Problem Statement

The aim of this assignment is to compare , analyze the behavior of the different numerical methods studied in class which are used to solve system of linear equations like (Gaussian-elimination, LU decomposition, Gaussian-Jordan and Gauss-Seidel) and discuss pitfalls of each one and its advantage .

Gauss

It is a direct method to solve a system of linear equations in a finite number of operations.

We represent this system in an augmented matrix like this.

$$\left[\begin{array}{ccc|c} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ a_{31} & a_{32} & a_{33} & b_3 \end{array}\right]$$

It mainly consists of two parts:

1-Forward Elimination: the target of this part is to convert the original matrix to triangular matrix like this:

$$\left[\begin{array}{ccc|c} a_{11} & a_{12} & a_{13} & b_1 \\ & a'_{22} & a'_{23} & b'_2 \\ & & a''_{33} & b''_3 \end{array}\right]$$

2-Backward substitution: to find values of (X1, X2……….,Xn) where n = number of equations.

$$\left. \begin{aligned} x_3 &= b''_3 / a''_{33} \\ x_2 &= (b'_2 - a'_{23}x_3) / a'_{22} \\ x_1 &= (b_1 - a_{12}x_2 - a_{13}x_3) / a_{11} \end{aligned} \right|$$

Gauss Procedure:

Inputs:

1- Coefficient = values of a’s.
2- Results = values of b’s.

outputs:

1- Solution table: contain matrices each one show how to eliminate element until you reach to the triangular matrix.
2- Final matrix: show the triangular matrix.
3- Solutions: to show the solutions of system.
4- Condition: Boolean equals to 1 if any error happen.

Algorithm

In this procedure we call two procedure : Forward Elimination, Backward substitution.

Pseudo Code:

```
[ solutionTable,finalMatrix,solutions,condition] = Gauss(coefficient,results){
        [solutionTable,help,condition]= forwardElimination(horzcat(coefficient,results));
        finalMatrix=help;
        if (condition==0)
           solutions=backSubistitution(help);
        else
          solutions=0;
}
```

Pitfalls of Gauss:

1-Total number of FLOPS – O(n3)
For large n, the operation count for Gauss Elimination is about n3. This means that as n doubled, the cost of solving the linear equations goes up by a factor of 8.

2-we may divide by zero and system can be solved.
And we can solve this problem using pivoting strategies.

Division by zero

It is possible that during both elimination and back-substitution phases a division by zero can occur.

For example:

$$2x_2 + 3x_3 = 8$$

$$4x_1 + 6x_2 + 7x_3 = -3 \qquad A = \begin{bmatrix} 0 & 2 & 3 \\ 4 & 6 & 7 \\ 2 & 1 & 6 \end{bmatrix}$$

$$2x_1 + x_2 + 6x_3 = 5$$

3-round off error :

$$\begin{bmatrix} 20 & 15 & 10 \\ -3 & -2.249 & 7 \\ 5 & 1 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 45 \\ 1.751 \\ 9 \end{bmatrix}$$

Solve it on a computer using 6 significant digits with chopping

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0.9625 \\ 1.05 \\ 0.999995 \end{bmatrix}$$

$$\begin{bmatrix} 20 & 15 & 10 \\ -3 & -2.249 & 7 \\ 5 & 1 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 45 \\ 1.751 \\ 9 \end{bmatrix}$$

Solve it on a computer using 5 significant digits with chopping

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0.625 \\ 1.5 \\ 0.99995 \end{bmatrix}$$

From this example, we note that for the same system there are different solutions
And this because of round off error.

We can avoid round off error using scaling strategy.

Pivoting strategies:

There Are Two Types Of Pivoting :

1-Partial Poivoting :

Interchange equations (rows) only.
Choose the largest coefficient in the column to become the pivot coefficient.

Exercise (Partial Pivoting)

$$\left[\begin{array}{ccccc|c} 1 & 2 & 5 & -1 & 7 & 1 \\ 0 & 0 & 33 & 2 & 15 & 2 \\ 0 & -4 & 5 & 6 & 1 & 3 \\ 0 & 6 & 25 & 99 & 2 & 4 \\ 0 & -8 & 5 & 0 & 10 & 5 \end{array}\right] \qquad \left[\begin{array}{ccccc|c} 1 & 2 & 5 & -1 & 7 & 1 \\ 0 & -8 & 5 & 0 & 10 & 5 \\ 0 & -4 & 5 & 6 & 1 & 3 \\ 0 & 6 & 25 & 99 & 2 & 4 \\ 0 & 0 & 33 & 2 & 15 & 2 \end{array}\right]$$

2-Complete pivoting:

Interchange both equations and unknown elements (pivots)
Choose the largest coefficient in the sub- matrix to become the pivot coefficient.

Exercise (Complete Pivoting)

$$\left[\begin{array}{ccccc|c} 1 & 2 & 99 & -1 & 7 & 1 \\ 0 & 0 & 33 & 2 & 15 & 2 \\ 0 & -4 & 5 & 6 & 1 & 3 \\ 0 & 6 & 25 & 88 & 2 & 4 \\ 0 & -8 & 5 & 0 & 10 & 5 \end{array}\right] \qquad \left[\begin{array}{ccccc|c} 1 & -1 & 99 & 2 & 7 & 1 \\ 0 & 88 & 25 & 6 & 2 & 4 \\ 0 & 6 & 5 & -4 & 1 & 3 \\ 0 & 2 & 33 & 0 & 15 & 2 \\ 0 & 0 & 5 & -8 & 10 & 5 \end{array}\right]$$

Scaling

Rescale all coefficients in a row to make the largest coefficient equal to 1.

Example:

$$2x_1 + 100{,}000x_2 = 100{,}000 \Rightarrow 0.00002x_1 + x_2 = 1$$

Sample Run

```
out1 =

    4.0000   -1.0000    2.0000    2.0000
         0    2.2500    2.5000    1.0000
    2.0000    1.0000    4.0000    1.0000
    4.0000   -1.0000    2.0000    2.0000
         0    2.2500    2.5000    1.0000
         0    1.5000    3.0000         0
    4.0000   -1.0000    2.0000    2.0000
         0    2.2500    2.5000    1.0000
         0         0    1.3333   -0.6667


out2 =

    4.0000   -1.0000    2.0000    2.0000
         0    2.2500    2.5000    1.0000
         0         0    1.3333   -0.6667


out3 =

    1.0000
    1.0000
   -0.5000


out4 =

     0
```

Gauss Jordan

Similar to the Gauss elimination except

1. Elimination is applied to all equations (excluding the pivot equation) instead of just the subsequent equations.
2. All rows are normalized by dividing them by their pivot elements.
3. No back substitution is required.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ a_{31} & a_{32} & a_{33} & b_3 \end{bmatrix} \Rightarrow \begin{bmatrix} a_{11} & \boxed{a_{12}} & a_{13} & b_1 \\ & a_{22}^{(1)} & a_{23}^{(1)} & b_2^{(1)} \\ & \boxed{a_{32}^{(1)}} & a_{33}^{(1)} & b_3^{(1)} \end{bmatrix} \Rightarrow \begin{bmatrix} a_{11} & & a_{13}^{(1)} & b_1^{(1)} \\ & a_{22}^{(1)} & a_{23}^{(1)} & b_2^{(1)} \\ & & a_{33}^{(2)} & b_3^{(2)} \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} a_{11} & & & b_1^{(2)} \\ & a_{22}^{(1)} & & b_2^{(2)} \\ & & a_{33}^{(2)} & b_3^{(2)} \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & & & b_1^{(2)}/a_{11} \\ & 1 & & b_2^{(2)}/a_{22}^{(1)} \\ & & 1 & b_3^{(2)}/b_{33}^{(2)} \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & & & c_1^{(3)} \\ & 1 & & c_2^{(3)} \\ & & 1 & c_3^{(3)} \end{bmatrix}$$

It is a variation of Gauss elimination. The major differences are:

– When an unknown is eliminated, it is eliminated from all other equations rather than just the subsequent ones.

– All rows are normalized by dividing them by their pivot elements.

– Elimination step results in an identity matrix.

– Consequently, it is not necessary to employ back substitution to obtain solution.

Advantages:

Gauss-Jordan (GJ) Elimination is prefered when the inverse of a matrix is required.

Pitfalls:

Almost 50% more arithmetic operations than Gaussian elimination.

Improvement strategies:

Same as those used in the Gauss elimination
– Use pivoting and scaling to avoid division-by- zero and to reduce round-off error.

Computing Cost

Which of Gauss elimination and Gauss-Jordan elimination involves more FLOPS?

	Gauss Elimination	Gauss-Jordan Elimination
Elimination Step	Forward Elimination – only needs to eliminate the coefficients below the diagonal. Cost ~ $2n^3/3$	Needs to eliminate coefficients below and above the diagonal. Cost ~ $2 * (2n^3/3)$
Substitution Step	Back Substitution Cost ~ $O(n^2)$	No substitution step
Total	$2n^3/3 + O(n^2)$	$4n^3/3$ (More costly when n is big)

Algorithm :

The same inputs and outputs of Gauss and here is the pseudo code .

