AE 675A : Introduction To Finite Element Methods

Assignment Report

1D – Post-Processing and Numerical Integration

AMAL M S

22101262

Department of Aerospace Engineering, IIT Kanpur

CONTENTS

- Question
- MATLAB Code
- Output
- Inference

QUESTION

Given below are the figure of 1D bars of length L units subjected to two different conditions.

Properties:

EA = 1;

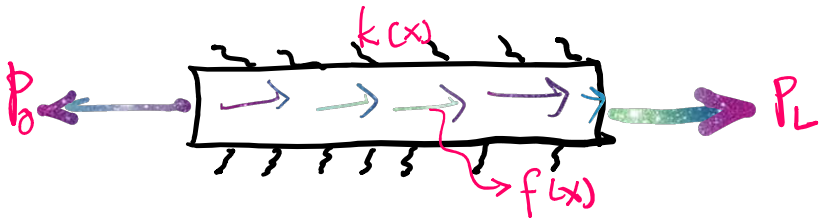
k = 0, 10;

f = 10, 10x, $10x^2$

Boundary Conditions are as shown in figure.

Plot the displacement field and post processed strain fields after solving using FEM with Numerical Integration.

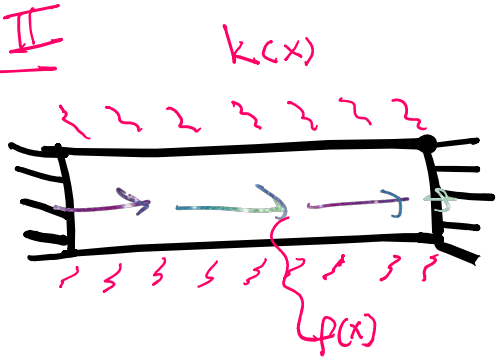
CASE I



$P_0 = 5$

$P_L = 10$

Case II



MATLAB CODE

READ ME

This code is only made for cases in problem 1 and problem 2 and solves using linear finite element basis functions.

When asked for question Number :

- Enter "1" : for first problem where both the end forces are given.

- Enter "2" : for second problem where both the ends are fixed.

- k can be 0 or any other number.
- axial force distribution can be a number or a function of x.

```
clear all
Q = input('Enter the question number :' );
```

```
if Q == 1


    syms x


    EA = input('Enter the value of EA:  ');
    k = input('spring constant assosciated with the member, k = ');
    x_0 = 0 ;
    x_L = input('Enter the length of the bar:  ');
    nel  = input('Enter the number of elements:  ');


    K = zeros(nel+1,nel+1);
    F = zeros(nel+1,1);


    h = (x_L - x_0)/nel;


    x1 = 0:h:(x_L-h);
    x2 = h:h:x_L;


    while true
        f = input('Distribution (use small letter "x" if it is a variable) of
axial force = ')
```

Boundary condition at x = 0

```
        P_0 = input('Give the force at the end x = 0 :  ')
```

