CHAPTER EIGHT

Transient Problems

Computer Implementation 8.1 *(Matlab)*

A transient analysis can be implemented in *Matlab* by a simple extension of the functions presented in earlier chapters. In this section the triangular element functions presented in Chapter 5 are extended to handle transient analysis of two dimensional boundary value problems.

MatlabFiles\Chap8\TransientBVPTriElement.m

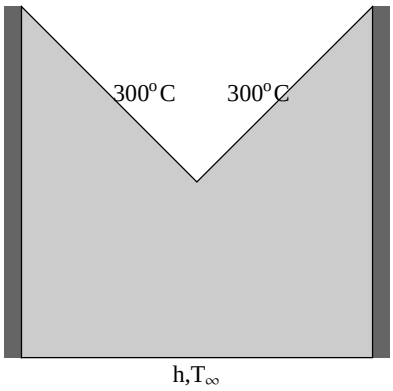
```
function [km, ke, rq] = TransientBVPTriElement(kx, ky, p, q, m, coord)
% [km, ke, rq] = TransientBVPTriElement(kx, ky, p, q, m, coord)
% Generates for a triangular element for 2d BVP
% kx, ky, p, q, m = parameters defining the BVP
% coord = coordinates at the element ends

x1=coord(1,1); y1=coord(1,2);
x2=coord(2,1); y2=coord(2,2);
x3=coord(3,1); y3=coord(3,2);
```

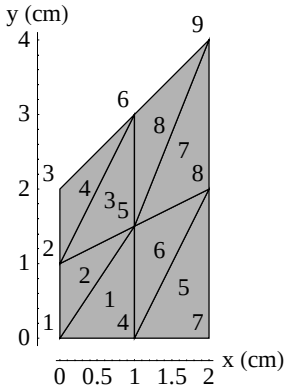
```
b1 = y2 - y3; b2 = y3 - y1; b3 = y1 - y2;
c1 = x3 - x2; c2 = x1 - x3; c3 = x2 - x1;
f1 = x2*y3 - x3*y2; f2 = x3*y1 - x1*y3; f3 = x1*y2 - x2*y1;
A = (f1 + f2 + f3)/2;
kxx = 1/(4*A)*kx*[b1^2, b1*b2, b1*b3;
   b1*b2, b2^2, b2*b3; b1*b3, b2*b3, b3^2];
kyy = 1/(4*A)*ky*[c1^2, c1*c2, c1*c3;
   c1*c2, c2^2, c2*c3; c1*c3, c2*c3, c3^2];
kp = -(p*A/12)*[2, 1, 1; 1, 2, 1; 1, 1, 2];
ke = kxx + kyy + kp;
rq = 1/3*q*A*[1; 1; 1];
km = m*A/12 * [2,1,1; 1,2,1; 1,1,2];
```

MatlabFiles\Chap8\BVPTriNBCTerm.m

```
function [ka, rb] = BVPTriNBCTerm(side, alpha, beta, coord)
% [ka, rb] = BVPTriNBCTerm(side, alpha, beta, coord)
% Generates kalpha and rbeta when NBC is specified along a side
% side = side over which the NBC is specified
% alpha and beta = coefficients specifying the NBC
% coord = coordinates at the element ends

x1=coord(1,1); y1=coord(1,2);
x2=coord(2,1); y2=coord(2,2);
x3=coord(3,1); y3=coord(3,2);
switch (side)
case 1
   L = sqrt((x2-x1)^2+(y2-y1)^2);
   ka = -alpha*L/6 * [2,1,0; 1,2,0; 0,0,0];
   rb = beta *L/2 * [1; 1; 0];
case 2
   L = sqrt((x2-x3)^2+(y2-y3)^2);
   ka = -alpha*L/6 * [0,0,0; 0,2,1; 0,1,2];
   rb = beta *L/2 * [0; 1; 1];
case 3
   L = sqrt((x3-x1)^2+(y3-y1)^2);
   ka = -alpha*L/6 * [2,0,1; 0,0,0; 1,0,2];
   rb = beta *L/2 * [1; 0; 1];
end
```