Pseudo Code:

```
[solutionTable,finalMatrix,solutions,condition] = GaussJordan(coefficient,results)
[solutionTable,finalMatrix,condition]=forwardElimination(horzcat(coefficient,results));
if (condition==0)
   diagonalCounter=2;
   length=size(finalMatrix);
   while(diagonalCounter<=length(1)){
      rowCounter=diagonalCounter-1;
      while (rowCounter>0){
           factor=-1*finalMatrix(rowCounter,diagonalCounter)/
                     finalMatrix(diagonalCounter,diagonalCounter);
          columnCounter=diagonalCounter;
          while (columnCounter<=length(2)){
                         finalMatrix( rowCounter,columnCounter)=factor*
                    finalMatrix(diagonalCounter ,columnCounter)+ finalMatrix(
                    rowCounter,columnCounter);
         columnCounter=columnCounter+1;
         }
           solutionTable=[solutionTable;finalMatrix];
         rowCounter=rowCounter-1;
      }

      diagonalCounter=diagonalCounter+1;
   }
 count=2;
 variables(1)=0;
 while (count<=length(1)){
 variables=[variables;0];
 count=count+1;
 }
  diagonalCounter=1;
  while ( diagonalCounter<=length(1)){
      variables(diagonalCounter)=finalMatrix(diagonalCounter,length(2))/
           finalMatrix(diagonalCounter,diagonalCounter);
      diagonalCounter=diagonalCounter+1;
  }
   solutions=variables;
else
   solutions=0;
```

Sample Run:

```
>> a=[2,1,4;1,2,3;4,-1,2];
b=[1;1.5;2];
[out1,out2,out3,out4]=GaussJordan(a,b)

out1 =

    4.0000   -1.0000    2.0000    2.0000
         0    2.2500    2.5000    1.0000
    2.0000    1.0000    4.0000    1.0000
    4.0000   -1.0000    2.0000    2.0000
         0    2.2500    2.5000    1.0000
         0    1.5000    3.0000         0
    4.0000   -1.0000    2.0000    2.0000
         0    2.2500    2.5000    1.0000
         0         0    1.3333   -0.6667
    4.0000         0    3.1111    2.4444
         0    2.2500    2.5000    1.0000
         0         0    1.3333   -0.6667
    4.0000         0    3.1111    2.4444
         0    2.2500         0    2.2500
         0         0    1.3333   -0.6667
    4.0000         0         0    4.0000
         0    2.2500         0    2.2500
         0         0    1.3333   -0.6667


out2 =

    4.0000         0         0    4.0000
         0    2.2500         0    2.2500
         0         0    1.3333   -0.6667

out3 =

    1.0000
    1.0000
   -0.5000


out4 =

     0
```

Gauss Seidel Method Report

Introduction

It is an iterative method used to solve a linear system of equations of n linear equations with unknown x:

A x = b

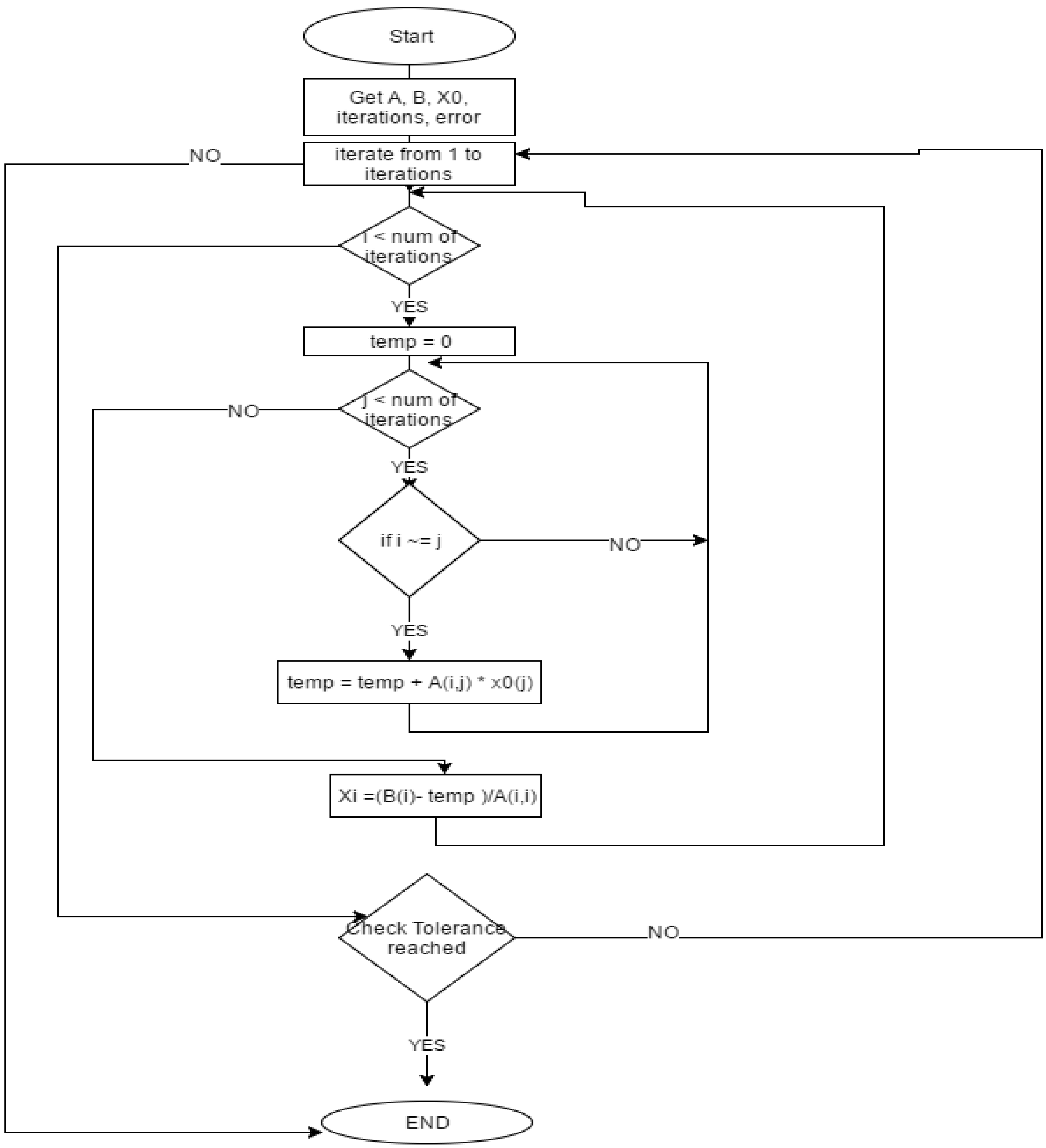
Though it can be applied to any matrix with non-zero elements on the diagonals, convergence is only guaranteed if the matrix is either diagonally dominant, or symmetric and positive definite.

pseudo-code

```
inputs : A,B,x0,iterations, error
repeat from 1 to iterations
   for i from 1 until (num of variables) do

      temp = 0
      for j from 1 until (num of variables) do
         if j ≠ i then
            temp = temp + A(i,j)*x0(j)
         end if
      end (j-loop)
        Xi = (B(i) - temp )/A(i,i)
   end (i-loop)
   check if tolerance is reached
end (repeat)
```

FlowChart



Start
Get A, B, X0, iterations, error
iterate from 1 to iterations
NO
i < num of iterations
YES
temp = 0
j < num of iterations
NO
YES
if i ~= j
NO
YES
temp = temp + A(i,j) * x0(j)
Xi =(B(i)- temp)/A(i,i)
Check Tolerance reached
NO
YES
END

Analysis for the behavior of different examples

Example :

$5x - y + z = 10$
$2x + 8y - z = 11$
$-x + y + 4z = 3$

X0 = [0 0 0]

Solution :

Iterations	xi	yi	zi	xi+1	yi+1	zi+1	time
1.0000	0	0	0	2.0000	0.8750	1.0313	0
2.0000	2.0000	0.8750	1.0313	1.9688	1.0117	0.9893	0.0313
3.0000	1.9688	1.0117	0.9893	2.0045	0.9975	1.0017	0.0357
4.0000	2.0045	0.9975	1.0017	1.9992	1.0004	0.9997	0.0053
5.0000	1.9992	1.0004	0.9997	2.0001	0.9999	1.0001	0.0010
6.0000	2.0001	0.9999	1.0001	2.0000	1.0000	1.0000	0.0002

Example :

$$8x_1 + 4x_2 - x_3 = 11$$
$$-2x_1 + 3x_2 + x_3 = 4$$
$$2x_1 - x_2 + 6x_3 = 7$$

Solution :

