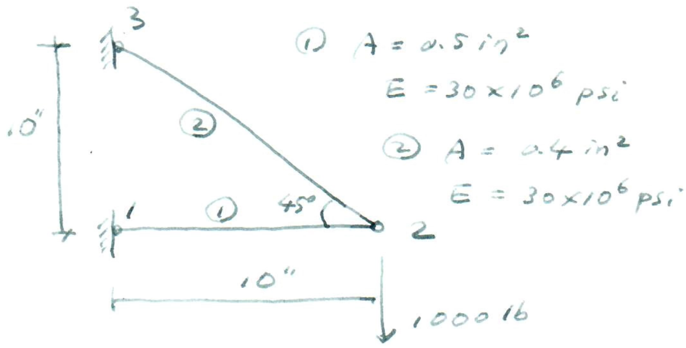
1/8

```
%------------------------------------------------------------------------------%
% Example 7.4.1
% to solve static 2-d truss structure
%
% Problem description
%   Find the deflection and stress of the truss made of two members
%   as shown in Fig. 7.4.1.
%
% Variable descriptions
%   k = element stiffness matrix
%   kk = system stiffness matrix
%   ff = system force vector
%   index = a vector containing system dofs associated with each element
%   gcoord = global coordinate matrix
%   disp = nodal displacement vector
%   elforce = element force vector
%   eldisp = element nodal displacement
%   stress = stress vector for every element
%   elprop = element property matrix
%   nodes = nodal connectivity matrix for each element
%   bcdof = a vector containing dofs associated with boundary conditions
%   bcval = a vector containing boundary condition values associated with
%           the dofs in 'bcdof'
%------------------------------------------------------------------------------%

%----------------------------
% control input data
%----------------------------

clear
nel=2;             % number of elements
nnel=2;            % number of nodes per element
ndof=2;            % number of dofs per node
nnode=3;           % total number of nodes in system
sdof=nnode*ndof;   % total system dofs
```




```
%----------------------------
% nodal coordinates
%----------------------------

gcoord(1,1)=0.0;  gcoord(1,2)=0.0;   % x, y-coordinate of node 1
gcoord(2,1)=10.0; gcoord(2,2)=0.0;   % x, y-coordinate of node 2
gcoord(3,1)=0.0;  gcoord(3,2)=10.0;  % x, y-coordinate of node 3

%---------------------------------------------
% material and geometric properties
%---------------------------------------------

elprop(1,1)=30000000;  % elastic modulus of 1st element
elprop(1,2)=0.4;       % cross-section of 1st element
elprop(2,1)=30000000;  % elastic modulus of 2nd element
```

2/8

```
elprop(2,2)=0.5;        % cross-section of 2nd element

%-----------------------------
% nodal connectivity
-----------------------------

nodes(1,1)=1;  nodes(1,2)=2;   % nodes associated with element 1
nodes(2,1)=2;  nodes(2,2)=3;   % nodes associated with element 2

%-----------------------------
% applied constraints
%-----------------------------

bcdof(1)=1;     % 1st dof (horizontal displ) is constrained
bcval(1)=0;     % whose described value is 0
bcdof(2)=2;     % 2nd dof (vertical displ) is constrained
bcval(2)=0;     % whose described value is 0
bcdof(3)=5;     % 5th dof (horizontal displ) is constrained
bcval(3)=0;     % whose described value is 0
bcdof(4)=6;     % 6th dof (vertical displ) is constrained
bcval(4)=0;     % whose described value is 0

%-----------------------------
% initialization to zero
%-----------------------------

 =zeros(sdof,1);                 % system force vector
kk=zeros(sdof,sdof);             % system stiffness matrix
index=zeros(nnel*ndof,1);        % index vector
elforce=zeros(nnel*ndof,1);      % element force vector
eldisp=zeros(nnel*ndof,1);       % element nodal displacement vector
k=zeros(nnel*ndof,nnel*ndof);    % element stiffness matrix
stress=zeros(nel,1);             % stress vector for every element
%-----------------------------
% applied nodal force
%-----------------------------

ff(4)=-1000;     % 2nd node has 1000 lb in downward direction

%---------------------------
% loop for elements
%---------------------------

for iel=1:nel     % loop for the total number of elements

nd(1)=nodes(iel,1);   % 1st connected node for the (iel)-th element
nd(2)=nodes(iel,2);   % 2nd connected node for the (iel)-th element

...=gcoord(nd(1),1); y1=gcoord(nd(1),2);   % coordinate of 1st node
x2=gcoord(nd(2),1); y2=gcoord(nd(2),2);   % coordinate of 2nd node
```

