CHAPTER FOUR

Trusses, Beams, and Frames

Computer Implementation 4.1 *(Matlab)*

The analysis of plane trusses can be performed conveniently by the two small *Matlab* functions (PlaneTruss-Element and PlaneTrussResults) presented in Chapter 1. Using these functions now we consider solution of the six bar truss shown in Figure 4.XXX. The steps are exactly those explained in similar examples in Chapter 1.

MatlabFiles\Chap4\SixBarTrussEx.m

```
% Six bar truss example
e = 200*10^3; A = 0.001*1000^2; P = 20000.;
alpha = pi/6;
nodes = 1000*[0, 0; 4, 0; 0, 3; 4, 3; 2, 2];
dof=2*length(nodes);
conn=[1,2; 2,5; 5,3; 2,4; 1,5; 5,4];
lmm = [1, 2, 3, 4; 3, 4, 9, 10; 9, 10, 5, 6;
    3, 4, 7, 8; 1, 2, 9, 10; 9, 10, 7, 8];
```

```
elems=size(lmm,1);
K=zeros(dof); R = zeros(dof,1);
debc = [1, 2, 5, 6, 7, 8];
ebcVals = zeros(length(debc),1);

%load vector
R = zeros(dof,1); R(3) = P*sin(alpha); R(4) = P*cos(alpha);

% Assemble global stiffness matrix
K=zeros(dof);
for i=1:elems
    lm=lmm(i,:);
    con=conn(i,:);
    k=PlaneTrussElement(e, A, nodes(con,:));
    K(lm, lm) = K(lm, lm) + k;
end
K
R
% Nodal solution and reactions
[d, reactions] = NodalSoln(K, R, debc, ebcVals)
results=[];
for i=1:elems
    results = [results; PlaneTrussResults(e, A, ...
            nodes(conn(i,:),:), d(lmm(i,:)))];
end
format short g
results
```

```
>> SixBarTrussEx

K =

  Columns 1 through 6

       85355       35355      -50000           0           0           0
       35355       35355           0           0           0           0
      -50000           0       85355      -35355           0           0
           0           0      -35355  1.0202e+005           0           0
           0           0           0           0       71554      -35777
           0           0           0           0      -35777       17889
           0           0           0           0           0           0
           0           0           0      -66667           0           0
      -35355      -35355      -35355       35355      -71554       35777
      -35355      -35355       35355      -35355       35777      -17889

  Columns 7 through 10
```

```
       0          0     -35355     -35355
       0          0     -35355     -35355
       0          0     -35355      35355
       0     -66667      35355     -35355
       0          0     -71554      35777
       0          0      35777     -17889
   71554      35777     -71554     -35777
   35777      84555     -35777     -17889
  -71554     -35777 2.1382e+005          0
  -35777     -17889          0 1.0649e+005


R =

       0
       0
   10000
   17321
       0
       0
       0
       0
       0
       0


d =

       0
       0
 0.21311
 0.24998
       0
       0
       0
       0
-0.0060971
0.012242


reactions =

  -10873
 -217.27
  874.27
 -437.13
```

```
    -1.7279
     -16666


results =

  5.3276e-005      10.655       10655
 -4.6334e-006    -0.92669     -926.69
 -4.8873e-006    -0.97746     -977.46
 -8.3326e-005     -16.665      -16665
  1.5363e-006     0.30727      307.27
  -9.659e-009  -0.0019318     -1.9318
```

Computer Implementation 4.2 *(Matlab)*

The analysis of space trusses can be performed conveniently by writing two small *Matlab* functions, one for defining the element stiffness matrix and the other for computing the element strain, stress, and the axial force.

MatlabFiles\Chap4\SpaceTrussElement.m

```
function k = SpaceTrussElement(e, A, coord)
% k = SpaceTrussElement(e, A, coord)
% Generates stiffness matrix of a space truss element
% e = modulus of elasticity
% A = Area of cross-section
% coord = coordinates at the element ends

x1=coord(1,1); y1=coord(1,2); z1=coord(1,3);
x2=coord(2,1); y2=coord(2,2); z2=coord(2,3);
L=sqrt((x2-x1)^2+(y2-y1)^2+(z2-z1)^2);
ls=(x2-x1)/L; ms=(y2-y1)/L; ns=(z2-z1)/L;
k = e*A/L*[ls^2, ls*ms, ls*ns, -ls^2, -(ls*ms), -(ls*ns);
    ls*ms, ms^2, ms*ns, -(ls*ms), -ms^2, -(ms*ns);
    ls*ns, ms*ns, ns^2, -(ls*ns), -(ms*ns), -ns^2;
    -ls^2, -(ls*ms), -(ls*ns), ls^2, ls*ms, ls*ns;
    -(ls*ms), -ms^2, -(ms*ns), ls*ms, ms^2, ms*ns;
    -(ls*ns), -(ms*ns), -ns^2, ls*ns, ms*ns, ns^2];
```