Iterations	xi	yi	zi	xi+1	yi+1	zi+1	time
1.0000	0	0	0	1.3750	2.2500	1.0833	0
2.0000	1.3750	2.2500	1.0833	0.3854	1.2292	1.2431	0.9896
3.0000	0.3854	1.2292	1.2431	0.9158	1.5295	1.1163	0.5304
4.0000	0.9158	1.5295	1.1163	0.7498	1.4611	1.1603	0.1660
5.0000	0.7498	1.4611	1.1603	0.7895	1.4729	1.1490	0.0397
6.0000	0.7895	1.4729	1.1490	0.7822	1.4718	1.1512	0.0073
7.0000	0.7822	1.4718	1.1512	0.7830	1.4716	1.1509	0.0008
8.0000	0.7830	1.4716	1.1509	0.7831	1.4717	1.1509	0.0001
9.0000	0.7831	1.4717	1.1509	0.7830	1.4717	1.1509	0.0001

Example :

2*x1+x2+1*x3 = 3

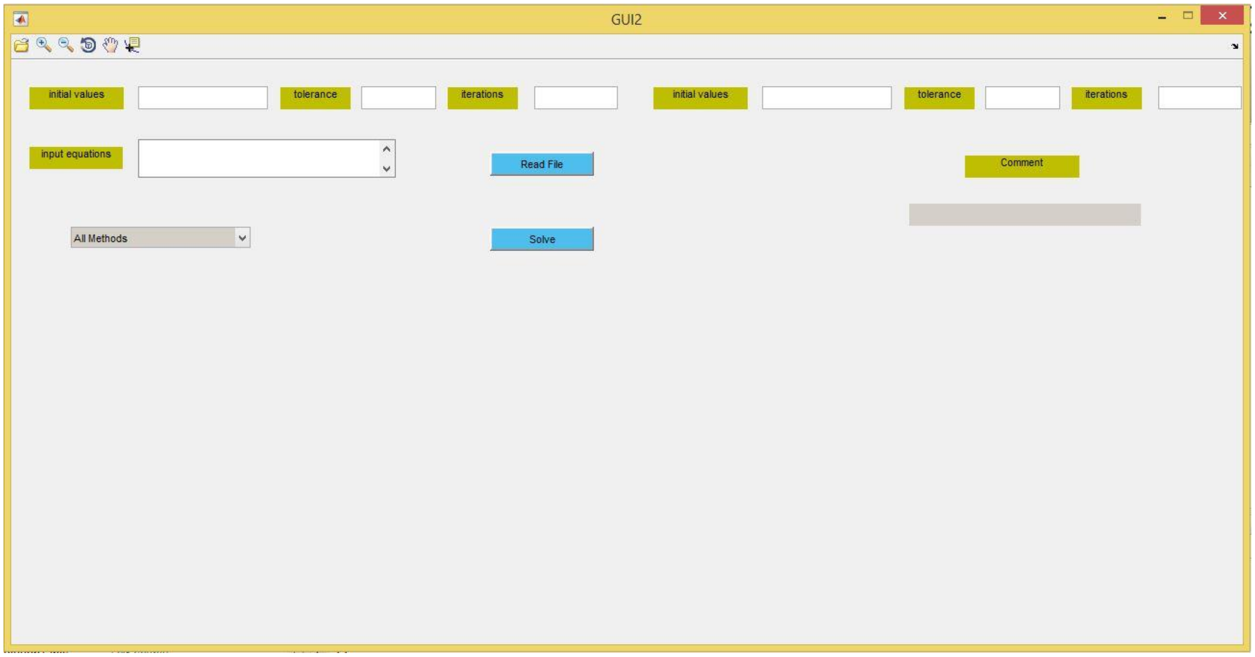
4*x1+2*x2+x3 = 4

x1+2*x2+3*x3 = 4

Solution :

Iterations	xi	yi	zi	xi+1	yi+1	zi+1	time
1.0000	1.0000	2.0000	1.0000	0	1.5000	0.3333	0
2.0000	0	1.5000	0.3333	0.5833	0.6667	0.6944	0.5833
3.0000	0.5833	0.6667	0.6944	0.8194	0.0139	1.0509	0.2361
4.0000	0.8194	0.0139	1.0509	0.9676	-0.4606	1.3179	0.1481
5.0000	0.9676	-0.4606	1.3179	1.0714	-0.8017	1.5107	0.1038
6.0000	1.0714	-0.8017	1.5107	1.1455	-1.0464	1.6491	0.0741
7.0000	1.1455	-1.0464	1.6491	1.1986	-1.2218	1.7483	0.0531
8.0000	1.1986	-1.2218	1.7483	1.2367	-1.3477	1.8195	0.0381
9.0000	1.2367	-1.3477	1.8195	1.2641	-1.4379	1.8706	0.0273
10.0000	1.2641	-1.4379	1.8706	1.2837	-1.5026	1.9072	0.0196
11.0000	1.2837	-1.5026	1.9072	1.2977	-1.5490	1.9334	0.0141
12.0000	1.2977	-1.5490	1.9334	1.3078	-1.5823	1.9523	0.0101
13.0000	1.3078	-1.5823	1.9523	1.3150	-1.6062	1.9658	0.0072
14.0000	1.3150	-1.6062	1.9658	1.3202	-1.6233	1.9755	0.0052
15.0000	1.3202	-1.6233	1.9755	1.3239	-1.6356	1.9824	0.0037
16.0000	1.3239	-1.6356	1.9824	1.3266	-1.6444	1.9874	0.0027
17.0000	1.3266	-1.6444	1.9874	1.3285	-1.6507	1.9909	0.0019
18.0000	1.3285	-1.6507	1.9909	1.3299	-1.6552	1.9935	0.0014
19.0000	1.3299	-1.6552	1.9935	1.3308	-1.6584	1.9953	0.0010
20.0000	1.3308	-1.6584	1.9953	1.3315	-1.6608	1.9967	0.0007
21.0000	1.3315	-1.6608	1.9967	1.3321	-1.6624	1.9976	0.0005

User Guide



GUI has some fileds and buttons as viewed in the photo , each one has a meaning or a function :

1. Input equations : user can enter input equations here which he want to solve , it can have spaces or tabs but must be in valid format containing x1,x2,…..,xn and their coefficients like :
   (x1+x2+2*x3 = 3 or x1 -2*x2 +2*x3 = 3)

2. Drop down list : user can select the desired method he wants to solve linear system using it by just selecting it

3. Initial value : user can enter initial value here in case of using iterative methods (it is only visible and able to be used in case of selecting an iterative method from drop down list options , otherwise it will be invisible)

   (Default value = [0,0,0,……] n zeros)

4. Tolerance : user can specify tolerance / Absolute error which is allowed in the final solutions of x1,x2,…,xn to increase accuracy

(Default value = 0.00001)

5. Iterations : user can specify maximum number of iterations allowed it iterative method if he cares more about running time versus accuracy (Default value = 50)

6. Initial value , tolerance and iterations in the right half side like the mentioned above but it is used to the second iterative method (Jacobi iterative) in case of using All Methods, otherwise it will be invisible always and we use the ones in the left half side only .

7. Read File Button : allow user to choose input file (extension : ".txt") to read parameters and input equations from it , the input file must be in valid and fixed format :
   - The first line must contain number of equations we want to solve .
   - The n next line each one will contain an equation till finishing all the equations.
   - The next line don't have any fixed format but for flexibility we make it ("key name") then next line you can write ("key value")
     for example :

```
{
        3
        x1+x2+x3 = 3
        x1+2*x2+x3 = 4
        x1+2*x2+3*x3 = 4
        initial
        1 2 1
        tolerance
        0.00004
        iterations
        20
}
```

8. Solve Button: to solve equations with the given parameters, if the parameters are valid then we will execute method call, otherwise we will show a message “invalid input” in comment field, if inputs are valid then we will look for solutions if we can get it then display the solutions in an appropriate view depending on the selected methods like (tables, matrix representation) we will see that in sample runs section.

   Note: after solving using any method we output the solutions in a text file which is located in same directory where running GUI exists.

