Chapter 6

Finite Element Formulation for Trusses

We know that each element in the truss (see Figure 2a) can only take either a tensile or compressive axial force. In that respect, truss elements are identical to bar elements. So, all the analysis done in Chapter 5 is applicable to truss elements also. There is, however, one difference and that is discussed in this chapter.

6.1 Stiffness matrix for truss elements

In axially loaded bars, each node can only have one degree of freedom, i.e., deformation along the axis. But, in trusses each node has two degrees of freedom, viz. deformations in *X* and *Y* directions. There should be a constraint between the *X* and *Y* deformations such that the combined deformation is along the axis of the element. Therefore, every truss element made up of two nodes, has four degrees of freedom. And, these four degrees of freedom should be expressible in terms of only two degrees of freedom along the axis. This is accomplished through the coordinate transformation.

Let us isolate one truss element. Let it have deformations q_1, q_2, q_3, and q_4 as shown in Figure 1. The first two are *X* and *Y* deformation at one node, and the last two are *X* and *Y* deformations at the other node. Let q_1' and q_2' be the deformations along the length of the truss in its undeformed pose. From the figure, we observe that

$$\begin{aligned} q_1' &= q_1 \cos\theta + q_2 \sin\theta \\ q_2' &= q_3 \cos\theta + q_4 \sin\theta \end{aligned} \tag{1}$$

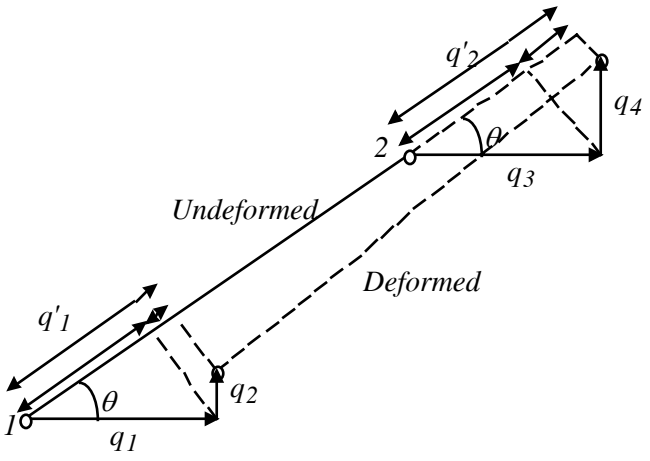

Figure 1 Transformation from 4 degrees of freedom to two axial degrees of freedom

The transformation from $\{q_1\ q_2\ q_3\ q_4\}$ to $\{q_1', q_2'\}$ can be represented in the matrix form as below.

$$\mathbf{q}' = \begin{Bmatrix} q_1' \\ q_2' \end{Bmatrix} = \begin{bmatrix} l & m & 0 & 0 \\ 0 & 0 & l & m \end{bmatrix} \begin{Bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{Bmatrix} = \mathbf{Lq} \tag{2}$$

where

$l = \cos\theta = \dfrac{x_2 - x_1}{L_e}$ and $m = \sin\theta = \dfrac{y_2 - y_1}{L_e}$ are the direction cosines of the unit vector along the element. The matrix $\mathbf{L}$ is the transformation matrix.

The *SE* for a truss element can now be written as

$$SE = \frac{1}{2}\mathbf{q}'^T\mathbf{K}_e'\mathbf{q}' = \frac{1}{2}\mathbf{q}^T\mathbf{L}^T\mathbf{K}_e'\mathbf{L}\mathbf{q} = \frac{1}{2}\mathbf{q}^T\mathbf{K}_e\mathbf{q} \tag{3}$$

where "prime" refers to quantities in the local coordinate system (i.e., along the length of the truss element, whereas "unprimed" quantities are in the global *X*-*Y* coordinate system. The matrix $\mathbf{L}$ is the transformation matrix. It is easy to see from Equation (3) that the stiffness matrix for a truss element in its inclined orientation is

$$\mathbf{K}_e = \mathbf{L}^T\mathbf{K}_e'\mathbf{L} \tag{4}$$

where

$$\mathbf{K}_e' = \frac{A_e E_e}{L_e}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{5}$$

$$L = \begin{bmatrix} l & m & 0 & 0 \\ 0 & 0 & l & m \end{bmatrix} \quad \text{with } l \text{ and } m \text{ defined as in Equation (2).} \tag{6}$$