Using these functions now we consider solution of the transient heat flow problem using an 8 element model. We assemble $\boldsymbol{M}$, $\boldsymbol{K}$, matrices and $\boldsymbol{R}$ vector from the corresponding element quantities in the usual manner. Finally the equations are expressed in a form suitable for the differential equation solver (ode23) in *Matlab.*





MatlabFiles\Chap8\HeatODE.m

```
% % Transient heat flow through V-groove
global Mf Kf Rf
h = 200; Tinf = 50; rho = 1600; cp = 800; t0=300;
nodes = (1/100)* [0, 0; 0, 1; 0, 2;
   1, 0; 1, 3/2; 1, 3; 2, 0; 2, 2; 2, 4];
kx = 3; ky = 3; p = 0; q = 0;
lmm = [1, 4, 5; 5, 2, 1; 2, 5, 6;
   6, 3, 2; 4, 7, 8; 8, 5, 4; 5, 8, 9; 9, 6, 5];
debc = [3,6,9]; ebcVals=t0*ones(length(debc),1);
dof=length(nodes); elems=size(lmm,1);
M=zeros(dof); K=zeros(dof); R = zeros(dof,1);

% Generate equations for each element and assemble them.
for i=1:elems
   lm = lmm(i,:);
   [m, k, r] = TransientBVPTriElement(kx, ky, p, q, rho*cp, nodes(lm,:));
   M(lm, lm) = M(lm, lm) + m;
   K(lm, lm) = K(lm, lm) + k;
   R(lm) = R(lm) + r;
end

% Compute and assemble NBC contributions
lm = lmm(1,:);
[k, r] = BVPTriNBCTerm(1, -h, h*Tinf, nodes(lm,:));
K(lm, lm) = K(lm, lm) + k;
R(lm) = R(lm) + r;

lm = lmm(5,:);
```

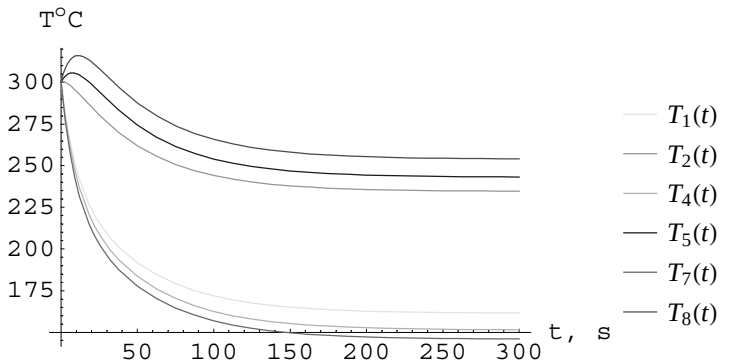
```
[k, r] = BVPTriNBCTerm(1, -h, h*Tinf, nodes(lm,:));
K(lm, lm) = K(lm, lm) + k;
R(lm) = R(lm) + r;
% Adjust for essential boundary conditions
dof = length(R);
df = setdiff(1:dof, debc);
Mf = M(df, df);
Kf = K(df, df);
Rf = R(df) - K(df, debc)*ebcVals;

% Setup and solve the resulting first order differential equations
d0 = t0*ones(length(Mf),1);
[t,d] = ode23('HeatODE',[0,300],d0);
plot(t,d(:,1),'-r',t,d(:,2),'-g',t,d(:,3),...
    '-c',t,d(:,4),'-m',t,d(:,5),'-y',t,d(:,6),'-k');
legend('T1', 'T2', 'T4', 'T5', 'T7', 'T8');
hold off
[t(1:10),d(1:10,:)]
```