MatlabFiles\Chap4\SpaceTrussResults.m

```
function results = SpaceTrussResults(e, A, coord, disps)
% results = SpaceTrussResults(e, A, coord, disps)
% Compute space truss element results
% e = modulus of elasticity
% A = Area of cross-section
```

```
% coord = coordinates at the element ends
% disps = displacements at element ends

x1=coord(1,1); y1=coord(1,2); z1=coord(1,3);
x2=coord(2,1); y2=coord(2,2); z2=coord(2,3);
L=sqrt((x2-x1)^2+(y2-y1)^2+(z2-z1)^2);
ls=(x2-x1)/L; ms=(y2-y1)/L; ns=(z2-z1)/L;
T=[ls,ms,ns,0,0,0; 0,0,0,ls,ms,ns];
d = T*disps;
eps= (d(2)-d(1))/L;
sigma = e.*eps;
force = sigma.*A;
results=[eps, sigma, force];
```

Using these functions now we consider solution of the three bar truss. The steps are exactly those explained in similar examples in Chapter 1.

MatlabFiles\Chap4\ThreeBarSpaceTrussEx.m

```
% Three bar space truss example
a1 = 200; a2 = 600; e = 200000; P = 20000;
nodes = 1000*[.96, 1.92, 0; -1.44, 1.44, 0; 0, 0, 0; 0, 0, 2];
dof=3*length(nodes);
conn=[1,4; 2,4; 3,4];
lmm = [1, 2, 3, 10, 11, 12;
    4, 5, 6, 10, 11, 12;
    7, 8, 9, 10, 11, 12];
debc = [1:9];
ebcVals = zeros(length(debc),1);

%load vector
R = zeros(dof,1); R(11) = -P;

% Assemble global stiffness matrix
K=zeros(dof);
for i=1:2
    lm=lmm(i,:);
    con=conn(i,:);
    k=SpaceTrussElement(e, a1, nodes(con,:));
    K(lm, lm) = K(lm, lm) + k;
end
lm=lmm(3,:);
con=conn(3,:);
k=SPaceTrussElement(e, a2, nodes(con,:));
K(lm, lm) = K(lm, lm) + k
```

```
% Nodal solution and reactions
[d, reactions] = NodalSoln(K, R, debc, ebcVals)
results=[];
for i=1:2
    results = [results; SpaceTrussResults(e, a1, ...
            nodes(conn(i,:),:), d(lmm(i,:)))];
end
results = [results; SpaceTrussResults(e, a2, ...
        nodes(conn(3,:),:), d(lmm(3,:)))];
format short g
results
```

```
>> ThreeBarSpaceTrussEx

K =

  Columns 1 through 6

     1459.7      2919.3     -3040.9           0           0           0
     2919.3      5838.6     -6081.9           0           0           0
    -3040.9     -6081.9      6335.3           0           0           0
          0           0           0      3566.7     -3566.7      4953.8
          0           0           0     -3566.7      3566.7     -4953.8
          0           0           0      4953.8     -4953.8      6880.3
          0           0           0           0           0           0
          0           0           0           0           0           0
          0           0           0           0           0           0
    -1459.7     -2919.3      3040.9     -3566.7      3566.7     -4953.8
    -2919.3     -5838.6      6081.9      3566.7     -3566.7      4953.8
     3040.9      6081.9     -6335.3     -4953.8      4953.8     -6880.3

  Columns 7 through 12

          0           0           0     -1459.7     -2919.3      3040.9
          0           0           0     -2919.3     -5838.6      6081.9
          0           0           0      3040.9      6081.9     -6335.3
          0           0           0     -3566.7      3566.7     -4953.8
          0           0           0      3566.7     -3566.7      4953.8
          0           0           0     -4953.8      4953.8     -6880.3
          0           0           0           0           0           0
          0           0           0           0           0           0
          0           0       60000           0           0      -60000
          0           0           0      5026.4     -647.45      1912.9
          0           0           0     -647.45      9405.4      -11036
          0           0      -60000      1912.9      -11036       73216
```

```
d =

         0
         0
         0
         0
         0
         0
         0
         0
         0
  -0.18705
    -2.592
   -0.3858


reactions =

    6666.7
     13333
    -13889
   -6666.7
    6666.7
   -9259.3
         0
         0
     23148


results =

   0.00050936       101.87        20375
   0.00033036       66.072        13214
   -0.0001929       -38.58       -23148
```

Computer Implementation 4.3 *(Matlab)*

The analysis of beams can be performed conveniently by writing two small *Matlab* functions, one for defining the element stiffness matrix and the other for computing the element displacement, bending moment and shear force. For each element the results are computed at element ends and at center of the element. By adjusting the increment of s in the for loop in the BeamResults function the results can be obtained at several points and used for plotting bending moment and shear force diagrams.

