# *Numerical System Project*


| | |
|---|---|
| حسن أيمن عبد المنعم أحمد النجار | 19015566 |
| حسن علي حسن إبراهيم عبد الواحد | 19015603 |
| عبد الرحمن إبراهيم عبد الحليم سعد | 19015880 |
| عمرو عبد الحميد محمد إبراهيم | 19016112 |

# Gauss Elimination:

## Pseudocode

**Pseudocode to perform Forward Elimination:**

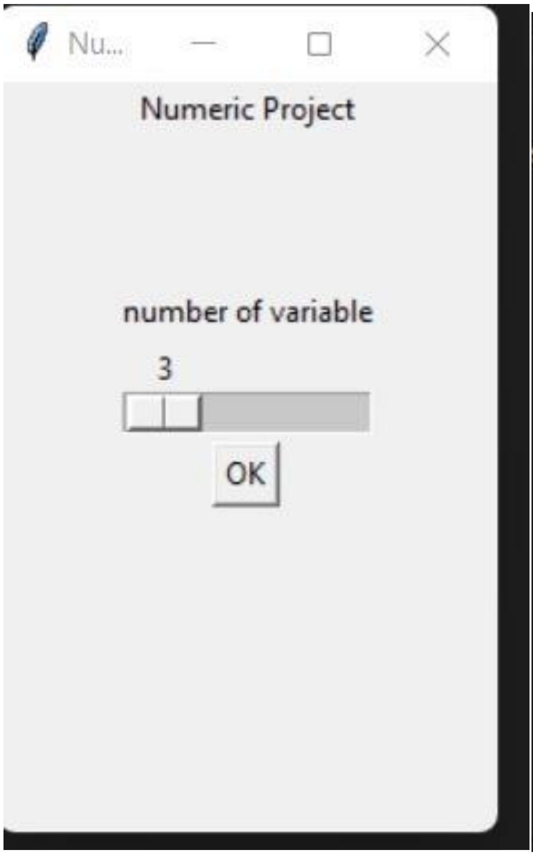
```
for k = 1 to n-1
 for i = k+1 to n
        factor = aik / akk
 for j = k+1 to n
        aij = aij – factor * akj
 bi = bi – factor * bk
```

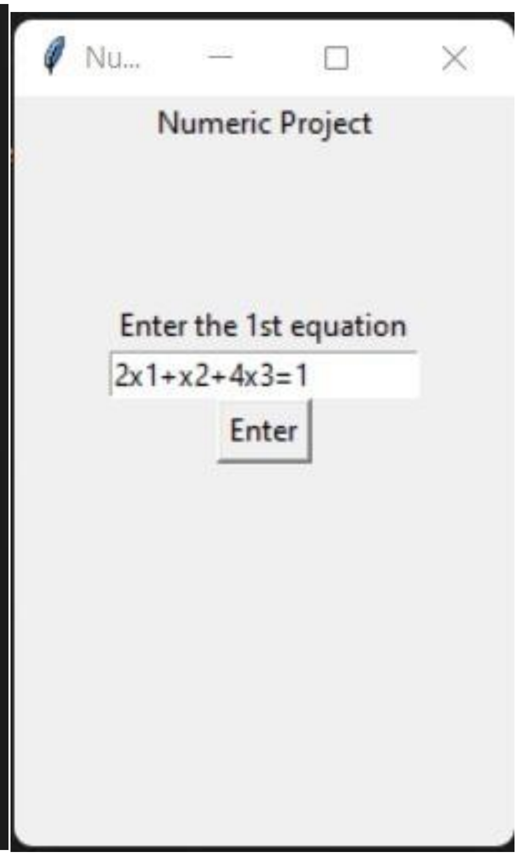
**Pseudocode to perform Back Substitution:**

