Computer Implementation 4.5 ***(Matlab)*** *Space frame (p. 293)*

The analysis of space 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