Numerical Analysis 2020/21
Project 2

Salma Salem 5553

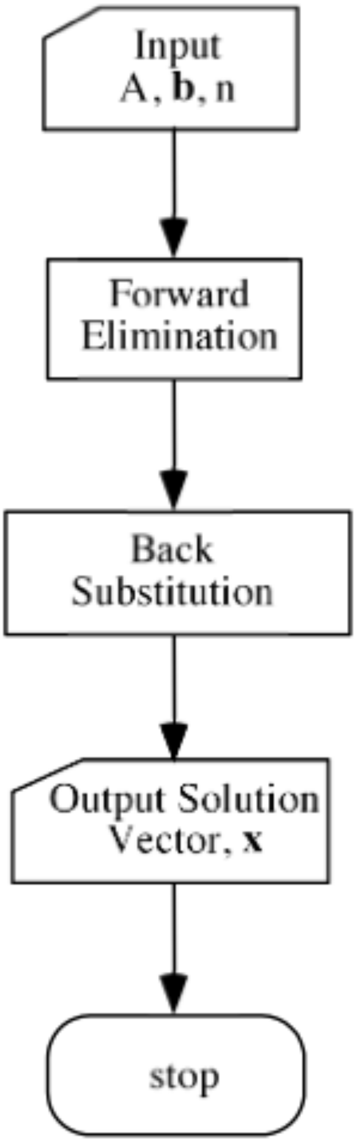
Malak Kassem 5979

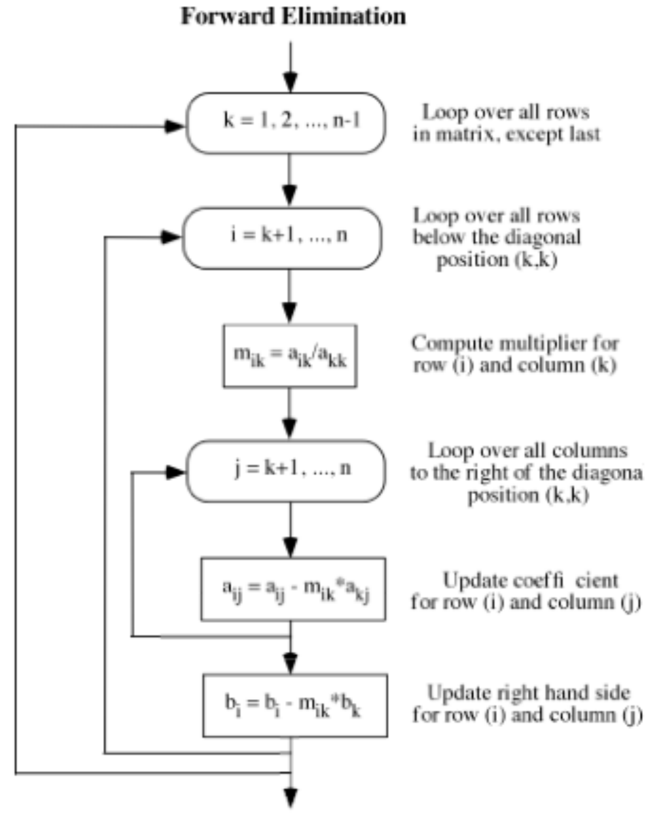
Maryam Yasser 5787

Gauss Elimination:

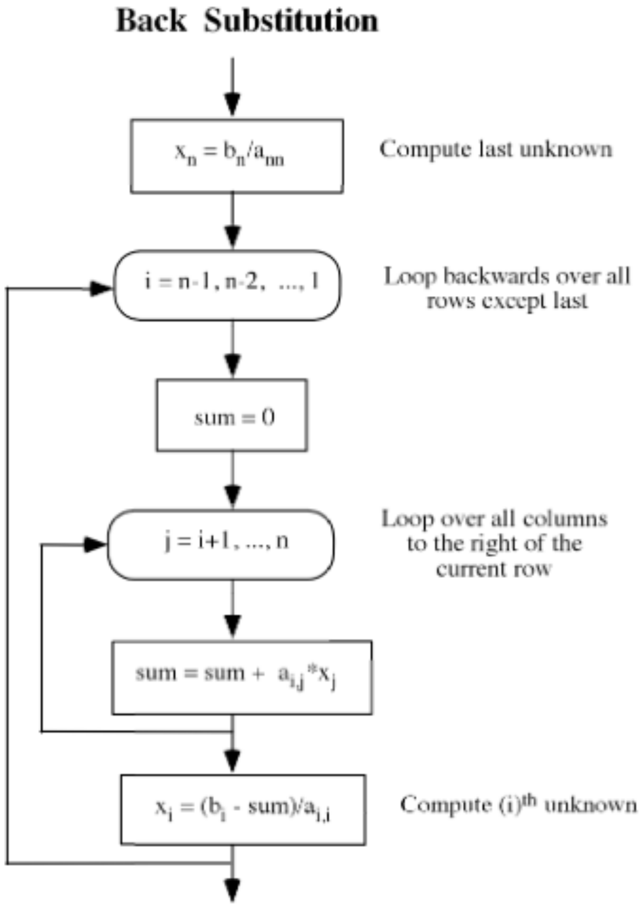
Gaussian elimination, also known as **row reduction**, is an algorithm in linear algebra for solving a system of linear equations. It is usually understood as a sequence of operations performed on the corresponding matrix of coefficients. This method can also be used to find the rank of a matrix, to calculate the determinant of a matrix, and to calculate the inverse of an invertible square matrix.

Flowchart:





Flowchart for forward elimination.



Flowchart for back substitution.

Our Code:

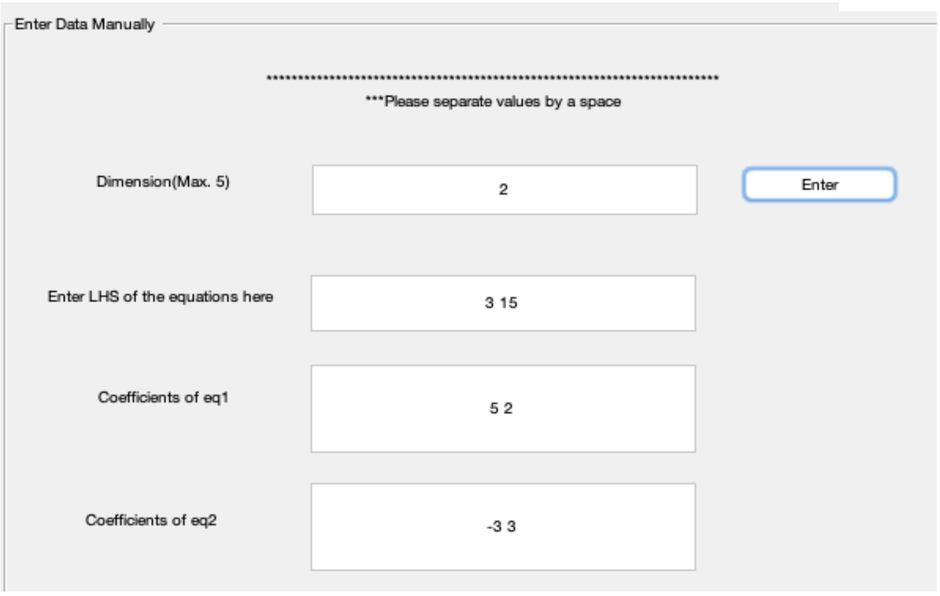
```
function x = Gauss_Elimination(array)

    A= array ;
    [m,n] = size(A);
    detA= array(:,1:n-1);
    if det(detA)==0
        msgbox('gauss elimination cannot be performed', 'Error','error');
        return;
    end
    for j = 1:m-1
        for i= j+1:m
            A(i,:)=A(i,:)- A(j,:)*(A(i,j)/A(j,j));
        end
    end
    for i=1:m
        if A(i,i)==0
            msgbox('The system has no solutions');
            return;
        end
    end
    if A(m,n-1) == 0 && A(m,n)==0
        msgbox('The system has an infinite number of solutions');
        return;
    end
    x = zeros(1,m);
    for s =m:-1:1
        c=0;
        for k=2:m
            c=c+A(s,k)*x(k);
        end
        x(s) = (A(s,n)-c)/A(s,s);
    end
    x=x';
    writetable(array2table(x),'outputGauss.txt')
```

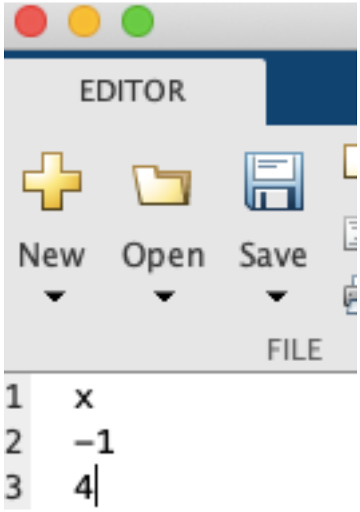
Hint: -ve numbers be written as -1 not - 1

Examples:

