```
%% This is a Pseudo Code on how the entire FEM 2D 1DOF code↙
works !!!

%% Main Script - FEM2D.m

Define all inputs constants
Define all Essential and Natural Boundary Conditions
Define element length along x and y direction
Define Guess Vector and initialize previous iteration solution↙
vector (GPU)

% Calculate NOD (connectivity Matrix) and GLXY (global coord of↙
all nodes)
Call "MeshR" function to define NOD and GLXY

Initiate iteration

while (iter is less than ITMAX) && (convergence not achieved)
    Initialize Global stiffness and force matrix
    for (iterate for all elements)
        if (applying for Direct ,Newtons's iteration)
            Define ELU for particular element from last and↙
last-to-last iteration
            Define ELXY (global coords of all element node)↙
from GLXY
        end

        % Calculate Elemental Stiffness (ELK) and force matrix↙
(ELF)
        Call "ELEMATRC2D" function to calculate ELK & ELF

        Assemble ELK and ELF into GLK and GLF using NOD matrix
    end

    % Apply Essential and Natural Boundary Condition on GLK and↙
```

```
GLF

    Call "BNDRYUNSYM" function to calculate updated GLK and GLF

    if NONLIN == 1 % that is for Direct iteration
        Update previous iteration solution (GPU)
        Calculate current iteration solution (GCU)
    elseif NONLIN == 2 % that is for Newton's iteration
        Update previous iteration solution
        Calculate current iteration solution by adding GLK\GLF↙
with previous solution
    end

    % Note for Newton's iteration,
    % One should apply the essential BCs for the very first↙
iteration.
    % Afterwards VSPV = 0

    if iter == 1 % after applying EBCs for 1st iteration,↙
change value of VSPV.
        if NONLIN > 1 % For Newton's Iteration only
            all elements in VSPV = 0
        end
    end

    Calculate Error using the formula for this iteration

    if error<epsilon
        that means it converged.
        get out of while loop
    end
end % end of while loop

if convergence ~= 1
    that means, iter > ITMAX.
```

```
    Hence print "Error: Did Not Converge".
end

if convergence == 1 % that is it converged
    print final solution
end

%%↙
--------------------------------------------------------------↙
--%

%           MeshR
% Pseudo code for MeshR Function

Define important constants - NXX, NYY, NXX1, NYY1, NEM (total↙
number of elements), NNM (total number of nodes)

% Special Case
if NPE == 8 % Quadratic element with 8 nodes / element
    NNM = NNM - NEM;
end

% Special Case of 9 noded element
if NPE == 9
    K0 = 1;
else
    K0 = 0;
end

Defining 1st element node 1, 2 and 3. % NOD(1,1) ; NOD(1,2) &↙
NOD(1,3)

if NPE == 9 % Special case of 9 noded element
    Defining 3rd node of 9 noded element.
end
```

```
Define 4th node

if NPE > 4 % For 8 noded and 9 noded element
    Defining node 5,6,7,8
    if NPE == 9 % Special 9 noded element case
        Defining 9th node
    end
end

if NY > 1 % Case when we have multiple elements along y↙
direction
    Defining NOD matrix for all elements above the 1st element↙
(in y direction) using the NOD defined matrix for 1st element↙
end.

if NX > 1 % case when we have multiple elements along x↙
direction
    for NI  = 2:NX
        for I = 1:NPE
            Defining NOD matrix for all elements right to the↙
first element (ie, in x direction) using the NOD defined matrix↙
for 1st element end.
        end

        for NJ = 2:NY
            Defining NOD matrix for all elements using↙
previously defined elements
        end
    end
end

% Generating Global Coords for all the nodes

Increase size of DX and DY by 1 and add 0 as their last↙
```

```
element.

