Implementation of a Beam Element in FEA using MATLAB

[Type the document subtitle]

Finite Element Analysis
EML 5226

Jainil N Desai
UFID 96950890

1) Objective:

Element Implemented: A two node iso-parametric beam element. It solves for the deflection of the beam according to the boundary conditions and applied loads.

I have implemented a Matlab code to solve a cantilever beam or a simply supported beam with point loads at any location of the beam. There may even be supports at any location other than cantilever that is fixed at right hand side for the cantilever beam. Also, for the simply supported beam, there may be supports at any point in between the beam.

2) Description:

a) Theory:

Equilibrium Equation:

Here, $\sum Fy = 0$

We assume no axial load. So

Vs+ds – Vs +f(s)ds = 0

So,

dVs/ds +f = 0

Also sum of bending moments = 0

Ms+ds –Ms + Vs+ds*ds + f ds * ds/2 = 0

Now taking $\lim_{ds-0}$

We get,

dMs/ds +Vs = 0

hence, $\frac{d^2m}{ds^2} = f$

deflection equation of a beam is

$$\frac{d^2w}{ds^s} = \frac{M(s)}{EI}$$

Here, w = deflection of the beam,

E = Young's Modulus

I = moment of area

M = a function of s

s = variable varying along length of beam

So, combining the equilibrium and deflection of beam, we get the basic constitutive equation as

$$\frac{d^2}{ds^2}\left(EI\frac{d^2w}{ds^2}\right) = f$$

This is called Euler-Bernoulli Beam equation

This is a boundary value problem with boundary conditions as,

- Slope = dw/ds is specified
- Moment at s = 0 is Mo

- V = shear force at s=0 = Vo
- w is a function of s

So, the weak form is

$$\int_0^L \left[\frac{d^2}{ds^2}\left(EI\frac{d^2w}{ds^2}\right) - f\right]\delta w\, ds = 0, after\ integrating\ by\ parts$$

And applying boundary conditions, we have the final equation as

$$\int_0^L \frac{d^2\delta w}{ds^2} EI\ \frac{d^2w}{ds^2} ds = \ \delta w1\, V1 + \ \delta\theta 1\, M1 + \ \delta w2\, V2 + \ \delta\theta 2\, M2 + \int_0^L f\delta w\, ds$$

This is called principle of virtual work.
Shape Function: the shape function defines the deflection as a function of s. They are
w(s) = w1N1(s) + θ1N2(s) + w2 N3(s) + θ2 N4(s)
where N1(s), N2(s), N3(s) and N4(s) are the shape functions and are defined as

Le = length of element
Zeta = s/Le
So,
N1(s) = 1- 3 $(zeta)^2$ + 2$(zeta)^3$
N2(s) = Le(zeta – 2$(zeta)^2$ + $(zeta)^3$)
N3(s) = 3$(zeta)^2$ – 2$(zeta)^3$
N4(s) = 4(-$(zeta)^2$ +$(zeta)^3$)

These are called hermite polynomials and the elements formulated using these are called Hermite Elements.

Using the above polynomials and the weak form of the beam equation, the elemental equation of a beam can be written as

[Ke]*{Xe} = {Fe}
Where,

$$[Ke] = \begin{pmatrix} 12 & 6Le & -12 & 6Le \\ 6Le & 4Le^2 & -6Le & 2Le^2 \\ -12 & -6Le & 12 & -6Le \\ 6Le & 2Le^2 & -6Le & 4Le^2 \end{pmatrix}$$

{Xe} = {w1 θ1 $w2$ θ2} = displacement matrix
And
{Fe} = {Fc} + {Fd}
{Fd} = forces due to uniformly distributed loads and {Fc} = general force and moment matrix

Hence assembling the above for all the elements of the entire beam, we can calculate the displacements and forces at each elements. We can use two approaches,

- penalty method

- Eliminating unknown rows and columns.

In the implementation of this element I have used the later method.