MatlabFiles\Chap4\BeamElement.m

```
function [ke, rq] = BeamElement(EI, q, coord)
% [ke, rq] = BeamElement(EI, q, coord)
% Generates equations for a beam element
% EI = beam stiffness
% q = distributed load
% coord = coordinates at the element ends

L=coord(2)-coord(1);
ke = [(12*EI)/L^3, (6*EI)/L^2, -((12*EI)/L^3), (6*EI)/L^2;
    (6*EI)/L^2, (4*EI)/L, -((6*EI)/L^2), (2*EI)/L;
    -((12*EI)/L^3), -((6*EI)/L^2), (12*EI)/L^3, -((6*EI)/L^2);
    (6*EI)/L^2, (2*EI)/L, -((6*EI)/L^2), (4*EI)/L];
rq = [(L*q)/2; (L^2*q)/12; (L*q)/2; -((L^2*q)/12)];
```

MatlabFiles\Chap4\BeamResults.m

```
function [v, bm, V] = BeamResults(EI, q, coord, dn)
% [v, bm, V] = BeamResults(EI, q, coord, dn)
% Generates beam element results
% EI = beam stiffness
% q = distributed load
% coord = coordinates at the element ends
% dn = nodal solution
L=coord(2)-coord(1);
v=[]; bm=[]; V=[];
% Change increment to get results at more points
for s=0:L/2:L
    n = [(2*s^3)/L^3 - (3*s^2)/L^2 + 1, s^3/L^2 - (2*s^2)/L + s, ...
            (3*s^2)/L^2 - (2*s^3)/L^3, s^3/L^2 - s^2/L];
    v = [v; [coord(1)+s, n*dn+(q*(L - s)^2*s^2)/(24*EI)]];
    dn2 = [(12*s)/L^3 - 6/L^2, (6*s)/L^2 - 4/L, 6/L^2 - (12*s)/L^3, ...
            (6*s)/L^2 - 2/L];
    bm = [bm; [coord(1)+s, EI*dn2*dn+(q*(L^2 - 6*s*L + 6*s^2))/(12)]];
    dn3 = [12/L^3, 6/L^2, -(12/L^3), 6/L^2];
    V = [V; [coord(1)+s, EI*dn3*dn+((q*(12*s - 6*L))/(12))]];
end
```

Using these functions now we consider solution of the three element model. The steps are exactly those used in other *Matlab* implementations.

MatlabFiles\Chap4\BeamEx.m

```
% Beam example
F = 18; q = 10; EI = (210*10^6)*4*10^(-4);
L=2; nodes =[0:L:3*L]; n=2*length(nodes);
debc=[1,2,7]; ebcVals=zeros(length(debc),1);
```

```
K=zeros(n); R = zeros(n,1); R(3)=-F;
% Generate equations for each element and assemble them.
for i=1:2
   lm=[2*(i-1)+1,2*(i-1)+2,2*(i-1)+3,2*(i-1)+4];
   [ke, rq] = BeamElement(2*EI, 0, nodes([i:i+1]));
   K(lm, lm) = K(lm, lm) + ke;
   R(lm) = R(lm) + rq;
end
for i=3
   lm=[2*(i-1)+1,2*(i-1)+2,2*(i-1)+3,2*(i-1)+4];
   [ke, rq] = BeamElement(EI, -q, nodes([i:i+1]));
   K(lm, lm) = K(lm, lm) + ke;
   R(lm) = R(lm) + rq;
end
K
R
% Nodal solution and reactions
d = NodalSoln(K, R, debc, ebcVals)
va=[]; bma=[]; Va=[];
for i=1:2
   lm=[2*(i-1)+1,2*(i-1)+2,2*(i-1)+3,2*(i-1)+4];
   [v, bm, V]=BeamResults(2*EI, 0, nodes([i:i+1]), d(lm));
   va = [va; v]; bma = [bma; bm]; Va = [Va; V];
end
for i=3
   lm=[2*(i-1)+1,2*(i-1)+2,2*(i-1)+3,2*(i-1)+4];
   [v, bm, V]=BeamResults(EI, -q, nodes([i:i+1]), d(lm));
   va = [va; v]; bma = [bma; bm]; Va = [Va; V];
end
va
bma
Va
```

```
>> BeamEx

K =

  Columns 1 through 6

     252000      252000     -252000      252000           0           0
     252000      336000     -252000      168000           0           0
    -252000     -252000      504000           0     -252000      252000
     252000      168000           0      672000     -252000      168000
          0           0     -252000     -252000      378000     -126000
          0           0      252000      168000     -126000      504000
          0           0           0           0     -126000     -126000
          0           0           0           0      126000       84000
```

```
 Columns 7 through 8

          0           0
          0           0
          0           0
          0           0
    -126000      126000
    -126000       84000
     126000     -126000
    -126000      168000


R =

          0
          0
        -18
          0
        -10
    -3.3333
        -10
     3.3333


d =

          0
          0
 -0.00021315
 -0.00013138
 -0.00034127
 1.3605e-005
          0
  0.00026899


va =

          0            0
          1 -7.3732e-005
          2  -0.00021315
          2  -0.00021315
          3  -0.00031346
          4  -0.00034127
          4  -0.00034127
```

```
   5  -0.00023944
   6           0


bma =

   0     -31.643
   1     -11.036
   2      9.5714
   2      9.5714
   3      12.179
   4      14.786
   4      14.786
   5      12.393
   6  -1.199e-014


Va =

   0      20.607
   1      20.607
   2      20.607
   2      2.6071
   3      2.6071
   4      2.6071
   4      2.6071
   5     -7.3929
   6     -17.393
```

