17406e-03
13	1.1132812500e+00	-1.21649e-03	1.11450e+00	4.78871e-04
14	1.1138916016e+00	-3.68788e-04	1.11450e+00	4.78871e-04
15	1.1138916016e+00	-3.68788e-04	1.11420e+00	5.50474e-05
16	1.1140441895e+00	-1.56869e-04	1.11420e+00	5.50474e-05
17	1.1141204834e+00	-5.09103e-05	1.11420e+00	5.50474e-05
18	1.1141204834e+00	-5.09103e-05	1.11416e+00	2.06863e-06
19	1.1141395569e+00	-2.44208e-05	1.11416e+00	2.06863e-06
20	1.1141490936e+00	-1.11761e-05	1.11416e+00	2.06863e-06
21	1.1141538620e+00	-4.55373e-06	1.11416e+00	2.06863e-06
22	1.1141562462e+00	-1.24255e-06	1.11416e+00	2.06863e-06
23	1.1141562462e+00	-1.24255e-06	1.11416e+00	4.13041e-07
24	1.1141568422e+00	-4.14754e-07	1.11416e+00	4.13041e-07
25	1.1141571403e+00	-8.56834e-10	1.11416e+00	4.13041e-07
26	1.1141571403e+00	-8.56834e-10	1.11416e+00	2.06092e-07
27	1.1141571403e+00	-8.56834e-10	1.11416e+00	1.02618e-07
28	1.1141571403e+00	-8.56834e-10	1.11416e+00	5.08803e-08
29	1.1141571403e+00	-8.56834e-10	1.11416e+00	2.50118e-08
30	1.1141571403e+00	-8.56834e-10	1.11416e+00	1.20775e-08
31	1.1141571403e+00	-8.56834e-10	1.11416e+00	5.61031e-09
32	1.1141571403e+00	-8.56834e-10	1.11416e+00	2.37674e-09
33	1.1141571403e+00	-8.56834e-10	1.11416e+00	7.59953e-10
34	1.1141571408e+00	-4.84405e-11	1.11416e+00	7.59953e-10
35	1.1141571408e+00	-4.84405e-11	1.11416e+00	3.55756e-10
36	1.1141571408e+00	-4.84405e-11	1.11416e+00	1.53658e-10
37	1.1141571408e+00	-4.84405e-11	1.11416e+00	5.26088e-11

## 2. Newton Method

- We converged the same value in less iteration yet since we calculated result of derivative in each iteration, this method is costly.

```
f(x) = x*sin(x)-1 = 0
k    x                                f(x)
0    5.0000000000e-01                -7.60287e-01
1    1.3280040340e+00                2.89054e-01
2    1.1039207287e+00                -1.42220e-02
3    1.1141535060e+00                -5.04808e-06
4    1.1141571409e+00                -7.80376e-13
5    1.1141571409e+00                2.22045e-16
```

## 3. Secant Method

- This method is more process efficient compared to the others but since we guess the result of derivative, it iterates little bit slower compared to the Newton's method.

```
f(x) = x*sin(x)-1 = 0
k    x                                f(x)
0    0.0000000000e+00                -1.0000000000e+00
1    -5.0000000000e+00                -5.7946213733e+00
2    1.0428352128e+00                -9.9161399813e-02
3    1.1480446254e+00                4.6974737160e-02
4    1.1142255810e+00                9.5050025877e-05
5    1.1141570118e+00                -1.7919821726e-07
6    1.1141571409e+00                5.2180482157e-13
7    1.1141571409e+00                2.2204460493e-16
```

## 4. Library Routine

```
x                                f(x)
1.11416e+00                    -8.77853e-13
```



d.  $x^3 - 2x - 5 = 0$

- Roots of this function are shown in the image, so we can converge the real value.

```
>> roots([1 -3 3 -1])

ans =

    1.0000 + 0.0000i
    1.0000 + 0.0000i
    1.0000 - 0.0000i
```

