ECSE 443 – ASSIGNMENT 1
MUHAMMAD TAHA – 260505597

Chapter 1

Exercises

1.2

a) absolute error = $3 - \pi = -0.1416$

relative error = $\frac{3-\pi}{\pi} = -0.004507$

b) absolute error = $3.14 - \pi = -1.59\times10^{-3}$

relative error = $\frac{3.14-\pi}{\pi} = -5.07\times10^{-4}$

c) absolute error = $\frac{22}{7} - \pi = -1.26\times10^{-3}$

relative error = $\frac{\frac{22}{7}-\pi}{\pi} = -4.03\times10^{-4}$

1.5

$$Cond = \frac{(|(x+\Delta x) - (y+\Delta y)) - (x-y)|/|x-y|}{|(|x+\Delta x| + |y+\Delta y|) - (|x|+|y|)|/(|x|+|y|)}$$

$$Cond = \frac{|\Delta x - \Delta y|/|x-y|}{(|\Delta x| - |\Delta y|)/(|x|+|y|)}$$

$$Cond = \frac{\frac{|\Delta x - \Delta y|}{\epsilon}}{\frac{|\Delta x| - |\Delta y|}{1}}$$

$$Cond = \frac{1}{\epsilon}$$

This shows that the subtraction is sensitive when ϵ is small, which translates to x ≈ y.

1.6

a) For x = 0.1, forward error = 1.666×10^{-4} and the relative error = 1.674×10^{-4}
For x = 0.5, forward error = 2.057×10^{-2} and the relative error = 2.359×10^{-2}
For x = 1.0, forward error = 1.585×10^{-1} and the relative error = 5.708×10^{-1}

b) For x = 0.1, forward error = -8.331×10^{-8} and the relative error = -8.373×10^{-8}
For x = 0.5, forward error = -2.589×10^{-4} and the relative error = -2.950×10^{-4}
For x = 1.0, forward error = -8.138×10^{-3} and the relative error = -1.490×10^{-2}

1.10

$$Expression = \frac{1}{1-x} - \frac{1}{1+x}$$

a) It is difficult to compute the numerical values for the given expression when x is close to 0.

b) The following rearrangements of the expression gives a more accurate result in floating point arithmetic:

$$Expression = \frac{1}{1-x} - \frac{1}{1+x}$$

$$Expression = \frac{1+x}{(1-x)(1+x)} - \frac{1-x}{(1+x)(1-x)}$$

$$Expression = \frac{1+x-1+x}{(1-x)(1+x)}$$

$$Expression = \frac{2x}{1-x^2}$$

1.22

a) $UFL = \beta \times L = (10)(-98) = -980$

b) $x - y = (6.87 - 6.81)\times 10^{-97} = 0.06\times 10^{-97} = 6\times 10^{-99}$

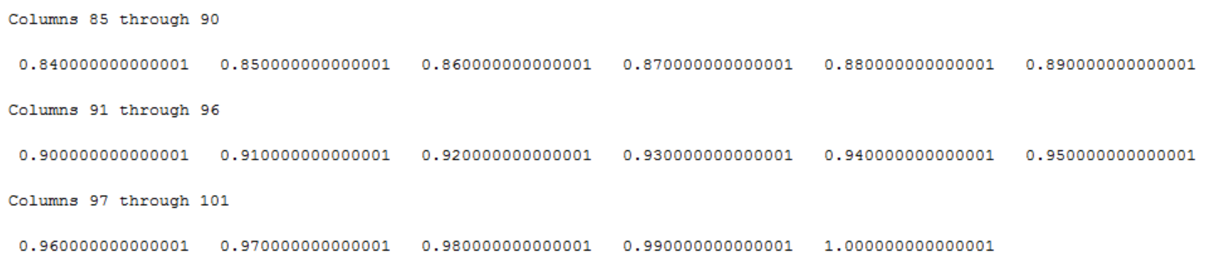
c) If the system permitted gradual overflow, the result of x-y would be 0.60×10^{-98}

Computer Problems

1.6

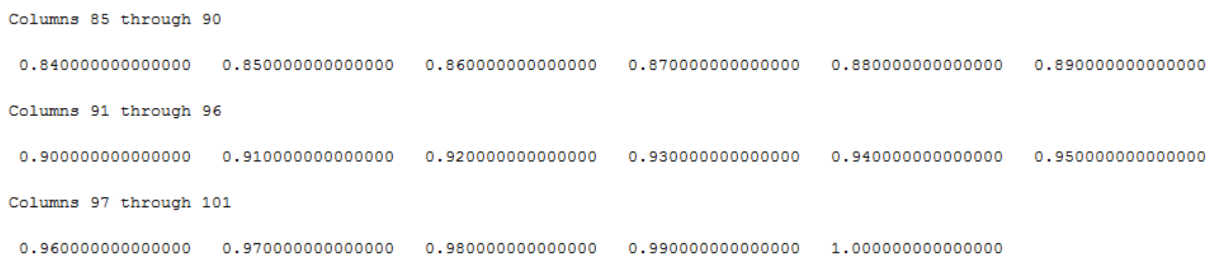
a) Method 2 is better as method 1 accumulates rounding error for large n.

