Solving linear equations with Methods "Gauss Elimination, Gauss Jordan Elimination, LU decomposition, Gauss Seidel iteration, Jacobi iteration" With JAVA

Team Members	Academic ID
Omar Muhammed El said Metmowh	**19016082**
Ahmed Rabea Salam Ali	**19015229**
Marwan Mahmoud Ibrahim Muhammed	**19016621**
Ussif Ashraf Ussif	**19016910**
Ibrahim Tarek Ibrahim Abdelaal	**19015167**

Numerical Computing Project

- **Flowchart or pseudo-code for some important functions:**

Gauss elimination

Forward

```
For K = 1 to N
        max = K
        For I = K+1 to N
                If (Ai,k > Amax,k) max = i
        If (Amax,k) == 0 break;
        SwapRows(A, max, k);
    SwapValues(B, max, k);
        For I = K+1 to N
                Factor = Ai,k / Ai,i
                For J = K+1 to N
                        Ai,j =  Ai,j – factor * Ai,k
                Next J
        Next I
Next K
```

Backward

```
For K = N-1 to 0
        For I = K-1        to 0
                Factor = Ai,k / Ak,k
                 For J = K to 0
                        Ai,j =  Ai,j – factor * Ak,j
```

LU decomposition

Doolittle Form

```
For I = 0 to N
        //Upper
        For K = I to N
                Sum = 0
                For J = 0 to I
                        Sum = Sum + (Li,j * Uj,k)
                Next J
        Ui,k = Ai,k – Sum
        Next K
        //Lower
        For K = I to N
                If I == K
                        Li,i = 1
                End of If
                Else
                        For J = 0 to I
                        Sum = Sum + (Lk,j * Uj,i)
                Next J
        Lk,i = (Ak,i – Sum) / Ui,i
        Next K
Next I
```

Croute Form

```
        For I = 0 to N
                //Lower
                For J = 0 to N
                        Sum = 0
                        If I > J
                                Lj,i = 0
                        End of If
                        Else
                                For K = 0 to I
                                        Sum = Sum + (Lj,k * Uk,i)
                                Next K
                Lj,i = Aj,i – Sum
                Next J
//Upper
For J = 0 to N
                Sum = 0
                If I > J
                        Uj,i = 0
                End of If
                If I == J
                        Ui,j = 1
                End of If
                Else
                For K = 0 to I
                        Sum = Sum + (Li,k * Uk,j)/Li,i
                Next J
        Ui,j = (Ai,k /Li,i) – Sum
        Next K
Next I
```

Cholesky Form

```
For I = 0 to N
        //Upper
        For K = I to N
                Sum = 0
                For J = 0 to I
                        Sum = Sum + (Li,j * Uj,k)
                Next J
        Ui,k = Ai,k – Sum
        Next K
        //Lower
        For K = I to N
                If I == K
                        Li,i = 1
                End of If
                Else
                        For J = 0 to I
                        Sum = Sum + (Lk,j * Uj,i)
                Next J
        Lk,i = (Ak,i – Sum) / Ui,i
        Next K
        //Diagonal
```

```
        Temp = Ui,i
        Di,i = Temp
        For J = 0 to N
                If I != J
                        Di,j=0
                End If
                Ui,j = Ui,j / Temp
        Next J
Next I
```