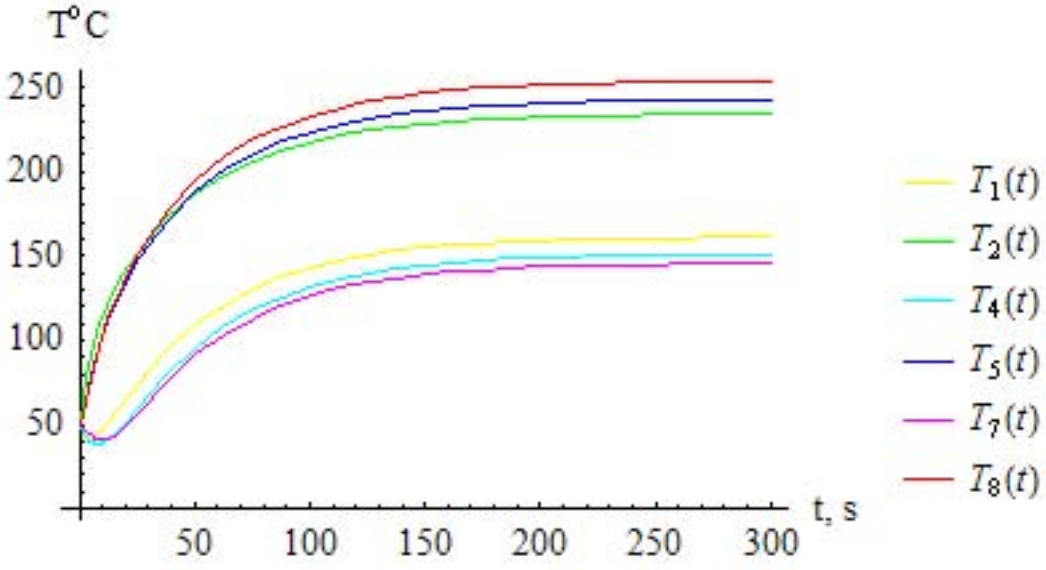
MatlabFiles\Chap8\HeatODE.m

```
function ddot = HeatODE(t, d)
% ddot = HeatODE(t, d)
% function to set up equations for a transient heat flow problem
global Mf Kf Rf
ddot = inv(Mf)*(Rf - Kf*d);
```

```
>> VGrooveHeatEx

ans =

  Columns 1 through 6

         0        300        300        300        300        300
    2.5083     282.52     300.31     281.59     303.27     278.91
    6.2557     262.96     298.19        260     305.61      255.9
      11.6     243.36     293.48     238.45     304.89      233.4
    17.131     229.23     288.35     223.63     301.15     217.43
    20.502     222.87     284.77     216.46     298.54     210.71
    23.873     217.31     281.67     210.73     295.47     204.67
    30.256     208.96     275.82     201.64     289.87     195.93
    33.809     204.96     273.13     197.74     286.61     191.56
    37.363     201.57     270.28        194     283.73     188.09

  Column 7

        300
```

307.69
313.87
316.24
314.47
312.01
309.33
303.65
300.6
297.47




T°C
250
200
150
100
50
50
100
150
200
250
300
t, s
$T_1(t)$
$T_2(t)$
$T_4(t)$
$T_5(t)$
$T_7(t)$
$T_8(t)$

Computer Implementation 8.2 ***(Matlab)*** *Modal analysis of a plane truss*

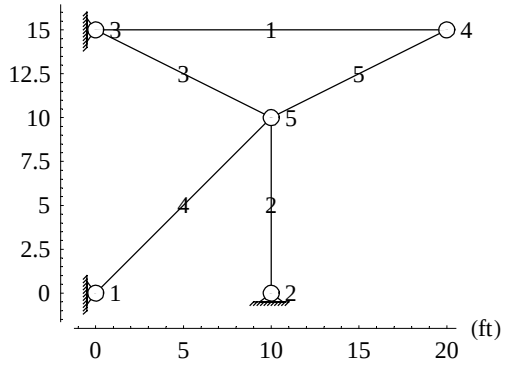
The following TransientPlaneTrussElement function returns the mass and the stiffness matrix of a plane truss element. To generate the mass matrix the function needs the mass density (ρ).

