Update needs for the code (11/02/24)

To do list:
- Edit grids
- Show the line after clicking the first point till the second one
- Select a node and line by a mouse click or getting in a box
- Define a material
- Define a section (shape, dimensions) and its material
- select a line or lines in the drafted model, click assign, select an element for the selected line
- select a node, assign a boundary condition: (fixed, roller, or pinned supports)
- select a node, apply a force (let's consider only directional force for now)
- select an element to apply a force (future work)
- run the analysis
- see the results (automated either original position or displaced position)
- see shear, axial, and moment diagrams
- design (future work)

In the code behind what it needs:
- coordinate matrix
- connection matrix
- boundary condition information (selected nodes and type of BC)
- applied force (magnitude and direction)

Using this data, problem is solved by a matrix operation, displacement and forces are obtained in the nodes. Then, force diagrams are plotted for the element. The next step is design (e.g., stating the displacement is within the allowed limits by standards/codes, if so, selecting the right amount of reinforced element for reinforced concrete section, if not, suggesting how much increase is needed for section dimensions (optimization problem which we can estimate without solving the problem in the future).

Example: A 1-story, 1-span structure is subjected to a 100 N horizontal tip force. The height of the story is 4 m, while the span is 5 m. The section dimensions are 0.5 m x 0.5 m for beam and columns. Calculate the displacement and rotations under this loading, and draw the axial, shear, and moment force diagrams.

Solution: the lateral displacement is 0.0026 m while the angle is 0.005 rad for Node 2. Moment diagram is as shown in the figure below.

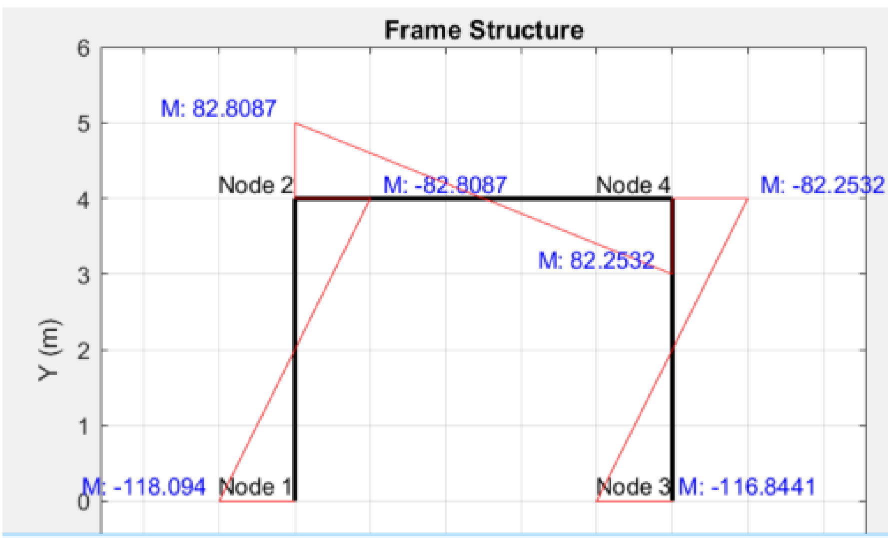

Figure #: moment diagram from MATLAB

Appendix A: Examples from the literature

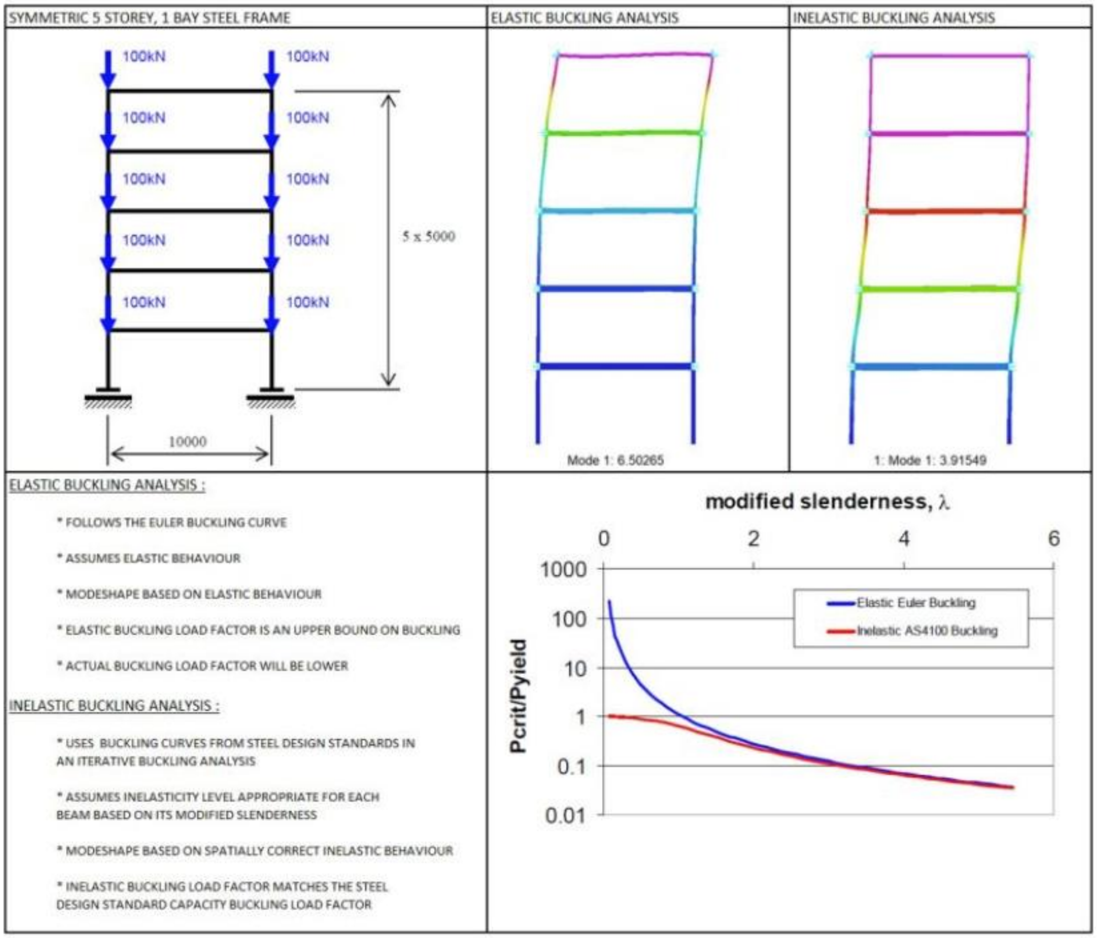

Figure #: illustration for how we can show the frame elements, boundary condition and force symbols, and displacement plot colored