b) The difference between methods 1 and 2 can be seen when a=0, b=1 and n=100. The following screenshots of the last few terms of the output intervals from both methods is shown below.



```
Columns 85 through 90

 0.840000000000001   0.850000000000001   0.860000000000001   0.870000000000001   0.880000000000001   0.890000000000001

Columns 91 through 96

 0.900000000000001   0.910000000000001   0.920000000000001   0.930000000000001   0.940000000000001   0.950000000000001

Columns 97 through 101

 0.960000000000001   0.970000000000001   0.980000000000001   0.990000000000001   1.000000000000001
```

Figure 1: Method 1

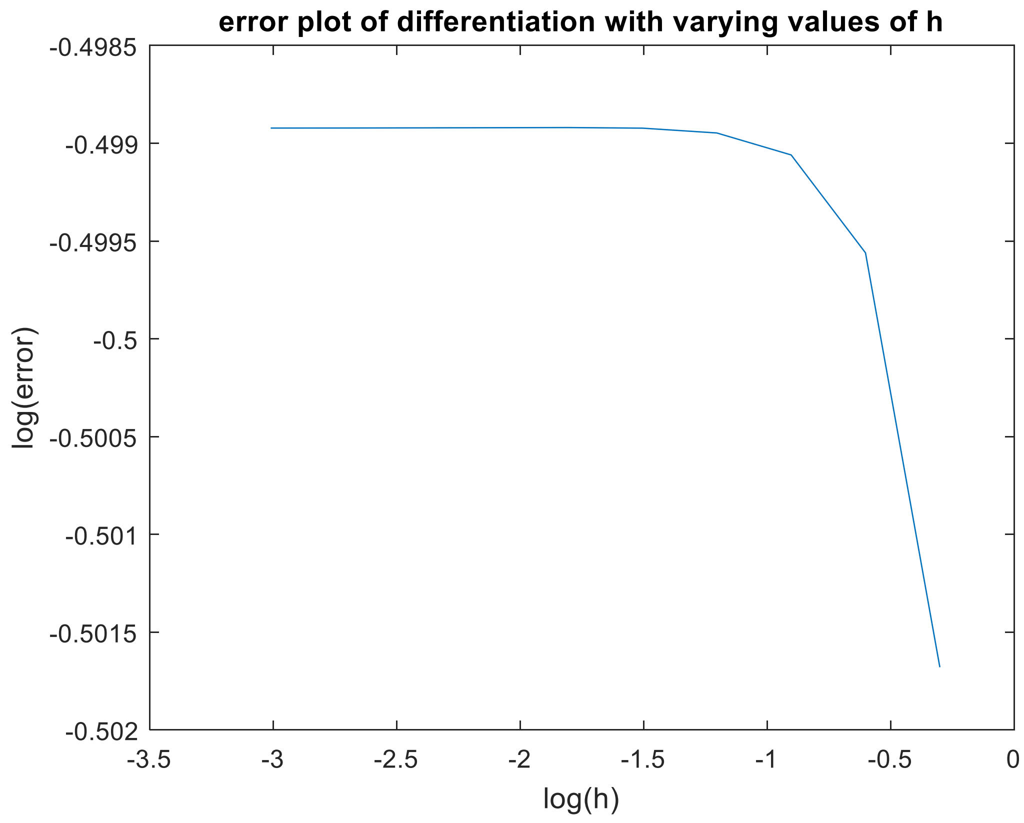


```
Columns 85 through 90

 0.840000000000000   0.850000000000000   0.860000000000000   0.870000000000000   0.880000000000000   0.890000000000000

Columns 91 through 96

 0.900000000000000   0.910000000000000   0.920000000000000   0.930000000000000   0.940000000000000   0.950000000000000

Columns 97 through 101

 0.960000000000000   0.970000000000000   0.980000000000000   0.990000000000000   1.000000000000000
```