GaussSeidel

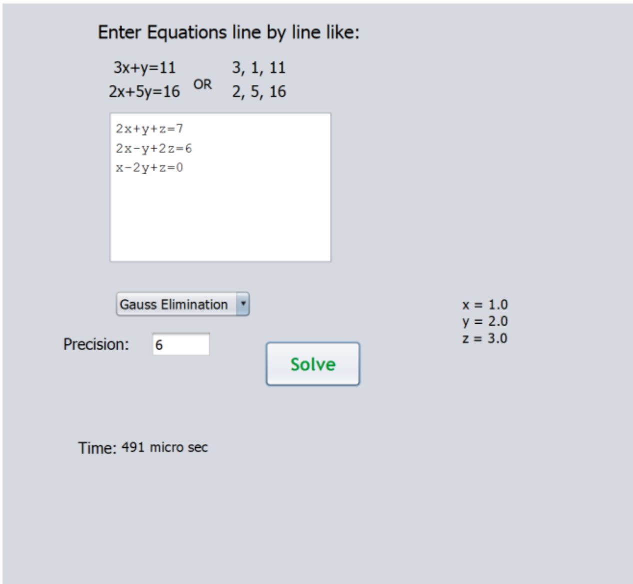
```
GaussSeidel(A, Y, initialGuess, iterations, epsilon) {
        n = initialGuess.length
        result = initial Guess
        prev = result
        for (i = 0 to i < iterations) {
                prev = result
                for (row = 0 to row < n) {
                        sum = Y[row]
                        for (column = 0 to column < n) {
                                if (column != row)
                                        sum -= A[row][column] * result[column]
        }
                                result[row] = sum / A[row][row]
                }
                if (absolute relative error < epsilon)
                        break
        }
}
```

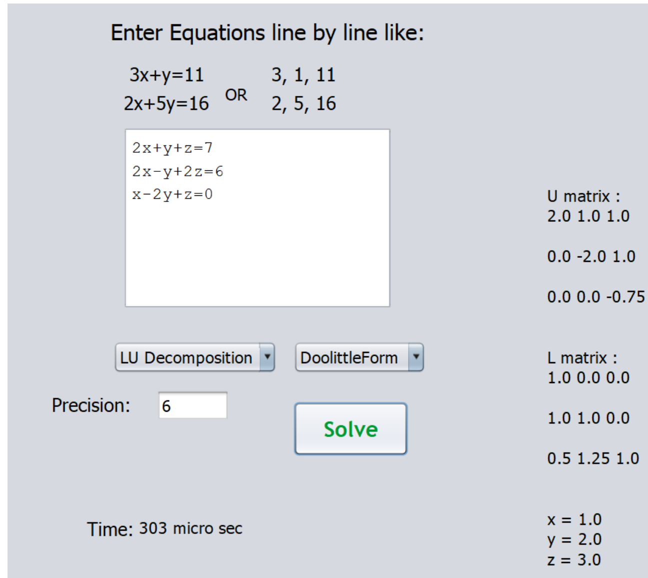
Jacobi

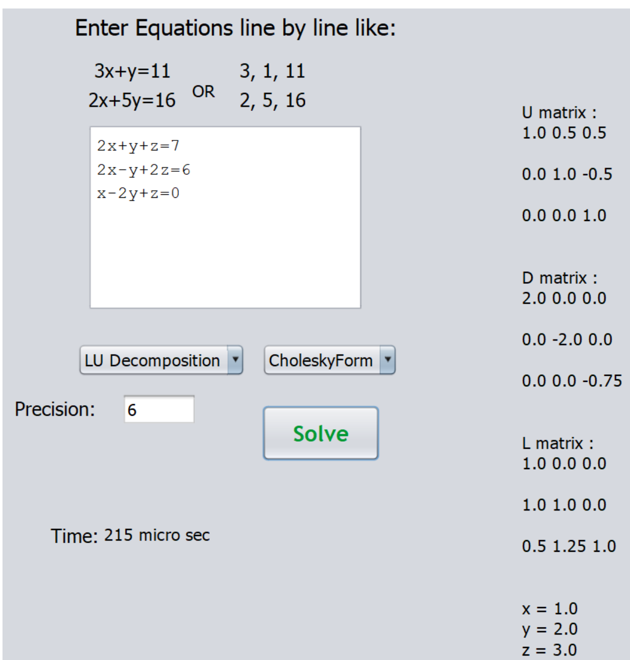
```
Jacobi (A, Y, initialGuess, iterations, epsilon) {

        n = initialGuess.length
        result = initial Guess
        prev = result
        for (i = 0 to i < iterations) {
                prev = result
                for (row = 0 to row < n) {
                        sum = Y[row]
                        for (column = 0 to column < n) {
                                if (column != row)
                                        sum -= A[row][column] * prev[column]                }
                                result[row] = sum / A[row][row]
                }
                if (absolute relative error < epsilon)
                        break
        }
}
```

