pre_processing_code.m

```
%% Pre-processing Code
%  (Generalized code for a structured mesh)
%   Note: This code is applicable only to the rectangular plates whose all
%         edges are simply supported.
%%
% This code divides the rectangular plate in 'n' elements in a structured
% mesh. These elements can have or have not equal dimensions and these
% elements are rectangular with 'a' as dimension in the x-direction and 'b'
% in y-direction. Further, value of load is to updated in the code named
% 'F_el'.

clear all
clc
%% Element and plate geometry (dimensions in 'm')

% ft, fb and c are thickness of different layers of plate
ft=0.002; fb=0.002; c=0.008; h_tot=ft+fb+(2*c);

% 'x_a' and 'y_b' are the dimensions of plate
x_a=5*h_tot; y_b=x_a;

% nel is the no. of elements in that particular direction
disp('Please enter an even number of elements in y-direction.')
nel_x=input('   # of divisions in x-direction: ');
nel_y=input('   # of divisions in y-direction: ');

% 'a' and 'b' are dimensions of the element
a=x_a/nel_x; b=y_b/nel_y;

%% Call of different sub-routines

interpolation_matrix_N;
K_el;
F_el_new;

%% Global stifness matrix [K] and load vector [F]
% There are 'nodes_global' global nodes with 17-DOF at each so total DOF of
% the problem are 'global_DOF'. Therefore size of [K] will be
% global_DOF x global_DOF. I'll use the code named "F_el" to get [F] by
% just changing the values in that code.

nodes_global=(nel_x+1)*(nel_y+1);   global_DOF=nodes_global*17;
K=zeros(global_DOF,global_DOF);
F=zeros(global_DOF,1);

% This outer loop constructs the global stiffness matrix and load vector
% using boolean matrix [A].

aa=1; bb=(nel_x+2)*17;
for ii=1:nel_x*nel_y                        % ii=1:total no. of elements
    Ael=zeros(68,global_DOF);
    if (ii>nel_x) && (mod(ii,nel_x)==1)     % This 'if' starts the next
        aa=aa+17;                           % row of elements.
    end
        for kk=1:17                                             % This inner loop remains the
            Ael(kk,aa)=1; Ael(kk+17,aa+17)=1;                   % same because it constructs
            Ael(kk+34,aa+bb)=1; Ael(kk+51,aa+(bb-17))=1;        % the [A] of a single element.
            aa=aa+1;
        end
    K=K+(Ael.'*Kel*Ael);
    F=F+(Ael.'*Fel(:,ii));
end
%% Boundary conditions
```

```
% First of all we'll apply BCs on the corner nodes which are node #
% 1, nel_x+1, nodes_global-nel_x, nodes_global
% aa is the global disp. number from where disp. of node 1 starts

aa=1;
for ii=1:4          % A rectangular plate can't have more than 04 corners.
    bb=0;
    if ii==2
        aa=17*(nel_x-1)+1; xx=aa;
    elseif ii==3
        aa=xx+17*((nodes_global-nel_x)-(nel_x+1)-1); xx=aa;
    elseif ii==4
        aa=xx+17*(nodes_global-(nodes_global-nel_x)-1);
    end
   for jj=1:17
       K(aa+bb,:)=[]; K(:,aa+bb)=[]; F(aa+bb,:)=[];
       % This inner loop remains same for every node that lies
       % on the corner. Just outer loop will be modified
       % when no of rows increase/decrease.
   end
end

% Now we'll apply BCs on the nodes at x=0 and x=a.
% aa is the global disp. number from where disp. of node # 'nel_x+2' starts

aa=17*(nel_x-1)+1;
for ii=1:2*(nel_y-1)
      bb=1;
    for jj=1:11         % This inner loop remains same for every node that
        if jj==3        % lies on the edge at x=0 and x=a. Just outer loop
            bb=2;       % will be modified when no of rows increase or
        elseif jj==4    % decrease.
            bb=3;
        elseif jj==6
            bb=4;
        elseif jj==7
            bb=5;
        elseif jj==11
            bb=6;
        end
        K(aa+bb,:)=[]; K(:,aa+bb)=[]; F(aa+bb,:)=[];
    end
    if mod(ii,2)~=0             % This 'if' ensures application of current
       aa=aa+6+17*(nel_x-1);    % BCs on the appropriate nodes which lie on
    elseif mod(ii,2)==0         % x=0 and x=a.
           aa=aa+6;
    end
end
% Now we'll apply BCs on the nodes at y=0 and y=b.
% aa is the global disp. number from where disp. of node # 2 starts

aa=1;
for ii=1:2*(nel_x-1)
    if ii==nel_x                                        % This 'if' switches the
       aa=6*(nel_x-1)+(nel_y-1)*(6+17*(nel_x-1)+6)+1;  % global nodes row from
    end                                                 % 1st to last.
       bb=0;
    for jj=1:11      % This inner loop remains same for every node that
        if jj==2     % lies on the edge at y=0 and y=b. Just outer loop
            bb=1;    % will be modified when no of rows increase.
        elseif jj==4
            bb=2;
        elseif jj==5
            bb=3;
        elseif jj==7
```

```
129             bb=4;
130         elseif jj==8
131             bb=5;
132         end
133         K(aa+bb,:)=[]; K(:,aa+bb)=[]; F(aa+bb,:)=[];
134     end
135     aa=aa+6;
136 end
137
138 %% Nodal Displacements
139
140 normal=(181e9)/(100*h_tot);
141 u=real(K\F);%*normal;
142
143
144
145
146
```