## 1. Bisection Method

- With this method, we can find an accurate answer but the converge rate is too small. It linearly converged.

f(x) = x^3-3*x^2+3*x-1 = 0				
k	a	f(a)	b	f(b)
0	-5.0000000000e+00	-2.16000e+02	5.00000e+00	6.40000e+01
1	0.0000000000e+00	-1.00000e+00	5.00000e+00	6.40000e+01
2	0.0000000000e+00	-1.00000e+00	2.50000e+00	3.37500e+00
3	0.0000000000e+00	-1.00000e+00	1.25000e+00	1.56250e-02
4	6.2500000000e-01	-5.27344e-02	1.25000e+00	1.56250e-02
5	9.3750000000e-01	-2.44141e-04	1.25000e+00	1.56250e-02
6	9.3750000000e-01	-2.44141e-04	1.09375e+00	8.23975e-04
7	9.3750000000e-01	-2.44141e-04	1.01562e+00	3.81470e-06
8	9.7656250000e-01	-1.28746e-05	1.01562e+00	3.81470e-06
9	9.9609375000e-01	-5.96046e-08	1.01562e+00	3.81470e-06
10	9.9609375000e-01	-5.96046e-08	1.00586e+00	2.01166e-07
11	9.9609375000e-01	-5.96046e-08	1.00098e+00	9.31323e-10
12	9.9853515625e-01	-3.14321e-09	1.00098e+00	9.31323e-10
13	9.9975585938e-01	-1.45519e-11	1.00098e+00	9.31323e-10
14	9.9975585938e-01	-1.45519e-11	1.00037e+00	4.91127e-11
15	9.9975585938e-01	-1.45519e-11	1.00006e+00	2.27374e-13
16	9.9990844727e-01	-7.67386e-13	1.00006e+00	2.27374e-13
17	9.9998474121e-01	-3.55271e-15	1.00006e+00	2.27374e-13
18	9.9998474121e-01	-3.55271e-15	1.00002e+00	1.19904e-14
19	9.9998474121e-01	-3.55271e-15	1.00000e+00	0.00000e+00
20	9.9999427795e-01	-2.22045e-16	1.00000e+00	0.00000e+00
21	9.9999427795e-01	-2.22045e-16	9.99999e-01	0.00000e+00
22	9.9999427795e-01	-2.22045e-16	9.99997e-01	0.00000e+00
23	9.9999427795e-01	-2.22045e-16	9.99995e-01	0.00000e+00
24	9.9999427795e-01	-2.22045e-16	9.99995e-01	0.00000e+00
25	9.9999427795e-01	-2.22045e-16	9.99995e-01	0.00000e+00
26	9.9999442697e-01	-2.22045e-16	9.99995e-01	0.00000e+00
27	9.9999450147e-01	-4.44089e-16	9.99995e-01	0.00000e+00
28	9.9999450147e-01	-4.44089e-16	9.99995e-01	0.00000e+00
29	9.9999450147e-01	-4.44089e-16	9.99995e-01	0.00000e+00
30	9.9999450147e-01	-4.44089e-16	9.99995e-01	0.00000e+00
31	9.9999450613e-01	-4.44089e-16	9.99995e-01	0.00000e+00
32	9.9999450846e-01	-2.22045e-16	9.99995e-01	0.00000e+00
33	9.9999450846e-01	-2.22045e-16	9.99995e-01	0.00000e+00
34	9.9999450904e-01	-4.44089e-16	9.99995e-01	0.00000e+00
35	9.9999450904e-01	-4.44089e-16	9.99995e-01	0.00000e+00
36	9.9999450918e-01	-2.22045e-16	9.99995e-01	0.00000e+00
37	9.9999450918e-01	-2.22045e-16	9.99995e-01	0.00000e+00

## 2. Newton Method

- We converged the same value in less iteration yet since we calculated result of derivative in each iteration, this method is costly.