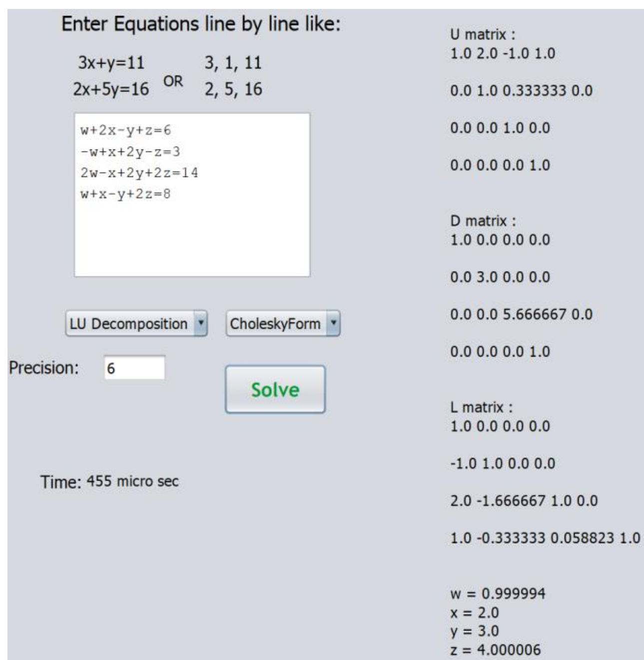
• Sample runs for each method

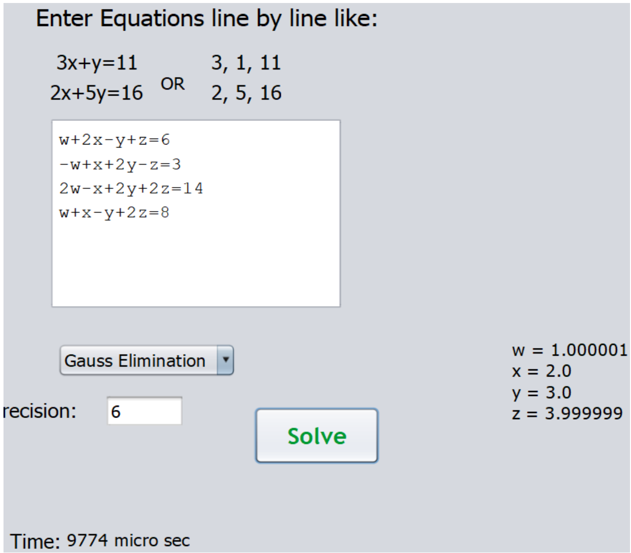
Normal samples

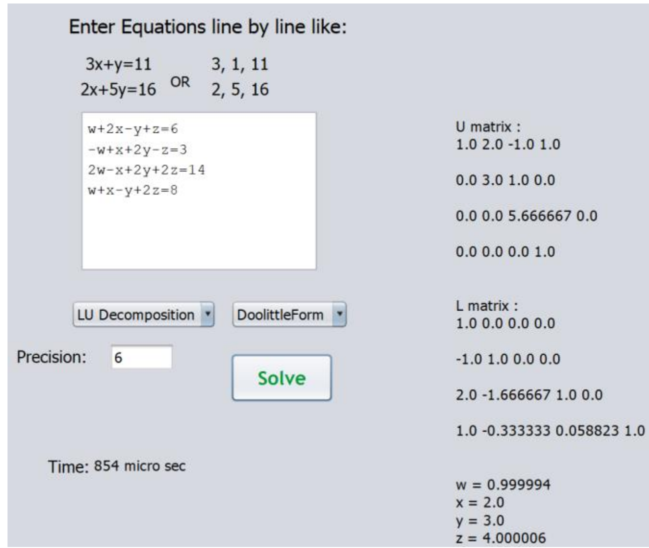


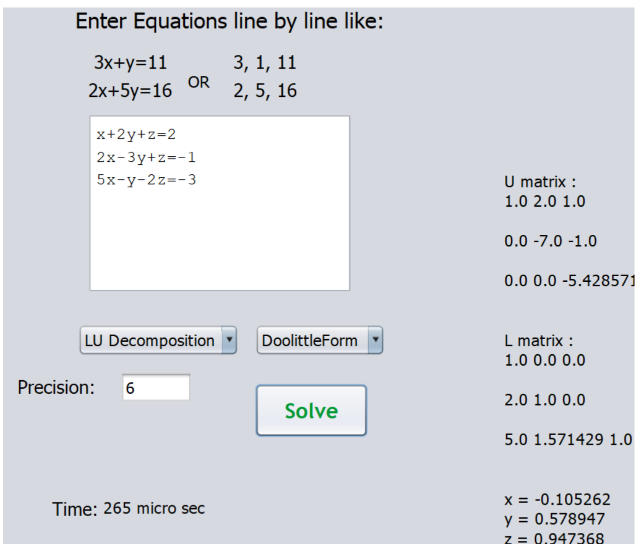


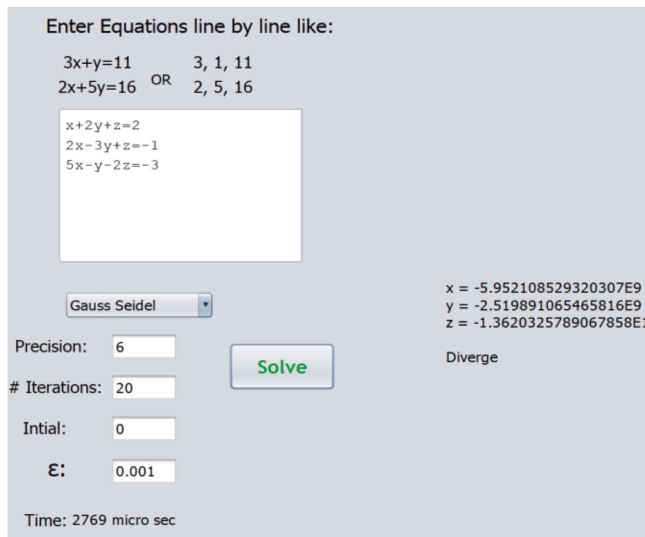