Computer Implementation 4.4 *(Matlab)*

The analysis of plane frames can be performed conveniently by writing two simple *Matlab* functions, one for defining the element stiffness matrix and the other for computing the element axial force, bending moment, and the shear force.

MatlabFiles\Chap4\PlaneFrameElement.m

```
function [ke, rq] = PlaneFrameElement(modulus, inertia, A, qs, qt, coord)
% [ke, rq] = PlaneFrameElement(modulus, inertia, A, qs, qt, coord)
% Generates equations for a plane frame element
% modulus = modulus of elasticity
% inertia = moment of inertia
% A = area of cross-section
% qs = distributed load along the element axis
% qt = distributed load normal to the element axis
```

```
% coord = coordinates at the element ends

EI=modulus*inertia; EA = modulus*A;
x1=coord(1,1); y1=coord(1,2);
x2=coord(2,1); y2=coord(2,2);
L=sqrt((x2-x1)^2+(y2-y1)^2);
ls=(x2-x1)/L; ms=(y2-y1)/L;

ke = [(EA*L^2*ls^2 + 12*EI*ms^2)/L^3, ((-12*EI + EA*L^2)*ls*ms)/L^3, ...
        (-6*EI*ms)/L^2, -((EA*L^2*ls^2 + 12*EI*ms^2)/L^3), ...
        ((12*EI - EA*L^2)*ls*ms)/L^3, (-6*EI*ms)/L^2;
    ((-12*EI + EA*L^2)*ls*ms)/L^3, (12*EI*ls^2 + EA*L^2*ms^2)/L^3, ...
        (6*EI*ls)/L^2, ((12*EI - EA*L^2)*ls*ms)/L^3, ...
        -((12*EI*ls^2 + EA*L^2*ms^2)/L^3), (6*EI*ls)/L^2;
    (-6*EI*ms)/L^2, (6*EI*ls)/L^2, (4*EI)/L, ...
        (6*EI*ms)/L^2, (-6*EI*ls)/L^2,(2*EI)/L;
    -((EA*L^2*ls^2 + 12*EI*ms^2)/L^3), ((12*EI -EA*L^2)*ls*ms)/L^3, ...
        (6*EI*ms)/L^2, (EA*L^2*ls^2 + 12*EI*ms^2)/L^3, ...
        ((-12*EI + EA*L^2)*ls*ms)/L^3, (6*EI*ms)/L^2;
    ((12*EI - EA*L^2)*ls*ms)/L^3, -((12*EI*ls^2 + EA*L^2*ms^2)/L^3),...
        (-6*EI*ls)/L^2, ((-12*EI + EA*L^2)*ls*ms)/L^3, ...
        (12*EI*ls^2 + EA*L^2*ms^2)/L^3, (-6*EI*ls)/L^2;
    (-6*EI*ms)/L^2, (6*EI*ls)/L^2, (2*EI)/L, (6*EI*ms)/L^2, ...
        (-6*EI*ls)/L^2, (4*EI)/L];
rq = [(L*(ls*qs - ms*qt))/2; (L*(ms*qs + ls*qt))/2;(L^2*qt)/12;
    (L*(ls*qs - ms*qt))/2; (L*(ms*qs + ls*qt))/2; -(L^2*qt)/12];
```