MatlabFiles\Chap8\TransientPlaneTrussElement.m

```
function [m, k] = TransientPlaneTrussElement(e, A, rho, coord)
% [m, k] = TransientPlaneTrussElement(e, A, rho, coord)
% Generates mass & stiffness matrices for a plane truss element
% rho = mass density
% e = modulus of elasticity
% A = area of cross-section
% coord = coordinates at the element ends

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;
k = e*A/L*[ls^2, ls*ms,-ls^2,-ls*ms;
   ls*ms, ms^2,-ls*ms,-ms^2;
   -ls^2,-ls*ms,ls^2,ls*ms;
```

```
   -ls*ms,-ms^2,ls*ms,ms^2];
m = ((rho*A*L)/6)*[2, 0, 1, 0; 0, 2, 0, 1;
   1, 0, 2, 0;0, 1, 0, 2];
```

Using this function we consider free vibration analysis of the plane truss structure shown. All members have the same cross-sectional area and are made of the same material, $\rho = 490\,\text{lb}/\text{ft}^3$, $E = 30\times10^6\,\text{lb}/\text{in}^2$ and $A = 1.25\,\text{in}^2$. The truss supports a rotating machine weighing 2000 lbs at its tip.



The weight of the machine is an added mass at node 4. This is defined as ma in the following line and is added directly to the global mass matrix corresponding to degrees of freedom 7 and 8 associated with node 4. The procedure for generating the global mass and stiffness matrix is exactly the same as that for the static analysis used in the previous chapters. Using in-lb units the calculations are as follows.

MatlabFiles\Chap8\ModalTrussEx.m

```
% Modal analysis of a plane truss
g = 386.4; e = 30*10^6; A = 1.25;
rho = (490/(12^3))/g; ma = 2000/g;
nodes = 12 * [0, 0; 10, 0; 0, 15; 20, 15; 10, 10];
conn = [3, 4; 2, 5; 5, 3; 1, 5; 5, 4];
elems = size(conn,1);
lmm=[];
for i=1:elems
   lmm = [lmm; [2*conn(i,1)-1, 2*conn(i,1),2*conn(i,2)-1, 2*conn(i,2)]];
end
debc = [1:6]; ebcVals=zeros(length(debc),1);
dof=2*size(nodes,1);
M=zeros(dof); K=zeros(dof);
% Add nodal masses to the global M matrix.
```

```
M(7,7)=ma; M(8,8) = ma;
% Generate equations for each element and assemble them.
for i=1:elems
   con = conn(i,:);
   lm = lmm(i,:);
   [m, k] = TransientPlaneTrussElement(e, A, rho, nodes(con,:));
   M(lm, lm) = M(lm, lm) + m;
   K(lm, lm) = K(lm, lm) + k;
end

% Adjust for essential boundary conditions
dof = length(R);
df = setdiff(1:dof, debc);
Mf = M(df, df);
Kf = K(df, df);

% Compute frequencies and mode shapes
[V, lam] = eig(Kf, Mf);
freq=sqrt(lam)
modeShapes = V
```

```
>> ModalTrussEx

freq =

    55.835        0        0        0
        0   235.46        0        0
        0        0   1598.4        0
        0        0        0   1971.6


modeShapes =

    0.099847   -0.42131   -0.01827   -0.036118
   -0.42305   -0.098538  0.0011071   -0.020645
  -0.041882   -0.18022     1.6041      1.8044
   -0.01476  -0.062462    -1.8123      1.6046
```

Computer Implementation 8.3 *(Matlab)* *Modal analysis of a plane frame*

The following TransientPlaneFrameElement function returns the mass and the stiffness matrix of a plane frame element. To generate the mass matrix the function needs the mass density (ρ).