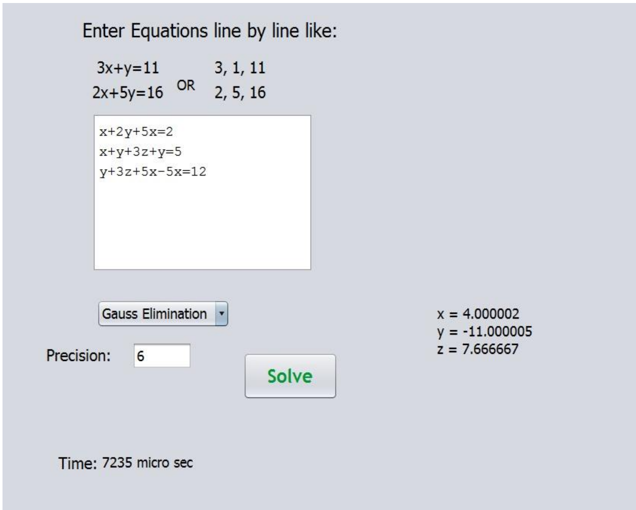

Enter Equations line by line like:
3x+y=11 3, 1, 11
OR
2x+5y=16 2, 5, 16
w+2x-y+z=6
-w+x+2y-z=3
2w-x+2y+2z=14
w+x-y+2z=8
Gauss Elimination
recision: 6
Solve
w = 1.000001
x = 2.0
y = 3.0
z = 3.999999
Time: 9774 micro sec