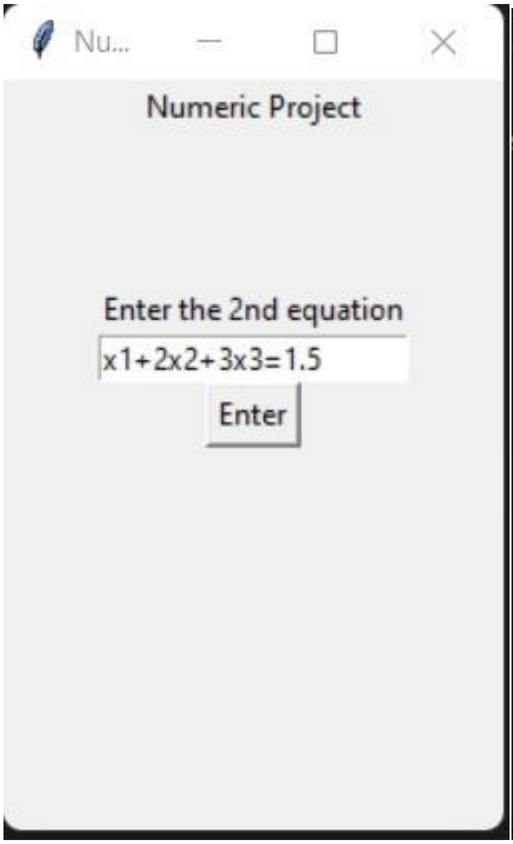
```
xn = bn / ann
for i = n-1 downto 1
 sum = 0
 for j = i+1 to n
        sum = sum + aij * xj
 xi = (bi – sum) / aii
```

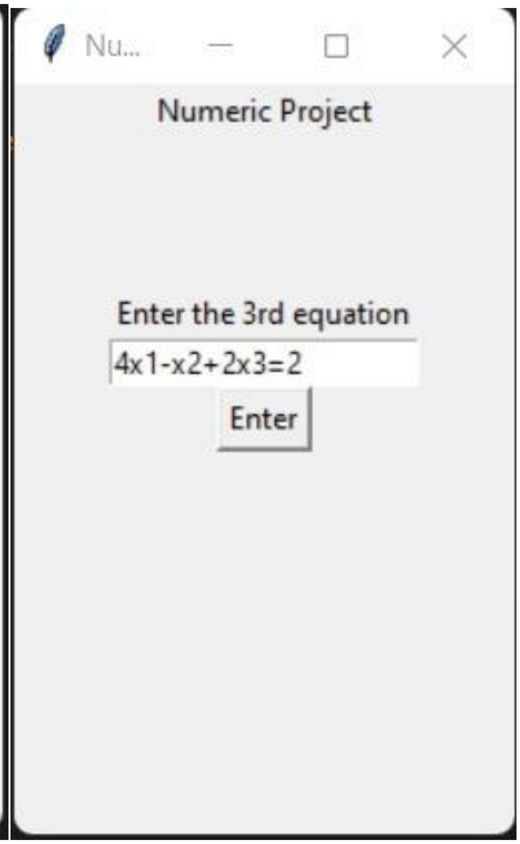
Total Cost : $2n^3/3 + O(n^2)$

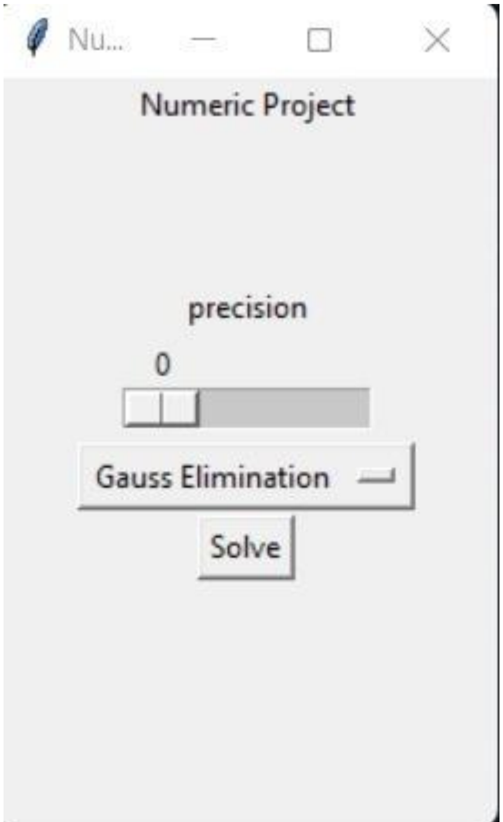
Runs:


Nu...
Numeric Project
number of variable
3
OK

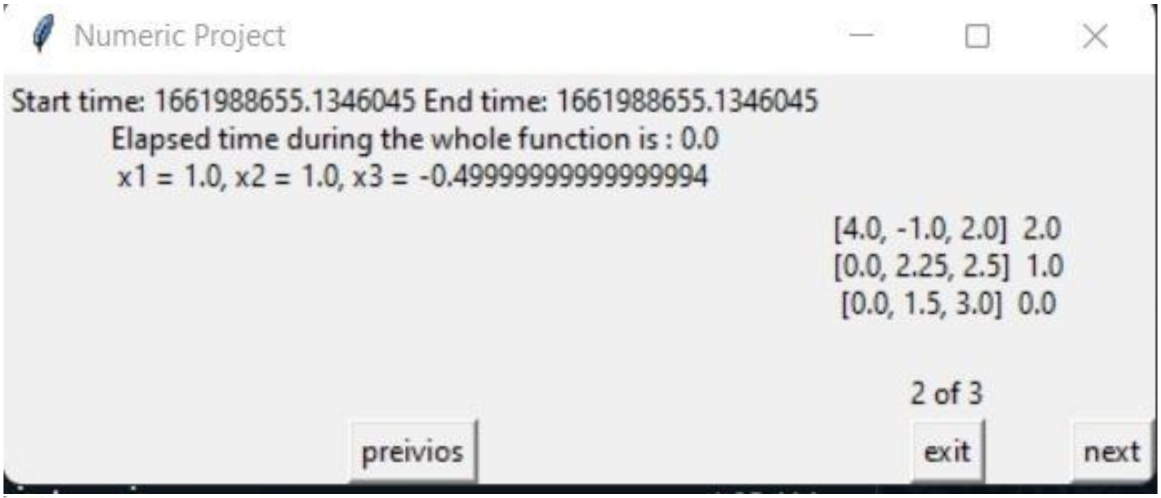

Nu...
Numeric Project
Enter the 1st equation
2x1+x2+4x3=1
Enter


Nu...
Numeric Project
Enter the 2nd equation
x1+2x2+3x3=1.5
Enter


Nu...
Numeric Project
Enter the 3rd equation
4x1-x2+2x3=2
Enter


Nu...
Numeric Project
precision
0
Gauss Elimination
Solve


Numeric Project
Start time: 1661988655.1346045 End time: 1661988655.1346045
Elapsed time during the whole function is : 0.0
x1 = 1.0, x2 = 1.0, x3 = -0.49999999999999994
[2.0, 1.0, 4.0] 2.0
[1.0, 2.0, 3.0] 1.0
[4.0, -1.0, 2.0] -0.6666666666666666
1 of 3
exit
next


Numeric Project
Start time: 1661988655.1346045 End time: 1661988655.1346045
Elapsed time during the whole function is : 0.0
x1 = 1.0, x2 = 1.0, x3 = -0.49999999999999994
[4.0, -1.0, 2.0] 2.0
[0.0, 2.25, 2.5] 1.0
[0.0, 1.5, 3.0] 0.0
2 of 3
preivios
exit
next