Figure 2: Method 2

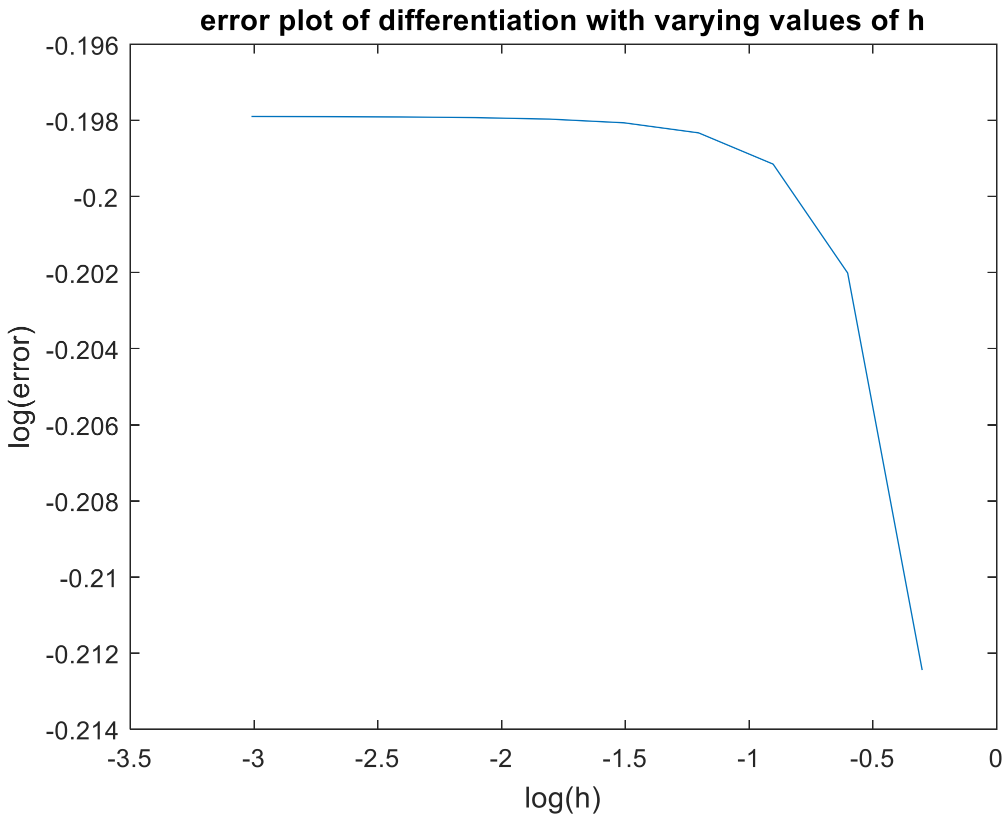
1.7

a) The error plot obtained from finite-difference formula is shown below.



The minimum value of the error is 0.3150, which corresponds to -0.5017 on the logarithmic scale. The approximation is less accurate as h decreases.

b) The error plot obtained from centered difference formula is shown below.



The minimum value of the error is 0.6131, which corresponds to -0.21244 on the logarithmic scale. The approximation is less accurate as h decreases.

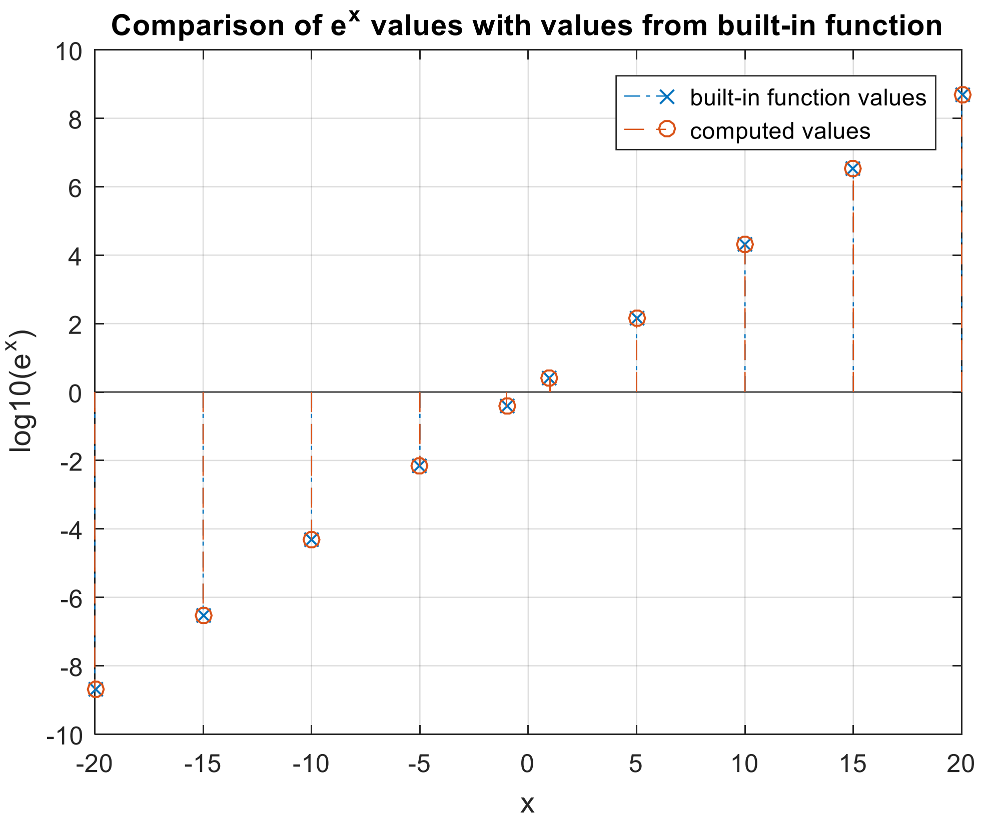
1.9

b) The stopping criterion to be used is the Lagrange form of the remainder term for the given Taylor series expression of e^x. The formula used for this remainder is as follows:

$$R_i = \frac{e^x(x^i)}{i!}$$

The criterion used an approximation for the remainder, which was to check when the remainder term is smaller than 10^{-6}.

c) The following graph shows the comparison of the computed values of e^x from the Taylor expression with the values computed by the built-in function for e^x in MATLAB.



d) For negative x, the corresponding positive value of x is used to find e^x, and then the result is achieved by computing $\frac{1}{e^x}$

e) The negative terms are the even terms and the positive terms are the odd terms in the series. The even and odd parts of the series could be separately added together and the resulting terms then added to arrive at the final solution. This could give a more accurate result for negative x, since by doing this any potential cancellation errors could be avoided.