Upon multiplication, Equation (4) gives the stiffness matrix of the truss element as

$$\mathbf{K}_e = \frac{A_e E_e}{L_e}\begin{bmatrix} l^2 & lm & -l^2 & -lm \\ lm & m^2 & -lm & -m^2 \\ -l^2 & -lm & l^2 & lm \\ -lm & -m^2 & lm & m^2 \end{bmatrix} \tag{7}$$

6.2 A note on the assembly of truss element stiffness matrices

As illustrated in Figure 2b in Chapter 5, the assembly of bar elements leads to a tri-diagonal global system stiffness matrix. The truss elements do not lead to a tri-diagonal matrix as the degrees of freedom are not adjacent to each other. Note that each truss element has two nodes, and each node has two degrees of

freedom, thus giving four degrees of freedom for the element. As shown in Equation (7), there is a 4 x 4 element matrix to be assembled. We have to first note the numbers of the global degrees of freedom of each of these four degrees of freedom, and then insert it appropriately in the global system matrix. An example will make it clear.

Consider the truss shown in Figure 2a. It has 7 elements, 5 nodes, and a total 10 degrees of freedom. The degrees of freedom are numbered as shown from 1 to 10. As illustrated in Figure 2b, the contribution of elements 3 and 4 will go to the entries with dark dots in the system matrix of size 10 x 10. Once we grasp this concept, the assembly of system matrices of even the 3-D finite elements can be accomplished easily in exactly the same way.