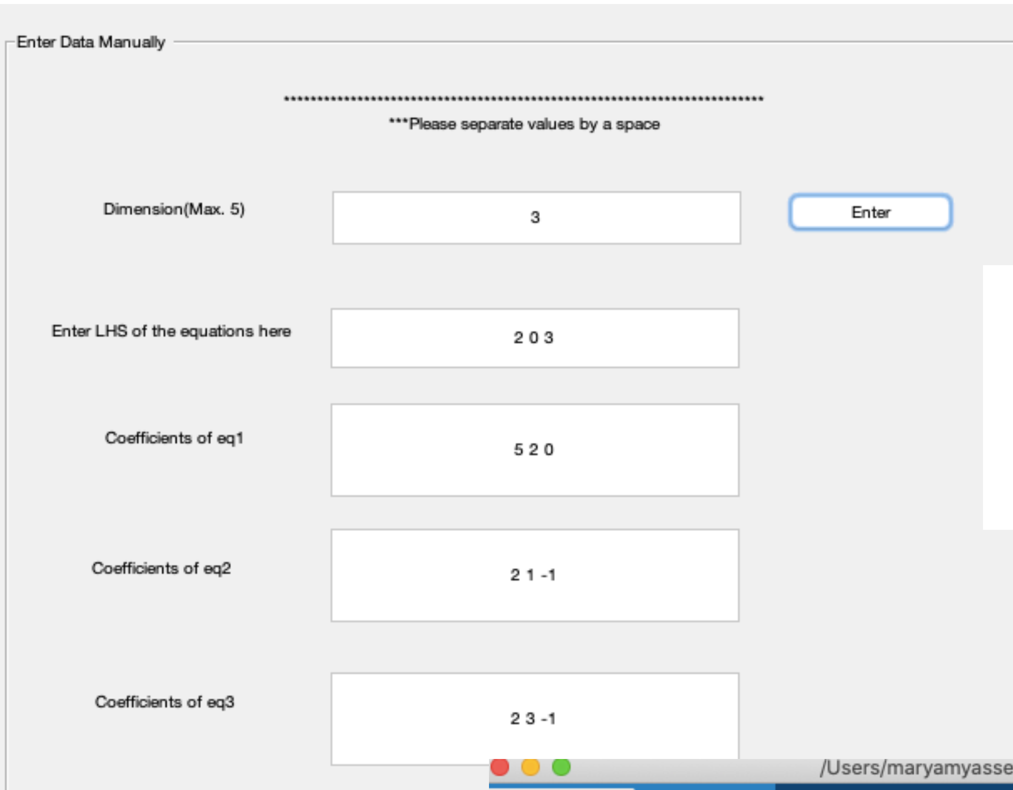
1)

$$
\begin{aligned}
5x + 2y &= 3 \\
-3x + 3y &= 15
\end{aligned}
$$

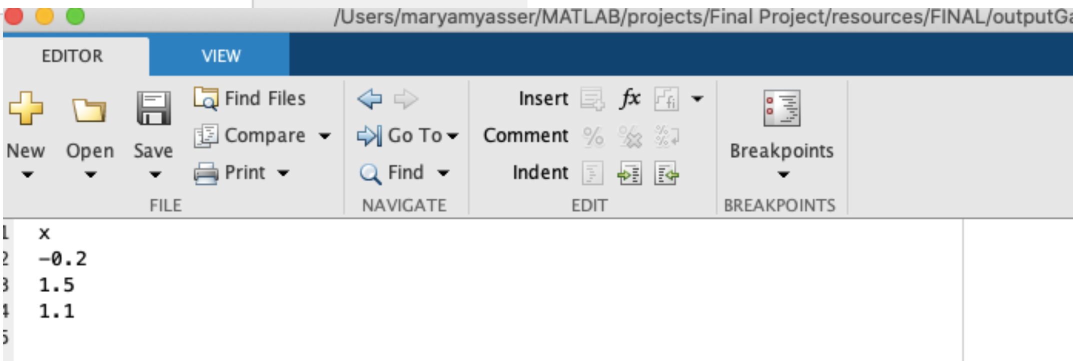




2)



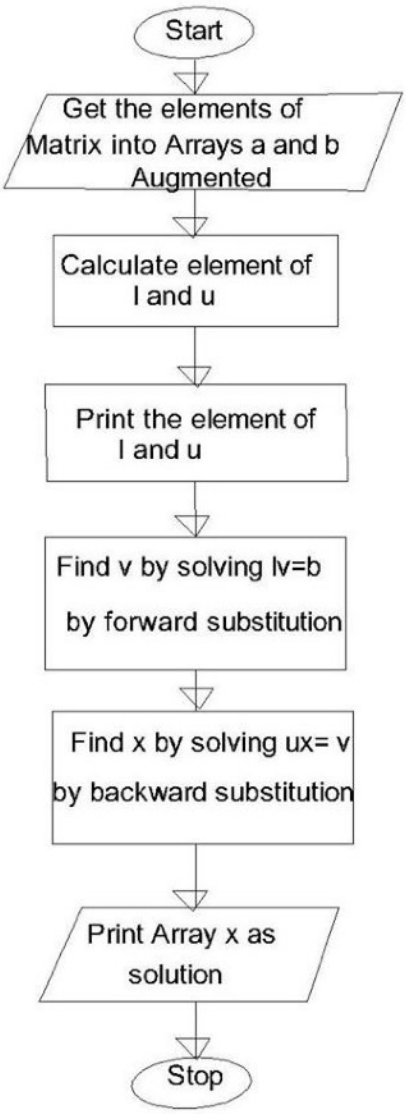
$$
\begin{aligned}
5x + 2y \quad\;\; &= 2 \\
2x + y - z &= 0 \\
2x + 3y - z &= 3
\end{aligned}
$$



LU DECOMPOSITION:

In numerical analysis and linear algebra, **lower–upper** (**LU**) **decomposition** or **factorization** factors a matrix as the product of a lower triangular matrix and an upper triangular matrix. The product sometimes includes a permutation matrix as well. LU decomposition can be viewed as the matrix form of Gaussian elimination. Computers usually solve square systems of linear equations using LU decomposition, and it is also a key step when inverting a matrix or computing the determinant of a matrix.

Flowchart

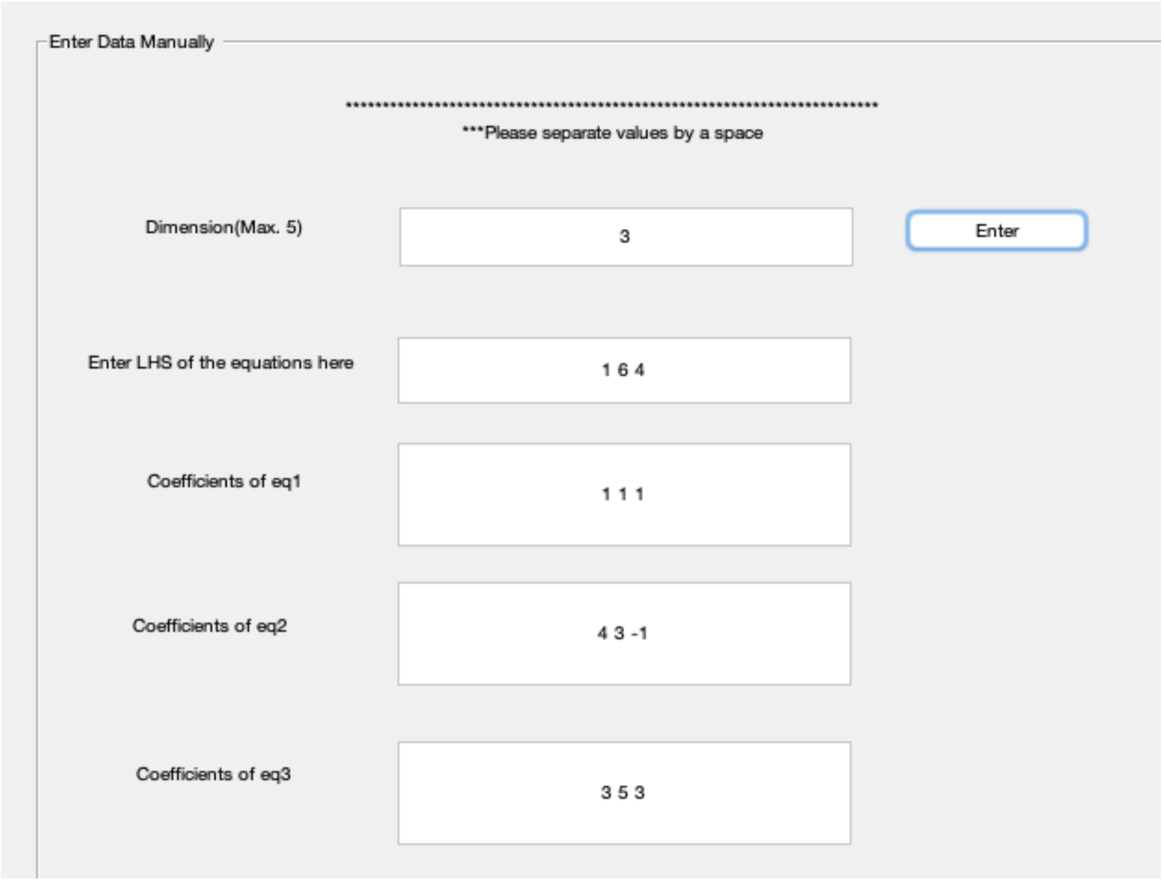


Our Code:

```
function x= LUdec(array)
    A= array;
    [ccc,n] = size(A);
    A= array(:,1:n-1);
    B = array(:,n);
    [m n]=size(A);
    if (m ~= n )
        disp ( 'LR2 error: Matrix must be square' );
        return;
    end;
  L=zeros(m,m);
  U=zeros(m,m);
  for i=1:m
      % Finding L
      for k=1:i-1
          L(i,k)=A(i,k);
          for j=1:k-1
            L(i,k)= L(i,k)-L(i,j)*U(j,k);
          end
          L(i,k) = L(i,k)/U(k,k);
      end
      % Finding U
      for k=i:m
          U(i,k) = A(i,k);
          for j=1:i-1
          U(i,k)= U(i,k)-L(i,j)*U(j,k);
          end
      end
  end
  for i=1:m
      L(i,i)=1;
  end
  y=zeros(m,1); % initiation for y
  y(1)=B(1)/L(1,1);
  for i=2:m
     %y(i)=B(i)-L(i,1)*y(1)-L(i,2)*y(2)-L(i,3)*y(3);
      y(i)=-L(i,1)*y(1);
      for k=2:i-1
           y(i)=y(i)-L(i,k)*y(k);
      end;
          y(i)=(B(i)+y(i))/L(i,i);
   end;
   % Now we use this y to solve Ux = y
   x=zeros(m,1);
   x(m)=y(m)/U(m,m);
   i=m-1;
   q=0;
   while  (i~= 0)
     x(i)=-U(i,m)*x(m);
      q=i+1;
         while (q~=m)
             x(i)=x(i)-U(i,q)*x(q);
             q=q+1;
         end;
       x(i)=(y(i)+x(i))/U(i,i);
       i=i-1;
   end
   output = x;
   writetable(array2table(output),'outputLU.txt');
```

Examples:

1)