Sample runs

First Example

Inputs {

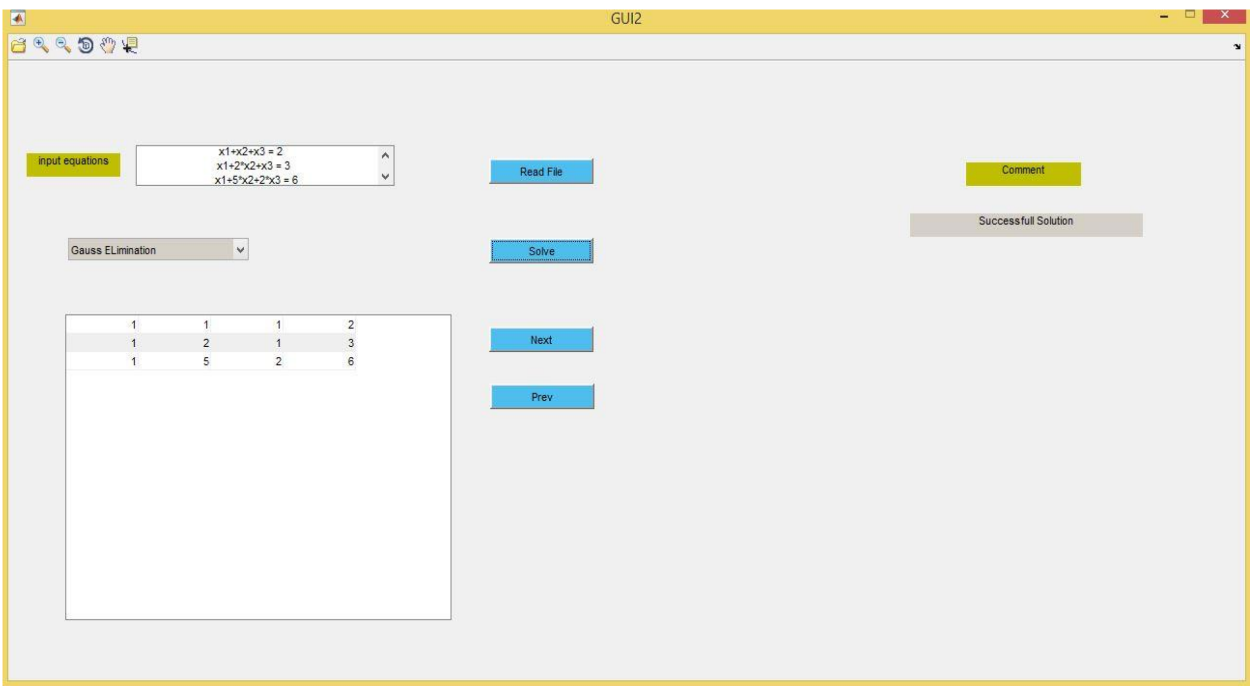
x1+x2+x3 = 2

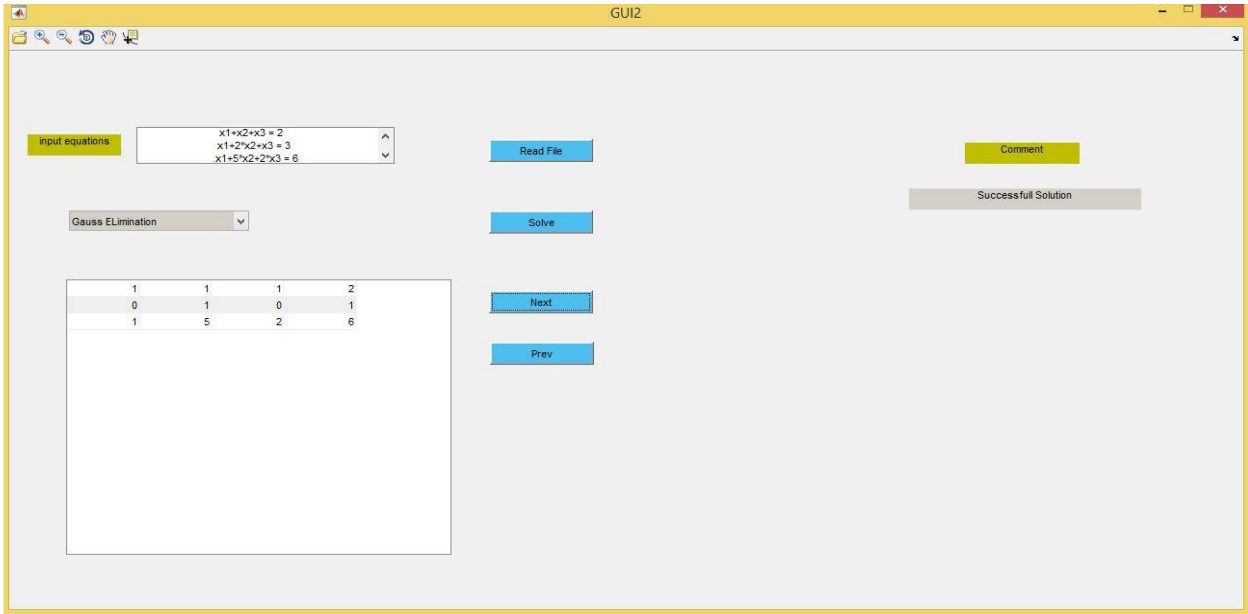
x1+2*x2+x3 = 3

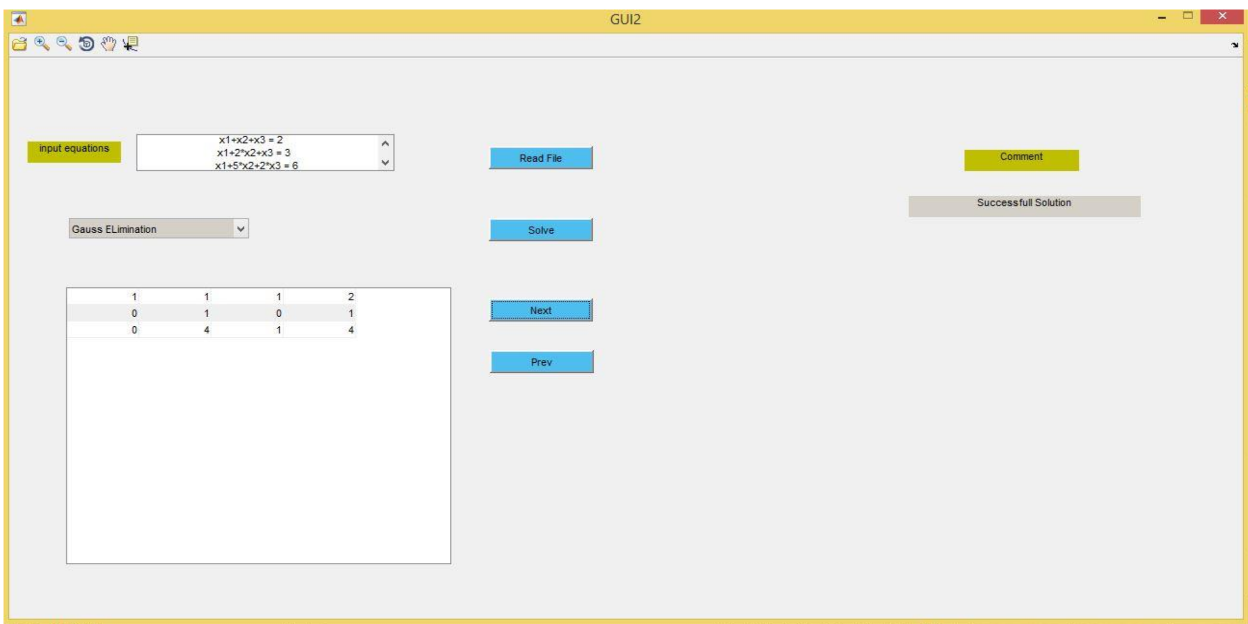
x1+5*x2+2*x3 = 6

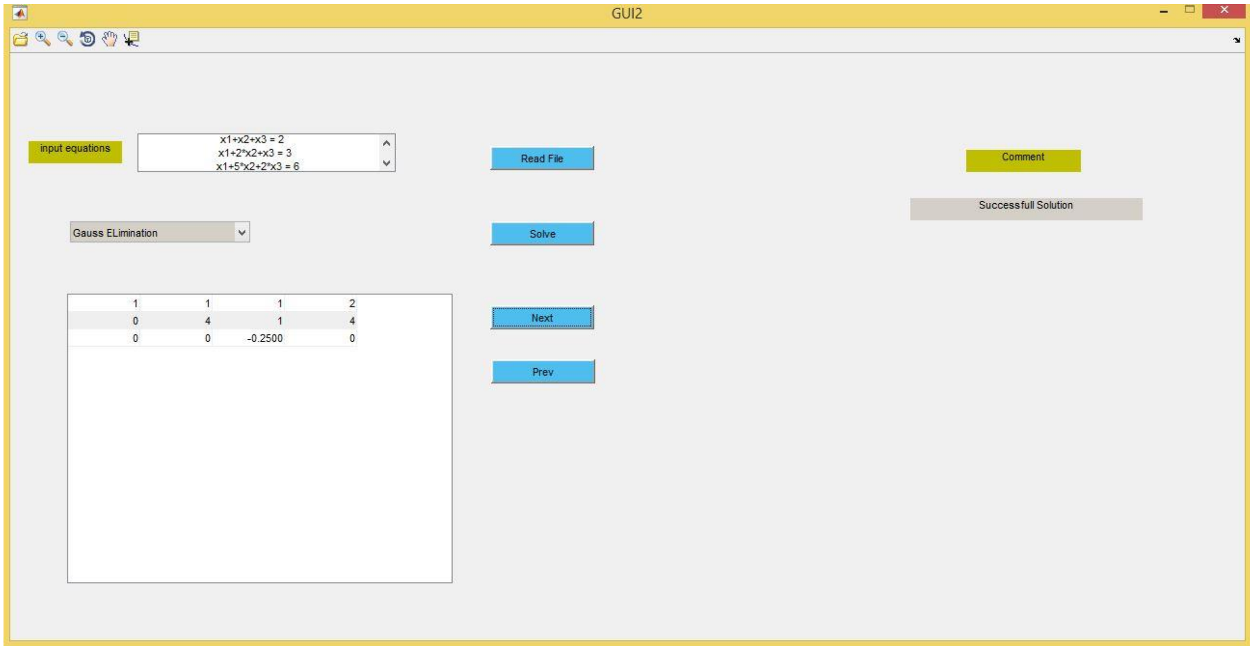
}

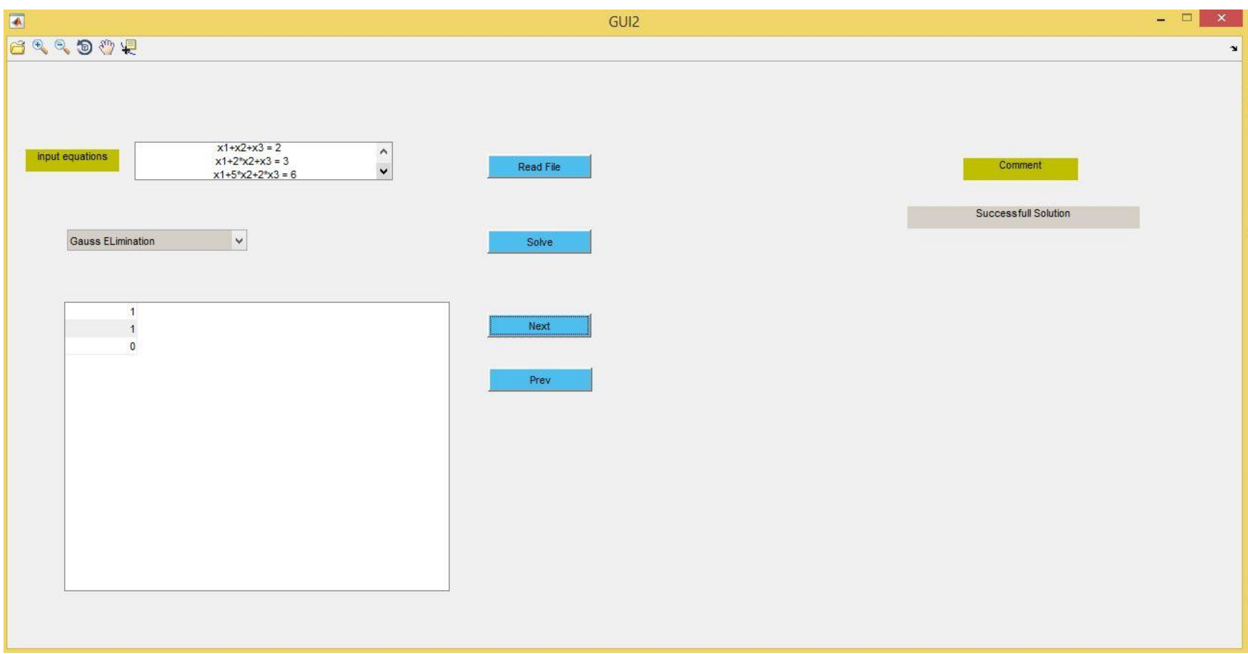
Solving using Gauss Elimination with showing every step using Next and Previous Buttons which are allowed and visible only in Direct Methods




GUI2
input equations
x1+x2+x3 = 2
x1+2*x2+x3 = 3
x1+5*x2+2*x3 = 6
Read File
Comment
Successfull Solution
Gauss ELimination
Solve
1 1 1 2
0 1 0 1
1 5 2 6
Next
Prev


GUI2
input equations
x1+x2+x3 = 2
x1+2*x2+x3 = 3
x1+5*x2+2*x3 = 6
Read File
Comment
Successfull Solution
Gauss ELimination
Solve
1 1 1 2
0 1 0 1
0 4 1 4
Next
Prev


GUI2
input equations
x1+x2+x3 = 2
x1+2*x2+x3 = 3
x1+5*x2+2*x3 = 6
Read File
Comment
Successfull Solution
Gauss ELimination
Solve
1 1 1 2
0 4 1 4
0 0 -0.2500 0
Next
Prev


GUI2
input equations
x1+x2+x3 = 2
x1+2*x2+x3 = 3
x1+5*x2+2*x3 = 6
Read File
Comment
Successfull Solution
Gauss ELimination
Solve
1
1
0
Next
Prev

The output will be :

outputGaussElimination.txt - Notepad

File Edit Format View Help

```
Original Matrix
1.0000000000  1.0000000000  1.0000000000  |  2.0000000000
1.0000000000  2.0000000000  1.0000000000  |  3.0000000000
1.0000000000  5.0000000000  2.0000000000  |  6.0000000000
-------------------------------------------------------------------
Step 1
1.0000000000  1.0000000000  1.0000000000  |  2.0000000000
0.0000000000  1.0000000000  0.0000000000  |  1.0000000000