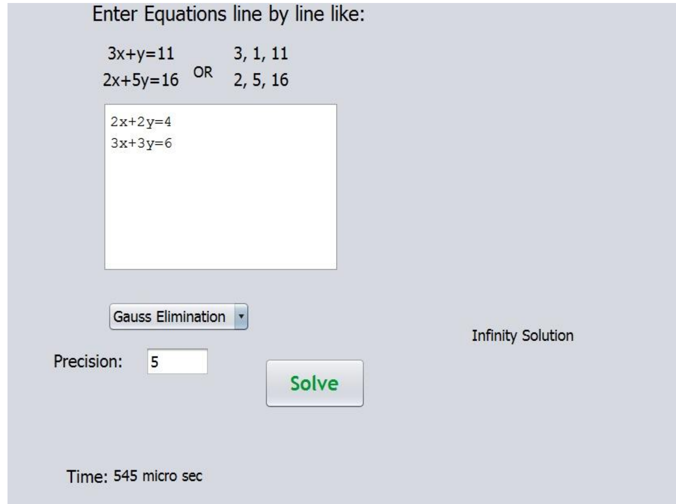

Enter Equations line by line like:
3x+y=11 3, 1, 11
OR
2x+5y=16 2, 5, 16
w+2x-y+z=6
-w+x+2y-z=3
2w-x+2y+2z=14
w+x-y+2z=8
U matrix :
1.0 2.0 -1.0 1.0
0.0 3.0 1.0 0.0
0.0 0.0 5.666667 0.0
0.0 0.0 0.0 1.0
LU Decomposition
DoolittleForm
L matrix :
1.0 0.0 0.0 0.0
-1.0 1.0 0.0 0.0
2.0 -1.666667 1.0 0.0
1.0 -0.333333 0.058823 1.0
Precision: 6
Solve
Time: 854 micro sec
w = 0.999994
x = 2.0
y = 3.0
z = 4.000006

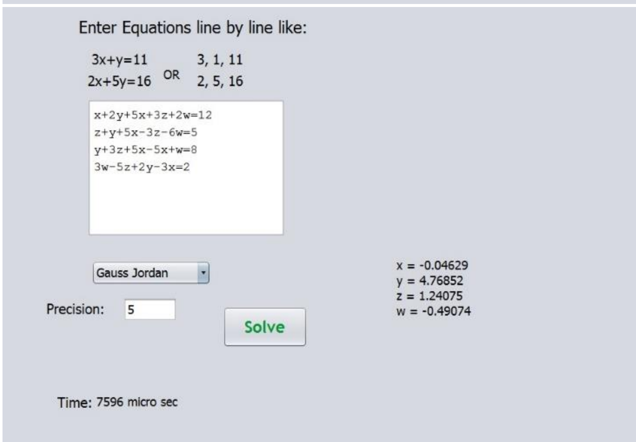

Enter Equations line by line like:
3x+y=11 3, 1, 11
OR
2x+5y=16 2, 5, 16
x+2y+z=2
2x-3y+z=-1
5x-y-2z=-3
U matrix :
1.0 2.0 1.0
0.0 -7.0 -1.0
0.0 0.0 -5.428571
LU Decomposition
DoolittleForm
L matrix :
1.0 0.0 0.0
2.0 1.0 0.0
5.0 1.571429 1.0
Precision: 6
Solve
Time: 265 micro sec
x = -0.105262
y = 0.578947
z = 0.947368

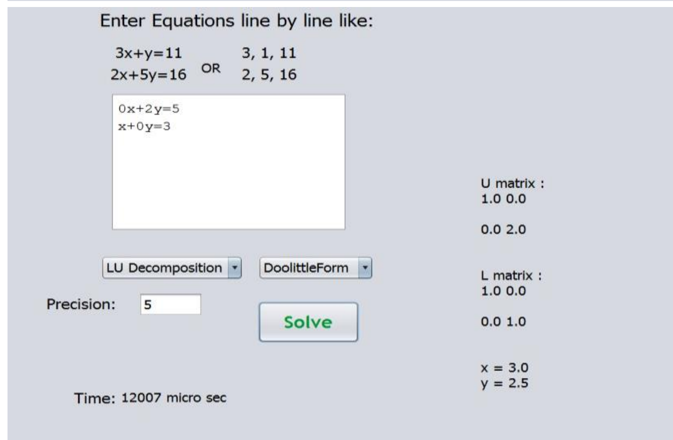

Enter Equations line by line like:
3x+y=11 3, 1, 11
OR
2x+5y=16 2, 5, 16
x+2y+z=2
2x-3y+z=-1
5x-y-2z=-3
x = -5.952108529320307E9
y = -2.519891065465816E9
z = -1.3620325789067858E1
Gauss Seidel
Precision: 6
Solve
Diverge
Iterations: 20
Intial: 0
ε: 0.001
Time: 2769 micro sec