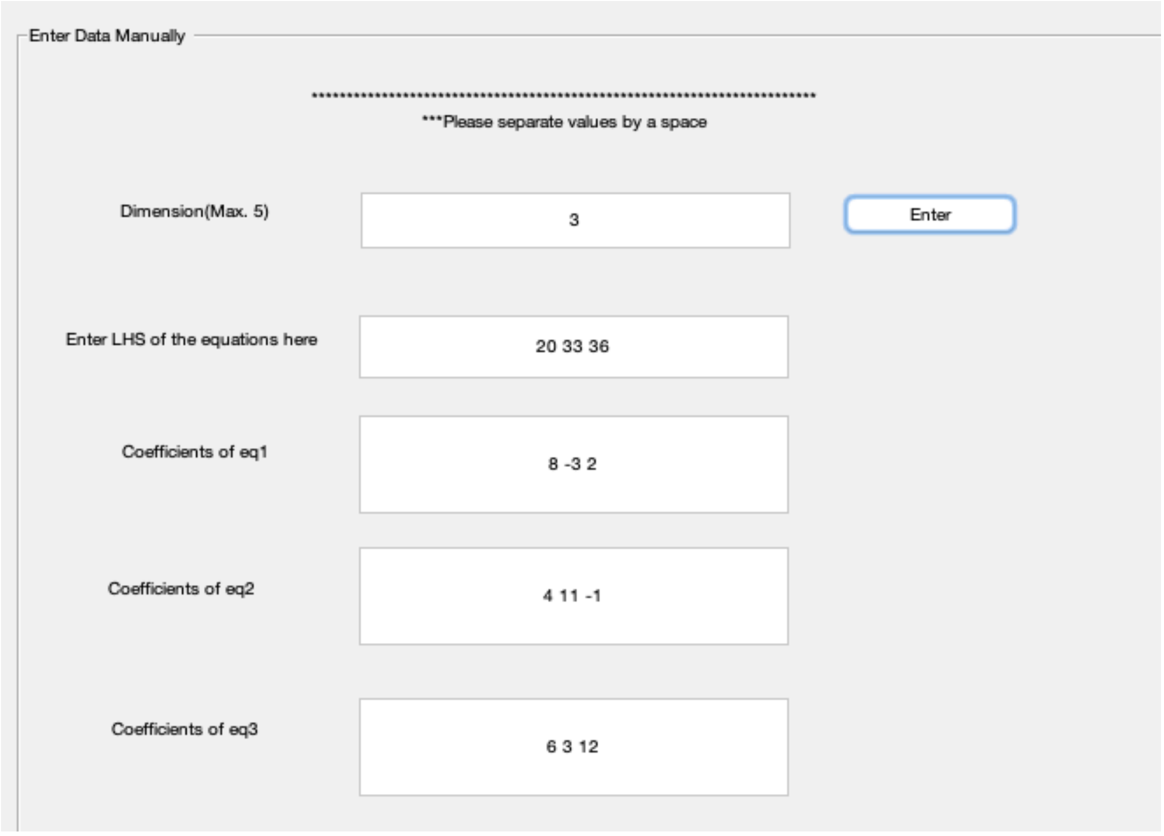

```
output
1
0.5
-0.5
```

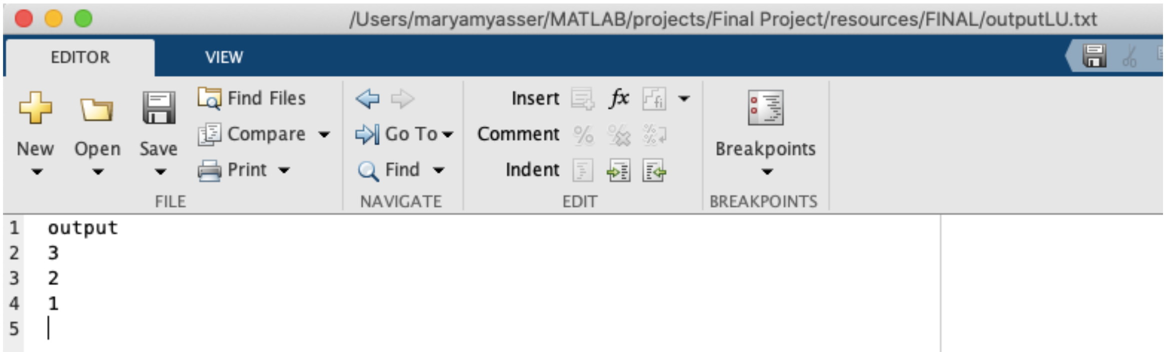
$$x_1 + x_2 + x_3 = 1$$

$$4x_1 + 3x_2 - x_3 = 6$$

$$3x_1 + 5x_2 + 3x_3 = 4$$

2)



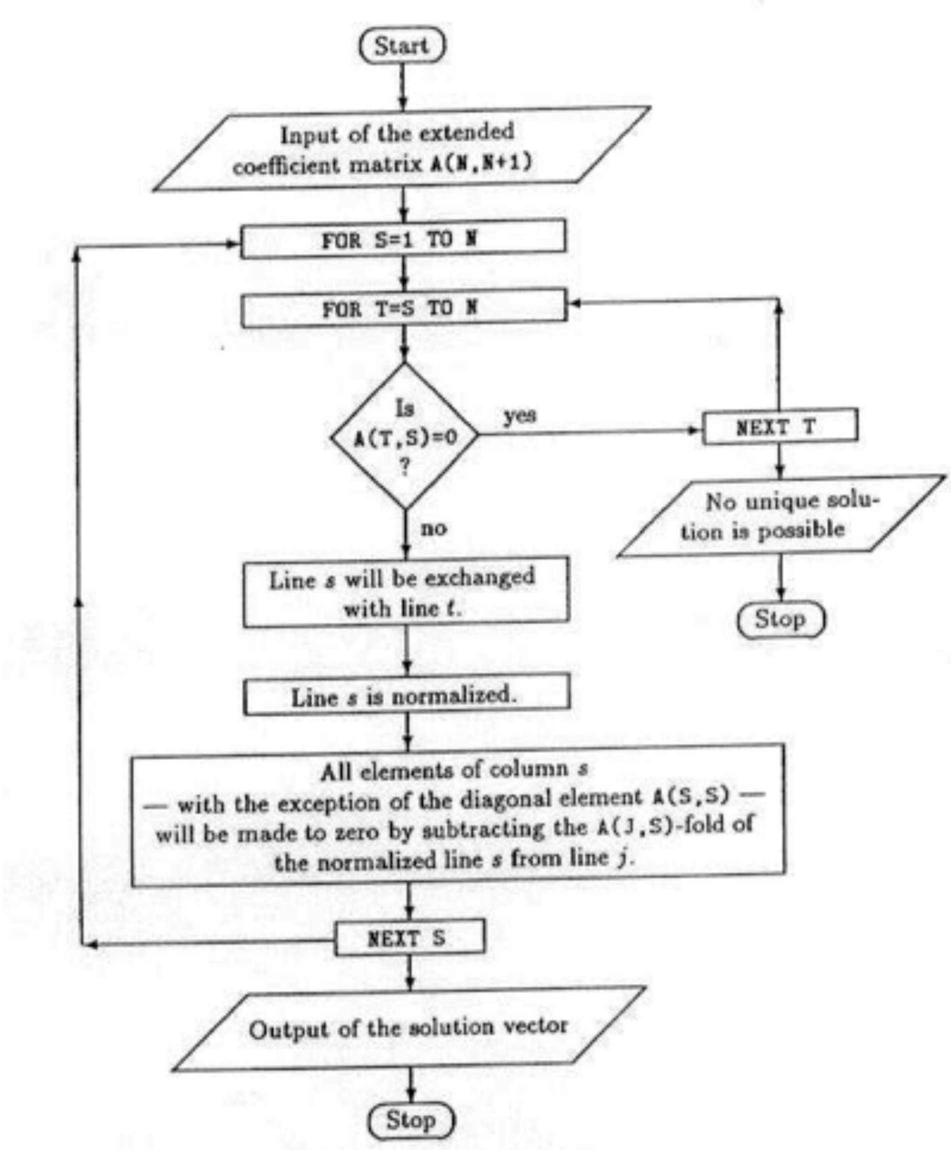


$$8x - 3y + 2z = 20$$
$$4x + 11y - z = 33$$
$$6x + 3y + 12z = 36$$

Gaussian Jordan

Gaussian Elimination helps to put a matrix in row echelon form, while Gauss-Jordan Elimination puts a matrix in reduced row echelon form. For small systems (or by hand), it is usually more convenient to use Gauss-Jordan elimination and explicitly solve for each variable represented in the matrix system. However, Gaussian elimination in itself is occasionally computationally more efficient for computers.

Flowchart

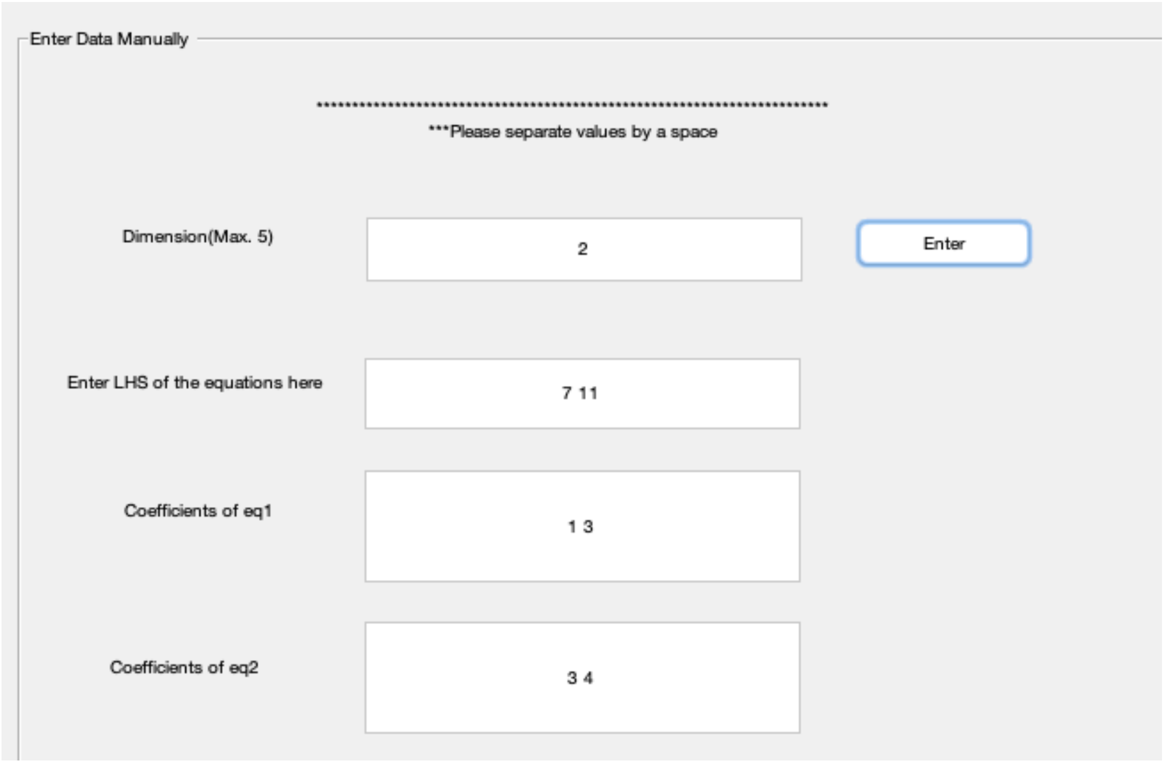


Our Code:

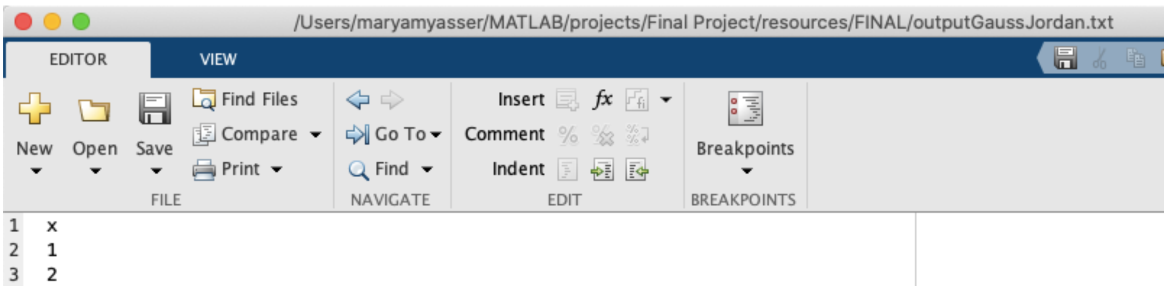
```
function x = Gauss_jordan(array)
    A= array ;
    [m,n] = size(A);
    detA= array(:,1:n-1);
    if det(detA)==0
        msgbox('gauss elimination cannot be performed', 'Error','error');
        return;
    end
    x = zeros(1,m);
    for j = 1:m-1
        for i= j+1:m
            A(i,:)=A(i,:) - A (j,:)*(A(i,j)/A(j,j));
        end
    end
    for j=m:-1:2
        for i=j-1:-1:1
             A(i,:)=A(i,:) - A (j,:)*(A(i,j)/A(j,j));
        end
    end
    for i=1:m
        if A(i,i)==0
            msgbox('The system has no solutions');
            return;
        end
    end
    if A(m,n-1) == 0 && A(m,n)==0
        msgbox('The system has an infinite number of solutions');
        return;
    end
    for s=1:m
        A(s,:)=A(s,:)/A(s,s);
        x(s) = A(s,n);
    end
    x=x';
    writetable(array2table(x),'outputGaussJordan.txt');
```

Examples:

1)

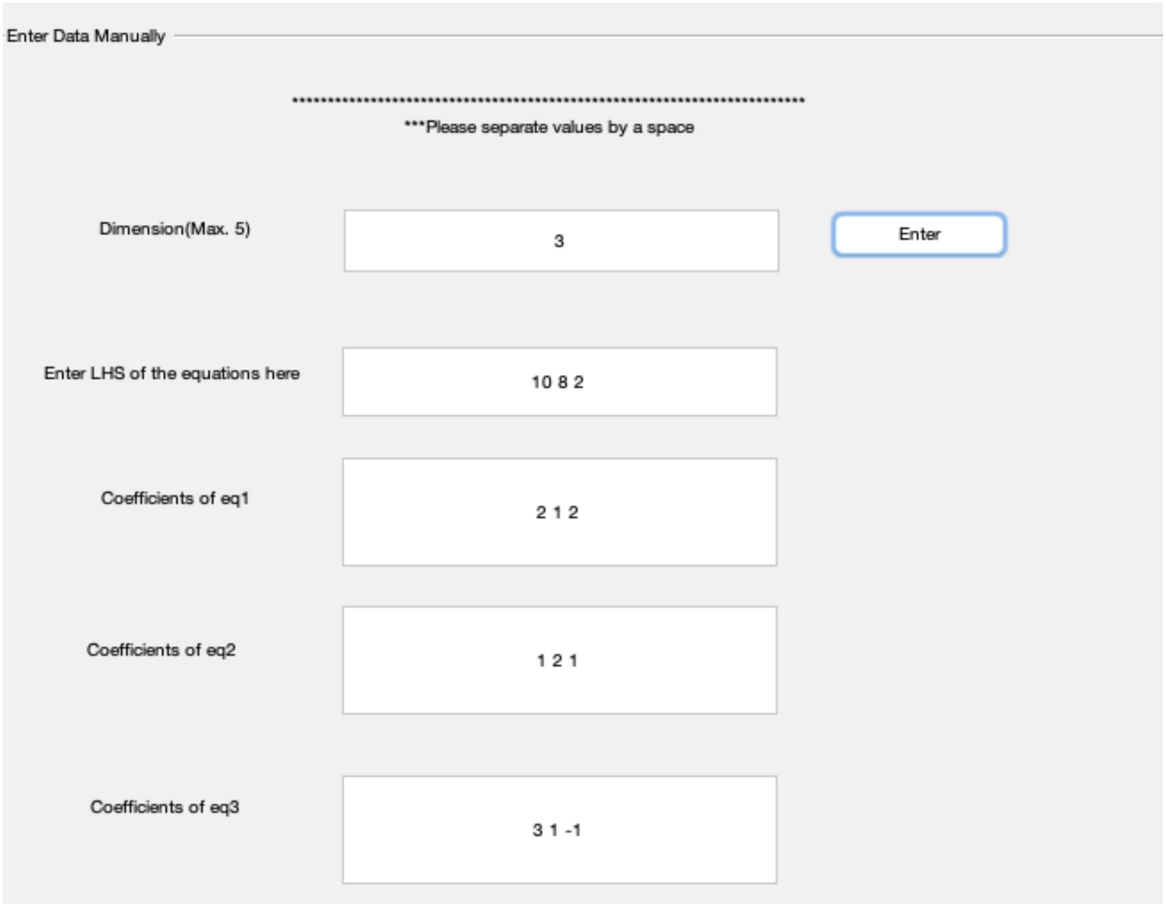


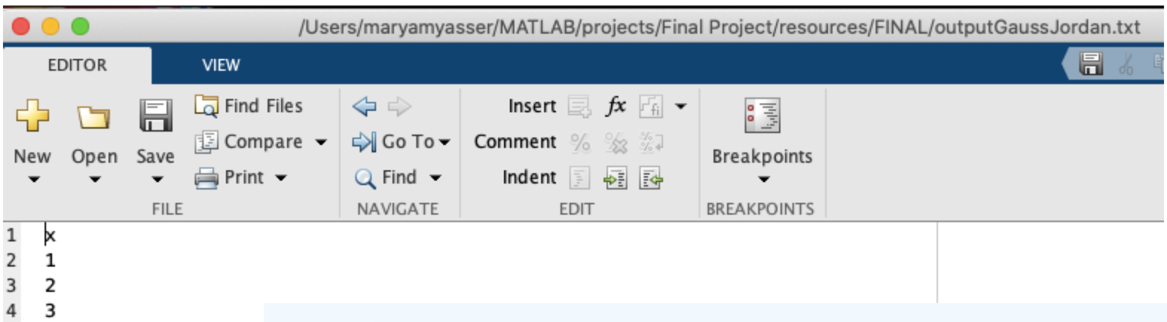
$$x + 3y = 7$$
$$3x + 4y = 11$$



2)

$$
\begin{aligned}
2x + y + 2z &= 10 \\
x + 2y + z &= 8 \\
3x + y - z &= 2
\end{aligned}
$$





Clearly, the solution reads $x = 1$, $y = 2$, and $z = 3$.

3)

LU -jordan -elimination

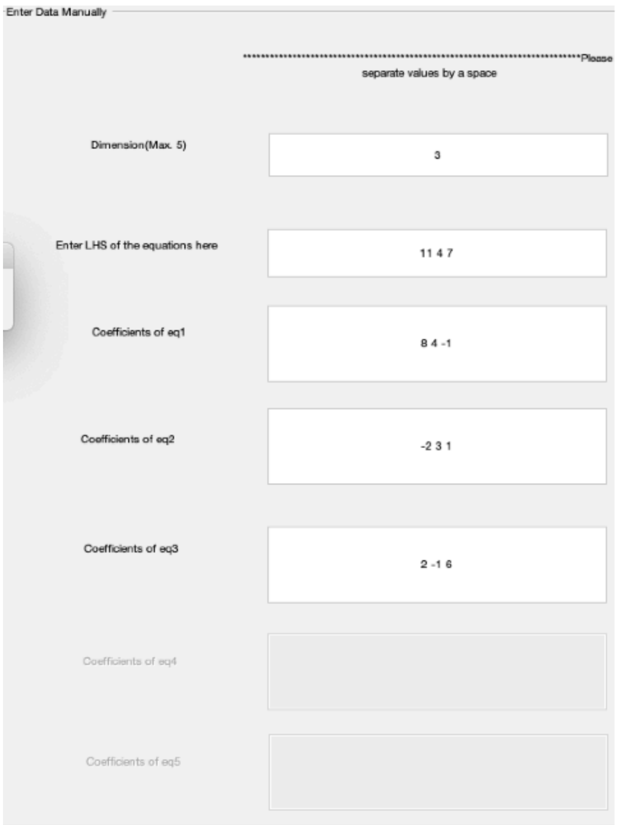
$8*x+4*y-1*z=11$

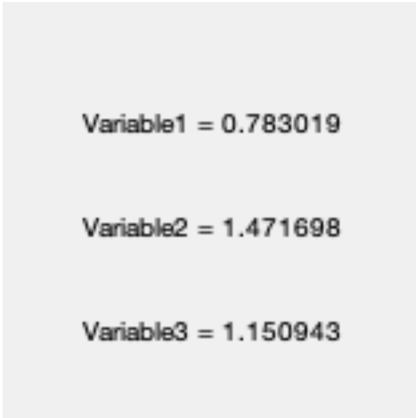
$-2*x+3*y+1*z=4$

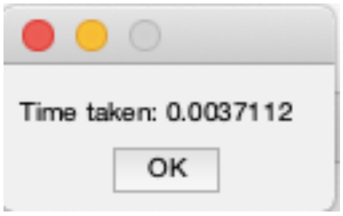
$2*x-1*y+6*z=7$

sol

$x=0.783, y=1.4717 and z=1.1509$







3)