Numeric Project
Start time: 1661988655.1346045 End time: 1661988655.1346045
Elapsed time during the whole function is : 0.0
x1 = 1.0, x2 = 1.0, x3 = -0.49999999999999994
[4.0, -1.0, 2.0] 2.0
[0.0, 2.25, 2.5] 1.0
[0.0, 0.0, 1.3333333333333335] -0.6666666666666666
3 of 3
preivios
exit

# guess-jordan:

## Pseudocode

Apply Gauss Jordan Elimination on Matrix A:

```
For i = 1 to n
        If Ai,i = 0
            Print "Mathematical Error!"
            Stop
        End If
        For j = 1 to n
            If i ≠ j
                Ratio = Aj,i/Ai,i
                For k = 1 to n+1
                    Aj,k = Aj,k - Ratio * Ai,k
                Next k
            End If
        Next j
    Next i
```

Obtaining Solution:

```
For i = 1 to n
        Xi = Ai,n+1/Ai,i
    Next i
```

Total-Cost: $4n^3/3$

# Pseudocode for LU Decomposition

```
LUDecomp(a, b, n, x, tol, er) {

     Declare s[n] // An n-element array for storing
     scaling factors
     Declare o[n] // Use as indexes to pivot rows.
     // oi or o(i) stores row number of the ith pivot
     row.
     er = 0
     Decompose(a, n, tol, o, s, er)
     if (er != -1)
     Substitute(a, o, n, b, x)
}

Decompose(a, n, tol, o, s, er) {
     for i = 1 to n { // Find scaling factors
          o[i] = i
          s[i] = abs(a[i,1])
          for j = 2 to n
               if (abs(a[i,j]) > s[i])
                    s[i] = abs(a[i,j])
     }
     for k = 1 to n-1 {
          Pivot(a, o, s, n, k) // Locate the kth pivot
          row
          // Check for singular or near-singular cases
          if (abs(a[o[k],k]) / s[o[k]]) < tol) {
               er = -1
               return
          }

     for i = k+1 to n {
     factor = a[o[i],k] / a[o[k],k]
     // Instead of storing the factors
     // in another matrix (2D array) L,
     // We reuse the space in A to store
     // the coefficients of L.
     a[o[i],k] = factor
```

```
    // Eliminate the coefficients at column j
    // in the subsequent rows
    for j = k+1 to n
        a[o[i],j] = a[o[i],j] - factor * a[o[k],j]
        }
        } // end of "for k" loop from previous page
        // Check for singular or near-singular cases
        if (abs(a[o[n],n]) / s[o[n]]) < tol)
            er = -1
}
```

# Psuedocode for finding the pivot

```
Pivot(a, o, s, n, k) {
    // Find the largest scaled coefficient in column k
    p = k // p is the index to the pivot row
    big = abs(a[o[k],k]) / s[o[k]])
    for i = k+1 to n {
        dummy = abs(a[o[i],k] / s[o(i)])
        if (dummy > big) {
            big = dummy
            p = i
        }
    }
    // Swap row k with the pivot row by swapping the
    // indexes. The actual rows remain unchanged
    dummy = o[p]
    o[p] = o[k]
    o[k] = dummy
}
```