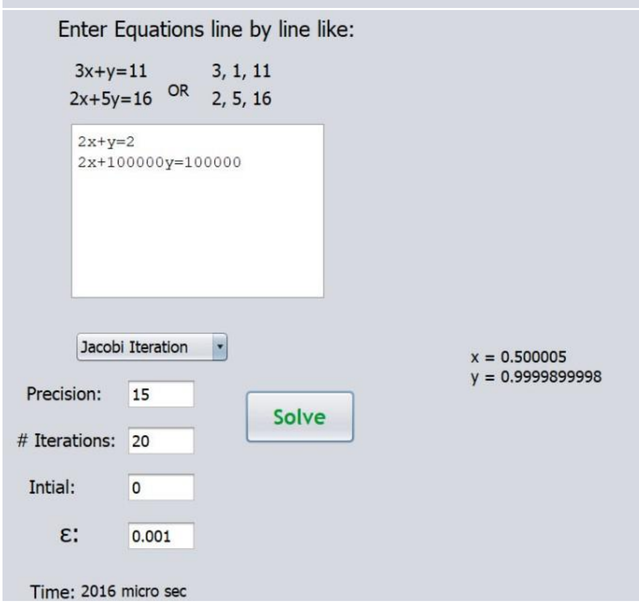
Tricky samples

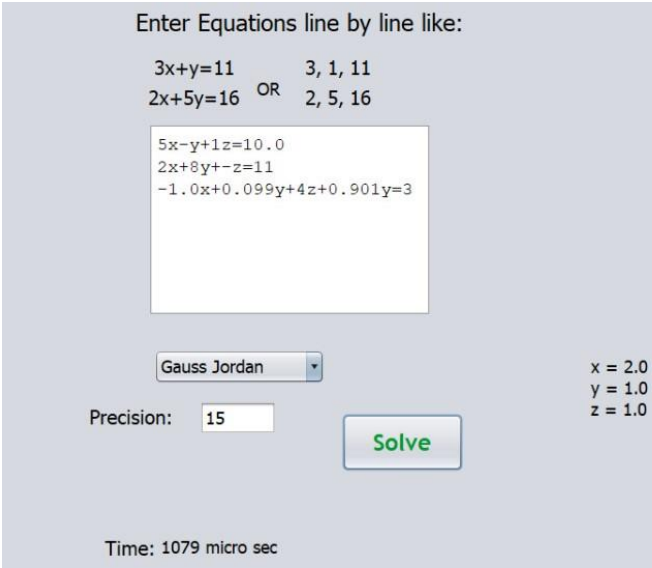


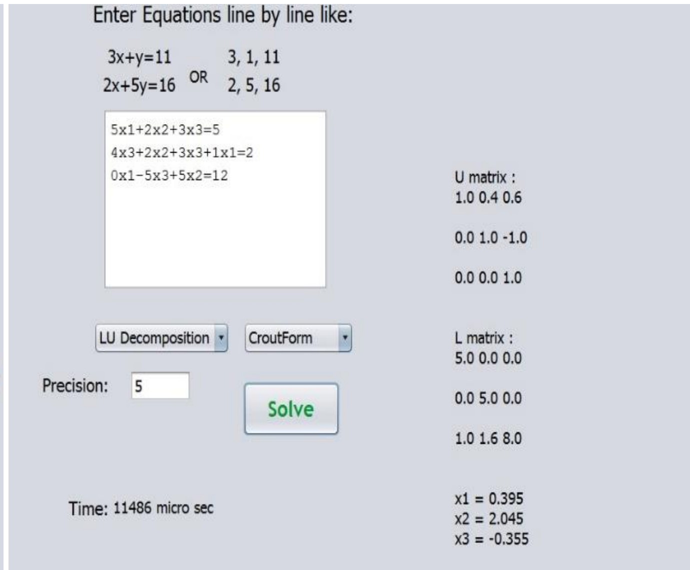


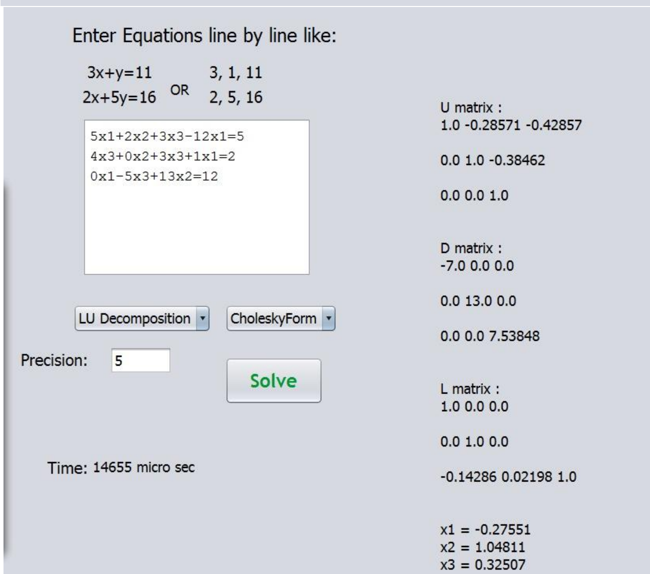


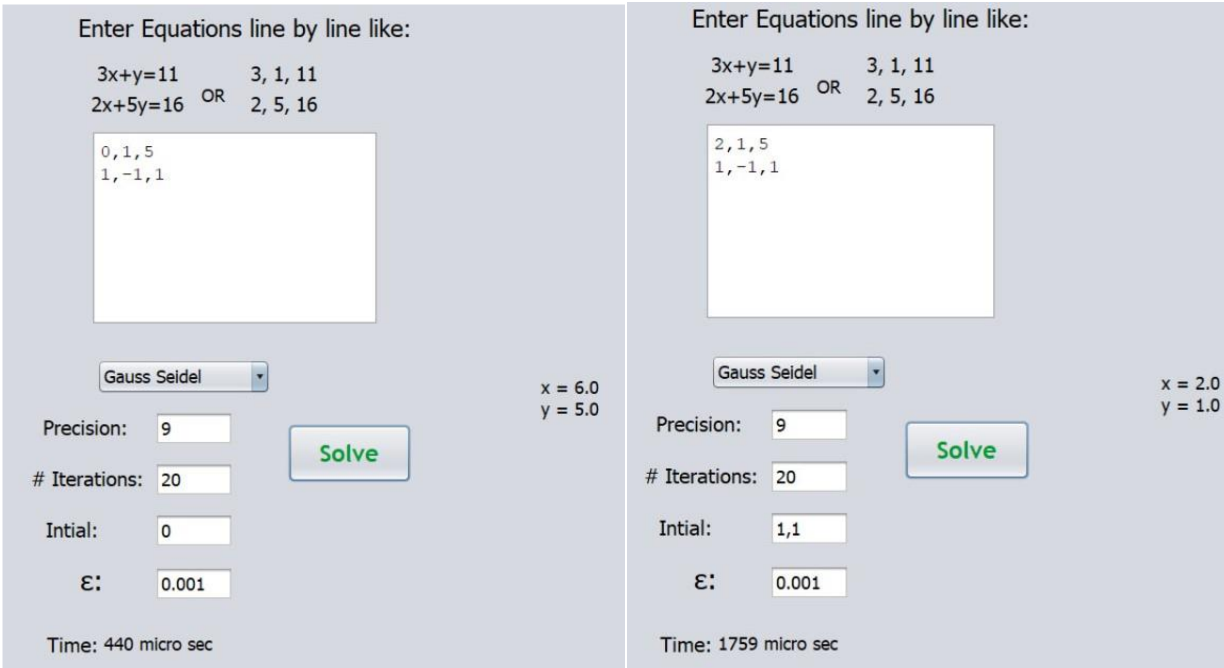





Enter Equations line by line like:
3x+y=11
2x+5y=16
OR
3, 1, 11
2, 5, 16
5x-y+1z=10.0
2x+8y+-z=11
-1.0x+0.099y+4z+0.901y=3
Gauss Jordan
Precision:
15
Solve
x = 2.0
y = 1.0
z = 1.0
Time: 1079 micro sec