jordan -elimination

x+y+1m=2

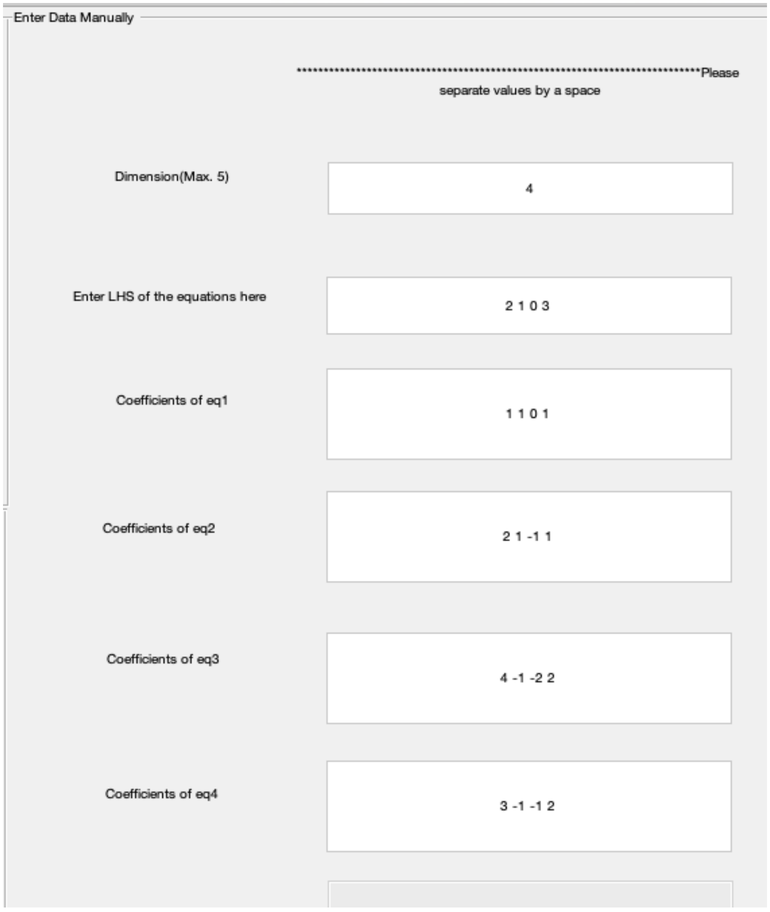
2x+1y-1z+1m=1

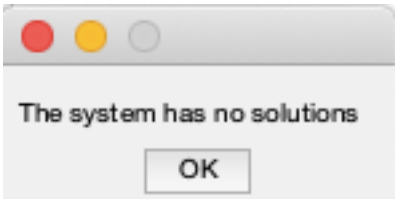
4x-1y-2z+2m=0

3x-1y-1z+2m=3

sol

no solution

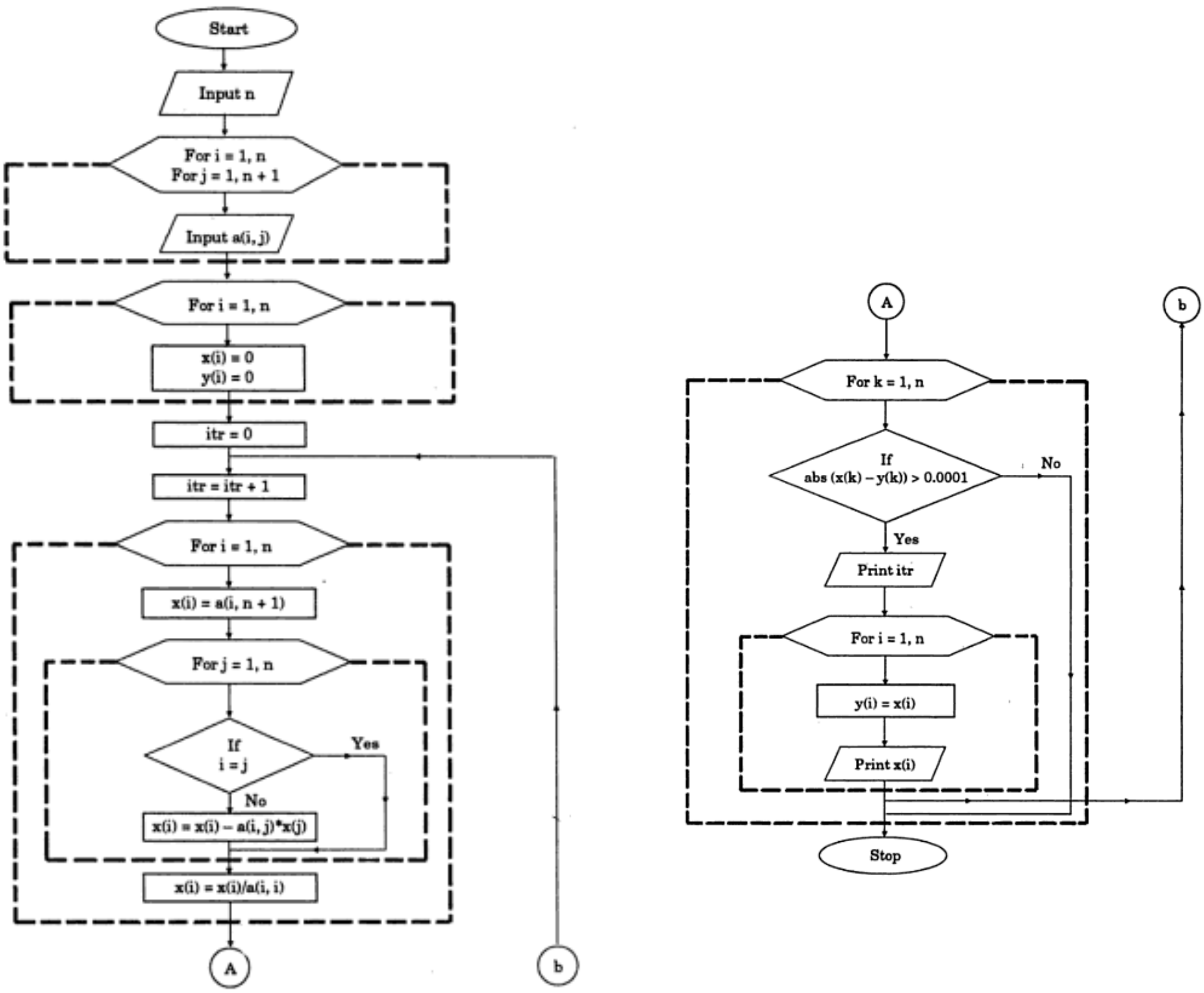




Gauss Seidel

Gauss–Seidel method, also known as the **Liebmann method** or the **method of successive displacement**, is an iterative method used to solve a system of linear equations. It is named after the German mathematicians Carl Friedrich Gauss and Philipp Ludwig von Seidel, and is similar to the Jacobi method. Though it can be applied to any matrix with non-zero elements on the diagonals, convergence is only guaranteed if the matrix is either strictly diagonally dominant,[1] or symmetric and positive definite.

Flowchart



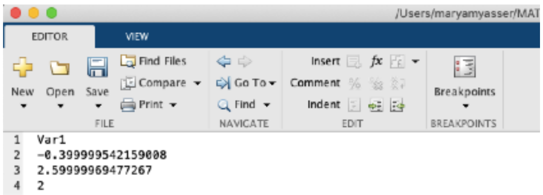
Our Code:

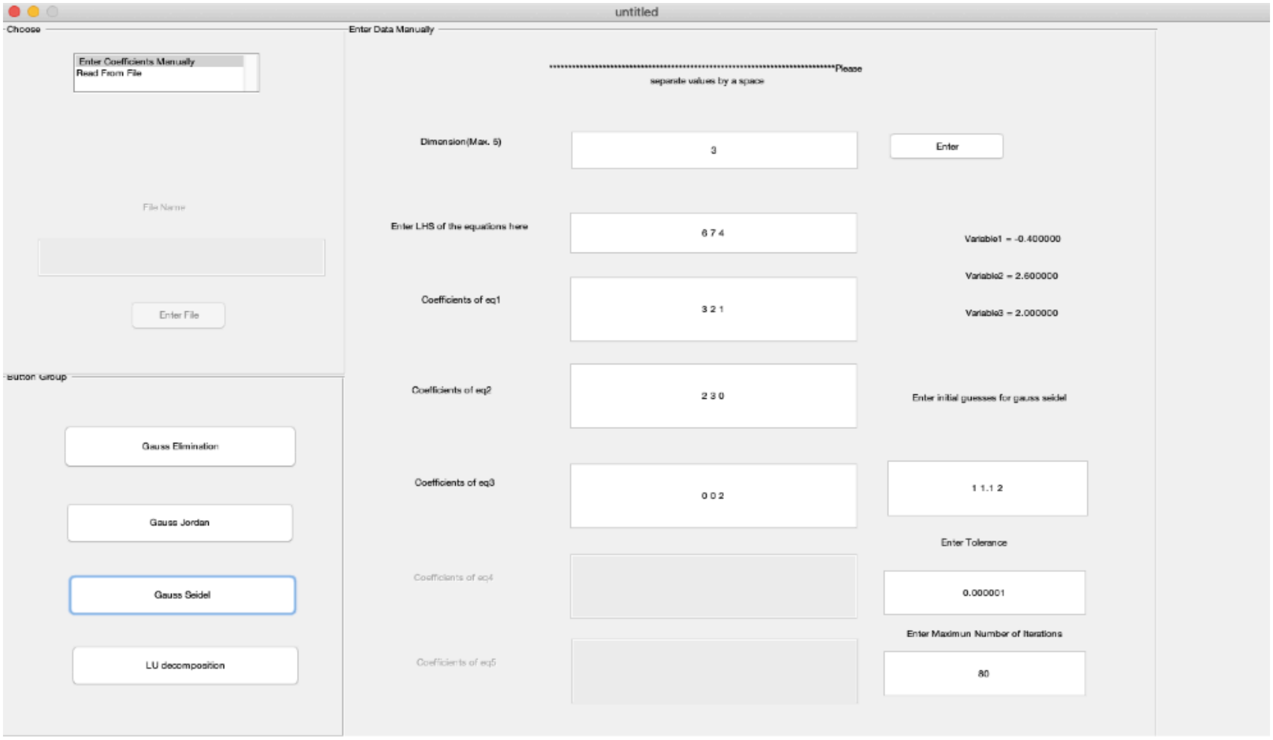
```
function x1 = Gauss_Seidel(array)
 A = array;
 n = length(A)-1;
 x1 = zeros(n);
 tol = 0.00001;
 m = 50;
 k = 1;
 while  k <= m
    err = 0;
    for i = 1 : n
       s = 0;
       for j = 1 : n
          s = s-A(i,j)*x1(j);
       end
       s = (s+A(i,n+1))/A(i,i);
       if abs(s) > err
         err  = abs(s);
       end
       x1(i) = x1(i) + s;
    end

    if err <= tol
      break;
    else
      k = k+1;
    end
 end
 output = x1(:,1);
 writetable(array2table(output),'outputGaussSeidel.txt');
```