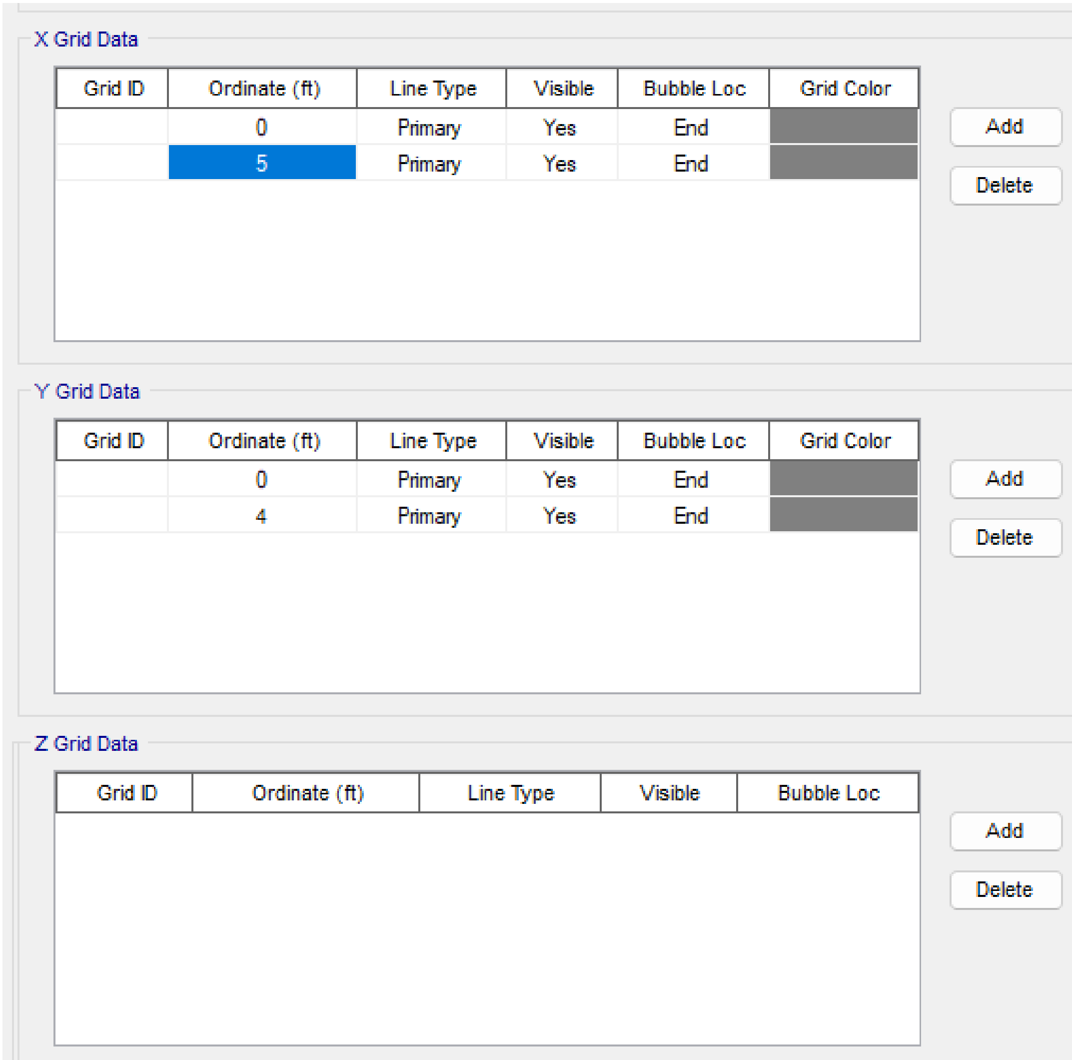

Figure #: Edit grids in Sap2000

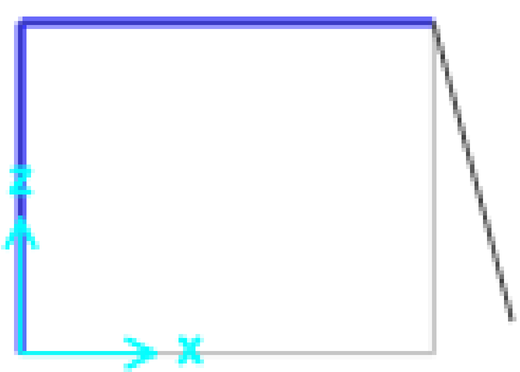

Figure #: Reference line after the first click

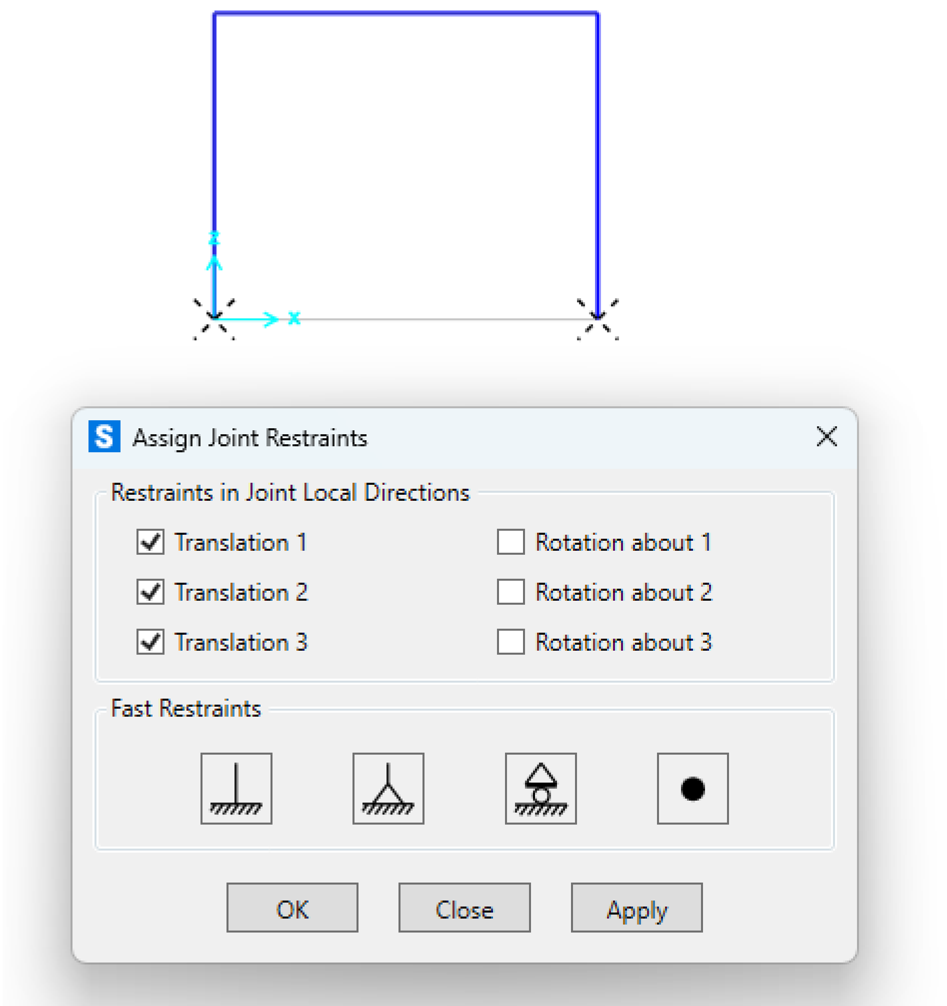

Figure #: Selected nodes and boundary condition options