$f(x) = x^3 - 3x^2 + 3x - 1 = 0$		
k	x	f(x)
0	5.0000000000e-01	-1.25000e-01
1	6.6666666667e-01	-3.70370e-02
2	7.7777777778e-01	-1.09739e-02
3	8.5185185185e-01	-3.25154e-03
4	9.0123456790e-01	-9.63418e-04
5	9.3415637860e-01	-2.85457e-04
6	9.5610425240e-01	-8.45799e-05
7	9.7073616827e-01	-2.50607e-05
8	9.8049077884e-01	-7.42540e-06
9	9.8699385256e-01	-2.20012e-06
10	9.9132923504e-01	-6.51887e-07
11	9.9421949003e-01	-1.93152e-07
12	9.9614632668e-01	-5.72301e-08
13	9.9743088446e-01	-1.69571e-08
14	9.9828725630e-01	-5.02432e-09
15	9.9885817088e-01	-1.48869e-09
16	9.9923878057e-01	-4.41092e-10
17	9.9949252042e-01	-1.30694e-10
18	9.9966168041e-01	-3.87241e-11
19	9.9977445368e-01	-1.14739e-11
20	9.9984963667e-01	-3.40017e-12

## 3. Secant Method

- This method is more process efficient compared to the others but since we guess the result of derivative, it iterates little bit slower compared to the Newton's method.

$f(x) = x^3 - 3x^2 + 3x - 1 = 0$		
k	x	f(x)
0	0.0000000000e+00	-1.0000000000e+00
1	-5.0000000000e+00	-2.1600000000e+02
2	2.3255813953e-02	-9.3184247928e-01
3	4.5020463847e-02	-8.7092788558e-01
4	3.5620107528e-01	-2.6683988264e-01
5	4.9365687216e-01	-1.2981795400e-01
6	6.2388588364e-01	-5.3205790634e-02
7	7.1432762572e-01	-2.3313352642e-02
8	7.8486386726e-01	-9.9572651632e-03
9	8.3745022998e-01	-4.2949595513e-03
10	8.7733792173e-01	-1.8455718443e-03
11	9.0739261688e-01	-7.9421271668e-04
12	9.3009639188e-01	-3.4158498968e-04
13	9.4723027097e-01	-1.4694492522e-04
14	9.6016561699e-01	-6.3208325381e-05
15	9.6992982477e-01	-2.7189916666e-05
16	9.7730072220e-01	-1.1695966614e-05
17	9.8286481462e-01	-5.0311402204e-06
18	9.8706503344e-01	-2.1641927166e-06
19	9.9023568199e-01	-9.3094869080e-07
20	9.9262913460e-01	-4.0045658789e-07

## 4. Library Routine

x	f(x)
1.00001e+00	0.00000e+00

1.c)

5.15

If an amount  $a$  is borrowed at interest rate  $r$  for  $n$  years, then the total amount to be repaid is given by

$$a(1+r)^n.$$

Yearly payments of  $p$  each would reduce this amount by

$$\sum_{i=0}^{n-1} p(1+r)^i = p \frac{(1+r)^n - 1}{r}.$$

The loan will be repaid when these two quantities are equal.

(a) For a loan of  $a = \$100,000$  and yearly payments of  $p = \$10,000$ , how long will it take to pay off the loan if the interest rate is 6 percent, i.e.,  $r = 0.06$ ?

(b) For a loan of  $a = \$100,000$  and yearly payments of  $p = \$10,000$ , what interest rate  $r$  would be required for the loan to be paid off in  $n = 20$  years? (c) For a loan of  $a = \$100,000$ , how large must the yearly payments  $p$  be for the loan to be paid off in  $n = 20$  years at 6 percent interest? You may use any method you like to solve the given equation in each case. For the purpose of this problem, we will treat  $n$  as a continuous variable (i.e., it can have fractional values).