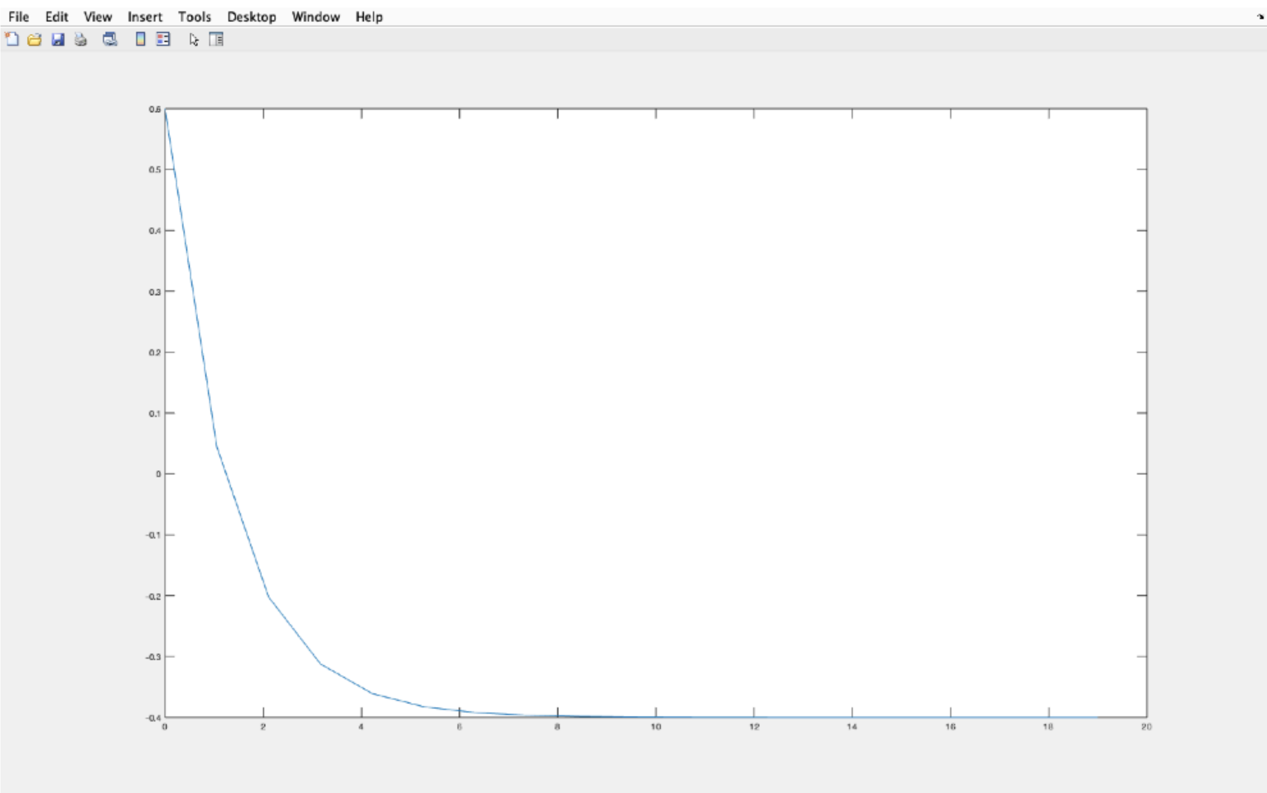
Examples:

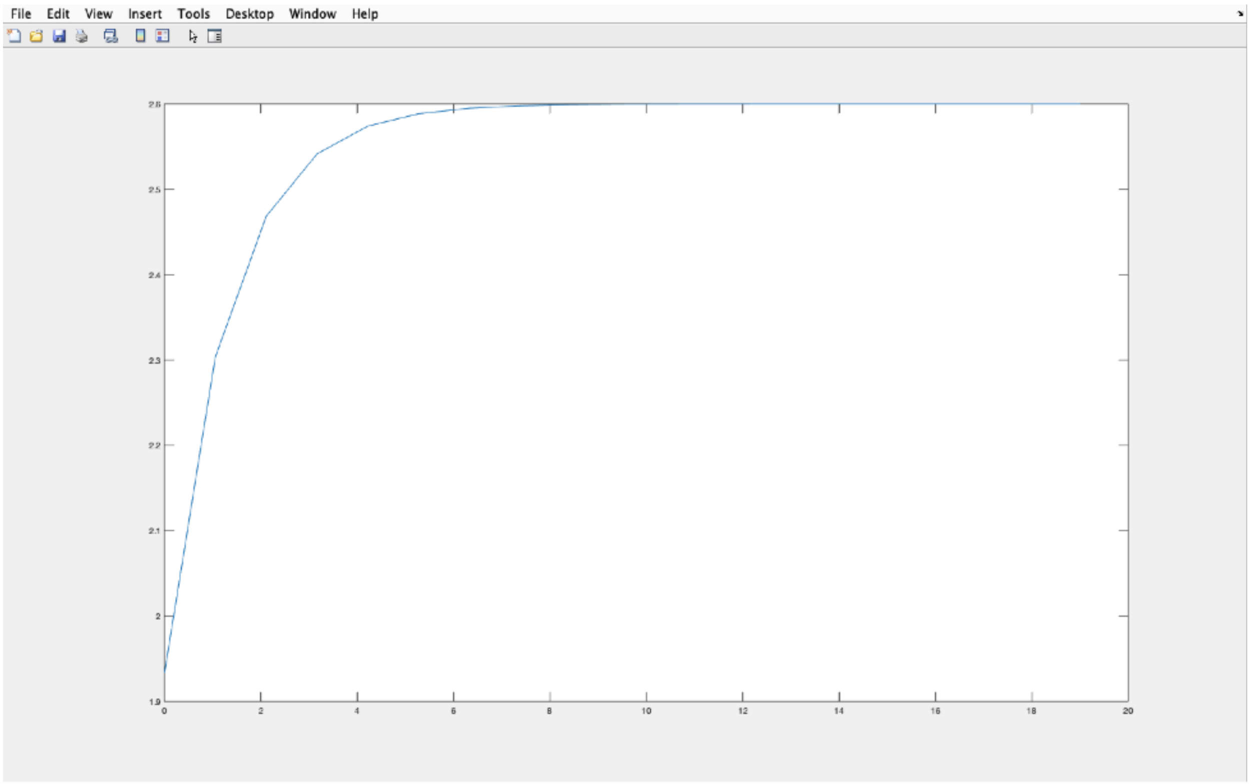
1)

```
3*a +2*b + c - 6
2*a +3*b - 7
2*c - 4
1 1.1 2
```