Figure #: Fixed boundary condition shown after assigning

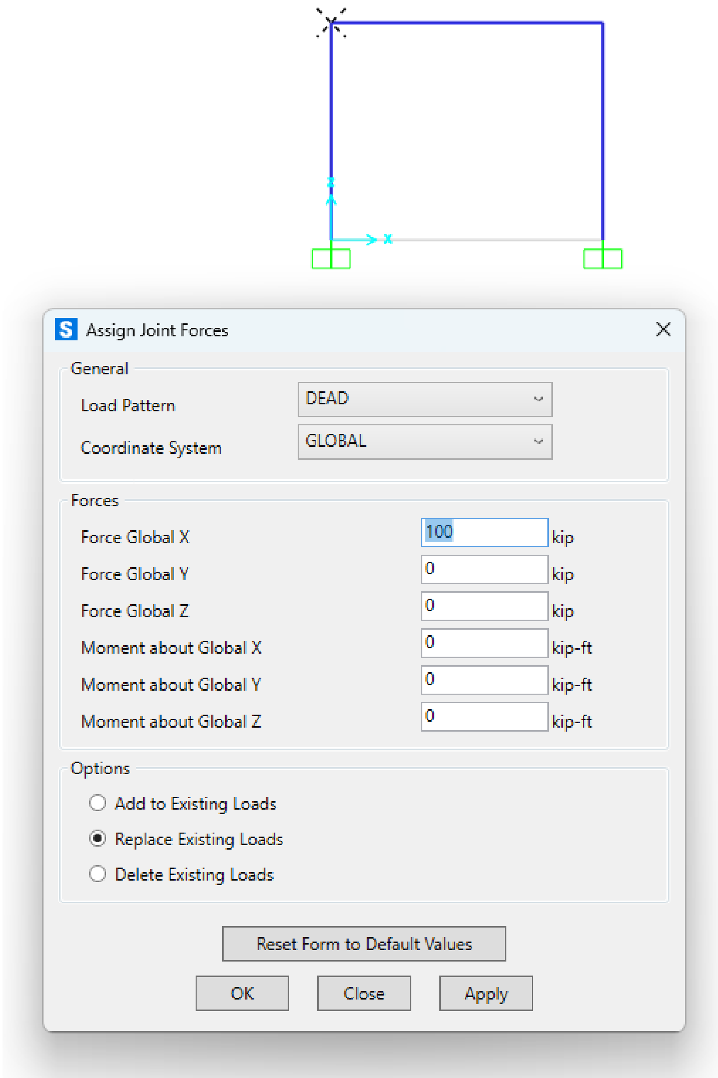

Figure #: Node selected to apply a force, and assigning a force

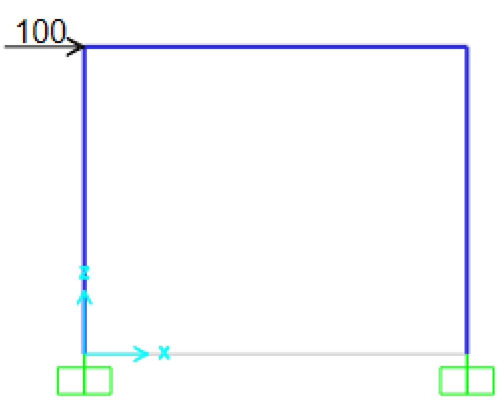

Figure #: Force illustration after assigning a force

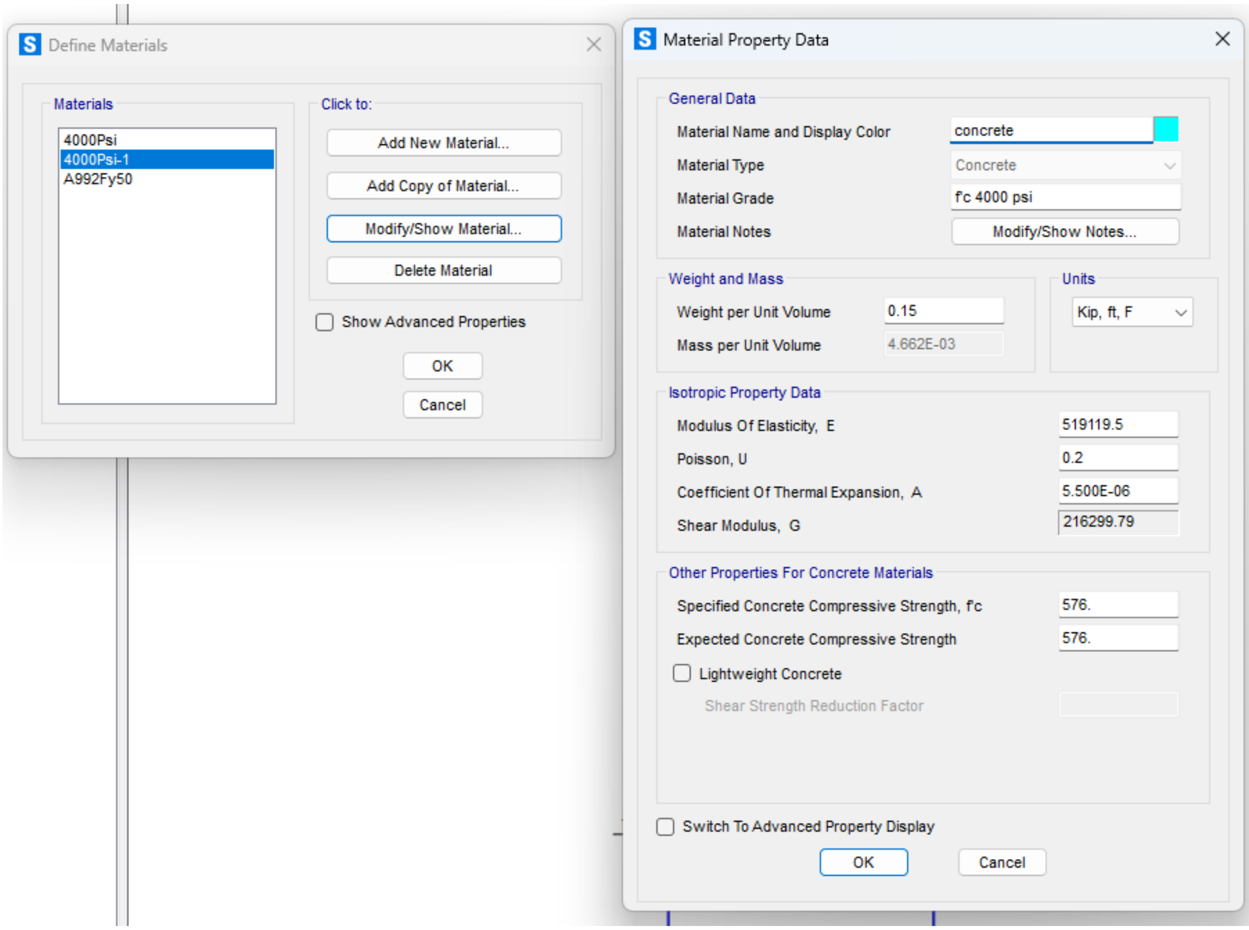