1.10

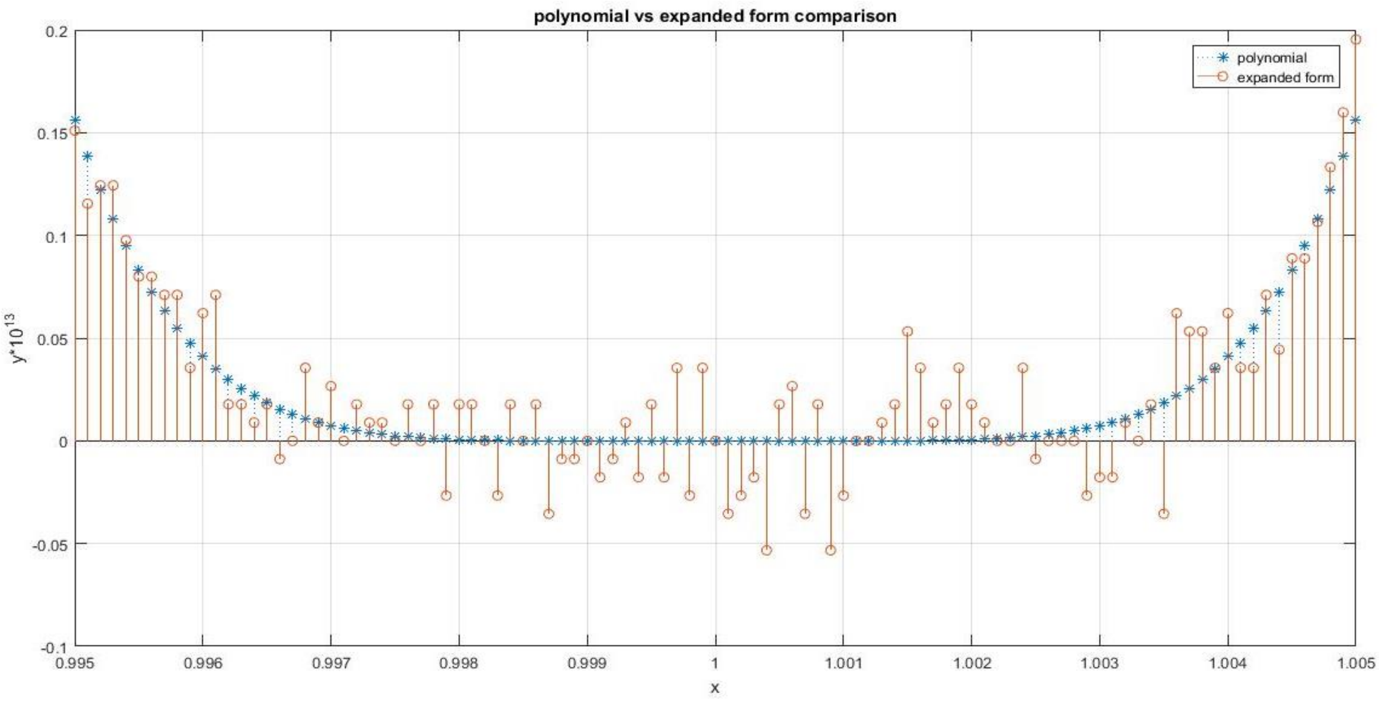
The following screenshots from MALTAB shows the output of the function with the values specified in question for testing. The function takes the input a, b, c in that order respectively.

```
>> quadraticSolver(6,5,-4)

ans =

   0.500000000000000
  -1.333333333333333

>> quadraticSolver(6*10^30,5*10^30,-4*10^30)

ans =

   0.500000000000000
  -1.333333333333333

>> quadraticSolver(0,1,1)

ans =

    -1
```

```
>> quadraticSolver(1,-10^5,1)

ans =

   1.0e+04 *

   9.999996614643589
   0.000000001000000

>> quadraticSolver(1,-4,3.999999)

ans =

   2.001000000000070
   1.998999999999930

>> quadraticSolver(10^-30, -10^30, 10^30)

ans =

   Inf
     1
```

1.14

The plot which compares the polynomial form of the expression against the expanded form is shown below:



The compact form of the polynomial will always give answer greater than or equal to 0, since the power 6, would make any negative result positive. On the other hand, the expanded form of the polynomial fixes this problem, but can introduce cancellation errors, therefore giving inaccurate answers.

Chapter 2

Exercises

2.7

a) det(A) = $1-(1+\epsilon)(1-\epsilon) = \epsilon^2$

b) The computed value of det(A) will be 0 when when $\epsilon < \sqrt{\epsilon_{mach}}$

c)
$$A = \begin{bmatrix} 1 & 1+\epsilon \\ 1-\epsilon & 1 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 0 \\ 1-\epsilon & 1 \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 1+\epsilon \\ 0 & \epsilon^2 \end{bmatrix} = LU$$

d) U is singular when $\epsilon < \sqrt{\epsilon_{mach}}$

2.13

The lower triangular system $L_1 x = b$, will first be solved via forward substitution and then the system $L_2 y = c - bx$ will be solved, also by forward substitution.