MatlabFiles\Chap4\PlaneFrameResults.m

```
function [f, bm, V] = PlaneFrameResults(modulus, inertia, A, ...
    qs, qt, coord, dn)
% [f, bm, V] = PlaneFrameResults(modulus, inertia, A, ...
%    qs, qt, coord, dn)
% Generates frame element results
% modulus = modulus of elasticity
% inertia = moment of inertia
% A = area of cross-section
% qs = distributed load along the element axis
% qt = distributed load normal to the element axis
% coord = coordinates at the element ends
% dn = nodal solution
% [f, bm, V] = [axial force, bending moment, shear]

EI=modulus*inertia; EA = modulus*A;
x1=coord(1,1); y1=coord(1,2);
x2=coord(2,1); y2=coord(2,2);
L=sqrt((x2-x1)^2+(y2-y1)^2);
```

```
ls=(x2-x1)/L; ms=(y2-y1)/L;

u = dn([1,4]);
v = dn([2,3,5,6]);

f=[]; bm=[]; V=[];
% Change increment to get results at more points
for s=0:L/2:L
   x = x1 + s*ls; y = y1+ s*ms;
   f = [f; [x,y,EA*(u(2)-u(1))/L]];
   dn2 = [(12*s)/L^3 - 6/L^2, (6*s)/L^2 - 4/L, 6/L^2 - (12*s)/L^3, ...
         (6*s)/L^2 - 2/L];
   bm = [bm; [x, y, EI*dn2*v+(qt*(L^2 - 6*s*L + 6*s^2))/(12)]];
   dn3 = [12/L^3, 6/L^2, -(12/L^3), 6/L^2];
   V = [V; [x, y, EI*dn3*v+((qt*(12*s - 6*L))/(12))]];
end
```

Using these functions now we consider solution of the two element model.

MatlabFiles\Chap4\PlaneFrameEx.m

```
% Plane frame example
e = 30000; a = 100; inertia = 1000; L = 15*12; q = 1/12;
nodes = [0, 0; L/sqrt(2), L/sqrt(2); L + L/sqrt(2), L/sqrt(2)];
conn=[1,2; 2,3];
lmm=[1,2,3,4,5,6; 4,5,6,7,8,9];
n=3*length(nodes);
debc=[1,2,3,7,8,9]; ebcVals=zeros(length(debc),1);
K=zeros(n); R = zeros(n,1);
% Generate equations for each element and assemble them.
for i=1
   lm=lmm(i,:);
   con=conn(i,:);
   [ke, rq] = PlaneFrameElement(e, inertia, a, 0, -q, nodes(con,:));
   K(lm, lm) = K(lm, lm) + ke;
   R(lm) = R(lm) + rq;
end
for i=2
   lm=lmm(i,:);
   con=conn(i,:);
   [ke, rq] = PlaneFrameElement(e, inertia, a, 0, 0, nodes(con,:));
   K(lm, lm) = K(lm, lm) + ke;
   R(lm) = R(lm) + rq;
end
K
R
```

```
% Nodal solution and reactions
d = NodalSoln(K, R, debc, ebcVals)
fa=[]; bma=[]; Va=[];
for i=1
   lm=lmm(i,:);
   con=conn(i,:);
   [f, bm, V]=PlaneFrameResults(e, inertia, a, 0, -q, ...
       nodes(con,:), d(lm));
   fa = [fa; f]; bma = [bma; bm]; Va = [Va; V];
end
for i=2
   lm=lmm(i,:);
   con=conn(i,:);
   [f, bm, V]=PlaneFrameResults(e, inertia, a, 0, 0, ...
       nodes(con,:), d(lm));
   fa = [fa; f]; bma = [bma; bm]; Va = [Va; V];
end
fa
bma
Va
```

```
>> PlaneFrameEx

K =

  Columns 1 through 6

     8364.2      8302.5     -3928.4     -8364.2     -8302.5     -3928.4
     8302.5      8364.2      3928.4     -8302.5     -8364.2      3928.4
    -3928.4      3928.4  6.6667e+005      3928.4     -3928.4  3.3333e+005
    -8364.2     -8302.5      3928.4       25031      8302.5      3928.4
    -8302.5     -8364.2     -3928.4      8302.5      8425.9      1627.2
    -3928.4      3928.4  3.3333e+005      3928.4      1627.2  1.3333e+006
          0           0           0      -16667           0           0
          0           0           0           0     -61.728     -5555.6
          0           0           0           0      5555.6  3.3333e+005

  Columns 7 through 9

          0           0           0
          0           0           0
          0           0           0
     -16667           0           0
          0     -61.728      5555.6
          0     -5555.6  3.3333e+005
      16667           0           0
          0      61.728     -5555.6
```

```
         0     -5555.6  6.6667e+005


R =

    5.3033
   -5.3033
      -225
    5.3033
   -5.3033
       225
         0
         0
         0


d =

         0
         0
         0
0.00060161
-0.0012547
0.00016851
         0
         0
         0


fa =

         0            0       10.027
     63.64        63.64       10.027
    127.28       127.28       10.027
    127.28       127.28      -10.027
    217.28       127.28      -10.027
    307.28       127.28      -10.027


bma =

         0            0      -288.14
     63.64        63.64       140.58
    127.28       127.28      -105.69
    127.28       127.28      -105.37
    217.28       127.28      -28.085
```

```
    307.28      127.28      49.199


Va =

         0           0       8.5136
     63.64       63.64       1.0136
    127.28      127.28      -6.4864
    127.28      127.28      0.85871
    217.28      127.28      0.85871
    307.28      127.28      0.85871
```