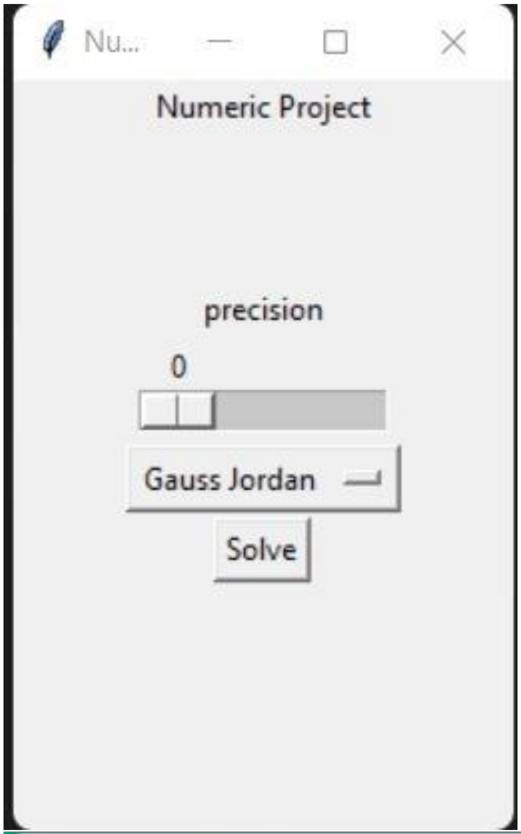
# Psuedocode for solving LUx = b

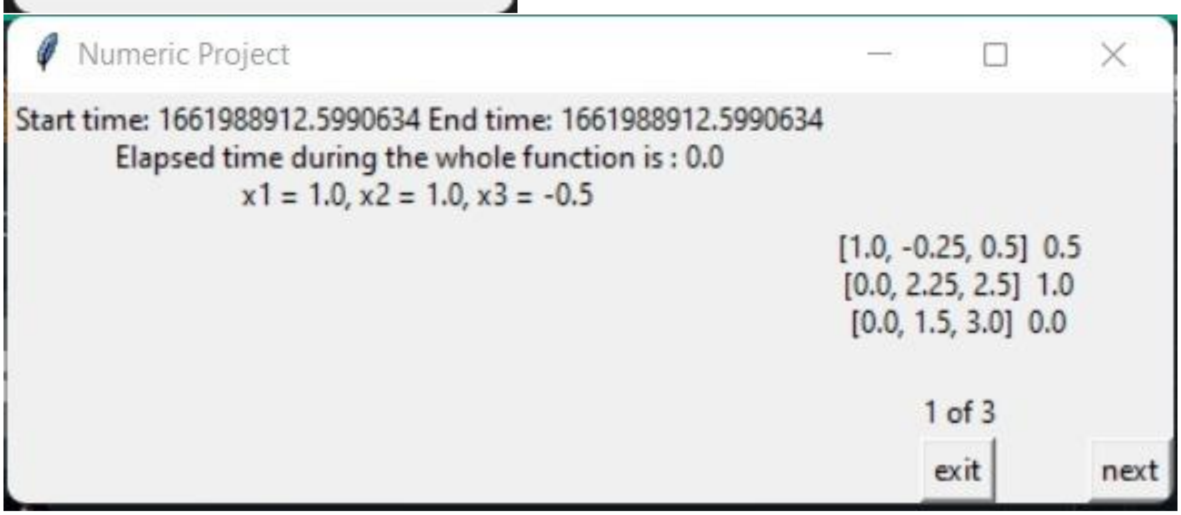
```
Substitute(a, o, n, b, x) {

    Declare y[n]
    y[o[1]] = b[o[1]]
    for i = 2 to n {
    sum = b[o[i]]
    for j = 1 to i-1
```

```
    sum = sum - a[o[i],j] * b[o[j]]
    y[o[i]] = sum
    }
    x[n] = y[o[n]] / a[o[n],n]
    for i = n-1 downto 1 {

        sum = 0
        for j = i+1 to n

            sum = sum + a[o[i],j] * x[j]
            x[i] = (y[o[i]] - sum) / a[o[i],i]

            }

}
```