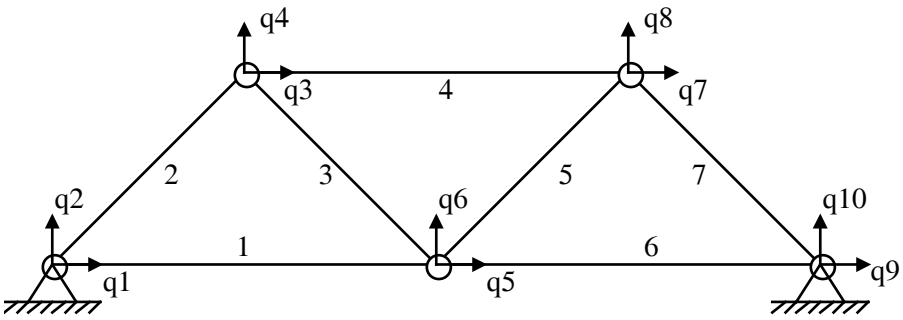

Figure 2a A truss with 7 elements, 5 nodes, and 10 degrees of freedom

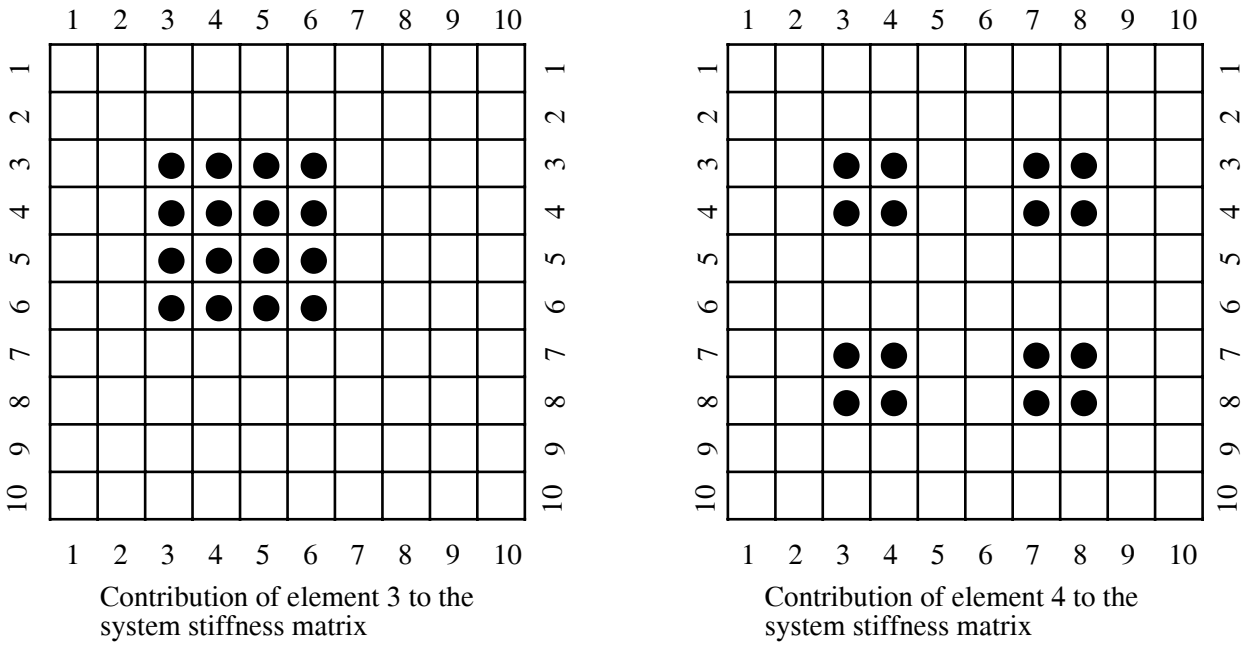

Figure 2b Illustration of assembly of the global stiffness matrix

6.3 Forces in truss elements

The discretization of trusses is according to the way truss members are connected with each other. There is no need to sub-divide the truss elements into smaller elements. Since we have nodes at each of the truss vertices (or "pin joint locations"), forces acting at the nodes can be directly taken in the global force vector in the X-Y coordinate system. Since truss elements can only take axial loads, the body and surface forces are not usually considered, nor are they a big concern in trusses. If the body and surface forces need to be considered, one has to "lump" on to the nodes directly as described in Section 2 of Chapter 5.

6.4 Computation of reaction forces, internal forces, and stresses

Deformations and ground reaction forces:

As in the case of bar elements, upon assembly of the system *PE* and application of the *MPE* principle, we get equilibrium equation

$$\mathbf{KQ} = \mathbf{F} \tag{8}$$

using which we can solve for unknown deformation **Q**. As discussed in Section 5.6, the matrix **K** will be singular when no boundary conditions are imposed on it. Let $\mathbf{K_b}$ be the stiffness matrix after imposing the boundary conditions (i.e., eliminating rows and columns corresponding to the degrees of freedom that are fixed to known values). If the imposed values on some degrees of freedom are non-zero, the force vector **F** will become $\mathbf{F_b}$ as shown in Section 5.6. We can then solve for $\mathbf{Q_b}$, the unknown degrees of freedom.

$$\mathbf{K}_b\mathbf{Q}_b = \mathbf{F}_b \tag{9}$$

After solving for $\boldsymbol{Q_b}$, we can construct the original, full $\boldsymbol{Q}$ by adding the imposed, known degree of freedom. Then, the ground reaction forces $\mathbf{F_R}$ are computed as follows.

$$\mathbf{F_R = KQ} \tag{10}$$

This $\mathbf{F_R}$ will also include the applied external forces.

Internal reactions, strains and stresses:

After computing **Q**, the element strains and stresses can easily be calculated. We need to do two things. For each element, we need to identify the corresponding four degrees of freedom and convert them to two degrees of freedom along the length of the element using Equation (2). The internal reaction forces in the element are computed using

$$\mathbf{f}_{R_e} = \frac{A_e E_e}{L_e}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\begin{Bmatrix} q_1' \\ q_2' \end{Bmatrix} \tag{11}$$

Strains and stresses are computed using

$$\varepsilon = \mathbf{Bq} \qquad \text{and} \qquad \sigma = E\varepsilon \tag{12}$$

which we derived in Chapter 4 (Refer to Equations (9) and (11) in Chapter 4).