Boundary condition at x = L

```
        P_L = input('Give the force at the end x = L :  ')


        if k == 0
            if int(f,0,x_L) + P_L - P_0 == 0
                break
            else
                fprintf('If spring constant assosciated with the member, k = 0,
then,\n there must be a force equilibrium balance to have a solution.\n Kindly
update the values of forces at both ends and the axial\n distribution of forces
accordingly.')
                continue
            end
        else
            break
        end
    end
```

```
    for e = 1:nel


        FF = zeros(2,1);
        KK = zeros(2);


        X1 = x1(e);
        X2 = x2(e);


        N1 = (X2 - x)/h;
        N2 = (x - X1)/h;
```

```
        N = [N1,N2];
        dN = diff(N);


        for i = 1:2
            FF(i) = FF(i) + Gauss_Legendre_Integration(f*N(i),1,X1,X2);
            for j = 1:2
                KK(i,j) = KK(i,j) + Gauss_Legendre_Integration(EA*dN(i)*dN(j) +
k*N(1)*N(2),2,X1,X2);
            end
        end
        for i = 1:2
            ig = e + (i-1);
            F(ig) = F(ig) + FF(i);
            for j = 1:2
                jg = e + (j-1);
                K(ig,jg) = K(ig,jg) + KK(i,j);
            end
        end
    end
    F(1) = F(1) + P_0;
    F(end) = F(end) + P_L;


    if k == 0
        K(1,:) = 0;
        K(:,1) = 0;
        K(1,1) = 1;
        fprintf('Since k = 0, with boundary condition type-2 at both ends\n there
are infinite possible solutions. Out of the infinite solutions possible, \n one
solution where the displacement at x = 0 is displayed!')
    end


    Alpha = K\F;


    for i = 1:nel


        X1 = x1(i);
        X2 = x2(i);


        N1 = (X2 - x)/h;
        N2 = (x - X1)/h;


        U(i) = Alpha(i)*N1 + Alpha(i+1)*N2;
```

```
    end
    % Strain
    Str = single(diff(U,x));
    xx = h/2:h:x_L


    for i = 1:nel
        if i == 1
            coeff(2*i-1:2*i) = polyfit(xx(1:2),Str(1:2),1);
        elseif i == nel
            coeff(2*i-1:2*i) = polyfit(xx(i-1:i),Str(i-1:i),1);
        else
            coeff(2*i-1:2*i) = polyfit(xx(i-1:i+1),Str(i-1:i+1),1);
        end
    end


    for i = 1:nel
        strain(i) = coeff(2*i-1)*x+coeff(2*i);
    end
```

Plotting

```
    figure();
    for i = 1:nel
        t = linspace(x1(i),x2(i),20);
        u = double(subs(U(i),x,t));
        plot(t,u,'r','Linewidth',0.2);
        hold on
    end
    title("Dispalcement field")
    xlabel("Length, x")
    ylabel("Displacement, u(x)")
    hold off
    % strain field
    for i = 1:nel
        t = linspace(x1(i),x2(i),20);
        uu = double(subs(strain(i),x,t));
        plot(t,uu,'r','Linewidth',0.2);
        hold on
    end
    title("Strain field")
    xlabel("Length, x")
    ylabel("strain (x)")


else


    syms x
```

```
    EA = input('Enter the value of EA:  ');
    k = input('spring constant assosciated with the member, k = ');
    x_0 = 0 ;
    x_L = input('Enter the length of the bar:  ');
    nel  = input('Enter the number of elements:  ');


    K = zeros(nel+1,nel+1);
    F = zeros(nel+1,1);


    h = (x_L - x_0)/nel;


    x1 = 0:h:(x_L-h);
    x2 = h:h:x_L;




    f = input('Distribution (use small letter "x" if it is a variable) of axial
force = ')
```

Boundary condition at x = 0

```
    U_0 = input('Give the displacement at the end x = 0 :  ')
```

Boundary condition at x = L

```
    U_L = input('Give the displacement at the end x = L :  ')
```

```
    for e = 1:nel


        FF = zeros(2,1);
        KK = zeros(2,2);
```

```
        X1 = x1(e);
        X2 = x2(e);


        N1 = (X2 - x)/h;
        N2 = (x - X1)/h;


        N = [N1,N2];
        dN = diff(N);


        for i = 1:2
            FF(i) = FF(i) + Gauss_Legendre_Integration(f*N(i),1,X1,X2);
            for j = 1:2
                KK(i,j) = KK(i,j) + Gauss_Legendre_Integration(EA*dN(i)*dN(j) +
k*N(1)*N(2),2,X1,X2);
            end
        end
        for i = 1:2
            ig = e + (i-1);
            F(ig) = F(ig) + FF(i);
            for j = 1:2
                jg = e + (j-1);
                K(ig,jg) = K(ig,jg) + KK(i,j);
            end
        end
    end
    K(1,:) = 0;
    F(1) = 0;
    K(:,1) = 0;
    K(1,1) = 1;


    K(:,end) = 0;
    K(end,:) = 0;
    F(end) = 0;
    K(end,end) = 1;
    Alpha = K\F;


    for i = 1:nel


        X1 = x1(i);
        X2 = x2(i);
```

```
        N1 = (X2 - x)/h;
        N2 = (x - X1)/h;


        U(i) = Alpha(i)*N1 + Alpha(i+1)*N2;
    end


    % Strain
    Str = single(diff(U,x));
    xx = h/2:h:x_L


    for i = 1:nel
        if i == 1
            coeff(2*i-1:2*i) = polyfit(xx(1:2),Str(1:2),1);
        elseif i == nel
            coeff(2*i-1:2*i) = polyfit(xx(i-1:i),Str(i-1:i),1);
        else
            coeff(2*i-1:2*i) = polyfit(xx(i-1:i+1),Str(i-1:i+1),1);
        end
    end


    for i = 1:nel
        strain(i) = coeff(2*i-1)*x+coeff(2*i);
    end
```