- What we borrowed is:

$$a \cdot (1+r)^n$$

- What we paid back:

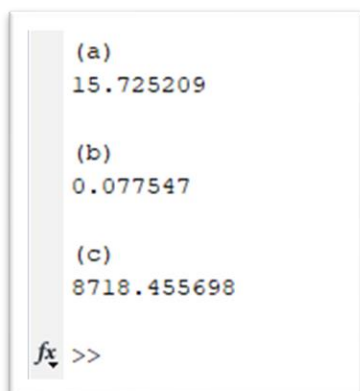
$$p \cdot \frac{(1+r)^n - 1}{r}$$

- When we solve this equation with each other

$$(1+r)^n \cdot (r \cdot a - p) - p = 0$$

This equation can be used to solve following problems. Since it is a one-dimensional problem, “fzero” function is going to be used to solve.

- The result is given below:



```
(a)
15.725209

(b)
0.077547

(c)
8718.455698

fx >>
```

1.d)

5.21

Lorenz derived a simple system of ordinary differential equations describing buoyant convection in a fluid as a crude model for atmospheric circulation. At steady state, the convective velocity  $x$ , temperature gradient  $y$ , and heat flow  $z$  satisfy the system of nonlinear equations

$$\begin{aligned}\sigma(y - x) &= 0, \\ rx - y - xz &= 0, \\ xy - bz &= 0,\end{aligned}$$

where  $\sigma$  (the Prandtl number),  $r$  (the Rayleigh number), and  $b$  are positive constants that depend on the properties of the fluid, the applied temperature gradient, and the geometry of the problem. Typical values Lorenz used are  $\sigma = 10$ ,  $r = 28$ , and  $b = 8/3$ . Write a program using Newton's method to solve this system of equations. You should find three different solutions.

---

- In this question, we have a system of ODE. In order to solve these equations with Newton's method, we need to calculate a Jacobian matrix for the derivatives.
- The function and Jacobian matrix defined as:

```
f = @(x,y,z) [sigma*(y-x); r*x-y-x*z; x*y-b*z];  
df = @(x,y,z) [-sigma sigma 0; r-z -1 -x; y x -b];
```

- The Newton method defined as:

```
while norm(step_size) > 1e-10 && k < 30  
  
    fx = f(x, y,z);  
    dfx = df(x, y,z);  
  
    step_size = - dfx\fx;  
  
    x = x + step_size(1);  
    y = y + step_size(2);  
    z = z + step_size(3);  
  
    k = k+1;  
end
```

- As the question said, we have 3 different solutions.

- For the first one we gaved the initial guess as [-1 -1 -1] and the result is:

```
x =  
    0  
  
y =  
    0  
  
z =  
    0
```

- For the first one we gaved the initial guess as [-10 -10 -10] and the result is:

```
x =  
-8.4853  
  
y =  
-8.4853  
  
z =  
27
```

- For the first one we gaved the initial guess as [10 10 10] and the result is:

```
x =  
8.4853  
  
y =  
8.4853  
  
z =  
27
```



1.e)

5.28

Newton's method can be used to compute the inverse of a nonsingular  $n \times n$  matrix  $A$ . If we define the function  $F: \mathbb{R}^{n \times n} \rightarrow \mathbb{R}^{n \times n}$  by

$$F(X) = I - AX$$

where  $X$  is an  $n \times n$  matrix, then  $F(X) = O$  precisely when  $X = A^{-1}$ . Because  $F'(X) = -A$ , Newton's method for solving this equation has the form

$$X_{k+1} = X_k - [F'(X_k)]^{-1}F(X_k) = X_k + A^{-1}(I - AX_k)$$

But  $A^{-1}$  is what we are trying to compute, so instead we use the current approximation to  $A^{-1}$ , namely  $X_k$ . Thus, the iteration scheme takes the form

$$X_{k+1} = X_k + X_k(I - AX_k)$$

(a) If we define the residual matrix

$$R_k = I - AX_k$$

and the error matrix

$$E_k = A^{-1} - X_k$$

show that

$$R_{k+1} = R_k^2 \text{ and } E_{k+1} = E_k A E_k$$

from which we can conclude that the convergence rate is quadratic, despite using only an approximate derivative.