6.5 Computer implementation of truss finite elements

We have discussed all the essential aspects of the truss elements and are now ready to implement it on the computer as a general purpose program. A Matlab script for FEM analysis of trusses is included at the end of this chapter. This script has the same general features as any FEM software. It is appropriate at this time to discuss the main features of a FEM software package. Referring to the Matlab script will be useful to fully understand the way FEM software works.

6.6 Components of FEM software packages

The three main components of a FEM software are: Pre-Processor, Analysis, and Post-Processor. The pre-processor takes care of the building of the geometric model of the structure. First one needs to create a graphical model of the structure, and then discretize it into elements. The process of discretization is called meshing. Usually, the finer the mesh is, the better is the accuracy. We saw this in Chapter 2 (Figure 12). The creation of the mesh implies defining nodes and their nodal coordinates, elements and element connectivities (i.e., which nodes make up the element). We also need to define the material properties and element properties (such as cross-section areas of elements).

The analysis module provides for specification of forces and displacement boundary conditions on the meshed model. It then computes the element stiffness matrices and force vectors, assembles them into global system matrix, imposes the boundary conditions, and finally solves for the unknown deformations.

The post-processor helps us visualize the results of the analysis. As discussed in the previous section, we can compute ground reaction forces, internal stresses and strains. All of these and the deformed model can be plotted graphically. Commercial FEM packages enable us to plot the contours of deformations, strains and stresses as if we did the analysis on a continuous model. The reason they are able to do this is because of the interpolating shape functions used in FEM formulations.

The Matlab script for trusses uses the same approach and should be studied carefully.

6.7 The Matlab script for truss FEM

Four input files are needed for truss.m. These files contain the geometric, physical and material property, and force information about the truss to be analyzed.

1) node.dat

This file contains the (X,Y) coordinates of all the node numbers in the following format.

Node#	X-coordinate	Y-Coordinate
1		
2		
3		
etc.		

2) elem.dat

This file contains the element information pertaining to the two nodes that form the element, the cross-section area and Young's modulus of the element.

Element#	Node#1	Node#2	Area	Young's modulus
1				
2				
3				
etc.				

3) forces.dat

This file contains the nodal forces to be applied on the truss. You need to specify the node number, the degree-of-freedom (dof) number (1 if force is in the X-direction, and 2 if the force is in the Y-direction), and the value of the force. If you have forces in the X and Y directions at the same node, then you need to put in two lines in the force, one for each direction.

Serial#	Node#	dof#	Force value
1			
2			
etc.			

4) disp.dat

This file contains the information about the displacement boundary conditions, i.e. which nodes and dof's are fixed to be zero. dof# is to be treated in the same way as you do in the case of force specification.

Serial#	Node#	dof#
1		
2		
etc.		