Plotting

```
    figure();
    % displacement field
    for i = 1:nel
        t = linspace(x1(i),x2(i),20);
        u = double(subs(U(i),x,t));
        plot(t,u,'r','Linewidth',0.2);
        hold on
    end
    title("Dispalcement field")
    xlabel("Length, x")
    ylabel("Displacement, u(x)")


    hold off
   % strain field
    for i = 1:nel
        t = linspace(x1(i),x2(i),20);
        uu = double(subs(strain(i),x,t));
        plot(t,uu,'r','Linewidth',0.2);
        hold on
```

```
    end
    title("Strain field")
    xlabel("Length, x")
    ylabel("strain (x)")
end
```

NUMERICAL INTEGRATION FUNCTION

```
function integration_result = Gauss_Legendre_Integration(Functn,p,a,b)


% This script is for computing definite integrals using Legendre-Gauss
Quadrature.


syms x


if rem(p,2) == 0
    nint = ((p+1)/2)+0.5;
else
    nint = (p+1)/2;
end
[points,weight]=Gauss_Legendre_quadrature(nint,a,b);
integration_result = 0;
for ii = 1:nint
    integration_result = integration_result+subs(Functn,x,points(ii))*weight(ii);
    ii = ii+1;
end


end
```

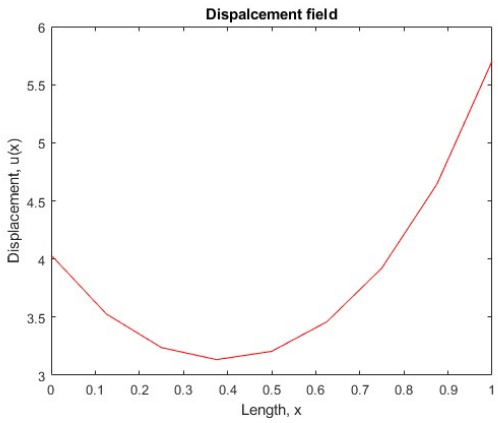
```
function [points,weight]=Gauss_Legendre_quadrature(N,a,b)
% Computes N Legendre-Gauss node points and weights  on an interval [a,b]


N1=N;
N2=N+1;
xu=linspace(-1,1,N1)';
% Initial guess
y=cos((2*(0:N-1)'+1)*pi/(2*(N-1)+2))+(0.27/N1)*sin(pi*xu*(N-1)/N2);
% Legendre-Gauss Vandermonde Matrix
```

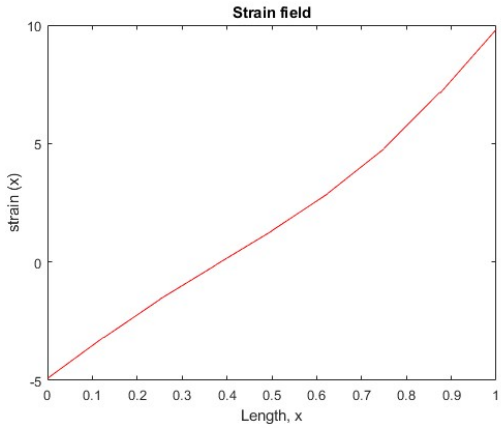
```
L=zeros(N1,N2);
% Derivative of LGVM
Lp=zeros(N1,N2);
% Compute the zeros of the N Legendre Polynomial
% using the recursion relation and the Newton-Raphson method
y0=2;
% Iterate until new points are uniformly within epsilon of old points
while max(abs(y-y0))>eps


    L(:,1)=1;
    Lp(:,1)=0;

    L(:,2)=y;
    Lp(:,2)=1;

    for k=2:N1
        L(:,k+1)=( (2*k-1)*y.*L(:,k)-(k-1)*L(:,k-1) )/k;
    end

    Lp=(N2)*( L(:,N1)-y.*L(:,N2) )./(1-y.^2);

    y0=y;
    y=y0-L(:,N2)./Lp;

end
% Linear map from[-1,1] to [a,b]
points=(a*(1-y)+b*(1+y))/2;
% Compute the weights
weight=(b-a)./((1-y.^2).*Lp.^2)*(N2/N1)^2;
end
```