(b) Write a program to compute the inverse of a given input matrix  $A$  using this iteration scheme. A reasonable starting guess is to take

$$X_0 = \frac{A^T}{\|A\|_1 \cdot \|A\|_{\inf}}$$

Test your program on some random matrices and compare its accuracy and efficiency with conventional methods for computing the inverse, such as LU factorization or Gauss-Jordan elimination.

---

a.

1. Proof of  $R_{k+1} = R_k^2$

- $R_{k+1} = I - A \cdot X_{k+1}$
- Using  $X_{k+1} = X_k + X_k(I - A \cdot X_k)$
- $R_{k+1} = I - A \cdot (X_k + X_k(I - A \cdot X_k))$
- $R_{k+1} = I - A \cdot X_k + AX_k(I - A \cdot X_k)$
- $R_{k+1} = I - A \cdot X_k + AX_k - (A \cdot X_k)^2$
- $R_{k+1} = (I - A \cdot X_k)^2$
- $R_{k+1} = R_k^2$

## 2. Proof of $E_{k+1} = E_k A E_k$

- $E_{k+1} = A^{-1} - X_{k+1}$
- Using  $X_{k+1} = X_k + X_k \cdot (I - A \cdot X_k)$
- $E_{k+1} = A^{-1} - X_k - X_k \cdot (I - A \cdot X_k)$  → Substitute the  $X_k$
- $E_{k+1} = A^{-1} - X_k - (I - X_k A) \cdot X_k$  → Multiply with  $x_k$  then take  $x_k$  parenthesis from the right side
- $E_{k+1} = A^{-1} - X_k - (A^{-1} A - X_k A) \cdot X_k$  → Substitute  $I = A^{-1} \cdot A$
- $E_{k+1} = A^{-1} - X_k - (A^{-1} - X_k) \cdot A X_k$  → Take  $A$  parenthesis from right side
- $E_{k+1} = (A^{-1} - X_k) \cdot (I - A X_k)$  → Take  $(A^{-1} - X_k)$  parenthesis
- $E_{k+1} = (A^{-1} - X_k) \cdot (A A^{-1} - A X_k)$  → Substitute  $I = A^{-1} \cdot A$
- $E_{k+1} = (A^{-1} - X_k) \cdot A \cdot (A^{-1} - X_k)$  → Substitute  $E_k = A^{-1} - X_k$
- $E_{k+1} = E_k \cdot A \cdot E_k$

b. In this part we applied the given theorem as:

```
X = A'/(norm(A,1)*norm(A,inf)); % best initial guess
R = I-A*X; % residual matrix
E = A'-X; % error matrix
```

And used the E as stopping criteria

- 1. The result is same for both method

```
>> A

A =

    -0.5000    9.0000   -9.0000
     1.0000   12.0000  -12.0000
    -1.0000   30.0000   30.0000

>> CP_5_28(A)
-----
Newtons method:

X =

    -0.8000    0.6000   -0.0000
     0.0200    0.0267    0.0167
    -0.0467   -0.0067    0.0167

-----
Library routine:

inv_A_library =

    -0.8000    0.6000         0
     0.0200    0.0267    0.0167
    -0.0467   -0.0067    0.0167

-----
fx >>
```

- 2. The result is same for both method

```
>> B = [2 3 1; 1 2 3; 3 1 2]
```

```
B =
```

2	3	1
1	2	3
3	1	2

```
>> CP_5_28(B)
```

```
-----  
Newton's method:
```

```
X =
```

0.0556	-0.2778	0.3889
0.3889	0.0556	-0.2778
-0.2778	0.3889	0.0556

```
-----  
Library routine:
```

```
inv_A_library =
```

0.0556	-0.2778	0.3889
0.3889	0.0556	-0.2778
-0.2778	0.3889	0.0556