Enter Equations line by line like:
3x+y=11
2x+5y=16
OR
3, 1, 11
2, 5, 16
5x1+2x2+3x3=5
4x3+2x2+3x3+1x1=2
0x1-5x3+5x2=12
U matrix :
1.0 0.4 0.6
0.0 1.0 -1.0
0.0 0.0 1.0
LU Decomposition
CroutForm
L matrix :
5.0 0.0 0.0
0.0 5.0 0.0
1.0 1.6 8.0
Precision:
5
Solve
Time: 11486 micro sec
x1 = 0.395
x2 = 2.045
x3 = -0.355


Enter Equations line by line like:
3x+y=11
2x+5y=16
OR
3, 1, 11
2, 5, 16
5x1+2x2+3x3-12x1=5
4x3+0x2+3x3+1x1=2
0x1-5x3+13x2=12
U matrix :
1.0 -0.28571 -0.42857
0.0 1.0 -0.38462
0.0 0.0 1.0
D matrix :
-7.0 0.0 0.0
0.0 13.0 0.0
0.0 0.0 7.53848
LU Decomposition
CholeskyForm
Precision:
5
Solve
L matrix :
1.0 0.0 0.0
0.0 1.0 0.0
-0.14286 0.02198 1.0
Time: 14655 micro sec
x1 = -0.27551
x2 = 1.04811
x3 = 0.32507



- ## Some features

The code achieved one of optional steps, our program saved the whole steps in a text file called "steps Of Solutions" that show the user the whole steps in all methods in arranged way.

The code handles the signs in our code to explain "–x = x", "--y= y" and so on.

The code handles if the variable appears in the equation more than one time to explain "2x+y-3x-5z" our code treats it as "-x+y-5z" and so on.

The code handles the variables name we can put any names of the variables and our code treats with it in normal way to explain the user can put the names of variables "x,y,z" or "x1,y1,z1" or and combinations of variables and with any order to explain the user can put "x+y=5 && y-x=1" for example.
Scaling is applied before each partial pivoting step.
The code supports LU decomposition in Cholskey form.

• Comparison between different methods

we will compare between methods with 3 systems of equations

first system: “2x+y+z=7, 2x-y+2z=6, x-2y+z=0”

second system: “w+2x-y+z=6, -w+x+2y-z=3, 2w-x+2y+2z=14, w+x-y+2z=8”

third system: “x+2y+z=2, 2x-3y+z=-1, 5x-y-2z=-3”

HINT: we try a method 10 times and take the average off the time in microseconds to be more accurate because the time effects with the memory state and the processor.

**The number of iterations: 20 and precision: 6.

College	1ST System	2nd System	3rd System
	Average Time(µs)	*Average Time(µs)*	*Average Time(µs)*
Gauss Elimination	488	848	512
Gauss Jordan Elimination	593	1028	627
LU Doolittle Form	253	463	294
LU Croute Form	219	539	318
LU Cholesky Form	290	487	336
Gauss Seidel	3028(Diverge)	4928(Diverge)	2769(Diverge)
Jacobi	4260(Diverge)	5206(Diverge)	4072(Diverge)

Comparison between different methods (time complexity, convergence, best and approximate errors)

• Data structure used

ArrayList : to store given equations
EQN: which is arraylist of equation members.
equation members: in which each coefficient is attached to its variable
for example: 2x+3y = 5
EQN --> [(2,x) , (3,y) , (5,null)]
equation members --> (2,x) , (3,y) , (5, null)