CHAPTER THREE

One Dimensional Boundary Value Problem

Matlab functions for solution of one dimensional boundary value problem

For *Matlab* solution of one dimensional boundary value problems using linear and quadratic element, the following functions can be defined and used in the manner shown in Chapter 2 for axial deformation problem.

MatlabFiles\Chap3\BVP1DLinElement.m

```
function [ke, re] = BVP1DLinElement(k, p, q, coord)
% [ke, re] = BVP1DLinElement(k, p, q, coord)
% Generates equations for a linear element for 1D BVP
% k,p,q = parameters defining the BVP
% coord = coordinates at the element ends

L=coord(2)-coord(1);
```

```
ke = [k/L - (L*p)/3, -(k/L) - (L*p)/6;
   -(k/L) - (L*p)/6, k/L - (L*p)/3];
re = [(L*q)/2; (L*q)/2];
```

MatlabFiles\Chap3\BVP1DQuadElement.m

```
function [ke, re] = BVP1DQuadElement(k, p, q, coord)
% [ke, re] = BVP1DQuadElement(k, p, q, coord)
% Generates equations for a quadratic element for 1D BVP
% k,p,q = parameters defining the BVP
% coord = coordinates at the element ends

L=coord(3)-coord(1);
ke = [(7*k)/(3*L) - (2*L*p)/15, (-8*k)/(3*L) - (L*p)/15, ...
      k/(3*L) + (L*p)/30;
   (-8*k)/(3*L) - (L*p)/15, (16*k)/(3*L) - (8*L*p)/15, ...
      (-8*k)/(3*L) - (L*p)/15;
   k/(3*L) + (L*p)/30, (-8*k)/(3*L) - (L*p)/15, ...
      (7*k)/(3*L) - (2*L*p)/15];
re = [(L*q)/6; (2*L*q)/3; (L*q)/6];
```

Several examples in the following sections illustrate the use of these functions.

Computer Implementation 3.1 ***(Matlab)*** *Solution using Matlab functions for 1D BVP*

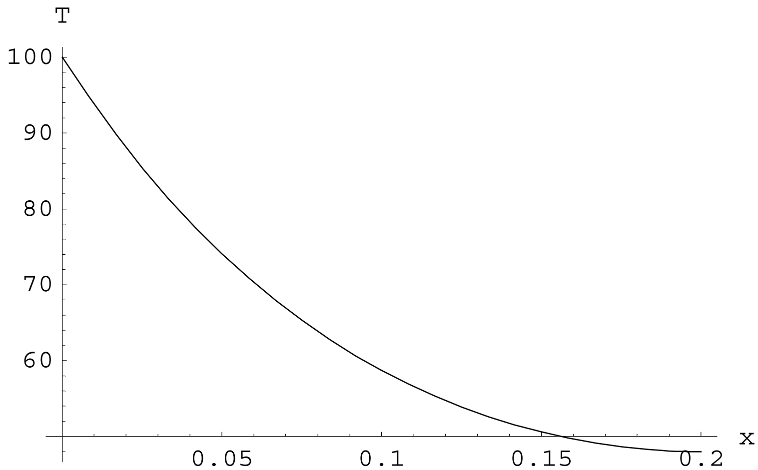
The tedious calculations to solve the previous heat flow problem can be conveniently carried out using *Matlab* functions presented earlier. Using these functions, and following procedures discussed in Chapter 1, the global equations for the four quadratic element model can be developed and assembled as follows.

MatlabFiles\Chap3\FinHeatFlowEx.m

```
% Heat flow through a fin
kx=237;w=3;t=0.3/100;A=w*t;h=30;L=20/100.;P=2*(w+t);Tinf=25;
alpha=-h*A/(kx*A); beta=h*A*Tinf/(kx*A);
k=kx*A; p=-h*P; q=h*P*Tinf;
nodes = [0:L/8:L];n=length(nodes);
K=zeros(n); R = zeros(n,1);
% Generate equations for each element and assemble them.
for i=1:4
   lm=[2*(i-1)+1,2*(i-1)+2,2*(i-1)+3];
   [ke, re] = BVP1DQuadElement(k,p,q, nodes(lm));
   K(lm, lm) = K(lm, lm) + ke;
   R(lm) = R(lm) + re;
end
% Adjust for NBC
```

```
K(n,n)=K(n,n)-alpha*k
R(n)=R(n)+beta*k
% Nodal solution and reactions
d = NodalSoln(K, R, [1], [100])
plot(nodes,d),title('Temperature distribution'), xlabel('x'),ylabel('T')

>> FinHeatFlowEx

K =

  Columns 1 through 7

  100.7412 -113.1594   13.9197         0         0         0         0
 -113.1594  232.3248 -113.1594         0         0         0         0
   13.9197 -113.1594  201.4824 -113.1594   13.9197         0         0
         0         0 -113.1594  232.3248 -113.1594         0         0
         0         0   13.9197 -113.1594  201.4824 -113.1594   13.9197
         0         0         0         0 -113.1594  232.3248 -113.1594
         0         0         0         0   13.9197 -113.1594  201.4824
         0         0         0         0         0         0 -113.1594
         0         0         0         0         0         0   13.9197

  Columns 8 through 9

         0         0
         0         0
         0         0
         0         0
         0         0
         0         0
 -113.1594   13.9197
  232.3248 -113.1594
 -113.1594  101.0112


R =

   37.5375
  150.1500
   75.0750
  150.1500
   75.0750
  150.1500
   75.0750
  150.1500
   44.2875
```

```
d =

  100.0000
   85.4383
   74.0844
   65.3306
   58.7175
   53.8905
   50.5968
   48.6592
   47.9772
```