Now using the above theory, I have implemented the element of beam to solve for the displacement and find the reactions.

b) The algorithm is as below:

1) Determine if the beam is a cantilever or a simply supported:
2) If a cantilever beam,
   a) Enter the cross section of beam, whether circular or rectangular and calculate I (moment of area). If the moment is available enter the value directly.
   b) Input the length of beam, number of elements we wish to divide the beam into, length of each element, etc.
   c) Calculate the element stiffness matrix and assemble that into global stiffness matrix.
   d) Input if there is a support at a node other than the fixed one for the cantilever.
   e) If there is a support, remove the row and column corresponding to the vertical displacement and store the reduced stiffness matrix as the displacement is constrained in y direction but angular displacement is allowed.
   f) After that remove the first and second row and column of the reduced global stiffness matrix as the angular and vertical displacements are constrained on the first node as there is a cantilever support.
   g) Input the forces or moments that are acting on the beam in order starting from first node. If no force or moment is acting, input zero at that node.
   h) After obtaining the reduced stiffness matrix and the force matrix, multiply the inverse of the reduced stiffness matrix and force matrix to obtain the nodal displacements.
   i) Calculate the shape functions and using the nodal displacements of the beam, find the displacements along the elements and plot the displacements to obtain the deflection of the beam.

3) Else if a simply supported beam,

   Repeat steps till e) above.
   f) Now remove the second and the last second row and column of the reduced stiffness matrix.

   Repeat steps g) through i) above and obtain the final deflection of the beam.

   End

   The matlab code is attached in appendix A

3) Results:

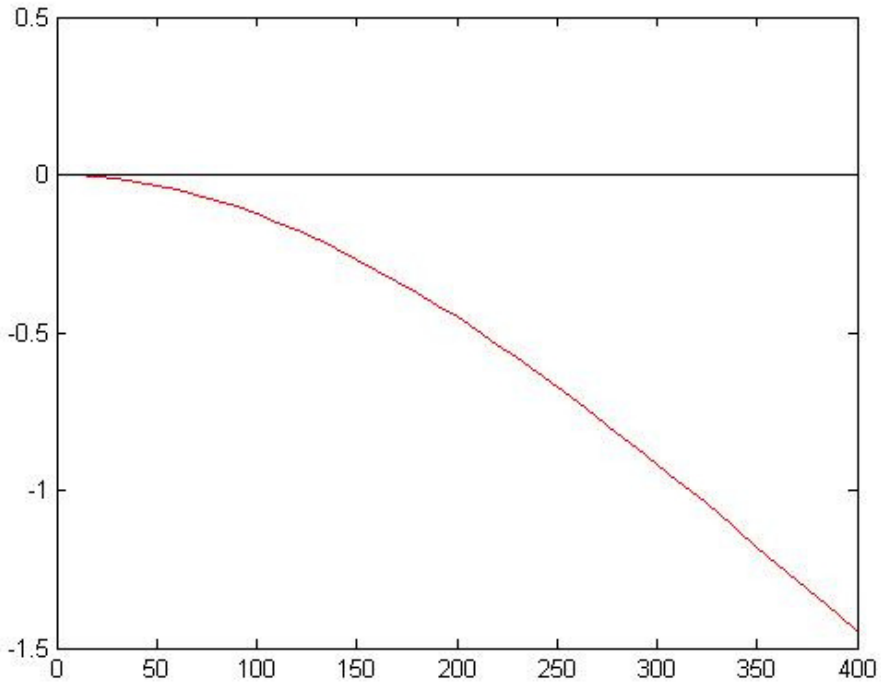
Now we test the above code with a simple problem of a cantilever beam and a simply supported beam.

a) Cantilever beam: consider a simple cantilever beam with a circular cross-section of 10 in diameter and a length of 400 in. The Young's Modulus of the beam is 30 x 10^6 Psi. There is a load of 1000 lb acting in the downward direction at the right end of the beam.

