```
%% Non-Linear FEM - Assignment 4
% Pseudo Code

Input Variable

Call MESH2DR function to generate mesh, ie, define NOD and GLXY matrices

initialize following array matrices
% GPU - Current Time Step and Current Iteration solution
% GLU - Current Time Step and Previous Iteration solution
% GLP - Previous Time Step solution
% GLV - Velocity matrix
% GLA - Acceleration matrix


if (ITEM == 1) % if dealing with parabolic equation
    if NSSV > 0 % ie, there are natural BCs
        VSSV = VSSV*DT; %
    end
end

Define A1, A2, A3, A4, A5, A6, A7, A8

for NT = 1:NTime % Start Time loop
    initiate iter variable
    while (iter < ITERMAX) && (convergence == 0) % Start iterative loop
        initiate GLK and GLF

        for I = 1:NEM % Iterating for every element
            Defining ELU and ELU0

            if ITEM == 2 % Hyperbolic equation
                Defining 1st and 2nd derivative of ELU wrt time
```

```
(ELU1 and ELU2)
            else % Parabolic equation
                initializing ELU1 and ELU2
            end

            Defining ELXY % ie, defining element coords

            Calling ELEMATRCS2D_Time function to calculate ELK↙
and ELF

            Assembling GLF and GLK matrices
        end

        Calling BNDRYUNSYM_Time to apply Natural and Essential↙
BCs

        Calculating solution for the current iteration

        increasing iter counter

        if iter == 1 %If for the first iteration
            if NONLIN == 2 % for Newton's iteration only
                VSPV(:) = 0; % After applying the EBCs once, we↙
no longer have to apply the EBC in Newton's iteration
            end
        end

        Calculating Error

        Checking if error is less than epsilon

        Calculating Velocity and Acceleration arrays for next↙
time step

        Updating Previous time step solution (GLP)
```

```
    end
end

% ------------------------------------------------- %

%                        ELEMATRCS2D_Time
% Pseudo Code for ELEMATRCS2D_Time

Defining all the constants of the Differential equation

Initializing ELK, ELK0, ELM, ELF, Tangent matrix (if NONLIN ==↙
2)

Calling Gaus_int function to get the Gauss Points and Gauss↙
Weights

for I = 1:NGPF
    for J = 1:NGPF % Starting the loop for integral calculation
        %Calling INTERPLN2D function for Shape functions (SFL),↙
global derivatives of the shape function, and Jacobian

        Initialize x,y,u,dux,duy
        for I = 1:NPE
            Calculate x, y, u, dux, duy
        end

        Evaluate the functions of Differential equation -
        Fxy, Axx, Ayy, A00

        if ITEM > 0 % For time dependent DE
            evalute C function
            % C is C0 for Parabolic
            % C is C1 for Hyperbolic
        end
```

```
        if ITEM > 0 % For time dependent DT
            %Calculate u, dux, duy, Axx, Ayy, for previous time↙
step value
        end

        Define ELK, ELF, ELM
        % ELM is C matric for Parabolic DE
        % ELM is M matrix for Hyperbolic DE

        if NONLIN > 1 % For Newton's Iteration
            Calculate tangent matrix
        end

        if ITEM == 1 % Parabolic Equation
            Calculate K-Hat and F-Hat
            Equate K-Hat as ELK and F-Hat as ELF
        elseif ITEM >1 % Hyperbolic Equation
            Calculate K-Hat and F-Hat
            Equate K-Hat as ELK and F-Hat as ELF
        end
    end
end
```