1.0000000000  5.0000000000  2.0000000000  |  6.0000000000
-------------------------------------------------------------------
Step 2
1.0000000000  1.0000000000  1.0000000000  |  2.0000000000
0.0000000000  1.0000000000  0.0000000000  |  1.0000000000
0.0000000000  4.0000000000  1.0000000000  |  4.0000000000
-------------------------------------------------------------------
Step 3
1.0000000000  1.0000000000  1.0000000000  |  2.0000000000
0.0000000000  4.0000000000  1.0000000000  |  4.0000000000
0.0000000000  0.0000000000  -0.2500000000  |  0.0000000000
-------------------------------------------------------------------
Solutions
1.0000000000
1.0000000000
0.0000000000
```

Second Example

Inputs {

x1+x2+x3 = 2

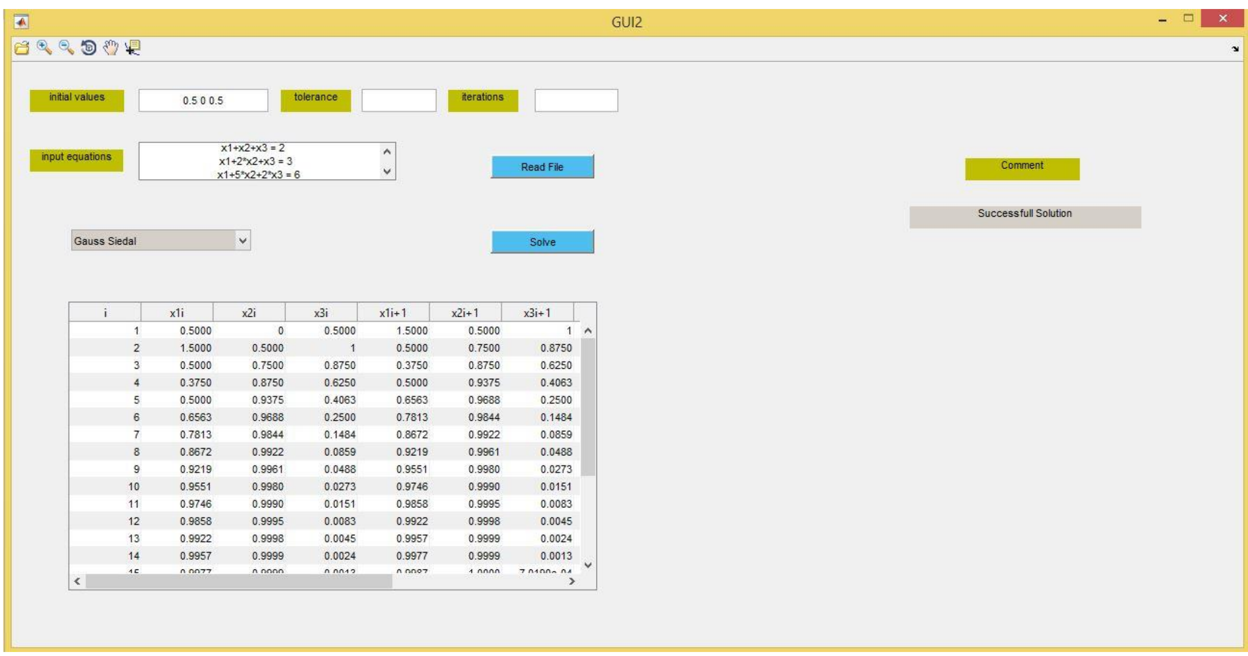
x1+2*x2+x3 = 3

x1+5*x2+2*x3 = 6

}

Solving using Gauss Seidel with initial value = [0.5 , 0 , 0.5]

Output data will be viewed in table format .



i	x1i	x2i	x3i	x1i+1	x2i+1	x3i+1
1	0.5000	0	0.5000	1.5000	0.5000	1
2	1.5000	0.5000	1	0.5000	0.7500	0.8750
3	0.5000	0.7500	0.8750	0.3750	0.8750	0.6250
4	0.3750	0.8750	0.6250	0.5000	0.9375	0.4063
5	0.5000	0.9375	0.4063	0.6563	0.9688	0.2500
6	0.6563	0.9688	0.2500	0.7813	0.9844	0.1484
7	0.7813	0.9844	0.1484	0.8672	0.9922	0.0859
8	0.8672	0.9922	0.0859	0.9219	0.9961	0.0488
9	0.9219	0.9961	0.0488	0.9551	0.9980	0.0273
10	0.9551	0.9980	0.0273	0.9746	0.9990	0.0151
11	0.9746	0.9990	0.0151	0.9858	0.9995	0.0083
12	0.9858	0.9995	0.0083	0.9922	0.9998	0.0045
13	0.9922	0.9998	0.0045	0.9957	0.9999	0.0024
14	0.9957	0.9999	0.0024	0.9977	0.9999	0.0013