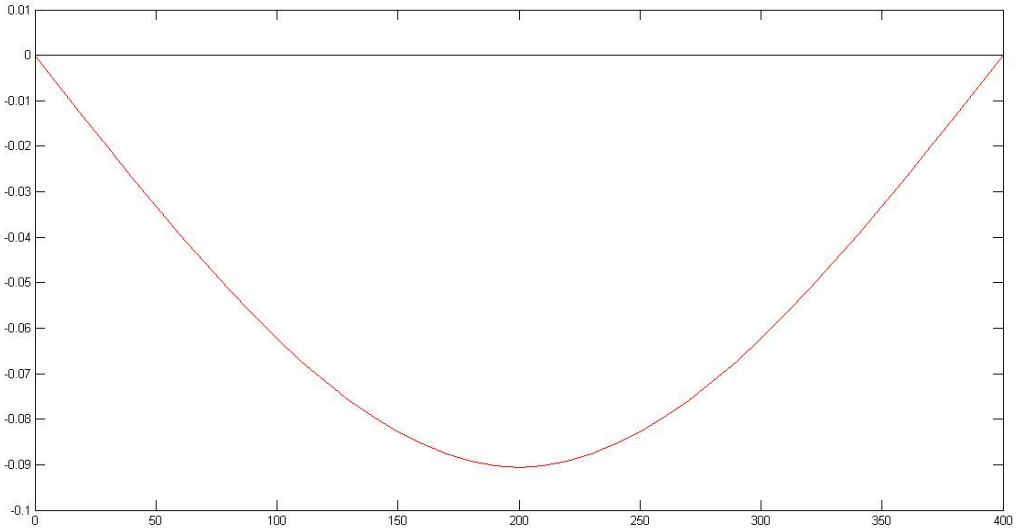
Analytical solution: The maximum deflection at the right end of the beam is,

Ymax = FL^3/(3EI),
Where, I = pi*(d/2)^4/42 = 490.8739 in^4, E = 30 * 10^6 psi
So, Ymax = 1000*400^3/(3*30*10^6*490.8739)
= 1.44866 m

Solution with Finite Element implemented:
Maximum displacement = 1.4487. So we get the same value till three decimal places. The matlab answer that we obtained is accurate till three decimals. Hence we see that the solution is valid. The deflection plot is attached here,



b) Simply supported beam: consider a simply supported beam with a circular cross-section of 10 in diameter and a length of 400 in. The Young's Modulus of the beam is 30 x 10^6 Psi. There is a load of 1000 lb acting in the downward direction at the center of the beam.
Analytical Solution:
The maximum deflection at the center of the beam is :
Ymax = FL^3/(48*EI)
= 1000*400^3 / (48 *30 * 10^6 * 490.8739)
= 0.090541 m
Solution with Finite Element implemented:
Maximum displacement = 0.905. So we get the same value till three decimal places. The matlab answer that we obtained is accurate till three decimals. Hence we see that the solution is valid. The deflection plot is attached here,



4) Conclusions:

We conclude that a simple beam may be solved for displacements and reactions easily by hand, but as the loads start varying and the analytical solution by hand becomes difficult. So we implement the finite element analysis to approximate the beam deflection. We saw that the shape function is used to interpolate the deflection at each point in between the element. The finite element solution of a beam element is a cubic polynomial while actual beam solution is of the 4^{th} order. The finite element model gives a stiffer beam. It actually forces the beam in to specific modes of deflection as per the shape functions and hence gives a more stiffer beams. Moreover as we increase the number of nodes, we get a better solution. Here we gained an insight on a beam element, how it behaves, how the stiffness matrix is derived and the final solution. Also the actual implementation of shape functions to get the total deflection of a structure by programming it. Also it gave a deeper sense of how an actual commercial Finite Element software would be getting the solutions and how to better control inputs to a solution of a problem by using FEA.