Matlab Script 3 for truss FEM

```
==================start of truss.m==================
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Finite Element MATLAB script for
% deflection and stress analysis of trusses.
% Written by: G.K. Ananthasuresh
% for MEAM 310 class in Spring 1997
% at the University of Pennsylvania.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
 clg                        % Clear graphics window
 clear all                  % Clear all variables
 clc                        % Clear command window
 hold off                   % No hold on the graphics window
%
% This script needs the following scripts to run
%    matcut.m         -->   trims a matrix
%    veccut.m         -->   trims a vector
%    femtruss.m       -->   FEA for trusses
% It also needs the following input files
%    node.dat         -->   nodal data
%    elem.dat         -->   element data
%    forces.dat       -->   force data
%    disp.dat         -->   displacement boundary condition data
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% READ INPUT from files
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Read in nodal and element connectivity data from
% files: nodes.dat and elem.dat
load node.dat
load elem.dat
%
% Read in force data from the file forces.dat
load forces.dat
% Read in displacement boundary condition data from the file disp.dat
load dispbc.dat
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% PRE-PROCESSING
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Identify the number of nodes, X and Y Coordinates of the nodes
NNODE = size(node,1);
nx = node(:,2);
ny = node(:,3);
%
% Identify the number of elements and form an element connectivity
array,
% the cross-section and Young's modulus arrays.
NELEM = size(elem,1);
```

```
ncon = elem(:,[2 3]);
A = elem(:,4);
E = elem(:,5);
%
% Arrange force information into a force vector, F
F = zeros(2*NNODE,1);           % Initialization
Nforce = size(forces,1);
for i = 1:Nforce,
   dof = (forces(i,2)-1)*2 + forces(i,3);
   F(dof) = forces(i,4);
end
%
% Displacement boundary conditions
Nfix = size(dispbc,1);
j = 0;
for i = 1:Nfix,
    j = j + 1; dispID(j) = (dispbc(i,2)-1)*2+dispbc(i,3);
end
dispID = sort(dispID);
%
% Compute the lengths of the elements
for ie=1:NELEM,
  eye = ncon(ie,1);
  jay = ncon(ie,2);
  L(ie) = sqrt ( (nx(jay) - nx(eye))^2 + (ny(jay) - ny(eye))^2 );
end
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SOLUTION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Call femtruss.m to solve for the following.
%    Deflections: U
%    Reaction forces at the constrained nodes: R
%    Internal forces in each truss member: Fint
%    Global stiffness matrix, Kglobal
%    Strain energy: SE
[U,R,Fint,Kglobal,SE] =
femtruss(A,L,E,nx,ny,ncon,NELEM,NNODE,F,dispID);
%
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% POST-PROCESSING
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Plotting
clg
for ip=1:NELEM,
  pt1 = ncon(ip,1); pt2 = ncon(ip,2);
  dx1 = U(2*(pt1-1)+1);
  dy1 = U(2*(pt1-1)+2);
  dx2 = U(2*(pt2-1)+1);
  dy2 = U(2*(pt2-1)+2);
```

```
  TorC = Fint(ip,5);
%
% Plot undeformed only
%
% Note: The elements in tension, compression, and no stress are
%          plotted in red, cyan, and white respectively.
%          Undeformed configuration is in dotted yellow.
%          The nodes are marked with little yellow circles in the
%          undeformed configuration.
%
  plot([nx(pt1) nx(pt2)], [ny(pt1) ny(pt2)],'--y');
  hold on
  plot([nx(pt1) nx(pt2)], [ny(pt1) ny(pt2)],'oy');
  if TorC == 1
      plot([nx(pt1)+dx1 nx(pt2)+dx2], [ny(pt1)+dy1 ny(pt2)+dy2],'-r');
  elseif TorC == -1
      plot([nx(pt1)+dx1 nx(pt2)+dx2], [ny(pt1)+dy1 ny(pt2)+dy2],'-c');
  else
     plot([nx(pt1)+dx1 nx(pt2)+dx2], [ny(pt1)+dy1 ny(pt2)+dy2],'-w');
  end
end
xlabel('X');
ylabel('Y');
axis('equal');
legend('--y','Undeformed','yo','Node','-r','Tensile stress','-c',...
      'Compressive stress','-w','No stress');



====================end of truss.m=======================


====================start of matcut.m====================
function D = mcut(C,i)
% To remove ith row and ith column from C (size: NxN) and
% return D (size: N-1xN-1)
% This function is useful in imposing boundary condition onto
% the global stiffness and force vectors.
[m,n] = size(C);
d1 = C(1:i-1,1:i-1);
d2 = C(1:i-1,i+1:n);
d3 = C(i+1:m,1:i-1);
d4 = C(i+1:m,i+1:n);
D = [d1 d2; d3 d4];
====================end of matcut.m======================


====================start of veccut.m====================
function D = veccut(C,i)
% To remove member i from C and return D with size 1 less.
% This function is useful in imposing the boundary conditions
% onto the global stiffness matrix and force vector.
```

```
[m,n] = size(C);
if m == 1
   d1 = C(1:i-1);
   d2 = C(i+1:n);
   D = [d1 d2];
end
if n == 1
   d1 = C(1:i-1);
   d2 = C(i+1:m);
   D = [d1; d2];
end
====================end of veccut.m====================
```

```
==================start of femtruss.m===============
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This m-file forms the finite element model of a truss