MatlabFiles\Chap8\TransientPlaneFrameElement.m

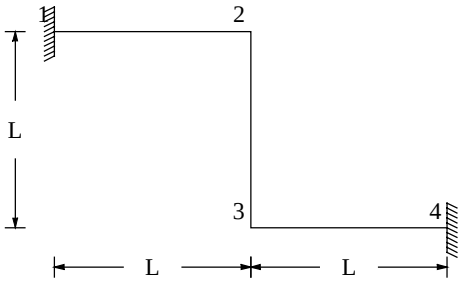
```
function [m, k, r] = TransientPlaneFrameElement(modulus, inertia, ...
   A, rho, qs, qt, coord)
% Plane frame element for dynamic analysis
% modulus = modulus of elasticity
% inertia = moment of inertia
% A = area of cross-section
% rho = mass density
% qs = distributed load along the element axis
% 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;

T = [ls, ms, 0, 0, 0, 0;
   -ms, ls, 0, 0, 0, 0;
   0, 0, 1, 0, 0, 0;
   0, 0, 0, ls, ms, 0;
   0, 0, 0, -ms, ls, 0;
   0, 0, 0, 0, 0, 1];
kl = [EA/L, 0, 0, -(EA/L), 0, 0;
   0, (12*EI)/L^3, (6*EI)/L^2, 0, ...
      -((12*EI)/L^3), (6*EI)/L^2;
    0, (6*EI)/L^2, (4*EI)/L, 0, ...
       -((6*EI)/L^2), (2*EI)/L;
    -(EA/L), 0, 0, EA/L, 0, 0;
    0, -((12*EI)/L^3), -((6*EI)/L^2), 0, ...
       (12*EI)/L^3, -((6*EI)/L^2);
    0, (6*EI)/L^2, (2*EI)/L, 0, ...
       -((6*EI)/L^2), (4*EI)/L];
ml = ((rho*A*L)/420)*[140,0, 0, 70, 0, 0;
   0, 156, 22*L, 0, 54, -13*L;
   0, 22*L, 4*L^2, 0, 13*L, -3*L^2;
   70, 0, 0, 140, 0, 0;
   0, 54, 13*L, 0, 156, -22*L;
   0, -13*L, -3*L^2, 0, -22*L, 4*L^2];
rl = [qs*(L/2); qt*(L/2); qt*(L^2/12);
      qs*(L/2); qt*(L/2); -qt*(L^2/12)];
m=T'*ml*T; k=T'*kl*T; r=T'*rl;
```

Using these functions now we consider modal analysis of the plane frame shown. The horizontal members carry a weight $200\ N/m$ in addition to their own weight. The other numerical data is as follows.

Use N-mm units.

$L = 600\,\text{mm}$; $E = 200\,\text{GPa} = 200,000\,N/\text{mm}^2$; $\rho = 7840\,\text{kg}/m^3 = 7.84\times10^{-6}\,\text{kg}/\text{mm}^3$; $A = 240\,\text{mm}^2$; I

Additional mass on horizontal members: $\frac{200}{9.81} = 20.3874\,\text{kg}/m = 0.0203874\,\text{kg}/\text{mm}$



MatlabFiles\Chap8\ModalFrameEx.m

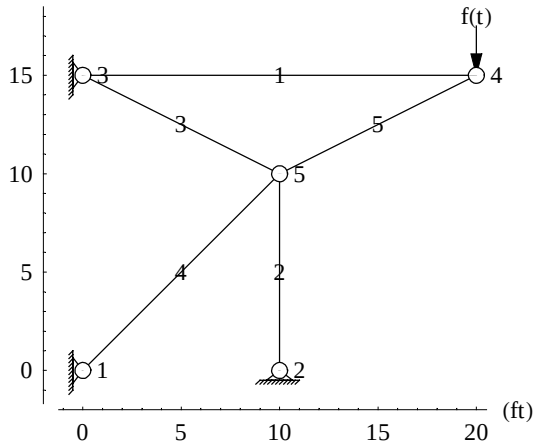
```
% Modal analysis of a plane frame
L = 1000; e = 200000; rho = 7.84*10^(-6); a = 240;
inertia = 2000; ma = 0.0203874;
nodes = [0, 0; L, 0; L, -L; 2*L, -L];
conn = [1,2; 2,3; 3,4];
elems = size(conn,1);
lmm=[];
for i=1:elems
   lmm = [lmm; [3*conn(i,1)-2, 3*conn(i,1)-1, 3*conn(i,1),...
            3*conn(i,2)-2, 3*conn(i,2)-1, 3*conn(i,2)]];
end
debc = [1,2,3,10,11,12]; ebcVals=zeros(length(debc),1);
dof=3*size(nodes,1);
M=zeros(dof); K=zeros(dof);

% Generate equations for each element and assemble them.
for i=1:2:elems
   con = conn(i,:);
   lm = lmm(i,:);
   [m, k, r] = TransientPlaneFrameElement(e, inertia, a, ...
      rho+ma/a, 0, 0, nodes(con,:));
   M(lm, lm) = M(lm, lm) + m;
   K(lm, lm) = K(lm, lm) + k;
end
```

```
for i=2
   con = conn(i,:);
   lm = lmm(i,:);
   [m, k, r] = TransientPlaneFrameElement(e, inertia, a, ...
      rho, 0, 0, nodes(con,:));
   M(lm, lm) = M(lm, lm) + m;
   K(lm, lm) = K(lm, lm) + k;
end

% Adjust for essential boundary conditions
dof = length(R);
df = setdiff(1:dof, debc);
Mf = M(df, df);
Kf = K(df, df);

% Compute frequencies and mode shapes
[V, lam] = eig(Kf, Mf);
freq=sqrt(lam)
modeShapes = V
```

```
>> ModalFrameEx

freq =

    0.57558         0         0         0         0         0
          0    4.2955         0         0         0         0
          0         0    4.4396         0         0         0
          0         0         0    76.951         0         0
          0         0         0         0    78.074         0
          0         0         0         0         0    204.62


modeShapes =

  2.4473e-007  -0.00012862   0.00014691      0.24834       0.2519     -0.01503
     0.26046  -0.00030283      0.26486    -0.051657    0.0039949      0.46663
   0.00022612    0.0015179    0.0022738  -0.00040806 -2.4546e-005    0.0025172
  2.4473e-007   0.00012862   0.00014691      0.24834      -0.2519      0.01503
     0.26046   0.00030283      0.26486    -0.051657   -0.0039949     -0.46663
  -0.00022612    0.0015179   -0.0022738   0.00040806 -2.4546e-005    0.0025172
```