Appendix I

```
clc
clear all
close all
g = input('Press 1 if cantilever beam /n Press 2 if Simply Supported Beam \n:');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%% CANTILEVER BEAM%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
if (g==1)
E = input('Enter Youngs Modulus of beam:');
j = input('Enter 1 for circular c/s, 2 for rectangular c/s or 3 to enter value of I
manually:');
if(j==1)
    d = input('Enter Diameter:');
    I = pi*(d/2)^4/4
elseif(j==2)
    b = input('Enter width of beam:');
    h= input('Enter height of beam:');
    I = b*h^3/12
elseif(j==3)
    I = input('Enter value of I manually:');
end
n = input('Enter number of elements:');
%L = length of beam
L = input('Enter Length of the beam: \n');
%input length if each element, l1, l2, ..., ln
%l = [3; 3];
for i=1:n
    fprintf('element %g', i);
l(i)=input('Enter length of element :');
end
%k = E*I/le^3*[12 6*le -12 6*le; 6*le 4*le^2 -6*le 2*le^2; -12 -6*le 12 -6*le; 6*le
2*le^2 -6*le 4*le^2]
%enter boundary condition at each node
%K2 = zeros(6,6);
if n == 1
   le = l(i);
    K = E*I/le^3*[12 6*le -12 6*le; 6*le 4*le^2 -6*le 2*le^2; -12 -6*le 12 -6*le;
6*le 2*le^2 -6*le 4*le^2];
    %K = wextend(2,'zpd',k,2, 'r');
else
for i = 1:n-1
    if (i == 1)
    le = l(i);
    k = E*I/le^3*[12 6*le -12 6*le; 6*le 4*le^2 -6*le 2*le^2; -12 -6*le 12 -6*le;
6*le 2*le^2 -6*le 4*le^2];
```

```
    K = wextend(2,'zpd',k,2, 'r');
    else
    K = wextend(2,'zpd',K,2, 'r');
    end
    le = l(i+1);
    k = E*I/le^3*[12 6*le -12 6*le; 6*le 4*le^2 -6*le 2*le^2; -12 -6*le 12 -6*le;
6*le 2*le^2 -6*le 4*le^2];
    K1 = wextend(2,'zpd',k,2*i, 'l');
    K= K+K1;
end
end
% BC
K2 = K;
ds = zeros((2*(n+1)),1);
for i = n+1:-1:2
    fprintf('Node %g \n', i);
    l = input('Press 1 if support is there for this node, 0 if no support is there:
\n');
    if (l == 1)
        K2((i-1)*2+1,:) = [];
        K2(:,(i-1)*2+1) = [];
        ds((i-1)*2+1) = 1;
    end
end
ds(1)=1;
ds(2)=1;
K2(1:2,:) = [];
K2(:,1:2) = [];

%BC
r = size(K2,1)
fprintf('Enter all the forces in order, first enter force and then moment for all
nodes');
for i = 1:r
    F(i) = input('');
end
a = 1;
%displacements
dis = inv(K2)* F'
for i = 1:2*(n+1)
    if(ds(i)==1)
        ds(i)=0;
    else
      ds(i)=dis(a);
     a=a+1;
```

```
    end
end
fprintf(' the displacements at the the nodes are: ')

fprintf('the reactions at the nodes are:')
Fr = K*ds
w=1;
N = [];
for i = 1:n
    l(i) = le;
    for s = 0:(0.1*le):le
        if (i~=1)&(s~=0)
        zeta = s/le;
        N1 = (1-3*zeta^2+2*zeta^3);
        N2 = le*(zeta-2*zeta^2+zeta^3);
        N3 = (3*zeta^2-2*zeta^3);
        N4 = le*(-zeta^2+zeta^3);
        N = vertcat(N, [N1, N2, N3, N4]);
        f = ds(((2*i)-1),1) * N1 + ds((2*i),1) * N2 + ds(((2*i)+1),1) * N3 +
ds(((2*i)+2),1) * N4;
        t(w)=f;
        w=w+1;
        elseif (i==1)
            zeta = s/le;
            N1 = (1-3*zeta^2+2*zeta^3);
        N2 = le*(zeta-2*zeta^2+zeta^3);
        N3 = (3*zeta^2-2*zeta^3);
        N4 = le*(-zeta^2+zeta^3);
        N = vertcat(N, [N1, N2, N3, N4]);
        f = ds(((2*i)-1),1) * N1 + ds((2*i),1) * N2 + ds(((2*i)+1),1) * N3 +