Figure #: Defining Material property (we can keep it as simple as only elasticity modulus and unit volume for now)

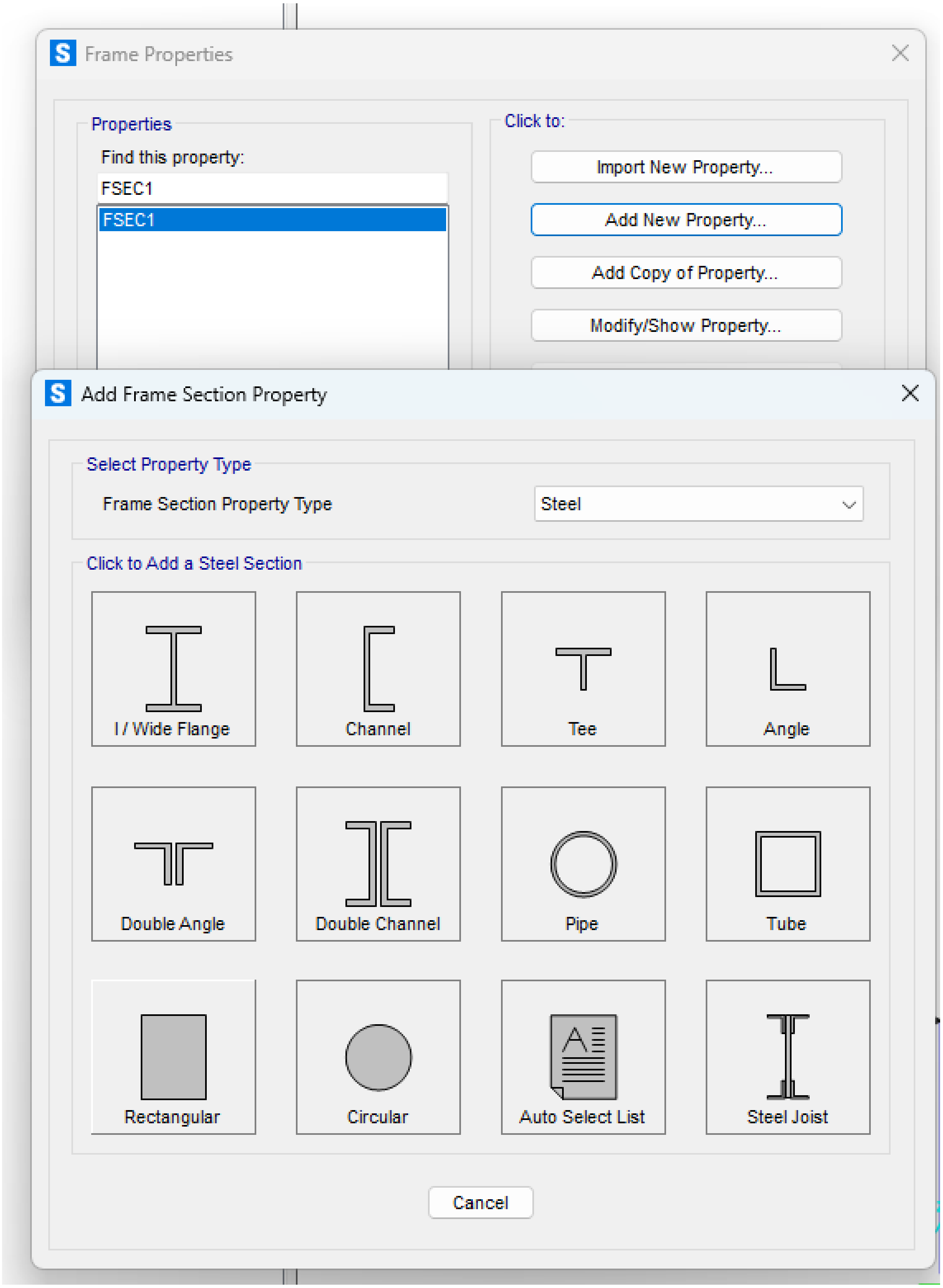

Figure #: Defining section (we can offer only rectangular, wide flange, and circular for now)