2.21

The formula to implement is as follows:

$$x = B^{-1}(2A+I)(C^{-1}+A)b$$

$$x = B^{-1}(2AC^{-1} + 2A^2 + C^{-1} + A)b$$

$$x = B^{-1}2AC^{-1}b + B^{-1}2A^2b + B^{-1}C^{-1}b + B^{-1}Ab$$

This can be implemented by first solving for y_1 in $Cy_1 = b$, and computing $y_2 = Ab$. From there, find $y_3 = y_1 + y_2$. Next find $y_4 = 2Ay_3 + y_3$. From there, x can be found by $Bx = y_3$.

2.22

Consider an arbitrary step during the process of LU factorization and let that step be step number m. The Gaussian elimination would require $2(n-m+1)$ multiplications and additions.

The total number of multiplications required is given as follows:

$$\sum_{k=1}^{n}(n-k+1)^2 = \sum_{k=1}^{n} k^2 = \frac{n(n+1)(2n+1)}{6} \approx \frac{2n^3}{6} \approx \frac{n^3}{3}$$

2.32

a) $\|A\|_{max} = max_{i,j}\left|a_{ij}\right|$

Property 1: As $A \neq 0$ then there is at least one non-zero a_{ij}. Hence, $\|A\|_{max} > 0$ for all $A \neq 0$.

Property 2: $||rA||_{max} = max_{i,j}\left|ra_{ij}\right| = |r|max_{i,j}\left|a_{ij}\right| = |r|||A||_{max}$

$||a_{ij} + b_{ij}||_{max} = max_{i,j}||a_{ij} + b_{ij}||$

$= max_{i,j}(|a_{ij}| + |b_{ij}|)$

$= max_{i,j}|a_{ij}| + max_{i,j}|b_{ij}|$

$= ||\boldsymbol{A}||_{max} + ||\boldsymbol{B}||_{max}$

Therefore, $||\boldsymbol{A}+\boldsymbol{B}||_{max} = ||\boldsymbol{A}||_{max} + ||\boldsymbol{B}||_{max}$

b) ||A||_F = $(\sum_{i.j} |a_{ij}|^2)^{1/2}$

Property 1: As $A \neq 0$ then there is at least one non-zero a_{ij}. Hence, $\|A\|_{max} > 0$ for all $A \neq 0$.

Property 2: $||rA||_F = (\sum_{i.j} |ra_{ij}|^2)^{\frac{1}{2}} = r(\sum_{i.j} |ra_{ij}|^2)^{\frac{1}{2}} = |r|||A||_F$

$$\left|\left|a_{ij} + b_{ij}\right|\right|_2^2 = \sum_{i,j}^{n} |a_{ij}|^2 + 2(\sum_{i,j}^{n} |a_{ij}|^2)^{\frac{1}{2}}(\sum_{i,j}^{n} |b_{ij}|^2)^{\frac{1}{2}} + \sum_{i,j}^{n} |b_{ij}|^2$$

$$= ((\sum_{i,j}^{n} |a_{ij}|^2)^{\frac{1}{2}} + (\sum_{i,j}^{n} |b_{ij}|^2)^{\frac{1}{2}})^2 = (\left||A|\right|_F + \left||B|\right|_F)^2$$

Hence $||\mathrm{A}+\mathrm{B}||_F = (\left||A|\right|_F + \left||B|\right|_F)^2$

2.37

a) $e_1^T B e_1 = \begin{bmatrix}1\\0\end{bmatrix}^T \begin{bmatrix}\alpha & a^T\\ a & A\end{bmatrix} \begin{bmatrix}1\\0\end{bmatrix} = \alpha$

As B is positive definite, α must also be positive. Now consider x to be a non-zero vector of size n. We have:

$$x^T A x = \begin{bmatrix}0\\x\end{bmatrix}^T \begin{bmatrix}\alpha & a^T\\ a & A\end{bmatrix} \begin{bmatrix}0\\x\end{bmatrix} > 0$$

Therefore, A must also be positive definite.

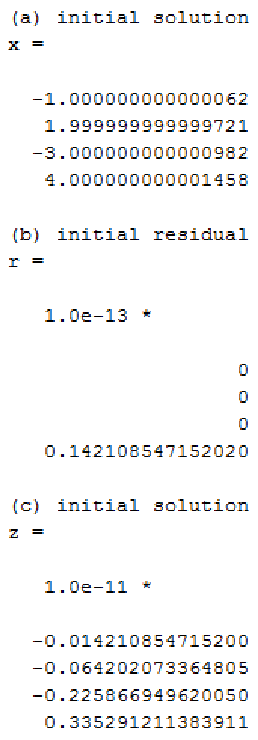
b) Let $LL^T = A - \frac{aa^T}{\sqrt{\alpha}}$. Therefore, the Cholesky factorization is as follows:

$$\begin{bmatrix}\sqrt{\alpha} & O\\ \dfrac{a}{\sqrt{\alpha}} & L\end{bmatrix} \begin{bmatrix}\sqrt{\alpha} & \dfrac{a^T}{\sqrt{\alpha}}\\ \dfrac{a}{\sqrt{\alpha}} & L^T\end{bmatrix} = \begin{bmatrix}\alpha & a^T\\ a & A\end{bmatrix}$$