File Edit View Insert Tools Desktop Window Help

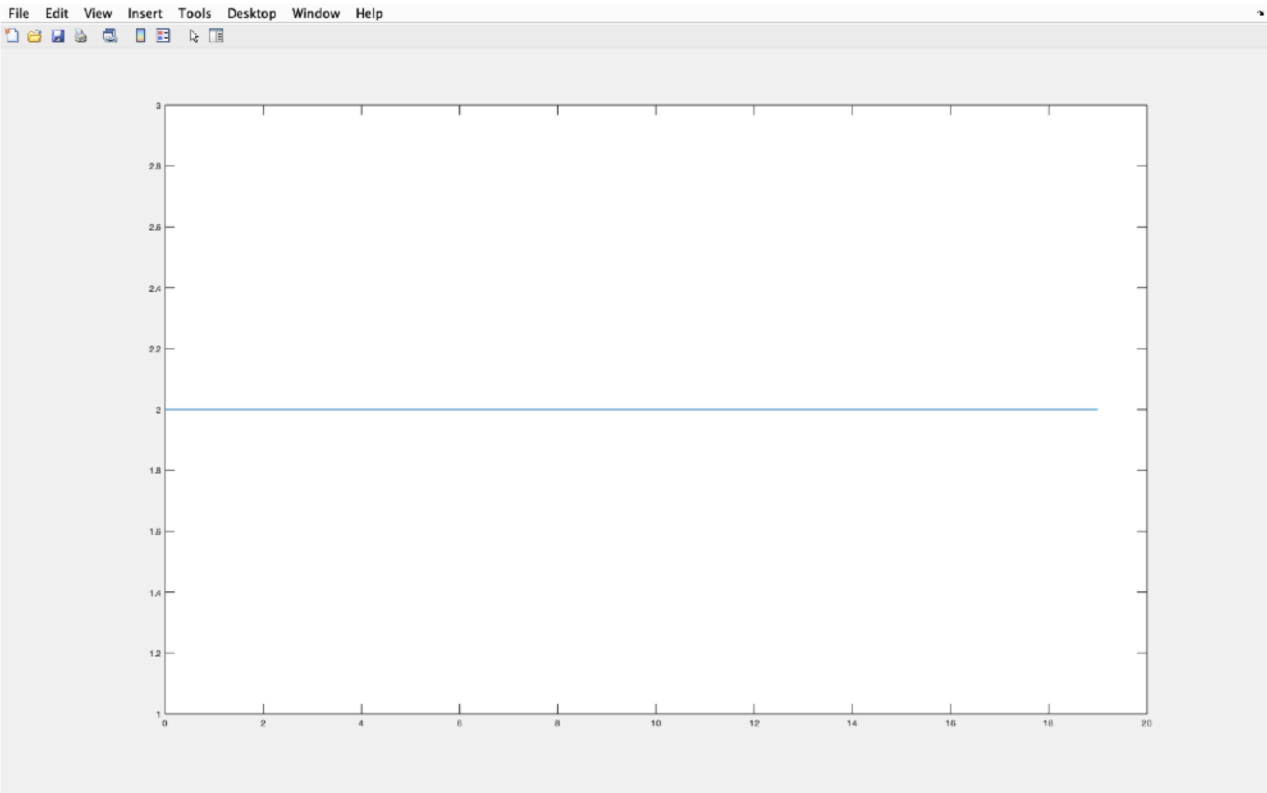

File Edit View Insert Tools Desktop Window Help

Figure 5

File Edit View Insert Tools Desktop Window Help

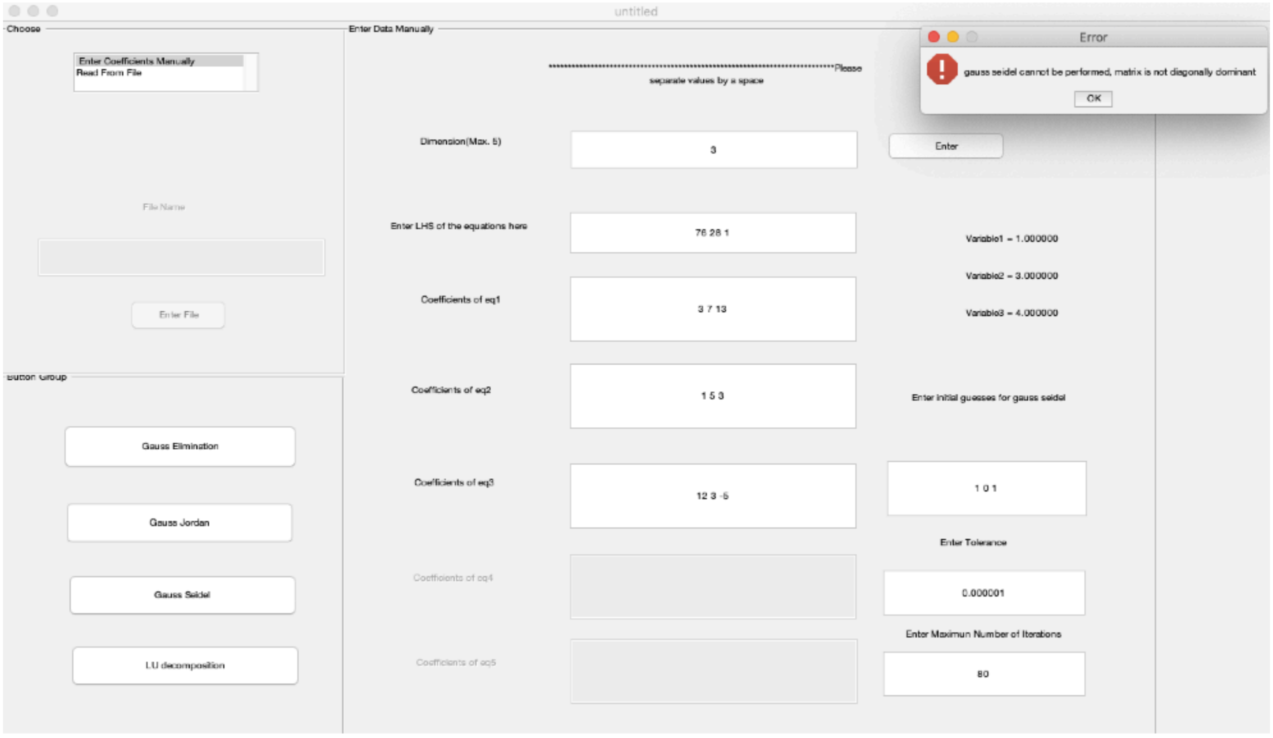
	1	2	3	4	5	6
1	0.6000	0.4000	1.9333	0.8333	2	0.8333
2	0.0444	0.5556	2.3037	0.5556	2	0.5556
3	-0.2025	0.2469	2.4683	0.2469	2	0.2469
4	-0.3122	0.1097	2.5415	0.1097	2	0.1097
5	-0.3610	0.0488	2.5740	0.0488	2	0.0488
6	-0.3827	0.0217	2.5884	0.0217	2	0.0217
7	-0.3923	0.0096	2.5949	0.0096	2	0.0096
8	-0.3966	0.0043	2.5977	0.0043	2	0.0043
9	-0.3985	0.0019	2.5990	0.0019	2	0.0019
10	-0.3993	8.4580e-04	2.5995	8.4580e-04	2	8.4580e-04
11	-0.3997	3.7591e-04	2.5998	3.7591e-04	2	3.7591e-04
12	-0.3999	1.6707e-04	2.5999	1.6707e-04	2	1.6707e-04
13	-0.3999	7.4254e-05	2.6000	7.4254e-05	2	7.4254e-05
14	-0.4000	3.3002e-05	2.6000	3.3002e-05	2	3.3002e-05
15	-0.4000	1.4667e-05	2.6000	1.4667e-05	2	1.4667e-05
16	-0.4000	6.5189e-06	2.6000	6.5189e-06	2	6.5189e-06

Time taken: 0.0016395 Number of iterations: 16

OK

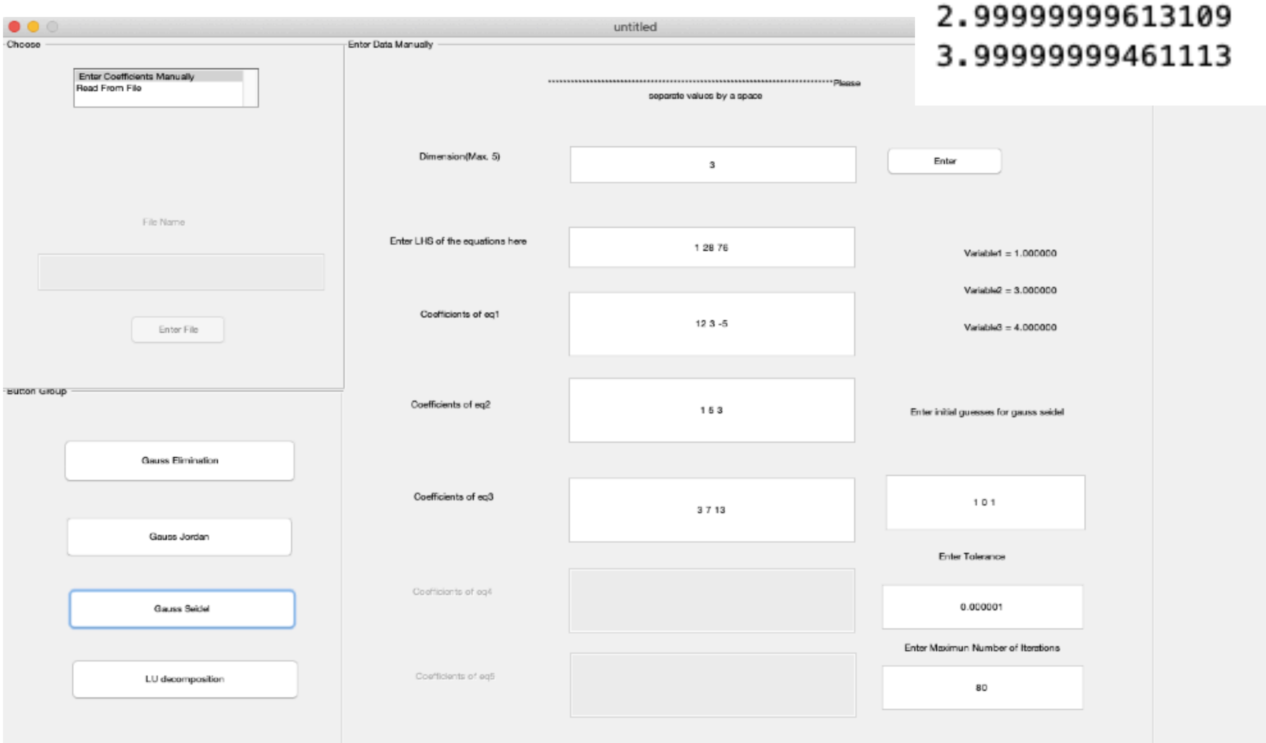
2)

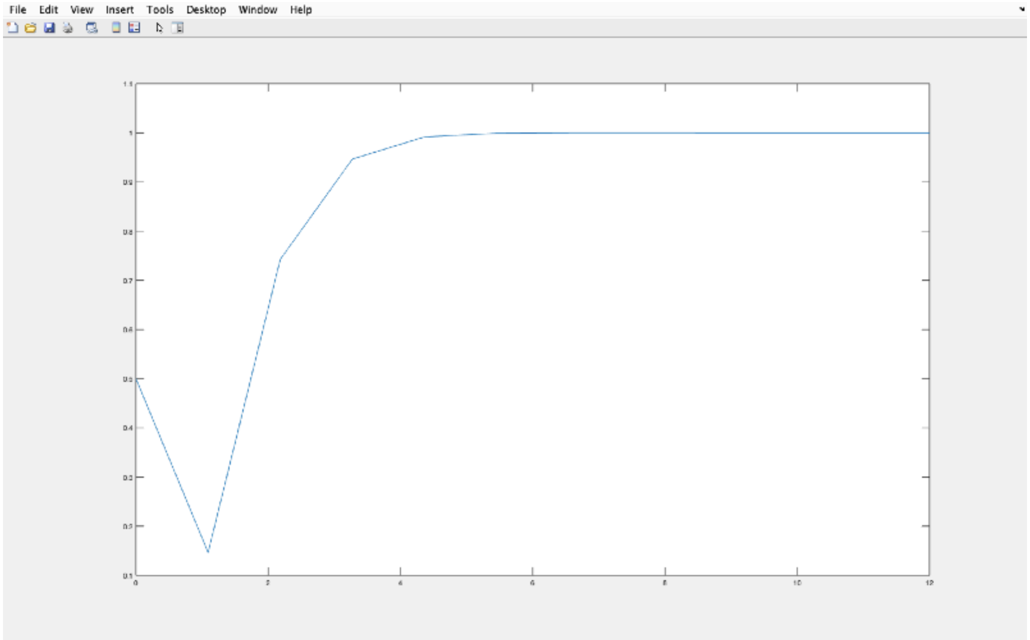
$$
\begin{aligned}
3x_1 + 7x_2 + 13x_3 &= 76 \\
x_1 + 5x_2 + 3x_3 &= 28 \\
12x_1 + 3x_2 - 5x_3 &= 1
\end{aligned}
$$