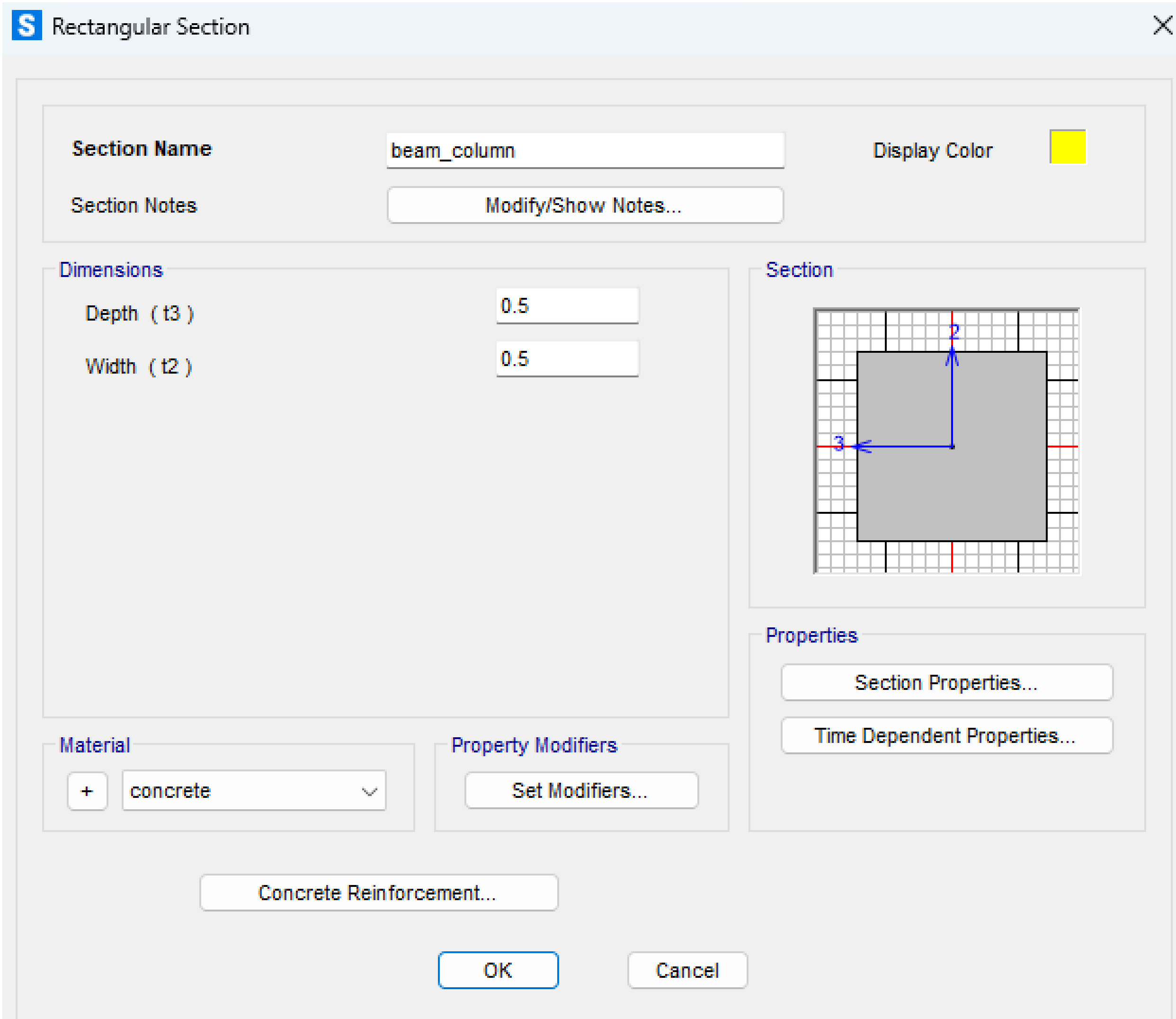

Figure #: Entering section dimensions and material selection

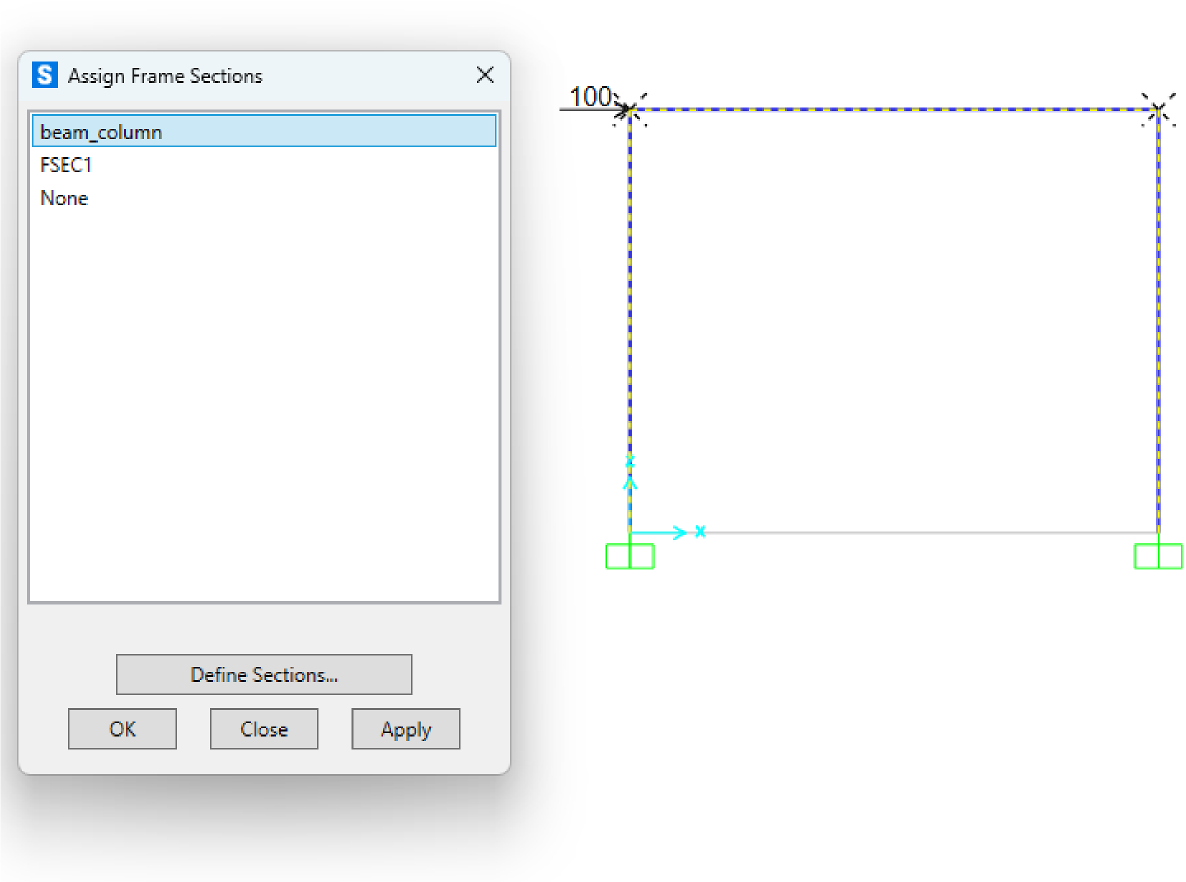

Figure #: Assign an element