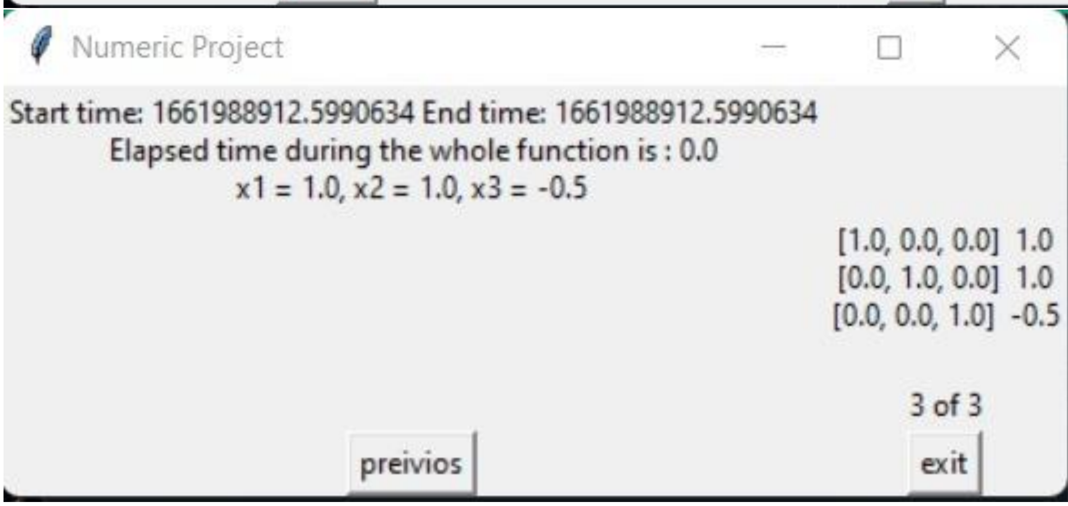
Total cost = O($n^3$) + $K$ * O($n^2$)

**Runs :**


Nu...
Numeric Project
precision
0
Gauss Jordan
Solve


Numeric Project
Start time: 1661988912.5990634 End time: 1661988912.5990634
Elapsed time during the whole function is : 0.0
x1 = 1.0, x2 = 1.0, x3 = -0.5
[1.0, -0.25, 0.5] 0.5
[0.0, 2.25, 2.5] 1.0
[0.0, 1.5, 3.0] 0.0
1 of 3
exit
next


Numeric Project
Start time: 1661988912.5990634 End time: 1661988912.5990634
Elapsed time during the whole function is : 0.0
x1 = 1.0, x2 = 1.0, x3 = -0.5
[1.0, 0.0, 0.7777777777777778] 0.6111111111111112
[0.0, 1.0, 1.1111111111111112] 0.4444444444444444
[0.0, 0.0, 1.3333333333333333] -0.6666666666666666
2 of 3
preivios
exit
next

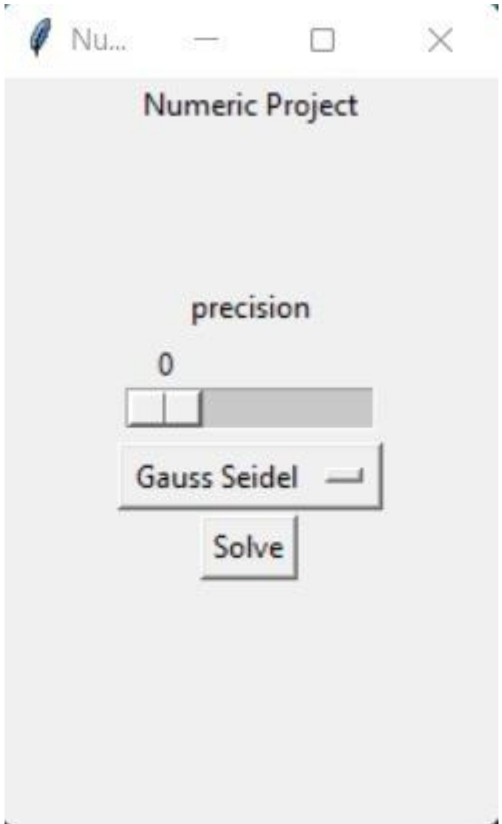

Numeric Project
Start time: 1661988912.5990634 End time: 1661988912.5990634
Elapsed time during the whole function is : 0.0
x1 = 1.0, x2 = 1.0, x3 = -0.5
[1.0, 0.0, 0.0] 1.0
[0.0, 1.0, 0.0] 1.0
[0.0, 0.0, 1.0] -0.5
3 of 3
preivios
exit