Computer Implementation 4.5 *(Matlab)*

The analysis of plane frames can be performed conveniently by writing two *Matlab* functions, one for defining the element stiffness matrix and the other for computing the element axial force, bending moment, and the shear force.

MatlabFiles\Chap4\SpaceFrameElement.m

```
function [ke, rq] = SpaceFrameElement(e, G, Ir, Is, J, A, qr, qs, coord)
% [ke, rq] = SpaceFrameElement(e, G, Ir, Is, J, A, qr, qs, coord)
% Generates equations for a space frame element
% e = modulus of elasticity
% G = shear modulus
% Ir, Is = moment of inertias about element r and s axes
% J = torsional rigity
% A = area of cross-section
% qr, qs = distributed loads along the element r and s axes
% coord = coordinates at the element ends

EIr=e*Ir; EIs=e*Is; GJ=G*J; EA = e*A;
n1=coord(1,1:3); n2=coord(2,1:3); n3=coord(3,1:3);
L=sqrt(dot((n2-n1),(n2-n1)));
ex = (n2 - n1)/L;
eyy = cross(n3 - n1, n2 - n1);
ey = eyy/sqrt(dot(eyy,eyy));
ez = cross(ex, ey);
H = [ex; ey; ez];
T = zeros(12);
T([1, 2, 3], [1, 2, 3]) = H;
T([4,5,6], [4,5,6]) = H;
T([7,8,9], [7,8,9]) = H;
T([10,11,12], [10,11,12]) = H;
TT = T';
```

```
ke = [EA/L, 0, 0, 0, 0, 0, -(EA/L), 0, 0, 0, 0, 0;
    0, (12*EIr)/L^3, 0, 0, 0, (6*EIr)/L^2, 0, -((12*EIr)/L^3), ...
        0,0, 0, (6*EIr)/L^2;
    0, 0, (12*EIs)/L^3, 0, -((6*EIs)/L^2), 0, ...
        0, 0, -((12*EIs)/L^3), 0, -((6*EIs)/L^2), 0;
    0, 0, 0, GJ/L, 0, 0, 0, 0, 0, -(GJ/L), 0, 0;
    0, 0, -((6*EIs)/L^2), 0, (4*EIs)/L, 0, 0, 0, (6*EIs)/L^2, ...
        0,(2*EIs)/L, 0;
    0, (6*EIr)/L^2, 0, 0, 0, (4*EIr)/L, 0, ...
        -((6*EIr)/L^2), 0, 0, 0, (2*EIr)/L;
    -(EA/L), 0, 0, 0, 0, 0, EA/L, 0, 0, 0, 0, 0;
    0, -((12*EIr)/L^3), 0, 0, 0, -((6*EIr)/L^2), ...
        0, (12*EIr)/L^3, 0, 0, 0, -((6*EIr)/L^2);
    0, 0, -((12*EIs)/L^3), 0, (6*EIs)/L^2, 0, 0, 0, ...
        (12*EIs)/L^3, 0, (6*EIs)/L^2, 0;
    0, 0, 0, -(GJ/L), 0, 0, 0, 0, 0, GJ/L, 0, 0;
    0, 0, -((6*EIs)/L^2), 0, (2*EIs)/L, 0, 0,0, (6*EIs)/L^2, ...
        0, (4*EIs)/L, 0;
    0, (6*EIr)/L^2, 0, 0, 0, (2*EIr)/L, 0, -((6*EIr)/L^2), 0, ...
        0, 0, (4*EIr)/L];
ke = TT*ke*T;
rq = TT*[0; (L*qs)/2; (L*qr)/2; 0; -((L^2*qr)/12); ...
        (L^2*qs)/12; 0; (L*qs)/2;
    (L*qr)/2; 0; (L^2*qr)/12; -((L^2*qs)/12)];
```