for the selected lines

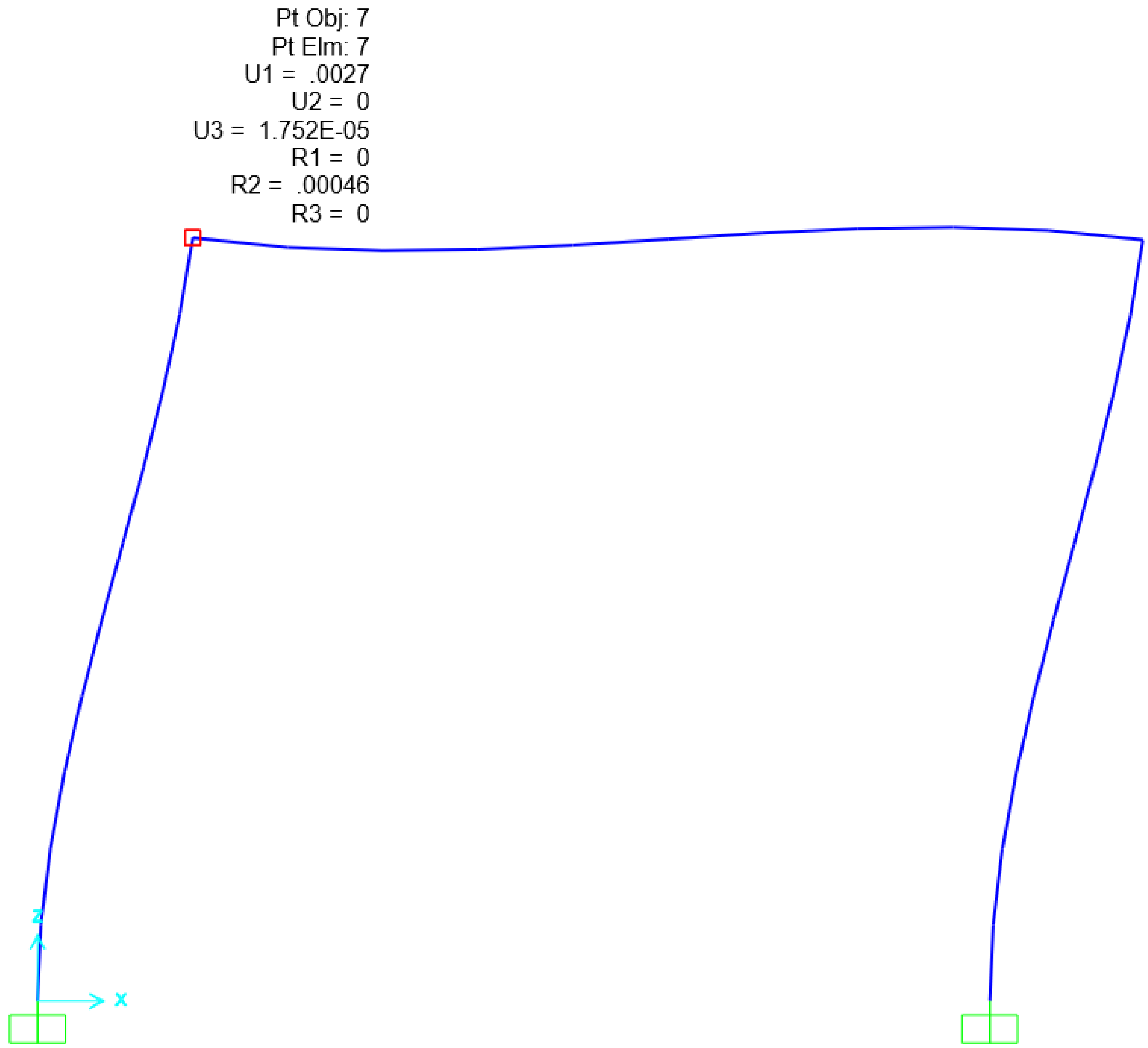

Figure #: Deformed shape with displacement or rotations roughly identical with MATLAB analysis results