Computer Implementation 3.2 ***(Matlab)*** *Solution of Buckling Problem*

The buckling problem can be implemented easily in *Matlab* by defining simple functions returning element $\boldsymbol{k}$ and $\boldsymbol{k}_p$ matrices as follows.

MatlabFiles\Chap3\BucklingLinElement.m

```
function [ke, kp] = BucklingLinElement(k, coord)
% [ke, kp] = BucklingLinElement(k, coord)
% Generates equations for a linear element for 1D Buckling
% k = bar stiffness (EI)
% coord = coordinates at the element ends

L=coord(2)-coord(1);
ke = k/L*[1, -1; -1, 1];
kp = [L/3, L/6; L/6, L/3];
```

MatlabFiles\Chap3\BucklingQuadElement.m

```
function [ke, kp] = BucklingQuadElement(k, coord)
% [ke, kp] = BucklingQuadElement(k, coord)
% Generates equations for a quadratic element for 1D Buckling
% k = bar stiffness (EI)
% coord = coordinates at the element ends

L=coord(3)-coord(1);
ke = [(7*k)/(3*L), -((8*k)/(3*L)), k/(3*L);
    -((8*k)/(3*L)), (16*k)/(3*L), -((8*k)/(3*L));
    k/(3*L), -((8*k)/(3*L)), (7*k)/(3*L)];
kp = [((2*L)/15), (L/15), -L/30;
    (L/15), ((8*L)/15), (L/15);
    -L/30, (L/15), ((2*L)/15)];
```

Using the BucklingQuadElement function, a solution using 4 quadratic is obtained as follows.

MatlabFiles\Chap3\BucklingEx.m

```
% Solution of Euler buckling using quadratic elements
L = 12*10.; EI = 10^6;
nodes = [0:L/8:L];n=length(nodes);
Ke=zeros(n); Kp=zeros(n);
% Generate equations for each element and assemble them.
for i=1:4
    lm=[2*(i-1)+1,2*(i-1)+2,2*(i-1)+3];
    [ke, kp] = BucklingQuadElement(EI, nodes(lm));
    Ke(lm, lm) = Ke(lm, lm) + ke;
    Kp(lm, lm) = Kp(lm, lm) + kp;
end
% Adjust for EBC
debc=[1,n];
df = setdiff(1:n, debc);
Kef = Ke(df, df)
Kep = Kp(df, df)
[v,e] = eig(Kef, Kep);
fprintf('Buckling load = %10.6g',e(1,1))
d = zeros(n,1);
d(df) = v(:,1)
plot(nodes,d),title('First buckling mode'), xlabel('x'),ylabel('v')


>> BucklingEx
```

```
Kef =

  1.0e+005 *

    1.7778   -0.8889         0         0         0         0         0
   -0.8889    1.5556   -0.8889    0.1111         0         0         0
         0   -0.8889    1.7778   -0.8889         0         0         0
         0    0.1111   -0.8889    1.5556   -0.8889    0.1111         0
         0         0         0   -0.8889    1.7778   -0.8889         0
         0         0         0    0.1111   -0.8889    1.5556   -0.8889
         0         0         0         0         0   -0.8889    1.7778


Kep =

    16     2     0     0     0     0     0
     2     8     2    -1     0     0     0
     0     2    16     2     0     0     0
     0    -1     2     8     2    -1     0
     0     0     0     2    16     2     0
     0     0     0    -1     2     8     2
     0     0     0     0     0     2    16

Buckling load =     685.74
d =

         0
    0.0494
    0.0913
    0.1193
    0.1292
    0.1193
    0.0913
    0.0494
         0
```