The output file will be :

outputGaussSeidel.txt - Notepad

File Edit Format View Help

i	x1i	x2i	x3i	x1i+1	x2i+1	x3i+1	Err1	Err2	Err3	Time
1.000000	0.500000	0.000000	0.500000	1.500000	0.500000	1.000000	0.000000	0.000000	0.000000	0.297012
2.000000	1.500000	0.500000	1.000000	0.500000	0.750000	0.875000	1.000000	0.250000	0.125000	0.345328
3.000000	0.500000	0.750000	0.875000	0.375000	0.875000	0.625000	0.125000	0.125000	0.250000	0.403858
4.000000	0.375000	0.875000	0.625000	0.500000	0.937500	0.406250	0.125000	0.062500	0.218750	0.377480
5.000000	0.500000	0.937500	0.406250	0.656250	0.968750	0.250000	0.156250	0.031250	0.156250	0.382833
6.000000	0.656250	0.968750	0.250000	0.781250	0.984375	0.148438	0.125000	0.015625	0.101563	0.377950
7.000000	0.781250	0.984375	0.148438	0.867188	0.992188	0.085938	0.085938	0.007813	0.062500	0.379280
8.000000	0.867188	0.992188	0.085938	0.921875	0.996094	0.048828	0.054688	0.003906	0.037109	0.376858
9.000000	0.921875	0.996094	0.048828	0.955078	0.998047	0.027344	0.033203	0.001953	0.021484	0.388700
10.000000	0.955078	0.998047	0.027344	0.974609	0.999023	0.015137	0.019531	0.000977	0.012207	0.392223
11.000000	0.974609	0.999023	0.015137	0.985840	0.999512	0.008301	0.011230	0.000488	0.006836	0.375784
12.000000	0.985840	0.999512	0.008301	0.992188	0.999756	0.004517	0.006348	0.000244	0.003784	0.420873
13.000000	0.992188	0.999756	0.004517	0.995728	0.999878	0.002441	0.003540	0.000122	0.002075	0.430783
14.000000	0.995728	0.999878	0.002441	0.997681	0.999939	0.001312	0.001953	0.000061	0.001129	0.378343
15.000000	0.997681	0.999939	0.001312	0.998749	0.999969	0.000702	0.001068	0.000031	0.000610	0.381664
16.000000	0.998749	0.999969	0.000702	0.999329	0.999985	0.000374	0.000580	0.000015	0.000328	0.369246
17.000000	0.999329	0.999985	0.000374	0.999641	0.999992	0.000198	0.000313	0.000008	0.000175	0.383738
18.000000	0.999641	0.999992	0.000198	0.999809	0.999996	0.000105	0.000168	0.000004	0.000093	0.375959
19.000000	0.999809	0.999996	0.000105	0.999899	0.999998	0.000055	0.000090	0.000002	0.000050	0.375555
20.000000	0.999899	0.999998	0.000055	0.999947	0.999999	0.000029	0.000048	0.000001	0.000026	0.379801
21.000000	0.999947	0.999999	0.000029	0.999972	1.000000	0.000015	0.000025	0.000000	0.000014	0.390914
22.000000	0.999972	1.000000	0.000015	0.999985	1.000000	0.000008	0.000013	0.000000	0.000007	0.385535
23.000000	0.999985	1.000000	0.000008	0.999992	1.000000	0.000004	0.000007	0.000000	0.000004	0.390370

Third Example

Inputs {

2*x1+x2+x3 = 2

2*x1+6*x2+x3 = 3

x1+4*x2-2*x3 = 6

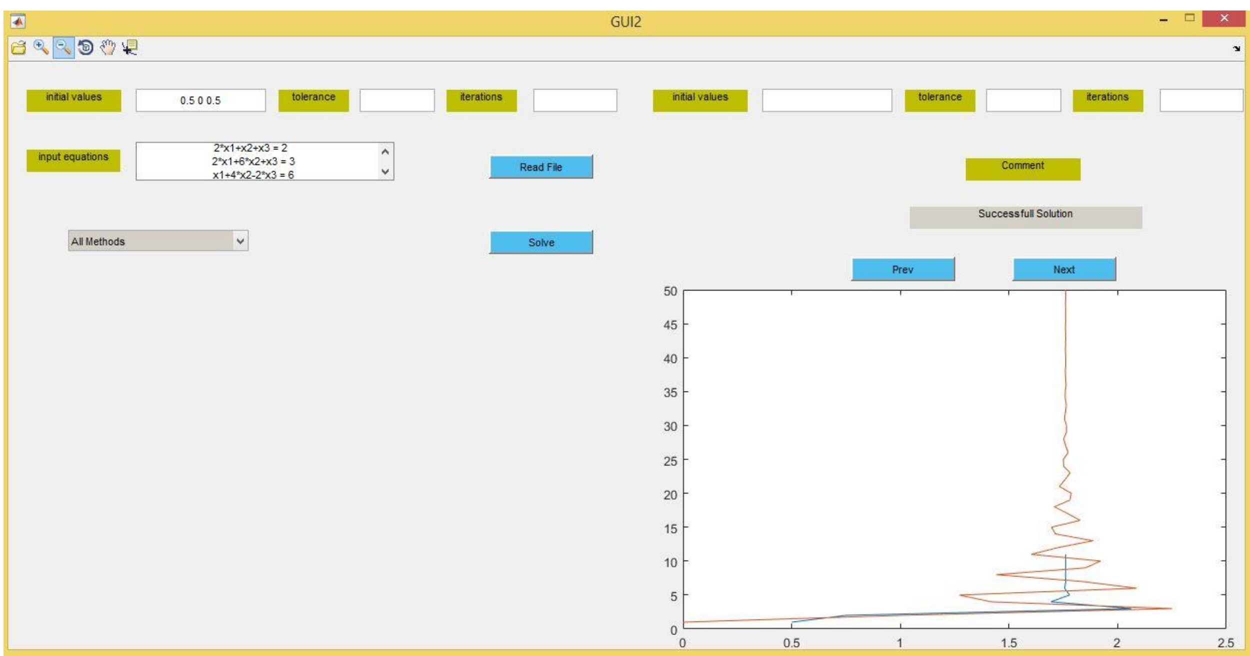
}

Solving using All Methods with initial value = [0.5 , 0 , 0.5]

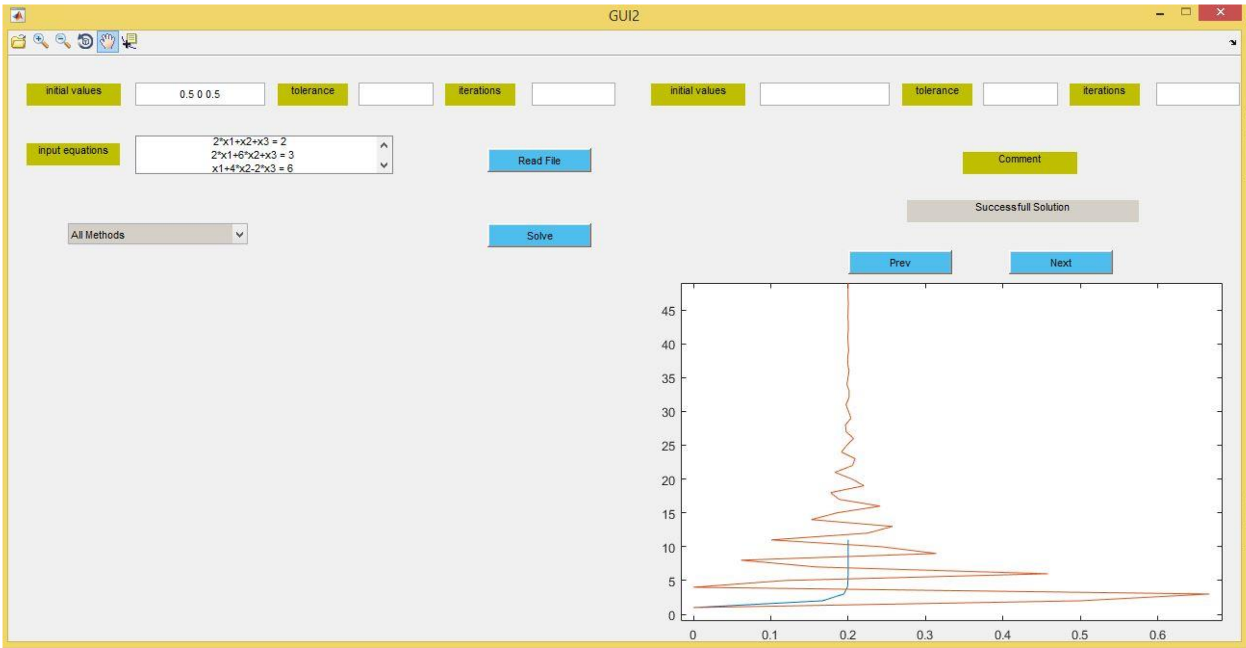
For Gauss Seidel Method.

