Regula Falsi-False position Method:

PSEUDOCODE:

```
1. Start
2. Define function f(x)
3. Input
        a. Lower and Upper guesses x0 and x1
        b. tolerable error e
4. If f(x0)*f(x1) > 0
        print "Incorrect initial guesses"
        goto 3
  End If
 5. Do
       x2 = x0 - ((x0-x1) * f(x0))/(f(x0) - f(x1))
        If f(x0)*f(x2) < 0
                x1 = x2
        Else
                x0 = x2
        End If

  While abs(f(x2) > e
  6. Print root as x2
7. Stop
```

Algorithm:

1. start

2. Define function f(x)

3. Choose initial guesses x0 and x1 such that f(x0)f(x1) < 0

4. Choose pre-specified tolerable error e.

5. Calculate new approximated root as:

   x2 = x0 - ((x0-x1) * f(x0))/(f(x0) - f(x1))

6. Calculate f(x0)f(x2)
   a. if f(x0)f(x2) < 0 then x0 = x0 and x1 = x2
   b. if f(x0)f(x2) > 0 then x0 = x2 and x1 = x1
   c. if f(x0)f(x2) = 0 then goto (8)

7. if |f(x2)|>e then goto (5) otherwise goto (8)

8. Display x2 as root.

9. Stop

Secant Method

Pseudocode:

```
1. Start
2. Define function as f(x)
3. Input:
     a. Initial guess x0, x1
     b. Tolerable Error e
     c. Maximum Iteration N
4. Initialize iteration counter step = 1
5. Do
     If f(x0) = f(x1)
          Print "Mathematical Error"
          Stop
      End If
   x2 = x1 - (x1 - x0) * f(x1) / ( f(x1) - f(x0) )
     x0 = x1
     x1 = x2
     step = step + 1
     If step > N
          Print "Not Convergent"
          Stop
```

End If

While abs f(x2) > e

6. Print root as x2

7. Stop

Algorithm:

1. Start

2. Define function as f(x)

3. Input initial guesses (x0 and x1),

   tolerable error (e) and maximum iteration (N)

4. Initialize iteration counter i = 1

5. If f(x0) = f(x1) then print "Mathematical Error"

   and goto (11) otherwise goto (6)

6. Calcualte x2 = x1 - (x1-x0) * f(x1) / (f(x1) - f(x0))

7. Increment iteration counter i = i + 1

8. If i >= N then print "Not Convergent"

   and goto (11) otherwise goto (9)

9. If |f(x2)| > e then set x0 = x1, x1 = x2

   and goto (5) otherwise goto (10)

10. Print root as x2

11. Stop

Gauss Jordan Method:

Pseudocode:

```
1. Start
2. Input the Augmented Coefficients Matrix (A):
        For i = 1 to n
                For j = 1 to n+1
                        Read Ai,j
                Next j
        Next i
3. 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
```

4. Obtaining Solution:

For i = 1 to n

Xi = Ai,n+1/Ai,i

Next i

5. Display Solution:

For i = 1 to n

Print Xi

Next i

6. Stop

Note: All array indexes are assumed to start from 1.

Algorithm:

1. Start

2. Input the number of unknowns, n

3. Input coefficients of augmented matrix as array, M[i][j]

4. For i = 1 to n

    If diagonal element, M[i][i]=0

        Print "Enter valid coefficients of augmented matrix!"

    End if

5. Do row operations to change the matrix to diagonal matrix

        For j = 1 to n

            If i $\neq$ j

                Ratio = M[j][i]/M[i][i]

                For k = 1 to n+1

                M[j][k] = M[j][k] - Ratio * M[i][k]

        End If

6. Print required solution

    For i = 1 to n

        Print X[i]

7. Stop

Gauss Jordan Method For Finding Inverse:

Pseudocode:

1. Start
2. Read Order of Matrix (n).
3. Read Matrix (A) of Order (n).
4. Augment and Identity Matrix of Order n to Matrix A.
5. Apply Gauss Jordan Elimination on Augmented Matrix (A).
6. Perform Row Operations to Convert the Principal Diagonal to 1.
7. Display the Inverse Matrix.
8. Stop.

Algorithm:

1. Start
2. Input order of square Matrix,n.
3. Input Matrix M:

```
For i = 1 to n
  For j = 1 to n
    Read M[i][j]
```

4. Augment Identity Matrix of Order n to Matrix M:

```
For i = 1 to n
  For j = 1 to n
     If i = j
      M[i][j+n] = 1
    Else
```

M[i][j+n] = 0

End If

5. Apply Gauss Jordan Elimination on Augmented Matrix M:

For i = 1 to n

If diagonal element, M[i][i]=0

Print “Enter valid coefficients of augmented matrix!”

End if

6. Do row operations to change the matrix to diagonal matrix

For j = 1 to n

If i ≠ j

Ratio = M[j][i]/M[i][i]

For k = 1 to 2*n

M[j][k] = M[j][k] - Ratio * M[i][k]

End If

7. Row Operation to Convert Principal Diagonal to 1.

For i = 1 to n

For j = n+1 to 2*n

M[i][j] = M[i][j]/M[i][i]

8. Display Inverse Matrix:

For i = 1 to n

For j = n+1 to 2*n

Print M[i][j]

9. Stop

Gauss Elimination Method:

Pseudocode:

1. Start
2. Read Number of Unknowns: n
3. Read Augmented Matrix (A) of n by n+1 Size
4. Transform Augmented Matrix (A)

   to Upper Triangular Matrix by Row Operations.
5. Obtain Solution by Back Substitution.
6. Display Result.
7. Stop

Algorithm:

1. Start
2. Input order of square Matrix,n.
3. Input the Coefficients of augmented Matrix M:

```
        For i = 1 to n
                For j = 1 to n+1
                        Read M[i][j]
```

4. Apply Gauss Elimination on Matrix M:

```
        For i = 1 to n-1 If diagonal element, M[i][i]=0
                Print "Enter valid coefficients of augmented matrix!"
    End if
                For j = i+1 to n
```

Ratio = M[j][i]/M[i][i]

for k = 1 to n+1

M[j][k] = M[j][k]- Ratio *M[i][k]

4. Back Substitution:

Xn = M[n][n+1]/M[n][n]

For i = n-1 to 1 (Step: -1)

Xi = M[i][n-1]

For j = i+1 to n

Xi = Xi –M[i][j]* Xj

Xi = Xi/ M[i][i]

5. Display Solution:

For i = 1 to n

Print Xi

6. Stop