MatlabFiles\Chap4\SpaceFrameResults.m

```
function [f, bmr, bms, bmt, Vr, Vs] = SpaceFrameResults(e, G, Ir, Is, J, A, qr, ...
    qs, coord, dn)
% [f, bmr, bms, bmt, Vr, Vs] = SpaceFrameResults(e, G, Ir, Is, J, A, qr, ...
%   qs, coord, dn)
% Computes results for a space frame element
% e = modulus of elasticity
% G = shear modulus
% Ir, Is = moment of inertias about element r and s axes
% J = torsional rigity
% A = area of cross-section
% qr, qs = distributed loads along the element r and s axes
% coord = coordinates at the element ends
% dn = nodal solution
% The output variables are
% f = axial force, bmr, bms = bending moments about r and s axes,
%   bmt = twisting moment, Vr, Vs = shear forces about r and s axes.

EIr=e*Ir; EIs=e*Is; GJ=G*J; EA = e*A;
n1=coord(1,1:3); n2=coord(2,1:3); n3=coord(3,1:3);
L=sqrt(dot((n2-n1),(n2-n1)));
```

```
ex = (n2 - n1)/L;
eyy = cross(n3 - n1, n2 - n1);
ey = eyy/sqrt(dot(eyy,eyy));
ez = cross(ex, ey);
H = [ex; ey; ez];
T = zeros(12);
T([1, 2, 3], [1, 2, 3]) = H;
T([4,5,6], [4,5,6]) = H;
T([7,8,9], [7,8,9]) = H;
T([10,11,12], [10,11,12]) = H;
TT = T';

dl = T*dn;
u = dl([1,7]); tw=dl([4,10]);
v = dl([2, 6, 8, 12]); w = dl([3, 5, 9, 11]);
f=[]; bmr=[]; bms=[]; bmt=[]; Vr=[]; Vs=[];
% Change increment to get results at more points
for s=0:L/2:L
   x = n1(1) + s*H(1,1); y = n1(2)+ s*H(1,2); z = n1(3)+ s*H(1,3);
   f = [f; [x,y,z, EA*(-u(1)+u(2))/L]];
   bmt = [bmt; [x,y,z, GJ*(-tw(1)+tw(2))/L]];
   dnv2 = [(12*s)/L^3 - 6/L^2, (6*s)/L^2 - 4/L, 6/L^2 - ...
         (12*s)/L^3, (6*s)/L^2 - 2/L];
   dnw2=[dnv2(1), -dnv2(2), dnv2(3), -dnv2(4)];
   bmr = [bmr; [x, y, z, EIr*dnv2*v+(qs*(L^2 - 6*s*L + ...
            6*s^2))/(12)]];
   bms = [bms; [x, y, z, -EIs*dnw2*w-(qr*(L^2 - 6*s*L + ...
            6*s^2))/(12)]];
   dnv3 = [12/L^3, 6/L^2, -(12/L^3), 6/L^2];
   Vs = [Vs; [x, y,z, EIr*dnv3*v+((qs*(12*s - 6*L))/(12))]];
   dnw3=[dnv3(1), -dnv3(2), dnv3(3), -dnv3(4)];
   Vr = [Vr; [x, y, z, EIs*dnw3*w+((qr*(12*s - 6*L))/(12))]];
end
```

Using these functions now we consider solution of the simple space frame model. The steps are exactly those used in other *Matlab* implementations.

Beams: $A = 3.2\,\text{in}^2; \quad J = 43\,\text{in}^4; \quad I_{\max} = I_r = 450\,\text{in}^4; \quad I_{\min} = I_s = 32\,\text{in}^2$

Columns: $A = 4\,\text{in}^2; \quad J = 60\,\text{in}^4; \quad I_{\max} = I_r = 650\,\text{in}^4; \quad I_{\min} = I_s = 54\,\text{in}^2$