Computer Implementation 8.4 ***(Matlab)*** *Transient analysis of a plane truss*

Consider solution of the plane truss structure All members have the same cross-sectional area and are of the same material, $\rho = 490\,\text{lb}/\text{ft}^3$, $E = 30\times10^6\,\text{lb}/\text{in}^2$ and $A = 1.25\,\text{in}^2$. The truss supports a rotating machine weighing 2000 lbs at its tip. Due to unbalance in the machine a harmonic force $P = 100\cos(7\pi t)$ lb is exerted on the truss.



The weight of the machine is an added mass at node 4. This is defined as ma in the following line and is added directly to the global mass matrix corresponding to degrees of freedom 7 and 8 associated with node 4. In defining the global load vector, a load of 1 unit is applied at degree of freedom 8. This global load vector is multiplied by $f(t) = 100\cos(7\pi t)$ before solving the differential equations of motion. The other procedure for generating the global mass, stiffness, and the load vector is exactly the same as that for the static analysis used in the previous chapters. Using in-lb units the calculations are as follows.

MatlabFiles\Chap8\TransientTrussEx.m

```
% Transient analysis of a plane truss
global Mf Kf Rf
g = 386.4; e = 30*10^6; A = 1.25;
rho = (490/(12^3))/g; ma = 2000/g;
nodes = 12 * [0, 0; 10, 0; 0, 15; 20, 15; 10, 10];
conn = [3, 4; 2, 5; 5, 3; 1, 5; 5, 4];
elems = size(conn,1);
lmm=[];
for i=1:elems
```