% system and solves it.
% Written by G.K. Ananthasuresh for MEAM 310
% class in Spring 1997.
% Inputs:
%     Ae    -->  A vector containing the cross-section areas of
%                all the elements
%     Le    -->  A vector containing the lengths of all the
%                elements
%     Ee    -->  A vector containing the Young's moduli of all
%                the elements
%     nx    -->  A vector containing the X-coordinates of all
%                the nodes
%     ny    -->  A vector containing the Y-coordinates of all
%                the nodes
%     ncon  -->  A matrix containing the nodal-connectivity of
%                all the elements
%     F     -->  A vector contating the applied external forces
%                dispID     -->  A vector containing the
%                displacement boundary condition information
%
% Outputs:
%     u     -->  The displacements of all the dof
%     Rdisp -->  The reaction forces at the dof which are
%                specified to be fixed
%     P     -->  A matrix containing the X and Y internal
%                reaction forces at the two
%                ends of each element, and flag that   tells you %
           if the element is in tension (1), compression %
      (-1) or no stress (0).
%     Ksing -->  The global stiffness matrix before the
%                application of the
%                           boundary conditions.
%     SE    -->  The strain energy in the entire truss
%
function [u,Rdisp,P,Ksing,SE] = femtruss(Ae, Le, Ee, nx, ny, ncon,
NELEM, NNODE, F, dispID)

K = zeros(2*NNODE,2*NNODE);           % Initialize global stiffness
matrix

for ie=1:NELEM,
  eye = ncon(ie,1);
  jay = ncon(ie,2);

% Form the transformation matrix, Lambda.
  L = Le(ie);
  A = Ae(ie);
  E = Ee(ie);
  lox = (nx(jay)-nx(eye))/L; mox = (ny(jay)-ny(eye))/L;
```

```
  loy = -mox; moy = lox;
  Lambda = [ lox mox  0  0    ; ...
              0   0  lox mox ];


  k = [ 1 -1; -1 1 ];                 % Local element stiffness matrix

  k = k*(A*E/L);

  klocal = Lambda' * k * Lambda;


% Form ID matrix to assemble klocal into the global system
% stiffness matrix, K.
  id1 = 2*(eye-1) + 1;
  id2 = id1 + 1;
  id3 = 2*(jay-1) + 1;
  id4 = id3 + 1;

  K(id1,id1) = K(id1,id1) + klocal(1,1);
  K(id1,id2) = K(id1,id2) + klocal(1,2);
  K(id2,id1) = K(id2,id1) + klocal(2,1);
  K(id2,id2) = K(id2,id2) + klocal(2,2);

  K(id1,id3) = K(id1,id3) + klocal(1,3);
  K(id1,id4) = K(id1,id4) + klocal(1,4);
  K(id2,id3) = K(id2,id3) + klocal(2,3);
  K(id2,id4) = K(id2,id4) + klocal(2,4);

  K(id3,id1) = K(id3,id1) + klocal(3,1);
  K(id3,id2) = K(id3,id2) + klocal(3,2);
  K(id4,id1) = K(id4,id1) + klocal(4,1);
  K(id4,id2) = K(id4,id2) + klocal(4,2);

  K(id3,id3) = K(id3,id3) + klocal(3,3);
  K(id3,id4) = K(id3,id4) + klocal(3,4);
  K(id4,id3) = K(id4,id3) + klocal(4,3);
  K(id4,id4) = K(id4,id4) + klocal(4,4);

end

% Store unlaltered K as Ksing before applying the boundary conditions.
  Ksing = K;

%det(K)
%inv(K);
%pause

% Imposing displacement boundary conditions
% ------------------------------------------
% dispID array contains the dof which are assigned zero values.
[sm,sn] = size(dispID);
```

```
Ndbc = sn;
for nd=1:Ndbc,
 K = matcut(K,dispID(nd)-nd+1);
 F = veccut(F,dispID(nd)-nd+1);
end

% To solve for unknown displacements.
U = inv(K)*F;
SE = 0.5*U'*K*U;

% Results
% ---------------
% "u" for all nodes (including those where values were
% specified)
u = zeros(2*NNODE,1);

for iu=1:Ndbc,
  u(dispID(iu)) = 12345.12345;
end

iuc = 0;
for iu=1:2*NNODE,
    if u(iu) == 12345.12345
     iuc = iuc+1;
  else
     u(iu) = U(iu-iuc);
  end
end

for iu=1:Ndbc,
  u(dispID(iu)) = 0;
end

dx = zeros(1,NNODE);
dy = zeros(1,NNODE);

for iu=1:NNODE,
  dx(iu) = u(2*(iu-1)+1);
  dy(iu) = u(2*(iu-1)+2);
end

%-----------------------------------------------
% Computation of reactions at constrained nodes
%-----------------------------------------------
R = Ksing*u;
Rdisp = zeros(1,Ndbc);
for iu=1:Ndbc,
  Rdisp(iu) = R(dispID(iu));
end

%----------------------------------------
% Computation of internal reaction forces
```

```
% and storing in P(NNODE,4)
%-----------------------------------------
M = zeros(NNODE,1);

for ie=1:NELEM,

  eye = ncon(ie,1);
  jay = ncon(ie,2);


% Form the transformation matrix, Lambda.
  L = Le(ie); A = Ae(ie);
  lox = (nx(jay)-nx(eye))/L; mox = (ny(jay)-ny(eye))/L;
  loy = -mox; moy = lox;
  Lambda = [ lox mox  0  0    ; ...
              0   0  lox mox ];


  k = [ 1 -1; -1 1 ];

  k = k*(A*E/L);

  klocal = Lambda' * k * Lambda;


% Form ID matrix to identify respective displacements.
  id1 = 2*(eye-1) + 1;
  id2 = id1 + 1;
  id3 = 2*(jay-1) + 1;
  id4 = id3 + 1;

  ulocal = [u(id1) u(id2) u(id3) u(id4)];

  Rint = klocal*ulocal';

  P(ie,1) = Rint(1);
  P(ie,2) = Rint(2);
  P(ie,3) = Rint(3);
  P(ie,4) = Rint(4);

  nxd1 = nx(eye) + u(id1);
  nyd1 = ny(eye) + u(id2);
  nxd2 = nx(jay) + u(id3);
  nyd2 = ny(jay) + u(id4);

  Ld = sqrt( (nxd2-nxd1)^2 + (nyd2-nyd1)^2 );

  if Ld > L P(ie,5) = 1; else P(ie,5) = -1; end
  if Ld==L P(ie,5) = 0; end

end
====================end of femtruss.m=================
```