MatlabFiles\Chap4\SpaceFrameEx.m

```
% Space frame example
ab = 3.2; Jb = 43; Irb = 450; Isb = 32;
ac = 4; Jc = 60; Irc = 650; Isc = 54;
```

```
q = 2./12; e = 29000.; G = 11200.;
L = 10.*12; h = 12.*12;
nodes = [0, 0, 0; 0, 0, h; L/2, 0, h; 0, L/2, h];
conn=[1,2,4; 2,3,4; 2,4,3];
lmm=[1:12; 7:18; [7:12 19:24]];
n=6*length(nodes);
debc=[1,2,3,13, 17,18, 20, 22, 24]; ebcVals=zeros(length(debc),1);
K=zeros(n); R = zeros(n,1);
% Generate equations for each element and assemble them.
for i=1
   lm=lmm(i,:);
   con=conn(i,:);
   [ke rq] = SpaceFrameElement(e, G, Irc, Isc, Jc, ac, 0, 0, ...
      nodes(con,:));
   K(lm, lm) = K(lm, lm) + ke;
   R(lm) = R(lm) + rq;
end
for i=2
   lm=lmm(i,:);
   con=conn(i,:);
   [ke rq] = SpaceFrameElement(e, G, Irb, Isb, Jb, ab, 0, q, ...
      nodes(con,:));
   K(lm, lm) = K(lm, lm) + ke;
   R(lm) = R(lm) + rq;
end
for i=3
   lm=lmm(i,:);
   con=conn(i,:);
   [ke rq] = SpaceFrameElement(e, G, Irb, Isb, Jb, ab, 0, -q, ...
      nodes(con,:));
   K(lm, lm) = K(lm, lm) + ke;
   R(lm) = R(lm) + rq;
end

% Nodal solution and reactions
format short g;
d = NodalSoln(K, R, debc, ebcVals)
fa=[]; bmra=[]; bmsa=[]; bmta=[]; Vra=[]; Vsa=[];
for i=1
   lm=lmm(i,:);
   con=conn(i,:);
   [f, bmr, bms, bmt, Vr, Vs]=SpaceFrameResults(e, G, Irc, Isc, ...
      Jc, ac, 0, 0, nodes(con,:), d(lm));
   fa = [fa; f]; bmra = [bmra; bmr];
   bmsa = [bmsa; bms]; bmta = [bmta; bmt];
   Vra = [Vra; Vr]; Vsa = [Vsa; Vs];
```

```
end
for i=2
   lm=lmm(i,:);
   con=conn(i,:);
   [f, bmr, bms, bmt, Vr, Vs]=SpaceFrameResults(e, G, Irb, Isb, ...
      Jb, ab, 0, q, nodes(con,:), d(lm));
   fa = [fa; f]; bmra = [bmra; bmr];
   bmsa = [bmsa; bms]; bmta = [bmta; bmt];
   Vra = [Vra; Vr]; Vsa = [Vsa; Vs];
end
for i=3
   lm=lmm(i,:);
   con=conn(i,:);
   [f, bmr, bms, bmt, Vr, Vs]=SpaceFrameResults(e, G, Irb, Isb, ...
      Jb, ab, 0, -q, nodes(con,:), d(lm));
   fa = [fa; f]; bmra = [bmra; bmr];
   bmsa = [bmsa; bms]; bmta = [bmta; bmt];
   Vra = [Vra; Vr]; Vsa = [Vsa; Vs];
end
fa
bmra
bmsa
bmta
Vra
Vsa
```

```
>> SpaceFrameEx

d =

          0
          0
          0
  0.00039863
 -0.00015917
 1.3104e-020
   0.0005754
  0.00011703
   -0.024828
 -0.00079971
  0.00033033
 1.3104e-020
          0
  0.00011703
   -0.041634
 -0.00079971
          0
```

```
         0
  0.0005754
         0
 -0.055715
         0
0.00033033
         0


fa =

         0         0         0       -20
         0         0        72       -20
         0         0       144       -20
         0         0       144  -0.88996
        30         0       144  -0.88996
        60         0       144  -0.88996
         0         0       144    -0.181
         0        30       144    -0.181
         0        60       144    -0.181


bmra =

         0         0         0 -2.8422e-014
         0         0        72     64.077
         0         0       144     128.15
         0         0       144     128.15
        30         0       144    -96.846
        60         0       144    -171.85
         0         0       144    -26.064
         0        30       144     198.94
         0        60       144     273.94


bmsa =

         0         0         0  3.5527e-015
         0         0        72    -13.032
         0         0       144    -26.064
         0         0       144  5.5511e-017
        30         0       144  2.0268e-016
        60         0       144  3.3307e-016
         0         0       144  2.2204e-016
         0        30       144 -2.0268e-016
         0        60       144 -5.5511e-016
```

bmta =

0	0	0	0
0	0	72	0
0	0	144	0
0	0	144	8.7025e-016
30	0	144	8.7025e-016
60	0	144	8.7025e-016
0	0	144	-1.7405e-015
0	30	144	-1.7405e-015
0	60	144	-1.7405e-015

Vra =

0	0	0	0.181
0	0	72	0.181
0	0	144	0.181
0	0	144	-5.2042e-018
30	0	144	-5.2042e-018
60	0	144	-5.2042e-018
0	0	144	1.0408e-017
0	30	144	1.0408e-017
0	60	144	1.0408e-017

Vsa =

0	0	0	0.88996
0	0	72	0.88996
0	0	144	0.88996
0	0	144	-10
30	0	144	-5
60	0	144	-3.5527e-015
0	0	144	10
0	30	144	5
0	60	144	-7.1054e-015