ds(((2*i)+2),1) * N4;
        t(w)=f;
        w=w+1;
        end
    end
end

w
x=1
for u = 0:(0.1*L/n):L
    lbh = u;
    lb(x) = lbh;
    lbd(x) = 0;
    x=x+1;
end
```

```
plot(lb, t, 'r', lb, lbd, 'k')

else


    %%%%%%%%%%%%%%%% SIMPLY SUPPORTED BEAM %%%%%%%%%%%%%%%%%%%%%%%%%%%%

    E = input('Enter Youngs Modulus of beam:');
j = input('Enter 1 for circular c/s, 2 for rectangular c/s or 3 to enter value of I
manually:');
if(j==1)
    d = input('Enter Diameter:');
    I = pi*(d/2)^4/4
elseif(j==2)
    b = input('Enter width of beam:');
    h= input('Enter height of beam:');
    I = b*h^3/12
elseif(j==3)
    I = input('Enter value of I manually:');
end
n = input('Enter number of elements:');
%L = length of beam
L = input('Enter Length of the beam: \n');
%input length if each element, l1, l2, ..., ln
%l = [3; 3];
for i=1:n
l(i)=input('Enter length of element :');
end
%k = E*I/le^3*[12 6*le -12 6*le; 6*le 4*le^2 -6*le 2*le^2; -12 -6*le 12 -6*le; 6*le
2*le^2 -6*le 4*le^2]
%enter boundary condition at each node
%K2 = zeros(6,6);
for i = 1:n-1
    if (i == 1)
    le = l(i);
    k = E*I/le^3*[12 6*le -12 6*le; 6*le 4*le^2 -6*le 2*le^2; -12 -6*le 12 -6*le;
6*le 2*le^2 -6*le 4*le^2];
    K = wextend(2,'zpd',k,2, 'r');
    else
    K = wextend(2,'zpd',K,2, 'r');
    end
    le = l(i+1);
    k = E*I/le^3*[12 6*le -12 6*le; 6*le 4*le^2 -6*le 2*le^2; -12 -6*le 12 -6*le;
6*le 2*le^2 -6*le 4*le^2];
```

```
    K1 = wextend(2,'zpd',k,2*i, 'l');
    K= K+K1;
end

% BC
K2 = K;
ds = zeros((2*(n+1)),1);
for i = n:-1:2
    fprintf('Node %g \n', i);
    l = input('Press 1 if support is there for this node, 0 if no support is there:
\n');
    if (l == 1)
        K2((i-1)*2+1,:) = [];
        K2(:,(i-1)*2+1) = [];
        ds((i-1)*2+1) = 1;
    end
end
ds(1)=1;
%ds(2)=1;
K2(1,:) = [];
K2(:,1) = [];

%BC
r = size(K2,1);
K2(r-1,:) = [];
K2(:,r-1) = [];
ds(r)=1;
r = size(K2,1);
fprintf('Enter all the forces in order, first enter force and then moment for all
nodes');
for i = 1:r
    F(i) = input('\n ');
end
a = 1;
%displacements
dis = inv(K2)* F'
for i = 1:2*(n+1)
    if(ds(i)==1)
        ds(i)=0;
    else
      ds(i)=dis(a);
     a=a+1;
    end
end
Fr = K*ds;
```

```
w=1
N = [];
for i = 1:n
    l(i) = le;
    for s = 0:(10):le
        if (i~=1)&(s~=0)
        zeta = s/le;
        N1 = (1-3*zeta^2+2*zeta^3);
        N2 = le*(zeta-2*zeta^2+zeta^3);
        N3 = (3*zeta^2-2*zeta^3);
        N4 = le*(-zeta^2+zeta^3);
        N = vertcat(N, [N1, N2, N3, N4]);
        f = ds(((2*i)-1),1) * N1 + ds((2*i),1) * N2 + ds(((2*i)+1),1) * N3 +
ds(((2*i)+2),1) * N4;
        t(w)=f;
        w=w+1;
        elseif (i==1)
            zeta = s/le;
            N1 = (1-3*zeta^2+2*zeta^3);
        N2 = le*(zeta-2*zeta^2+zeta^3);
        N3 = (3*zeta^2-2*zeta^3);
        N4 = le*(-zeta^2+zeta^3);
        N = vertcat(N, [N1, N2, N3, N4]);
        f = ds(((2*i)-1),1) * N1 + ds((2*i),1) * N2 + ds(((2*i)+1),1) * N3 +
ds(((2*i)+2),1) * N4;
        t(w)=f;
        w=w+1;
        end
    end
end

w
x=1
for u = 0:(10):L
    lbh = u;
    lb(x) = lbh;
    lbd(x) = 0;
    x=x+1;
end
plot(lb, t, 'r', lb, lbd, 'k')
end
```