if NPE == 8 % Case of Quadratic 8 noded element case
    % Note - 1st element 1st node  == Global Node 1
    for NI = 1:NEY1
        Defining coords of nodes of NIth element
        for NJ = 1:NX
            Defining coords all nodes to the right of 1st nodal↙
element on x axis (only the first row of nodes)
        end
        if NI<=NY
            Defining coords of all nodes above 1st node on y↙
axis (only the first column of the nodes)
            for II = 1:NX
                Defining coords of all nodes in horizontal↙
fashion
            end
        end
    end
else % Case of 9 noded or 4 noded element
    for NI = 1:NEY1
        for NJ = 1:NEX1
            Defining coord of nodes of NIth element
            if NJ<NEX1
                if IEL == 2 % Special case of 9 noded element
                    Changing XC to include the middle 9th node
                end
            end
        end


%%↙
-----------------------------------------------------------------↙
--%
```

```
%                   ELEMATRC2D
% Pseudo code for ELEMATRC2D Function

Call "Gaus_int" function to get Gauss Points and Weights

Define Differential Equation Coefficients

Initialize ELK and ELF

if NONLIN > 1 % That is for Newton's Iteration
    Initialize Tangent Matrix
end

% Since 2D, there is 2 integral for the 2 perpendicular↙
directions
% Hence, we will implement Gaussian Quadrature twice

for NI = 1:NGPF % For 1st Gaussian Quadrature
    for NJ = 1:NGPF % For 2nd Gaussian Quadrature
        Call "INTERPLN2D" function to get interpolation↙
functions (SFL), their global derivative (GDSFL) and↙
determinant of Jacobian

        Define Gaussian Quadrature constant (CONST)

        Initialize X, Y, U, dU/dX, dU/dY

        for I = 1:NPE
            if NONLIN>0 % that is for Newton's and direct↙
iteration method
                Calculate U, dU/dX, dU/dY
                % U = sum(ELU*SFL) ; dU/dX = sum(ELU*d(SFL)↙
/dX); du/dY = sum(ELU,d(SFL)/dY)
            end
            Define local node coordinates.
```

```
        end
        Define A11, A22 and FXY

        for I = 1:NPE
            Define ELF
            %ELF = sum(SFL*FXY). % Multiply with CONST to↙
eliminate the need to integrate
            for J = 1:NPE
                Define ELK
            end
        end
        if NONLIN > 1 %If using Newton's Iteration
            Define Tangent Matrix (TANG)
        end
    end
end

% Compute Residual vector and finalize tangent matrix

if NONLIN > 1 % If using Newton's iteration
    for I = 1:NPE
        for J = 1:NPE
            Calculating Residual.
            Equating residual to ELF, since we will export ELF↙
from this function

            Adding calculated ELK with TANG to completely↙
calculate the tangent matrix
            Equating TANG to ELK since we will export ELK from↙
this function.
        end
    end
end

%%↙
```

```
-----------------------------------------------------------------↙
--%
%                INTERPLN2D

Define local coordinates of master element

if NPE == 4 % Considering the case of 4 noded element
    for I = 1:NPE
        Defining interpolation function
        Defining d(SFL)/dx in local coordinate
        Defining d(SFL)/dy in local coordinates
    end
elseif NPE == 8 % Case of 8 noded element
    for I = 1:NPE
        if I <= 4 % Subcase of the 4 nodes placed at the 4↙
vertices of the master element
            Defining SFL, DSFL wrt x and DSFL wrt y (local↙
coord)
        else
            if I <= 6 % For nodes 5 and 6
                Defining SFL, DSFL wrt x and DSFL wrt y (local↙
coord)
            else % For nodes 7 & 8
                Defining SFL, DSFL wrt x and DSFL wrt y (local↙
coord)
            end
        end
    end
elseif NPE == 9 % Case of 9 noded element
    for I = 1:NPE
        if I <= 4
            Defining SFL, DSFL wrt x and DSFL wrt y (local↙
coord)
        else
            if I <= 6 % For nodes 5 and 6
```

```
                Defining SFL, DSFL wrt x and DSFL wrt y (local↙
coord)
            else
                if I <= 8 % For nodes 7 & 8
                    Defining SFL, DSFL wrt x and DSFL wrt y↙
(local coord)
                else % For node 9
                    Defining SFL, DSFL wrt x and DSFL wrt y↙
(local coord)
                end
            end
        end
    end
end

% Compute Jacobian

Initializing Jacobian Matrix (GJ)
for I = 1:2
    for J = 1:2
        for K = 1:NPE
            Calculating Jacobian Matrix
        end
    end
end

Calculating GJ inverse
Calculating Determinant of Jacobian matrix

Initializing GDSFL Matrix
for I = 1:2
    for J = 1:NPE
        for K = 1:2
            Calculating GDSFL
        end
```

```
    end
end

%%↙
---------------------------------------------------------------↙
--%
%                BNDRYUNSYM

if NSPV ~= 0 % Check if there is any defined Essential BCs
    for NP = 1:NSPV % Looping for all the given EBCs
        Equating the row in GLK of the particular DOF = 0
        Equating the diagonal element of the row = 1
        Equating the element of GLF of corresponding DOF with↙
given EBC value

        % Note - This process is independent of the type of↙
solver used.
        % Newton' iteration will be satisfied in the main↙
script

    end
end

if NSSV ~= 0 % Check if there is any defineed Natural BCs
    for NS = 1:NSSV
        Adding NBCs value to GLF
    end
end

%%↙
---------------------------------------------------------------↙
--%
%                Gaus_int

Takes Number of Gaus Point as input
```

```
output is 1D array of all the Gauss Points and Gauss Weight for↙
the given number of gaus points
```