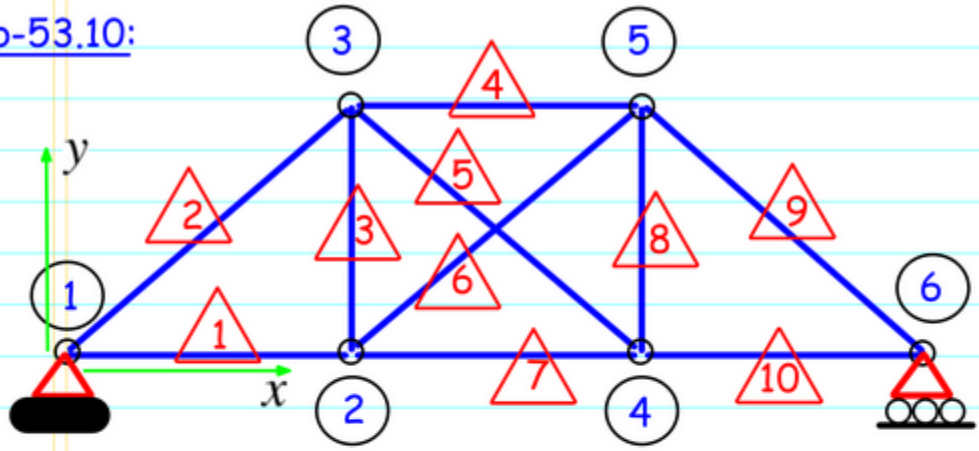
The report problem for which the FEA code was written is shown below. The Matlab code written to solve this problem is shown on the following pages.

53-19c

Pb-53.10:



Length of members 2-3 and 4-5 (truss height):

$$L_{23} = L_{45} = 1 \qquad (1)$$

L_{23} = L_{45} = 1

Length of members 1-2, 2-4, 4-6 (truss length):

$$L_{12} = L_{24} = L_{46} = 1 \qquad (2)$$

L_{12} = L_{24} = L_{46} = 1

Area of cross section: $A = 1/2$ (3)

A = 1/2

Young's modulus: $E = 5$ (4)

E = 5

Mass density: $\rho = 2$ (5)