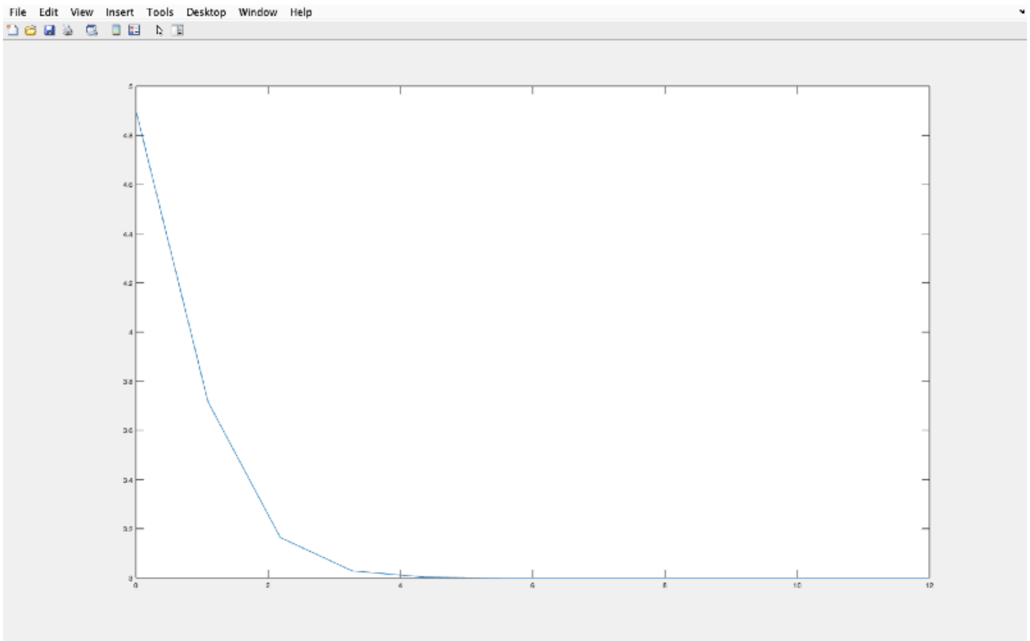


BUT AFTER CHANGING THE ORDER OF THE EQU. :

1.00000003237925
2.99999999613109
3.99999999461113




File Edit View Insert Tools Desktop Window Help


File Edit View Insert Tools Desktop Window Help

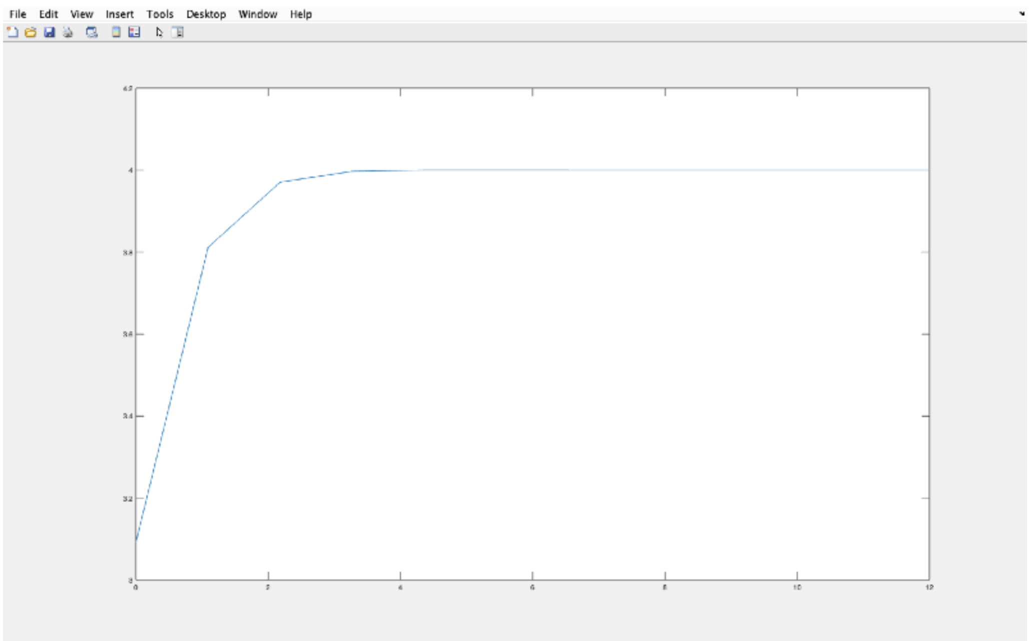

File Edit View Insert Tools Desktop Window Help

Figure 4

	1	2	3	4	5	6
1	0.5000	0.5000	4.9000	4.9000	3.0923	4.9000
2	0.1468	0.3532	3.7153	1.1847	3.8118	1.1847
3	0.7428	0.5960	3.1644	0.5960	3.9708	0.5960
4	0.9468	0.2040	3.0281	0.2040	3.9971	0.2040
5	0.9918	0.0450	3.0034	0.0450	4.0001	0.0450
6	0.9992	0.0074	3.0001	0.0074	4.0001	0.0074
7	1.0000	8.3096e-04	2.9999	8.3096e-04	4.0000	8.3096e-04
8	1.0000	1.0388e-05	3.0000	5.1485e-05	4.0000	5.1485e-05
9	1.0000	2.5421e-05	3.0000	2.5421e-05	4.0000	2.5421e-05
10	1.0000	8.5401e-06	3.0000	8.5401e-06	4.0000	8.5401e-06

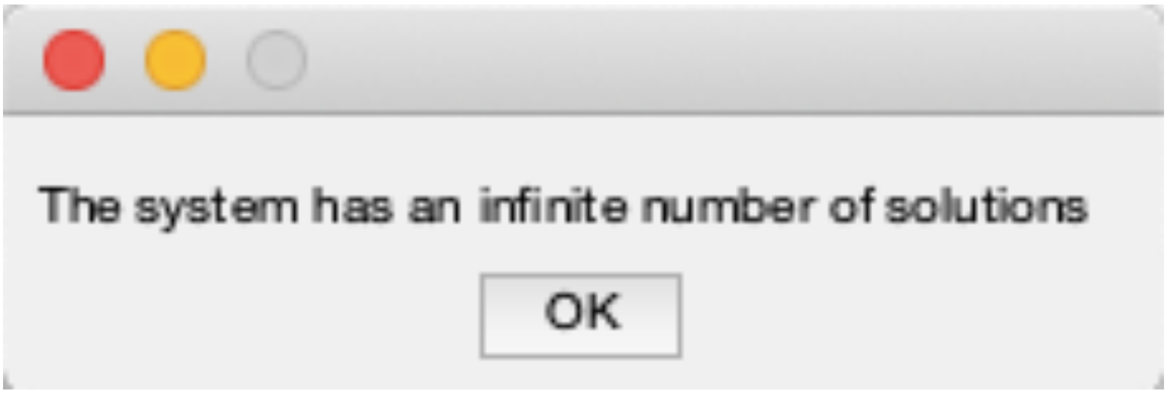
Time taken: 0.0064426 Number of iterations: 10

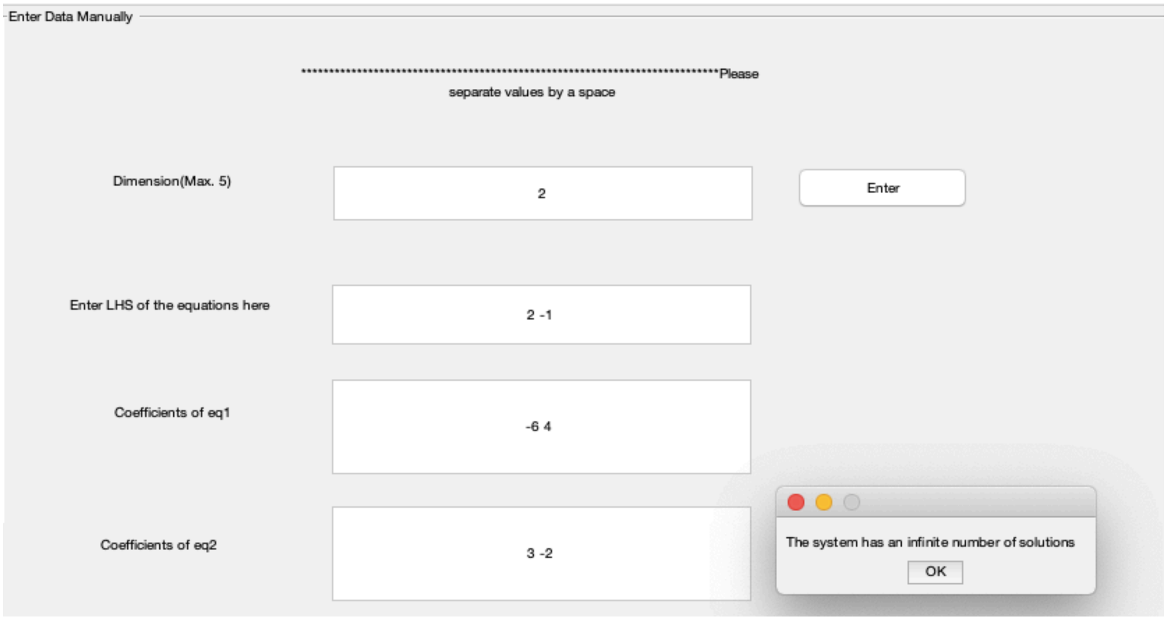
OK

Problematic functions:

1) The following system has infinite number of solution and therefore for all methods we get the same error.

```
-6*a + 4*b -2
3*a - 2*b +1
```





2) The following system no solution and therefore for all methods we get the same error.

```
-4*a + 10*b -6
2*a -5*b -3
```