The Curves between number of iterations and approximate root at this iterations will be plotted in case of using all methods and will be like :

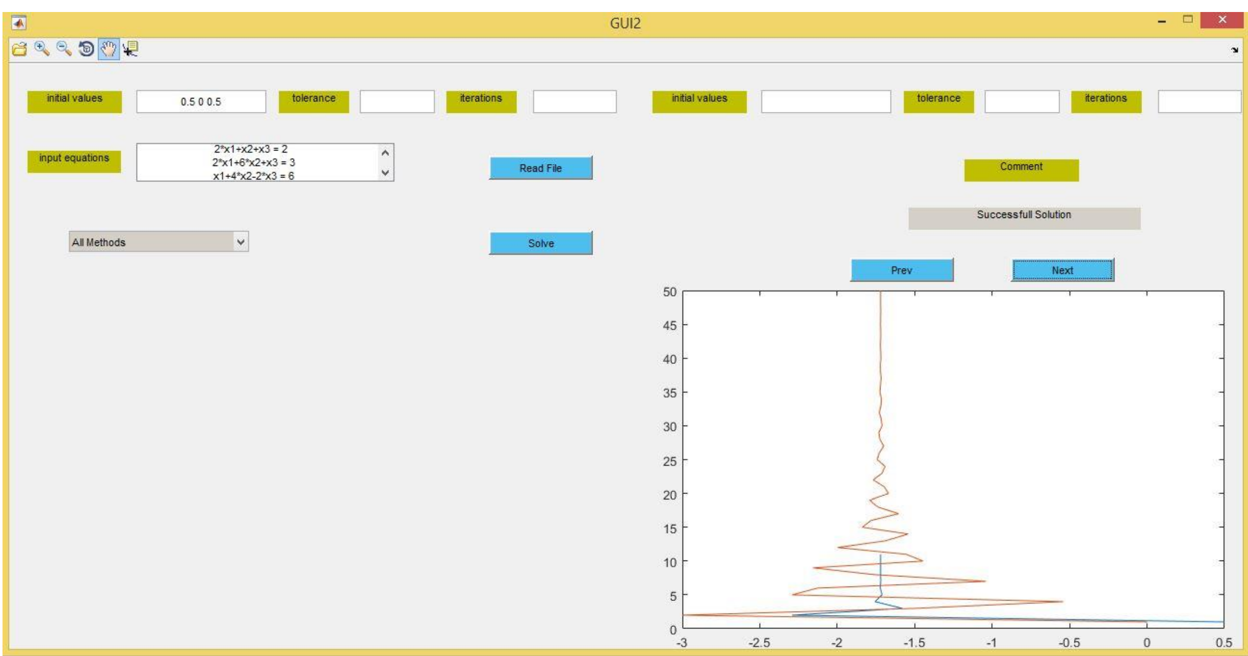
<u>Curve for x1</u>



Curve for x2



Curve for x3



The output Files :

- Gauss Elimination File :

outputGaussElimination.txt - Notepad

File Edit Format View Help

```
Original Matrix
2.0000000000  1.0000000000  1.0000000000  |  2.0000000000
2.0000000000  6.0000000000  1.0000000000  |  3.0000000000
1.0000000000  4.0000000000  -2.0000000000  |  6.0000000000
-------------------------------------------------------------------
Step 1
2.0000000000  1.0000000000  1.0000000000  |  2.0000000000
0.0000000000  5.0000000000  0.0000000000  |  1.0000000000
1.0000000000  4.0000000000  -2.0000000000  |  6.0000000000
-------------------------------------------------------------------
Step 2
2.0000000000  1.0000000000  1.0000000000  |  2.0000000000
0.0000000000  5.0000000000  0.0000000000  |  1.0000000000
0.0000000000  3.5000000000  -2.5000000000  |  5.0000000000
-------------------------------------------------------------------
Step 3
2.0000000000  1.0000000000  1.0000000000  |  2.0000000000
0.0000000000  5.0000000000  0.0000000000  |  1.0000000000
0.0000000000  0.0000000000  -2.5000000000  |  4.3000000000
-------------------------------------------------------------------
Solutions
1.7600000000
0.2000000000
-1.7200000000
```

- Gauss Jordan File :

outputGaussJordan.txt - Notepad

File Edit Format View Help

```
Original Matrix
2.0000000000  1.0000000000  1.0000000000  |  2.0000000000
2.0000000000  6.0000000000  1.0000000000  |  3.0000000000
1.0000000000  4.0000000000  -2.0000000000  |  6.0000000000
-------------------------------------------------------------------
Step 1
2.0000000000  1.0000000000  1.0000000000  |  2.0000000000
0.0000000000  5.0000000000  0.0000000000  |  1.0000000000
1.0000000000  4.0000000000  -2.0000000000  |  6.0000000000
-------------------------------------------------------------------
Step 2
2.0000000000  1.0000000000  1.0000000000  |  2.0000000000
0.0000000000  5.0000000000  0.0000000000  |  1.0000000000
0.0000000000  3.5000000000  -2.5000000000  |  5.0000000000
-------------------------------------------------------------------
Step 3
2.0000000000  1.0000000000  1.0000000000  |  2.0000000000
0.0000000000  5.0000000000  0.0000000000  |  1.0000000000
0.0000000000  0.0000000000  -2.5000000000  |  4.3000000000
-------------------------------------------------------------------
Step 4
2.0000000000  0.0000000000  1.0000000000  |  1.8000000000
0.0000000000  5.0000000000  0.0000000000  |  1.0000000000
0.0000000000  0.0000000000  -2.5000000000  |  4.3000000000
-------------------------------------------------------------------
Step 5
2.0000000000  0.0000000000  1.0000000000  |  1.8000000000
0.0000000000  5.0000000000  0.0000000000  |  1.0000000000
0.0000000000  0.0000000000  -2.5000000000  |  4.3000000000
-------------------------------------------------------------------
Step 6
2.0000000000  0.0000000000  0.0000000000  |  3.5200000000
0.0000000000  5.0000000000  0.0000000000  |  1.0000000000
0.0000000000  0.0000000000  -2.5000000000  |  4.3000000000
-------------------------------------------------------------------
Solutions
```


```
-------------------------------------------------------------------
Step 6
2.0000000000  0.0000000000  0.0000000000  |  3.5200000000
0.0000000000  5.0000000000  0.0000000000  |  1.0000000000
0.0000000000  0.0000000000  -2.5000000000  |  4.3000000000
-------------------------------------------------------------------
Solutions
```


```
1.7600000000
0.2000000000
-1.7200000000
```

- LU Decomposition File :

outputLUDecomposition.txt - Notepad

File Edit Format View Help

```
Matrix A
2.0000000000  1.0000000000  1.0000000000
2.0000000000  6.0000000000  1.0000000000
1.0000000000  4.0000000000  -2.0000000000
--------------------------------------------------------------------
Step 1 to get U
3.0000000000  1.0000000000  1.0000000000
0.0000000000  6.0000000000  0.0000000000
1.0000000000  4.0000000000  -1.0000000000
--------------------------------------------------------------------
Step 2 to get U
3.0000000000  1.0000000000  1.0000000000
0.0000000000  6.0000000000  0.0000000000
0.0000000000  3.5000000000  -1.5000000000
--------------------------------------------------------------------
Matrix U
3.0000000000  1.0000000000  1.0000000000
0.0000000000  6.0000000000  0.0000000000
0.0000000000  0.0000000000  -1.5000000000
--------------------------------------------------------------------
Matrix L
1.0000000000  0.0000000000  0.0000000000
1.0000000000  1.0000000000  0.0000000000
0.5000000000  0.7000000000  1.0000000000
--------------------------------------------------------------------
Y Solutions
2.0000000000
1.0000000000
4.3000000000
--------------------------------------------------------------------
X Solutions
1.7600000000
0.2000000000
-1.7200000000
```