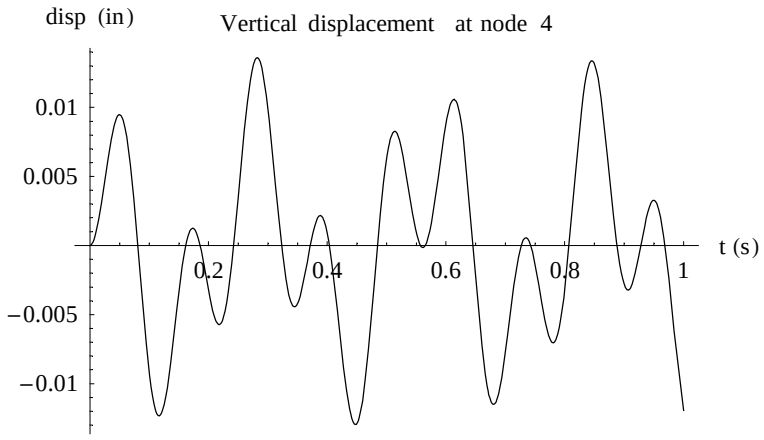
```
    lmm = [lmm; [2*conn(i,1)-1, 2*conn(i,1),2*conn(i,2)-1, 2*conn(i,2)]];
end
debc = [1:6]; ebcVals=zeros(length(debc),1);
dof=2*size(nodes,1);
M=zeros(dof); K=zeros(dof);
R = zeros(dof,1); R(8)=1;
% Add nodal masses to the global M matrix.

M(7,7)=ma; M(8,8) = ma;
% Generate equations for each element and assemble them.
for i=1:elems
    con = conn(i,:);
    lm = lmm(i,:);
    [m, k] = TransientPlaneTrussElement(e, A, rho, nodes(con,:));
    M(lm, lm) = M(lm, lm) + m;
    K(lm, lm) = K(lm, lm) + k;
end

% Adjust for essential boundary conditions
dof = length(R);
df = setdiff(1:dof, debc);
Mf = M(df, df);
Kf = K(df, df);
Rf = R(df) - K(df, debc)*ebcVals;

% Setup and solve the resulting first order differential equations
u0 = zeros(length(Mf),1);
v0 = zeros(length(Mf),1);
[t,d] = ode23('TrussODE',[0,1],[u0; v0]);
plot(t,d(:,2)); xlabel('time'); ylabel('Disp');
title('Vertical displacement at node 4');
```

MatlabFiles\Chap8\TrussODE.m

```
function ddot = TrussODE(t, d)
% ddot = TrussODE(t, d)
% function to set up equations for a transient truss problem
global Mf Kf Rf
ft = 100*cos(7*pi*t);
n=length(d);
u = d(1:n/2);
v = d(n/2+1:n);
vdot = inv(Mf)*(Rf*ft - Kf*u);
udot = v;
% format short g
% soln=[t, ft, u(1), udot(1), vdot(1)]
ddot = [udot; vdot];
```

Executing the script file TransientTrussEx a plot of the vertical displacement at node 4. The script file can be modified to see other quantities of interest.

```
>> TransientTrussEx
```



Computer Implementation 8.5 *(Matlab)* *Transient analysis of a plane frame*

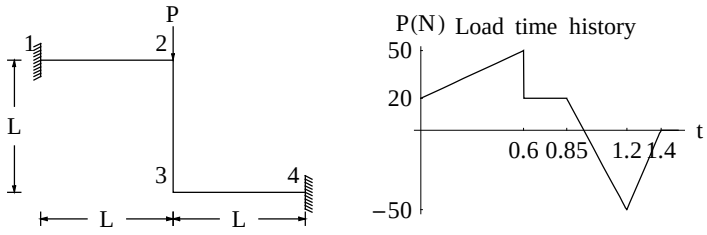
Consider solution of the plane frame. The load P varies with time as shown in the load-time history. The horizontal members carry a weight $200\,N/m$ in addition to their own weight. The other numerical data is as follows.

Use N-mm units.

$L = 600\,\text{mm}$; $E = 200\,\text{GPa} = 200,000\,N/\text{mm}^2$; $\rho = 7840\,\text{kg}/m^3 = 7.84\times10^{-6}\,\text{kg}/\text{mm}^3$; $A = 240\,\text{mm}^2$; I

Additional mass on horizontal members: $\frac{200}{9.81} = 20.3874\,\text{kg}/m = 0.0203874\,\text{kg}/\text{mm}$

f[t] = Which[t ≤ 0.6, 50 t + 20, t ≤ .85, 20, t ≤ 1.2, 190 – 200 t, t ≤ 1.4, 250 t – 350, t > 1.4, 0];