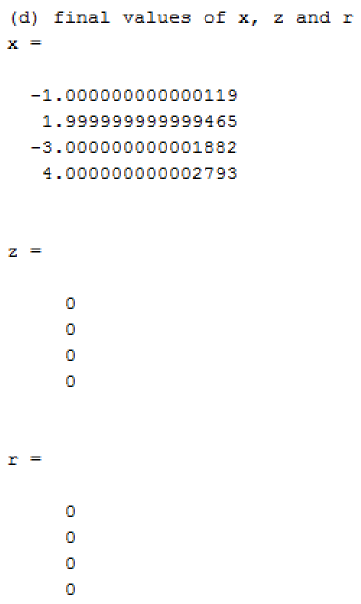
Computer Problems

2.5

The following snapshots show the results obtained in MATLAB for each part of the question



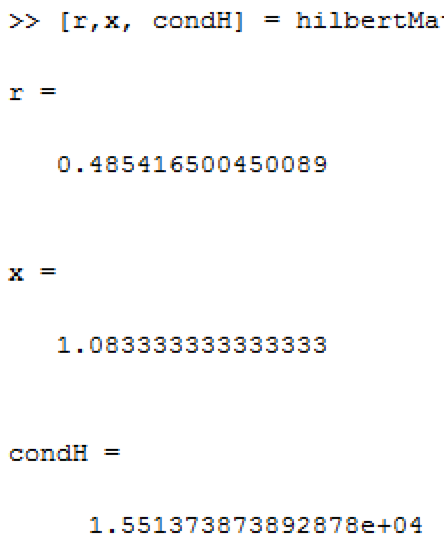
```
(a) initial solution
x =

  -1.000000000000062
   1.999999999999721
  -3.000000000000982
   4.000000000001458

(b) initial residual
r =

   1.0e-13 *

                   0
                   0
                   0
   0.142108547152020

(c) initial solution
z =

   1.0e-11 *

  -0.014210854715200
  -0.064202073364805
  -0.225866949620050
   0.335291211383911
```



```
(d) final values of x, z and r
x =

  -1.000000000000119
   1.999999999999465
  -3.000000000001882
   4.000000000002793


z =

     0
     0
     0
     0


r =

     0
     0
     0
     0
```

2.6

At n = 4, the error exceeds 100%, as shown in the snapshot below, where r is the residual-norm, x is the error-norm and condH is cond(H).

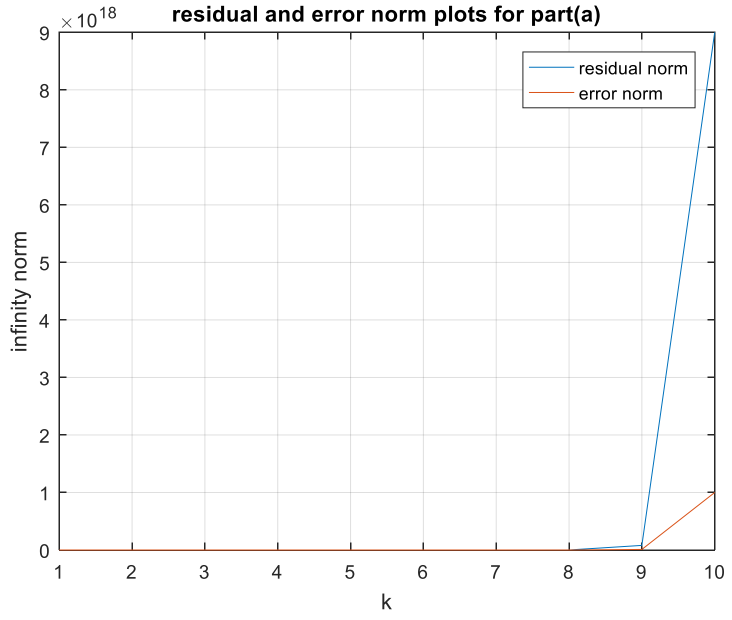


```
>> [r,x, condH] = hilbertMatrix(4)

r =

   0.485416500450089


x =

   1.083333333333333


condH =

     1.551373873892878e+04
```

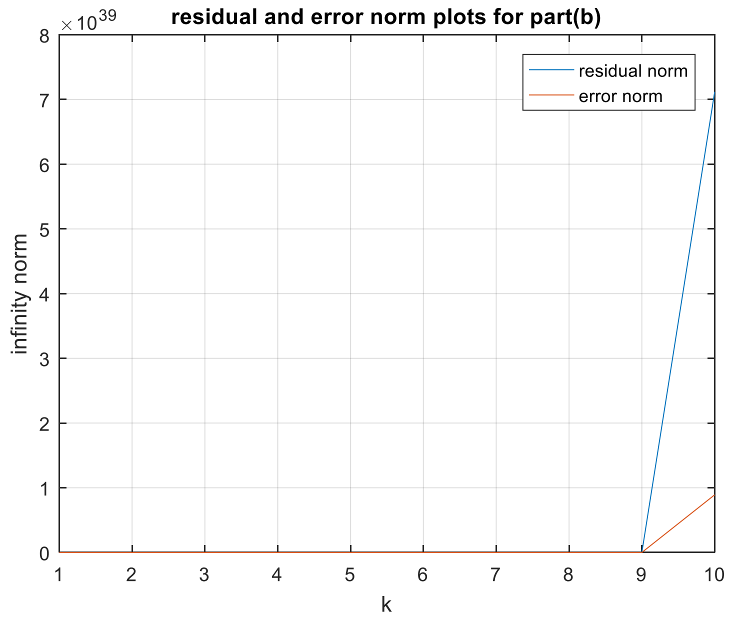
The number of correct digits in the solution can be approximated by: $16 - log_{10}(cond(H))$

2.9

a) The accuracy of solution decreases with decreasing value of ϵ, as can be seen in the plot below.



b) The accuracy of solution decreases with decreasing value of ϵ, as can be seen in the plot below.