Exercise 6.1

Compute the deflection of point 1 in Figure 3. Use the truss-element Finite Element MATLAB script supplied to you and also verify the result through a hand-calculation.

The hand calculation should be done in one of the two ways. One method is to do all the calculations involved in the FEM code by hand. This will enable you to fully understand the procedure. The other is to use the Castigliano's theorem or a similar analytical method that you would have learnt in your Statics/Strength-of-materials course.

Answer: Horizontal deflection = 281E-6 mm, and vertical deflection = 1.008E-3 mm.

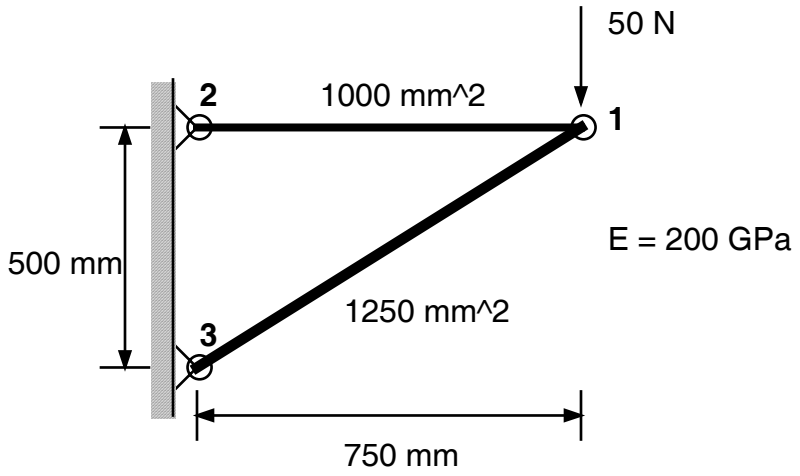

Figure 3 A simple truss

Exercise 6.2: Maxwell's theorem

Use the truss FEM Matlab script to verify an interesting theorem by James Clerk Maxwell for statically determinate pin-jointed frameworks which states that