\rho = 2

Do all work with your own matlab FE code, then verify the results with CALFEM; display both results clearly for easy comparison.

```
%TrussGEP Script
%Solves the Generalized Eigenvalue Problem for the truss and animates the
%mode shapes.

%Assign node coordinates for truss
%Rows are the global node numbers
%First column is x-coord for that node
%Second row is y-coord for that node
coord = [0, 0;
         1, 0;
         1, 1;
         2, 0;
         2, 1;
         3, 0];

%Number of nodal points
numnp = size(coord,1);

%Number of space dimensions
ndim = size(coord,2);
ndof = ndim; %Node degrees of freedom

%Number of global displacements
numdisp = numnp*ndim;

%Assign element connectivity for truss
%Rows are the element number
%First column is global node that is local node 1 for that element (row)
%Second column is global node that is local node 2 for that element (row)
conn = [1, 2;
        1, 3;
        2, 3;
        3, 5;
        3, 4;
        2, 5;
        2, 4;
        4, 5;
        5, 6;
        4, 6];

%Number of elements
nel = size(conn,1);

%Number of element nodes (2 nodes per bar)
nen = size(conn,2);
edof = ndof*nen; %Number of element deg of freedom (4 for a 2D bar element)

%Prescribed (known displacements)
%dknown: col 1 is the global displacement that is know, col 2 is the value
%for that known displacemnt
dknown = [1, 0;
          2, 0;
          12, 0];
dunk = 1:2*numnp; %Initialize dunk as all possible displacements
dunk = setdiff(dunk,dknown(:,1)); %Unknown disps are those that are not known
```

```
%Disp matrix has rows corresponding to elements and two columns (one for x
%disp and one for y disp). It relates the global node numbers to the
%displacements at that node
disp = 1:2*numnp;
disp = reshape(disp,[ndof,numnp])';

%Assign element properties (Young's Modulus, Cross Sectional Area, Density)
E(1:nel) = 5; %young's modulus
A(1:nel) = 0.5; %cross sectional area
rho(1:nel) = 2; %Density of elements

%Element lengths
%Rows are elements and columns are (x,y) for vector b/w that elements nodes
ecoord = [coord(conn(:,1),1) - coord(conn(:,2),1),
          coord(conn(:,1),2) - coord(conn(:,2),2)];
L(1:nel) = sqrt(ecoord(:,1).^2 + ecoord(:,2).^2); %compute length of elements

%Mass of elements
mass(1:nel) = rho.*A.*L; %mass = density*area*length

%Create Location Master Matrix relating global dof to local dof
lmm = zeros(nel, edof);%Preallocate memory for LMM
%Assign elements of LMM for all elements
tmp1 = (conn(:,1)-1)*2; %Number of disp dofs before node 1 or element e
lmm(:,1) = tmp1 + 1; %Global displacement for first local displacement
lmm(:,2) = tmp1 + 2; %Global displacement for second local disp
tmp2 = (conn(:,2)-1)*2; %Number of disp dofs before node 2 of element e
lmm(:,3) = tmp2 + 1; %Global displacement for first local displacement
lmm(:,4) = tmp2 + 2; %Global displacement for second local disp

%Element stiffnesses
ke = E.*A./L; %k is a 1 x nel matrix with element stiffness for each element

%Create Director Cosine Matrix
l = ecoord(:,1)./L'; %first director cosine
m = ecoord(:,2)./L'; %second director cosine
p = l.*m; %product of director cosine

%Element stiffness matrix in global coordinates
k = zeros(edof,edof,nel); %Preallocate element stiffness matrix
K = zeros(numdisp,numdisp); %Preallocate global stiffness matrix
for e = 1:nel %Loop through all elements
   k_11 = ke(e)*[l(e)^2, p(e); p(e), m(e)^2]; %stiffness submatrix
   k(:,:,e) = [k_11, -k_11; -k_11, k_11]; %local stiffness matrix from
                                          %submatrix
    for i = 1:edof %Loop through all rows in element stiffness matrix
        for j = 1:edof %Inner loop through columns in elem stiffness matrix
                r = lmm(e,i); %Use LMM to determine row in K
                c = lmm(e,j); %Use LMM to determine column in K
                K(r,c) = K(r,c) + k(i,j,e); %Assemble k(i,j,e) into position
                                            %K(r,c)
        end
    end
end
```

```
%Lumped mass matrix
mlump = zeros(numnp,1);%initialize lumped mass
%Loop through each of the elements and assign half of mass to each node
for e = 1:nel %Loop through each element
    for i = 1:nen %Loop through each node at ends of element
    tmp = conn(e,i); %Extract node number for element
    mlump(tmp) = mlump(tmp) + 0.5*mass(e); %Add half of element mass to
                                            %lumped mass matrix
    end
end

%Produce global mass matrix
%Element stiffness matrix in global coordinates
M = zeros(numdisp); %Preallocate global mass matrix
for n = 1:numnp %Loop through all nodes
    for i = 1:ndof %Loop through all node dofs
       tmp = disp(n,i); %dof number
       M(tmp,tmp) = mlump(n); %Place node 'n's lumped mass into the global
                              %Mass matrix row and column corresponding to
                              %dof number at that node
    end
end

%Solve generalized eigenvalue problem
b = dknown(:,1); %Boundary conditions
[L,X] = eigen(K,M,b); %CALFEM eigen value function to give eigenvalues L and
                      %eigenvectors X

%Determine fundamental circular frequency: (omega = sqrt(eigenvalue))
omega = L.^0.5;

%Determine fundamental period (period = 2pi/omega)
T = (omega./(2*pi)).^-1;

%Deformed mode is animated in red
hTruss = figure; %New figure
vibmode = 1; %Vibration mode (number corresponds to eigenvalue and
            %eigenvector) that will be animated. Change this number to
            %animate other mode shapes
title(sprintf('Vibration Mode %d',vibmode));
numAnim = 1; %Number of animation loops
num = 25; %Divide period into fifty time points to plot
timeStep = T(vibmode)/num; %Animation time step increment
counter = 1;
for jj = 1:numAnim %Loop through number of periods to animate
    for kk = 0:timeStep:T(vibmode) %Increment time from 0 to one full period
        clf(hTruss); %Clear the figure
        vib = X(:,vibmode).*sin(omega(vibmode)*kk); %Compute phi*sin(omega*t)
        coordAn = coord + reshape(vib,[ndof numnp])'; %Animation node
                                                %coordinates at this time step
        for i = 1:nel %Loop through all the elements
            %Plot the original truss system in green
            ecoordX = [coord(conn(i,1),1),coord(conn(i,2),1)]; %Element x-
                                                               %coords
```

```
            ecoordY = [coord(conn(i,1),2),coord(conn(i,2),2)]; %element y-
                                                               %coords
            axis([0 3.5 -0.4 1.4]); %Set axis scale to reasonable scale
                                    %axis([xmin xmax ymin ymax])
            plot(ecoordX,ecoordY,'g','LineWidth',2); %Plot original truss
                                                     %system
            hold on

            %Plot the animated truss system with vibration deformation
            ecoordAnX = [coordAn(conn(i,1),1),coordAn(conn(i,2),1)]; %element
                                                                   %x-coords
            ecoordAnY = [coordAn(conn(i,1),2),coordAn(conn(i,2),2)]; %element
                                                                   %y-coords
            axis([-0.5 3.5 -0.4 1.4]) %Set axis scale to reasonable scale
            plot(ecoordAnX,ecoordAnY,'r','LineWidth',4); %Plot deformed truss
            hold on
        end
        pause(0.02) %Seconds to pause between plotting time steps
        counter = counter + 1;
    end
end
```