3/8

```
leng=sqrt((x2-x1)^2+(y2-y1)^2);  % element length

if (x2-x1)==0;
if y2>y1;
   beta=2*atan(1);       % angle between local and global axes
else
   beta=-2*atan(1);
end
else
beta=atan((y2-y1)/(x2-x1));
end

el=elprop(iel,1);               % extract elastic modulus
area=elprop(iel,2);             % extract cross-sectional area

index=feeldof(nd,nnel,ndof);  % extract system dofs for the element

k=fetruss2(el,leng,area,0,beta,1); % compute element matrix

kk=feasmbl1(kk,k,index);           % assemble into system matrix

end

%----------------------------------------------------
%  apply constraints and solve the matrix
%----------------------------------------------------

[kk,ff]=feaplyc2(kk,ff,bcdof,bcval);  % apply the boundary conditions

disp=kk\ff;   % solve the matrix equation to find nodal displacements

%---------------------------------------------------
%  post computation for stress calculation
%---------------------------------------------------

for iel=1:nel         % loop for the total number of elements

nd(1)=nodes(iel,1);   % 1st connected node for the (iel)-th element
nd(2)=nodes(iel,2);   % 2nd connected node for the (iel)-th element

x1=gcoord(nd(1),1); y1=gcoord(nd(1),2);  % coordinate of 1st node
x2=gcoord(nd(2),1); y2=gcoord(nd(2),2);  % coordinate of 2nd node

leng=sqrt((x2-x1)^2+(y2-y1)^2);  % element length

if (x2-x1)==0;
if y2>y1;
   beta=2*atan(1);       % angle between local and global axes
 se
   beta=-2*atan(1);
end
```

```
else
beta=atan((y2-y1)/(x2-x1));
end

 =elprop(iel,1);                % extract elastic modulus
area=elprop(iel,2);             % extract cross-sectional area

index=feeldof(nd,nnel,ndof);  % extract system dofs for the element

k=fetruss2(el,leng,area,0,beta,1); % compute element matrix

for i=1:(nnel*ndof)           % extract displacements associated with
eldisp(i)=disp(index(i));     % (iel)-th element
end

elforce=k*eldisp;             % element force vector
stress(iel)=sqrt(elforce(1)^2+elforce(2)^2)/area; % stress calculation

if ((x2-x1)*elforce(3)) < 0;
stress(iel)=-stress(iel);
end

end

%----------------------------
% print fem solutions
%----------------------------


num=1:1:sdof;
displ=[num'; disp]          % print displacements

numm=1:1:nel;
stresses=[numm' stress]    % print stresses

%--------------------------------------------------------------------
```

5/8

```
function [index]=feeldof(nd,nnel,ndof)
%----------------------------------------------------------
% Purpose:
%     Compute system dofs associated with each element

% Synopsis:
%     [index]=feeldof(nd,nnel,ndof)
%
% Variable Description:
%     index - system dof vector associated with element "iel"
%     iel - element number whose system dofs are to be determined
%     nnel - number of nodes per element
%     ndof - number of dofs per node
%-----------------------------------------------------------

 edof = nnel*ndof;
   k=0;
   for i=1:nnel
     start = (nd(i)-1)*ndof;
       for j=1:ndof
         k=k+1;
         index(k)=start+j;
       end
   end
```

6/8

```
function [k,m]=fetruss2(el,leng,area,rho,beta,ipt)

%--------------------------------------------------------------
% Purpose:
    Stiffness and mass matrices for the 2-d truss element
%    nodal dof {u_1 v_1 u_2 v_2}
%
% Synopsis:
%    [k,m]=fetruss2(el,leng,area,rho,beta,ipt)
%
% Variable Description:
%    k - element stiffness matrix (size of 4x4)
%    m - element mass matrix (size of 4x4)
%    el - elastic modulus
%    leng - element length
%    area - area of truss cross-section
%    rho - mass density (mass per unit volume)
%    beta - angle between the local and global axes                ipt = 1: consistent
mass matrix
%          positive if the local axis is in the ccw direction from
%          the global axis
%    ipt = 1 - consistent mass matrix
%        = 2 - lumped mass matrix
%--------------------------------------------------------------------------

% stiffness matrix

 c=cos(beta); s=sin(beta);
 k= (area*el/leng)*[ c*c   c*s  -c*c  -c*s;...
                     c*s   s*s  -c*s  -s*s;...
                    -c*c  -c*s   c*c   c*s;...
                    -c*s  -s*s   c*s   s*s];


% consistent mass matrix

 if ipt==1

   m=(rho*area*leng/6)*[ 2*c*c+2*s*s  0  c*c+s*s  0;...
                         0  2*c*c+2*s*s  0  c*c+s*s;...
                         c*c+s*s  0  2*c*c+2*s*s  0;...
                         0  c*c+s*s  0  2*c*c+2*s*s];

% lumped mass matrix

 else

   m=(rho*area*leng/2)*[ c*c+s*s  0  0  0;...
                         0  c*c+s*s  0  0;...
                         0  0  c*c+s*s  0;...
                         0  0  0  c*c+s*s];

 end
```

7/8

```
function [kk]=feasmbl1(kk,k,index)
%----------------------------------------------------------
% Purpose:
%     Assembly of element matrices into the system matrix

% Synopsis:
%     [kk]=feasmbl1(kk,k,index)
%
% Variable Description:
%     kk - system matrix
%     k  - element matri
%     index - d.o.f. vector associated with an element
%-----------------------------------------------------------


 edof = length(index);
 for i=1:edof
   ii=index(i);
     for j=1:edof
       jj=index(j);
         kk(ii,jj)=kk(ii,jj)+k(i,j);
     end
 end
```

8/8

```
function [kk,ff]=feaplyc2(kk,ff,bcdof,bcval)

%----------------------------------------------------------
% Purpose:
     Apply constraints to matrix equation [kk]{x}={ff}
%
% Synopsis:
%    [kk,ff]=feaplybc(kk,ff,bcdof,bcval)
%
% Variable Description:
%    kk - system matrix before applying constraints
%    ff - system vector before applying constraints
%    bcdof - a vector containging constrained d.o.f
%    bcval - a vector containing contained value
%
%    For example, there are constraints at d.o.f=2 and 10
%    and their constrained values are 0.0 and 2.5,
%    respectively.  Then, bcdof(1)=2 and bcdof(2)=10; and
%    bcval(1)=1.0 and bcval(2)=2.5.
%-----------------------------------------------------------

 n=length(bcdof);
 sdof=size(kk);

 for i=1:n
    c=bcdof(i);
    for j=1:sdof
       kk(c,j)=0;
    end

    kk(c,c)=1;
    ff(c)=bcval(i);
 end
```