$$\sum_{i=1}^{m} l_i^t f_i^t - \sum_{j=1+m}^{n} l_j^c f_j^c = \text{Constant}$$

where l_i^t = length of the bar in tension

f_i^t = internal axial force in the *i th* bar in tension

and superscript *"c"* refers to compression.

There are *m* bars in tension out of the total of *n* bars.

This theorem has a great significance in the area of structural optimization.

Verify this theorem for the following pin-jointed framework (Figure 4) for the applied set of loads. Run the Matlab script several times by changing the configuration of the truss (i.e., add more elements, move elements around, add nodes, etc.), but do **not** change the points of application of the loads and supports, nor the values of the loads. The underlined part is the requirement for this theorem to hold. Also, make sure that the structure remains statically determinate. You should see that *Maxwell's Constant* shown above will have the same value for all those cases. Assume that E = 1.0E5, and A = 1 for the areas of cross-section of the bar.

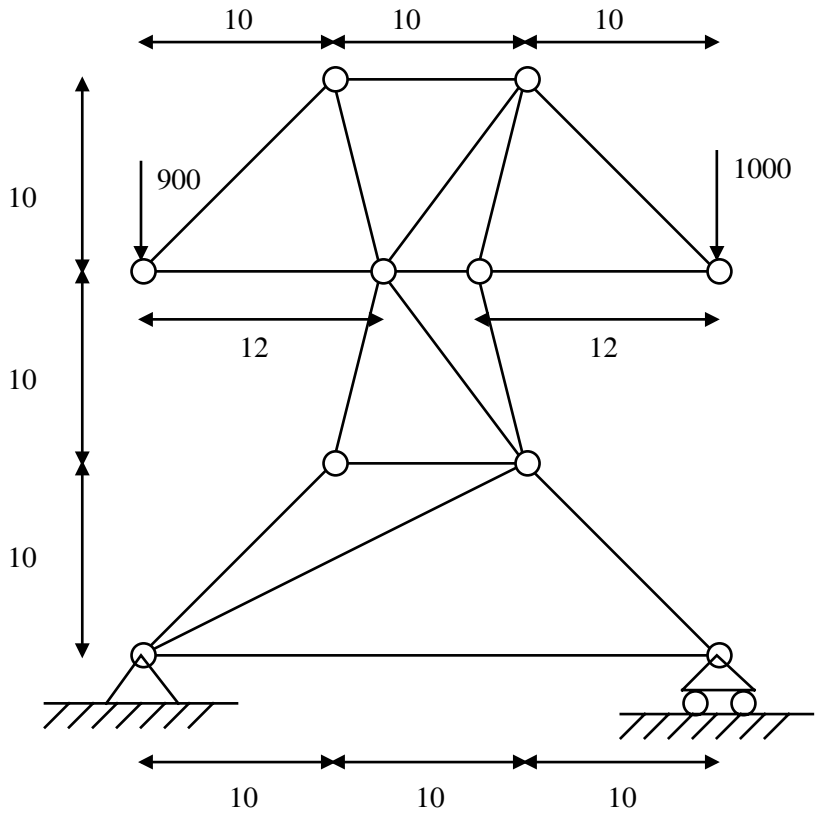

Figure 4 A truss to verify the Maxwell's theorem

(Not to scale and simply take the numbers without worrying about the units.)

Exercise 6.3

The Matlab script for truss FEM can only take displacement boundary conditions that have zero value. Modify it to have non-zero displacement specification. (The change is really simple. Refer to Section 5.6).