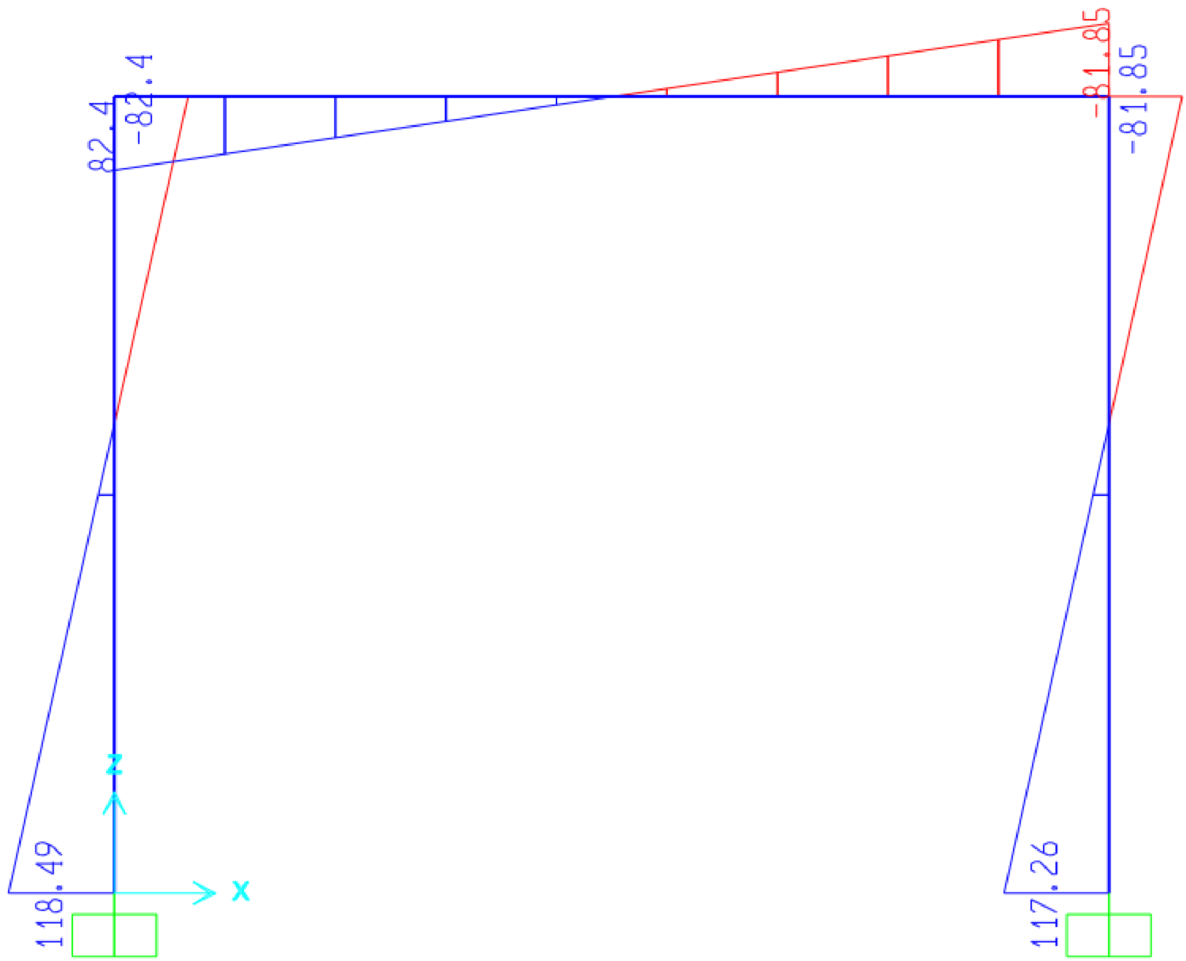

Figure #: SAP2000 Moment diagram

Appendix B: MATLAB code

```
clear all; clc; close all;
% ref. for matrix operations: https://www.mathworks.com/matlabcentral/fileexchange/160218-matrix-method-in-
structural-analysis-ch-5-plane-frame?s_tid=prof_contriblnk

% _______________________INPUT__________________________________________
 a=1;      %  a=story no;
 b=4;      %  b=height of the story - m
 c=1;      %  c=span no;
 d=5;      %  d=span length - m


% M= kg
% K1= N/m
% K2= N/m
% Fy= N

coords= [0    0;
        0    4;
        5    0;
        5    4]

connec= [1    2;
        3    4;
        2    4];

 n_mem=size(connec,1);
 n_nodes=size(coords,1);

 E=30e6*ones(n_mem,1);        % Elasticity modulus (N/m2)
 kesit_x=0.5;                 % section dimensions (m)
 kesit_y=0.5;

 I=((kesit_x*kesit_y^3)/12)*ones(n_mem,1); %
 Alan=kesit_x*kesit_y*ones(n_mem,1);

 L=zeros(n_mem,1);

for i=1:n_mem
   start_x=coords(connec(i,1),1);
   start_y=coords(connec(i,1),2);
   end_x=coords(connec(i,2),1);
   end_y=coords(connec(i,2),2);
   l_y=end_y-start_y;
   l_x=end_x-start_x;
   L(i)=sqrt((l_y)^2+(l_x)^2);

   % find the angle of the elements

   if (l_x~=0)&(l_y~=0),aci(i)=atan(abs(l_y)/abs(l_x));aci(i)=aci(i)*180/pi;

      if (l_x>0)&(l_y>0),aci(i)=aci(i);end
```

```
        if (l_x<0)&(l_y>0),aci(i)=-aci(i)+180;end
        if (l_x<0)&(l_y<0),aci(i)=aci(i)+180;end
        if (l_x>0)&(l_y<0),aci(i)=-aci(i);end

    end

    if (l_x==0)&(l_y>0),aci(i)=90;end
    if (l_x==0)&(l_y<0),aci(i)=270;end
    if (l_x>=0)&(l_y==0),aci(i)=0;end
    if (l_x<0)&(l_y==0),aci(i)=180;end
    aci(i)=aci(i)/180*pi;

end

% LOCAL STIFFNESS MATRIX

for i=1:length(connec)

    kle{i}=[rijitlikmatrisi(E(i),I(i),Alan(i),L(i))];
end

% TRANSFORMATION MATRIX

for i=1:n_mem
    r{i}=donusum(aci(i));
end

for i=1:n_mem
    k{i}=r{i}'*kle{i}*r{i};
end

%  DOFS

DOFS=zeros(n_nodes,3);
for i=1:n_nodes
    for j=1:3
        DOFS(i,j)=(i-1)*3+j;
        last_DOF=DOFS(i,j);
    end
end

% MATCHING

for i=1:n_mem
    start_node=connec(i,1);
    end_node=connec(i,2);
    start_node_dofs=DOFS(start_node,:);
    end_node_dofs=DOFS(end_node,:);
    matc{i}=[start_node_dofs,end_node_dofs];
end


REST_DOFS=[1 2 3 7 8 9];
```

```
ALL_DOFS=1:last_DOF;
UNREST_DOFS=[setdiff(ALL_DOFS,REST_DOFS)];

ext_force=zeros(n_nodes*3,1);
ext_force(4)=100; % N
% GLOBAL STIFFNESS MATRIX

K=zeros(last_DOF,last_DOF);

    for i=1:n_mem
      K(matc{i},matc{i})=K(matc{i},matc{i})+k{i};
    end

    K1=K(UNREST_DOFS,UNREST_DOFS);
    K2=K(UNREST_DOFS,REST_DOFS);
    K3=K(REST_DOFS,UNREST_DOFS);
    K4=K(REST_DOFS,REST_DOFS);
    Knew=[K1 K2; K3 K4];

F1=ext_force(UNREST_DOFS);
F2=ext_force(REST_DOFS);

% static analysis
D1=K1\-F1;
D2=zeros(length(REST_DOFS),1);
D=[D1;D2];
dofs_new=[UNREST_DOFS,REST_DOFS];

% get the global displacement
D_global=D(dofs_new);

for i = 1:length(dofs_new)
  D_global(dofs_new(i)) = D(i);
end
F_g=K*D_global+ext_force;

% get the element displacement
for i=1:n_mem
d_el_g{i}=D_global(matc{i});
d_el_l{i}=r{i}*d_el_g{i};
f_el_l{i}=kle{i}*d_el_l{i};
f_el_g{i}=r{i}'*f_el_l{i}
end

%% plot the diagrams
for i = 1:size(connec, 1)
  % Get the node indices for the current element
  node1 = coords(connec(i, 1), :);
  node2 = coords(connec(i, 2), :);