2.11

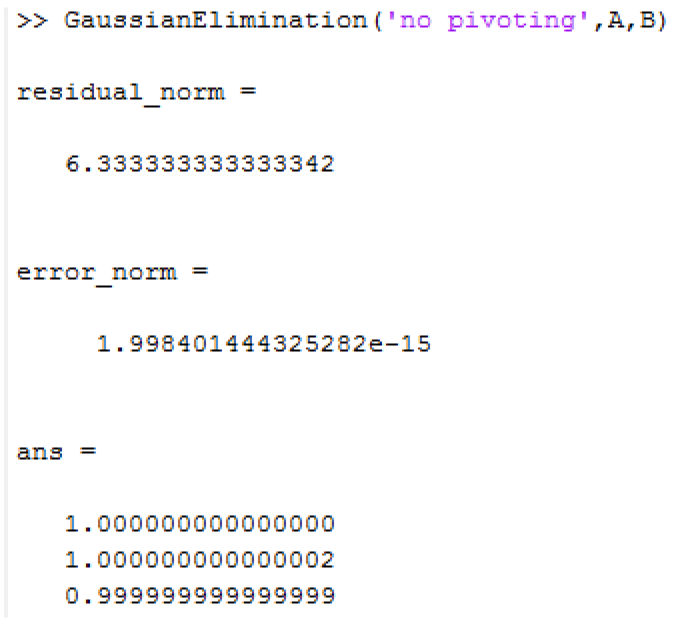
a) The implementation is included in the folder “computer problems” in “2.11”.

b) The following matrices were chosen for this problem:

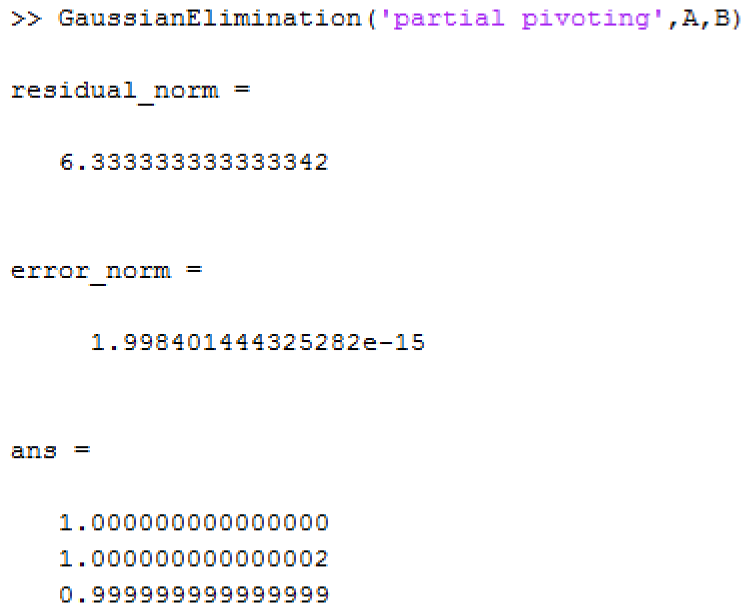
$$A = \begin{bmatrix} 9 & 3 & 4 \\ 3 & 6 & 8 \\ 2 & 5 & 6 \end{bmatrix}$$

$$B = \begin{bmatrix} 16 \\ 17 \\ 13 \end{bmatrix}$$

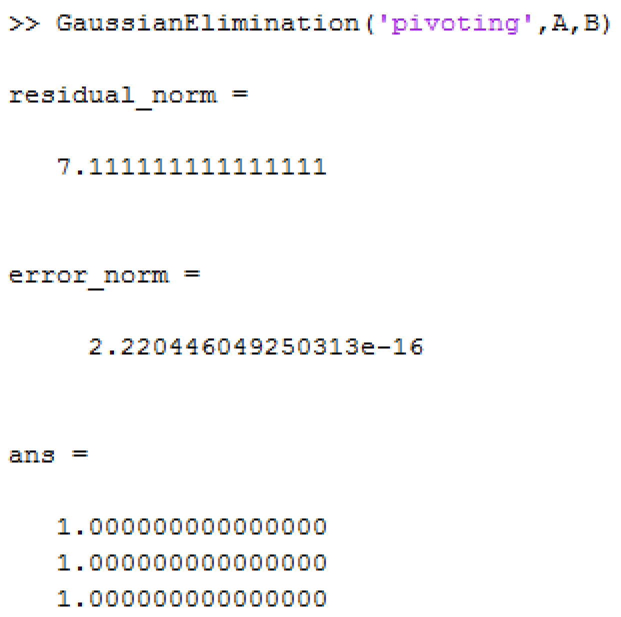
This would give a solution of ones. The following screenshots show the result obtained, along with residual and error norms for all three algorithms.



```
>> GaussianElimination('no pivoting',A,B)

residual_norm =

   6.333333333333342


error_norm =

     1.998401444325282e-15


ans =

   1.000000000000000
   1.000000000000002
   0.999999999999999
```



```
>> GaussianElimination('partial pivoting',A,B)

residual_norm =

   6.333333333333342


error_norm =

     1.998401444325282e-15


ans =

   1.000000000000000
   1.000000000000002
   0.999999999999999
```



```
>> GaussianElimination('pivoting',A,B)

residual_norm =

   7.111111111111111


error_norm =

     2.220446049250313e-16


ans =

   1.000000000000000
   1.000000000000000
   1.000000000000000
```

c) The following matrices give a result where partial pivoting is significantly inaccurate than complete pivoting:

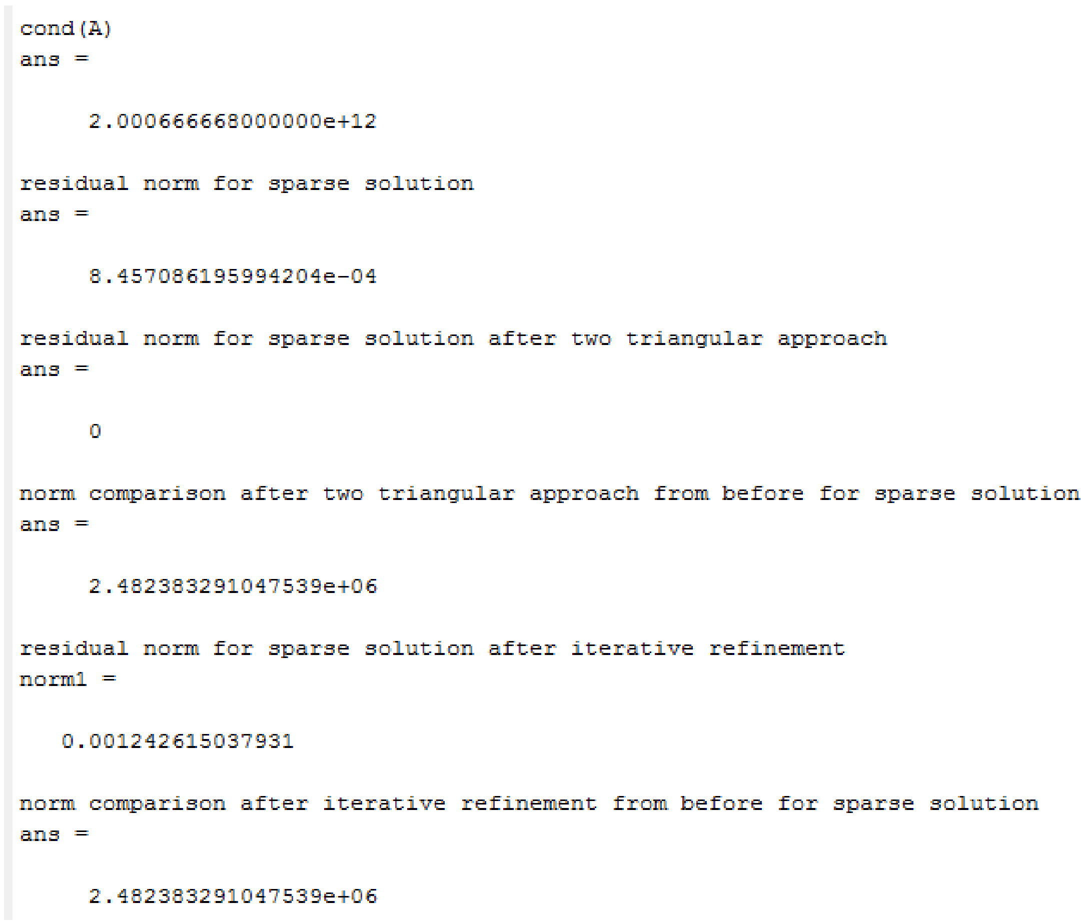
$$A = \begin{pmatrix} \frac{1}{3} & -\frac{2}{3} & \frac{1}{3} \\ 0 & \frac{2}{3} & \frac{2}{3} \\ -\frac{2}{3} & \frac{4}{3} & \frac{2}{3} \end{pmatrix} \qquad B = \begin{pmatrix} 1 \\ 4 \\ 2 \end{pmatrix}$$

2.17

a) Figure below shows the results obtained in terms of difference of solution obtained, along with the time requirement for each method.

```
norm(Xsparse - Xdense)
ans =

    0.0142

time taken by sparse solution
t1 =

    0.0650

time taken by dense solution
t2 =

    0.0230
```

b) The following snapshots were taken from MATLAB after execution of the program, which included solving the system using band solver, two triangular solves and with iterative refinement. Residual and error norms were calculated for each methodology.



```
cond(A)
ans =

     2.000666668000000e+12

residual norm for sparse solution
ans =

     8.457086195994204e-04

residual norm for sparse solution after two triangular approach
ans =

     0

norm comparison after two triangular approach from before for sparse solution
ans =

     2.482383291047539e+06

residual norm for sparse solution after iterative refinement
norm1 =

   0.001242615037931

norm comparison after iterative refinement from before for sparse solution
ans =

     2.482383291047539e+06
```

As can be seen, the accuracy improves after iterative refinement.