MatlabFiles\Chap8\TransientFrameEx.m

```
% Transient analysis of a plane frame
global Mf Kf Rf
L = 1000; e = 200000; rho = 7.84*10^(-6); a = 240;
inertia = 2000; ma = 0.0203874;
nodes = [0, 0; L, 0; L, -L; 2*L, -L];
conn = [1,2; 2,3; 3,4];
elems = size(conn,1);
lmm=[];
for i=1:elems
   lmm = [lmm; [3*conn(i,1)-2, 3*conn(i,1)-1, 3*conn(i,1),...
            3*conn(i,2)-2, 3*conn(i,2)-1, 3*conn(i,2)]];
end
debc = [1,2,3,10,11,12]; ebcVals=zeros(length(debc),1);
dof=3*size(nodes,1);
M=zeros(dof); K=zeros(dof);
R = zeros(dof,1); R(5)=1;

% Generate equations for each element and assemble them.
for i=1:2:elems
   con = conn(i,:);
   lm = lmm(i,:);
   [m, k, r] = TransientPlaneFrameElement(e, inertia, a, ...
      rho+ma/a, 0, 0, nodes(con,:));
   M(lm, lm) = M(lm, lm) + m;
   K(lm, lm) = K(lm, lm) + k;
   R(lm) = R(lm) + r;
```

```
end
for i=2
   con = conn(i,:);
   lm = lmm(i,:);
   [m, k, r] = TransientPlaneFrameElement(e, inertia, a, ...
      rho, 0, 0, nodes(con,:));
   M(lm, lm) = M(lm, lm) + m;
   K(lm, lm) = K(lm, lm) + k;
   R(lm) = R(lm) + r;
end

% Adjust for essential boundary conditions
dof = length(R);
df = setdiff(1:dof, debc);
Mf = M(df, df);
Kf = K(df, df);
Rf = R(df) - K(df, debc)*ebcVals;

% Setup and solve the resulting first order differential equations
u0 = zeros(length(Mf),1);
v0 = zeros(length(Mf),1);
[t,d] = ode23('FrameODE',[0,10],[u0; v0]);
plot(t,d(:,2)); xlabel('time'); ylabel('Disp');
title('Vertical displacement at node 2');
```

MatlabFiles\Chap8\FrameODE.m

```
function ddot = FrameODE(t, d)
% ddot = FrameODE(t, d)
% function to set up equations for a transient frame problem
global Mf Kf Rf
if t <= 0.6
   ft = 50*t + 20;
elseif t <= .85
      ft = 20;
   elseif t <= 1.2
         ft = 190 - 200*t;
      elseif t <= 1.4
            ft = 250*t - 350;
         else
            ft = 0;
         end
      end
   end
end
n=length(d);
u = d(1:n/2);
```

```
v = d(n/2+1:n);
vdot = inv(Mf)*(Rf*ft - Kf*u);
udot = v;
% format short g
% soln=[t, ft, u(2), udot(2), vdot(2)]
ddot = [udot; vdot];
```

Executing the script file TransientFrameEx a plot of the vertical displacement at node 2. The script file can be modified to see other quantities of interest.

```
>> TransientFrameEx
```

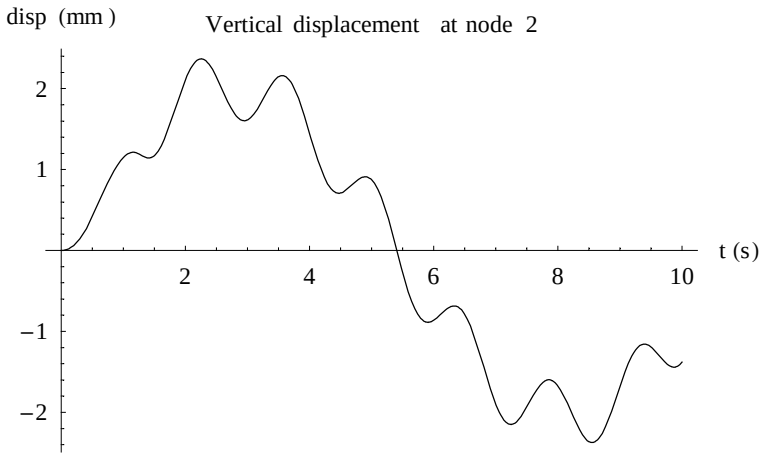


Figure 8.1. Displacement time history

Once the nodal displacements are known, any other quantity of interest, say bending moments in elements, can easily be computed using the same functions as those used in the static analysis case. Since the displacements are functions of time, obviously the element quantities are also functions of time.