- Gauss Seidel File :

outputGaussSeidel.txt - Notepad

File Edit Format View Help

i	x1i	x2i	x3i	x1i+1	x2i+1	x3i+1	Err1	Err2	Err3	Time
1.000000	0.500000	0.000000	0.500000	0.750000	0.166667	-2.291667	0.000000	0.000000	0.000000	0.280881
2.000000	0.750000	0.166667	-2.291667	2.062500	0.194444	-1.579861	1.312500	0.027778	0.711806	0.378859
3.000000	2.062500	0.194444	-1.579861	1.692708	0.199074	-1.755498	0.369792	0.004630	0.175637	0.408618
4.000000	1.692708	0.199074	-1.755498	1.778212	0.199846	-1.711203	0.085503	0.000772	0.044295	0.464212
5.000000	1.778212	0.199846	-1.711203	1.755679	0.199974	-1.722212	0.022533	0.000129	0.011009	0.536877
6.000000	1.755679	0.199974	-1.722212	1.761119	0.199996	-1.719449	0.005440	0.000021	0.002763	0.533473
7.000000	1.761119	0.199996	-1.719449	1.759727	0.199999	-1.720138	0.001392	0.000004	0.000689	0.424804
8.000000	1.759727	0.199999	-1.720138	1.760069	0.200000	-1.719966	0.000343	0.000001	0.000173	0.499120
9.000000	1.760069	0.200000	-1.719966	1.759983	0.200000	-1.720009	0.000087	0.000000	0.000043	0.409948
10.000000	1.759983	0.200000	-1.720009	1.760004	0.200000	-1.719998	0.000021	0.000000	0.000011	0.520069
11.000000	1.760004	0.200000	-1.719998	1.759999	0.200000	-1.720001	0.000005	0.000000	0.000003	0.449175

- Jacobi Iterative File :

outputJacobiIterative.txt - Notepad

File Edit Format View Help

i	x1i	x2i	x3i	x1i+1	x2i+1	x3i+1	Err1	Err2	Err3	Time
1.000000	0.000000	0.000000	0.000000	1.000000	0.500000	-3.000000	0.000000	0.000000	0.000000	0.242039
2.000000	1.000000	0.500000	-3.000000	2.250000	0.666667	-1.500000	1.250000	0.166667	1.500000	0.458919
3.000000	2.250000	0.666667	-1.500000	1.416667	0.000000	-0.541667	0.833333	0.666667	0.958333	0.489714
4.000000	1.416667	0.000000	-0.541667	1.270833	0.118056	-2.291667	0.145833	0.118056	1.750000	0.568842
5.000000	1.270833	0.118056	-2.291667	2.086806	0.458333	-2.128472	0.815972	0.340278	0.163194	0.555227
6.000000	2.086806	0.458333	-2.128472	1.835069	0.159144	-1.039931	0.251736	0.299190	1.088542	0.403346
7.000000	1.835069	0.159144	-1.039931	1.440394	0.061632	-1.764178	0.394676	0.097512	0.724248	0.461806
8.000000	1.440394	0.061632	-1.764178	1.851273	0.313899	-2.156539	0.410880	0.252267	0.392361	0.417241
9.000000	1.851273	0.313899	-2.156539	1.921320	0.242332	-1.446566	0.070047	0.071566	0.709973	0.429901
10.000000	1.921320	0.242332	-1.446566	1.602117	0.100654	-1.554675	0.319203	0.141678	0.108109	0.436206
11.000000	1.602117	0.100654	-1.554675	1.727011	0.225074	-1.997633	0.124894	0.124419	0.442957	0.433233
12.000000	1.727011	0.225074	-1.997633	1.886280	0.257269	-1.686348	0.159269	0.032195	0.311285	0.469826
13.000000	1.886280	0.257269	-1.686348	1.714539	0.152298	-1.542323	0.171740	0.104971	0.144025	0.440361
14.000000	1.714539	0.152298	-1.542323	1.695012	0.185541	-1.838134	0.019527	0.033243	0.295811	0.432531
15.000000	1.695012	0.185541	-1.838134	1.826297	0.241352	-1.781412	0.131284	0.055811	0.056722	0.468466
16.000000	1.826297	0.241352	-1.781412	1.770030	0.188136	-1.604149	0.056266	0.053215	0.177264	0.425963
17.000000	1.770030	0.188136	-1.604149	1.708006	0.177348	-1.738712	0.062024	0.010789	0.134563	0.517977
18.000000	1.708006	0.177348	-1.738712	1.780682	0.220450	-1.791301	0.072676	0.043102	0.052589	0.454373
19.000000	1.780682	0.220450	-1.791301	1.785426	0.204990	-1.668759	0.004744	0.015460	0.122542	0.489219
20.000000	1.785426	0.204990	-1.668759	1.731885	0.182985	-1.697308	0.053541	0.022005	0.028549	0.484976
21.000000	1.731885	0.182985	-1.697308	1.757162	0.205590	-1.768088	0.025277	0.022605	0.070780	0.547858
22.000000	1.757162	0.205590	-1.768088	1.781249	0.208961	-1.710240	0.024088	0.003371	0.057849	0.541107
23.000000	1.781249	0.208961	-1.710240	1.750639	0.191290	-1.691454	0.030610	0.017671	0.018786	0.538663
24.000000	1.750639	0.191290	-1.691454	1.750082	0.198362	-1.742100	0.000558	0.007072	0.050646	0.608378
25.000000	1.750082	0.198362	-1.742100	1.771869	0.206989	-1.728234	0.021787	0.008627	0.013866	0.548001
26.000000	1.771869	0.206989	-1.728234	1.760622	0.197416	-1.700087	0.011246	0.009573	0.028147	0.505299
27.000000	1.760622	0.197416	-1.700087	1.751335	0.196474	-1.724857	0.009287	0.000942	0.024770	0.475961
28.000000	1.751335	0.196474	-1.724857	1.764191	0.203698	-1.731385	0.012856	0.007224	0.006528	0.501723
29.000000	1.764191	0.203698	-1.731385	1.763844	0.200500	-1.710509	0.000348	0.003197	0.020876	0.526536
30.000000	1.763844	0.200500	-1.710509	1.755004	0.197137	-1.717077	0.008839	0.003363	0.006569	0.888799
31.000000	1.755004	0.197137	-1.717077	1.759970	0.201178	-1.728224	0.004966	0.004041	0.011146	0.718667
32.000000	1.759970	0.201178	-1.728224	1.763523	0.201381	-1.717659	0.003553	0.000202	0.010565	0.525600
33.000000	1.763523	0.201381	-1.717659	1.758139	0.198435	-1.715477	0.005384	0.002945	0.002181	0.484615
34.000000	1.758139	0.198435	-1.715477	1.758521	0.199867	-1.724060	0.000382	0.001431	0.008582	0.407639
35.000000	1.758521	0.199867	-1.724060	1.762097	0.201170	-1.721006	0.003576	0.001303	0.003053	0.299097

30.000000	1.763844	0.200500	-1.710509	1.755004	0.197137	-1.717077	0.008839	0.003363	0.006569	0.888799
31.000000	1.755004	0.197137	-1.717077	1.759970	0.201178	-1.728224	0.004966	0.004041	0.011146	0.718667
32.000000	1.759970	0.201178	-1.728224	1.763523	0.201381	-1.717659	0.003553	0.000202	0.010565	0.525600
33.000000	1.763523	0.201381	-1.717659	1.758139	0.198435	-1.715477	0.005384	0.002945	0.002181	0.484615
34.000000	1.758139	0.198435	-1.715477	1.758521	0.199867	-1.724060	0.000382	0.001431	0.008582	0.407639
35.000000	1.758521	0.199867	-1.724060	1.762097	0.201170	-1.721006	0.003576	0.001303	0.003053	0.299097
36.000000	1.762097	0.201170	-1.721006	1.759918	0.199469	-1.716613	0.002178	0.001701	0.004394	0.391191
37.000000	1.759918	0.199469	-1.716613	1.758572	0.199463	-1.721103	0.001347	0.000006	0.004490	0.525877
38.000000	1.758572	0.199463	-1.721103	1.760820	0.200660	-1.721789	0.002248	0.001197	0.000686	0.498912
39.000000	1.760820	0.200660	-1.721789	1.760564	0.200025	-1.718270	0.000256	0.000635	0.003519	0.681253
40.000000	1.760564	0.200025	-1.718270	1.759123	0.199524	-1.719668	0.001442	0.000501	0.001398	0.523137
41.000000	1.759123	0.199524	-1.719668	1.760072	0.200237	-1.721392	0.000950	0.000714	0.001723	0.389077
42.000000	1.760072	0.200237	-1.721392	1.760577	0.200208	-1.719490	0.000505	0.000029	0.001902	0.400020
43.000000	1.760577	0.200208	-1.719490	1.759641	0.199723	-1.719296	0.000936	0.000485	0.000194	0.452170
44.000000	1.759641	0.199723	-1.719296	1.759787	0.200002	-1.720735	0.000146	0.000280	0.001439	0.493436
45.000000	1.759787	0.200002	-1.720735	1.760366	0.200194	-1.720102	0.000579	0.000191	0.000633	0.500173
46.000000	1.760366	0.200194	-1.720102	1.759954	0.199895	-1.719430	0.000412	0.000299	0.000672	0.446500
47.000000	1.759954	0.199895	-1.719430	1.759767	0.199920	-1.720233	0.000187	0.000025	0.000803	0.450034
48.000000	1.759767	0.199920	-1.720233	1.760156	0.200116	-1.720276	0.000389	0.000196	0.000043	0.453435
49.000000	1.760156	0.200116	-1.720276	1.760080	0.199994	-1.719689	0.000077	0.000122	0.000587	0.440747
50.000000	1.760080	0.199994	-1.719689	1.759848	0.199922	-1.719972	0.000232	0.000072	0.000283	0.444436