  % Plot the element as a line between the two nodes
  plot([node1(1), node2(1)], [node1(2), node2(2)], 'k-', 'LineWidth', 2);
  hold on
```

```
end
max_x_dim= max(coords(:,1));
max_y_dim= max(coords(:,2));
xlim([-2 max_x_dim+2]);
ylim([-2 max_y_dim+2]);

% Set axis properties
xlabel('X (m)');
ylabel('Y (m)');
title('Frame Structure');
axis equal;
grid on;

% Optionally, mark the nodes
for i = 1:size(coords, 1)
   text(coords(i, 1), coords(i, 2), ['Node ' num2str(i)], 'VerticalAlignment', 'bottom', 'HorizontalAlignment', 'right');
end


%% plot the moment diagram
%% column 1
f_node=coords(connec(1,1),:);
s_node=coords(connec(1,2),:);

mom_node_1=f_node;
mom_node_1(1,1)=mom_node_1(1,1)+1*f_el_g{1}(3)/abs(f_el_g{1}(3));
mom_node_2=s_node;
mom_node_2(1,1)=mom_node_2(1,1)+1*f_el_g{1}(6)/-abs(f_el_g{1}(6));
% plot coor-1,mom-1,mom-2,coor-2

% Plot the element as a line between the two nodes
plot([f_node(1), mom_node_1(1)],[f_node(2), mom_node_1(2)], 'r')
hold on
text(mom_node_1(2)-2, mom_node_1(2), ['M: ' num2str(f_el_g{1}(3))], 'VerticalAlignment', 'bottom',
'HorizontalAlignment', 'center', 'Color', 'b');
plot([mom_node_1(1),mom_node_2(1)],[mom_node_1(2),mom_node_2(2)], 'r')
plot([mom_node_2(1),mom_node_2(1)],[mom_node_2(1),mom_node_2(1)], 'r')
plot([mom_node_2(1),s_node(1)],[mom_node_2(2),s_node(2)], 'r')
text(mom_node_2(2)-2, s_node(2), ['M: ' num2str(f_el_g{1}(6))], 'VerticalAlignment', 'bottom',
'HorizontalAlignment', 'center', 'Color', 'b');

%% column 2
f_node=coords(connec(2,1),:);
s_node=coords(connec(2,2),:);

mom_node_1=f_node;
mom_node_1(1,1)=mom_node_1(1,1)+1*f_el_g{2}(3)/abs(f_el_g{2}(3));
mom_node_2=s_node;
mom_node_2(1,1)=mom_node_2(1,1)+1*f_el_g{2}(6)/-abs(f_el_g{2}(6));
% plot coor-1,mom-1,mom-2,coor-2

% Plot the element as a line between the two nodes
plot([f_node(1), mom_node_1(1)],[f_node(2), mom_node_1(2)], 'r')
```

```
hold on
text(mom_node_1(1)+2,mom_node_1(2), ['M: ' num2str(f_el_g{2}(3))], 'VerticalAlignment', 'bottom',
'HorizontalAlignment', 'center', 'Color', 'b');

plot([mom_node_1(1),mom_node_2(1)],[mom_node_1(2),mom_node_2(2)], 'r')
plot([mom_node_2(1),mom_node_2(1)],[mom_node_2(1),mom_node_2(1)], 'r')
plot([mom_node_2(1),s_node(1)],[mom_node_2(2),s_node(2)], 'r')
text(mom_node_2(1)+1,mom_node_2(2), ['M: ' num2str(f_el_g{2}(6))], 'VerticalAlignment', 'bottom',
'HorizontalAlignment', 'center', 'Color', 'b');

%% beam
f_node=coords(connec(3,1),:);
s_node=coords(connec(3,2),:);

mom_node_1=f_node;
mom_node_1(1,2)=mom_node_1(1,2)+1*f_el_g{3}(3)/abs(f_el_g{3}(3));
mom_node_2=s_node;
mom_node_2(1,2)=mom_node_2(1,2)+1*f_el_g{3}(6)/-abs(f_el_g{3}(6));
% plot coor-1,mom-1,mom-2,coor-2

% Plot the element as a line between the two nodes
plot([f_node(1), mom_node_1(1)],[f_node(2), mom_node_1(2)], 'r')
hold on
text(mom_node_1(1)-1, mom_node_1(2), ['M: ' num2str(f_el_g{3}(3))], 'VerticalAlignment', 'bottom',
'HorizontalAlignment', 'center', 'Color', 'b');

plot([mom_node_1(1),mom_node_2(1)],[mom_node_1(2),mom_node_2(2)], 'r')
plot([mom_node_2(1),mom_node_2(1)],[mom_node_2(1),mom_node_2(1)], 'r')
plot([mom_node_2(1),s_node(1)],[mom_node_2(2),s_node(2)], 'r')
text(mom_node_2(1)-1, mom_node_2(2), ['M: ' num2str(f_el_g{3}(6))], 'VerticalAlignment', 'bottom',
'HorizontalAlignment', 'center', 'Color', 'b');
```