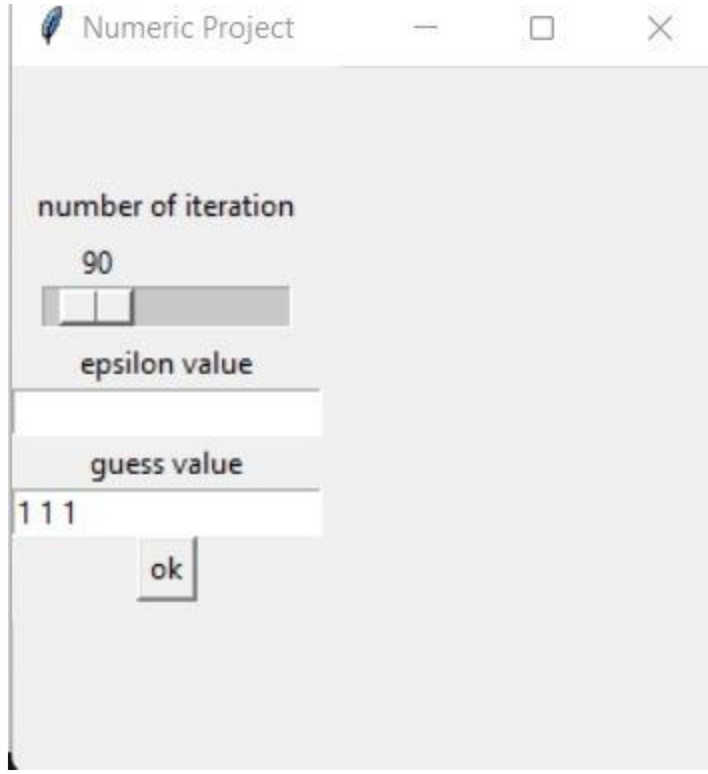
# PseudoCode for Guess Sediel:

```
Inputs: A, b
Output: φ

Choose an initial guess φ to the solution
repeat until convergence
    for i from 1 until n do
        σ ← 0
        for j from 1 until n do
            if j ≠ i then
                σ ← σ + a_ij φ_j
            end if
        end (j-loop)
        φ_i ← (1/a_ii)(b_i − σ)
    end (i-loop)
    check if convergence is reached
end (repeat)
```

**Total Cost : O(n^2)**

**Runs:**


Nu...
Numeric Project
precision
0
Gauss Seidel
Solve


Numeric Project
number of iteration
90
epsilon value
guess value
1 1 1
ok

# PseudoCode for Jacobi Iterative:

**Step 1.** Read the coefficients aij, i,j = 1, 2, …, n and the right hand vector bi, i= 1, 2, …, n of the system of equations and error tolerance ϵ.
**Step 2.** Rearrange the given equations, if possible, such that the system becomes diagonally dominant.
**Step 3.** Rewrite the ith equation as

$$x_i = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1, j\neq i}^{n} a_{ij}x_j\right) for\ i = 1, 2, \ldots, n$$

**Step 4.** Set the initial solution as

$$x_i = 0,\ i = 1, 2, \ldots, n$$

**Step 5.** Calculate the new value $x_i^{(n)}$ of $x_i$ as

$$x_i{}^{(n)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1, j\neq i}^{n} a_{ij}x_j\right) for\ i = 1, 2, \ldots, n$$

**Step 6.** If $| x_i – x_i^{(n)}| < \epsilon$ for all i, then goto Step 7 else $x_i = x_i^{(n)}$ for all i and goto step 5.
**Step 7.** Print $x_i^{(n)}$ , i = 1, 2, …, n as solution.

**Total Cost : O(n^2)**

## Runs:


# Data structure used:

We used Just Arrays

Figures for gui plotting