OUTPUT

Case 1:

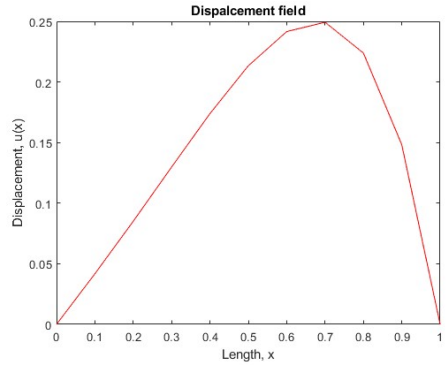
```
Enter the question number :1
Enter the value of EA:  1
spring constant assosciated with the member, k = 10
Enter the length of the bar:  1
Enter the number of elements:  8
Distribution (use small letter "x" if it is a variable) of axial force = 10
Give the force at the end x = 0 :  5
Give the force at the end x = L :  10
```

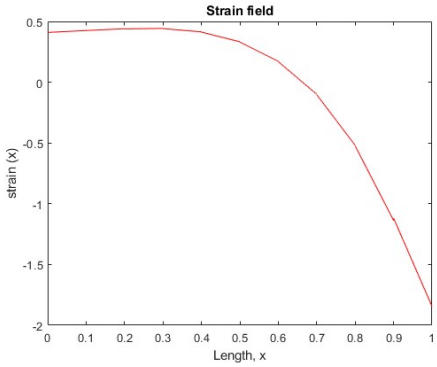




Case 2

```
Enter the question number :2
Enter the value of EA:  1
spring constant assosciated with the member, k = 10
Enter the length of the bar:  1
Enter the number of elements:  10
Distribution (use small letter "x" if it is a variable) of axial force = 10*x^2
Give the displacement at the end x = 0 :  0
Give the displacement at the end x = L :  0
```





INFERENCE

- From the previous assignment report we can compare the displacement field when using the symbolic integration to that of numerical integration implemented in this assignment. The result shows that, if we use correct point-rule, we can obtain results as accurate as symbolic integration

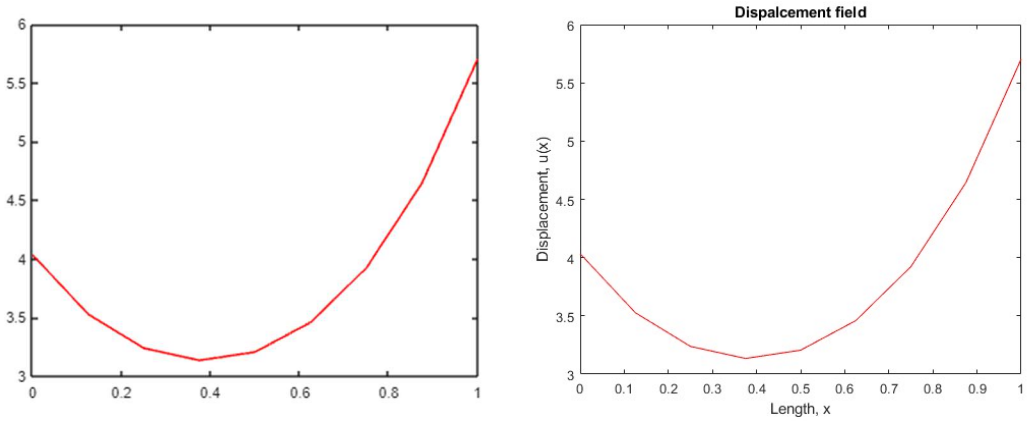


Figure 1: Case1: (a) symbolic integration (b) numerical integration

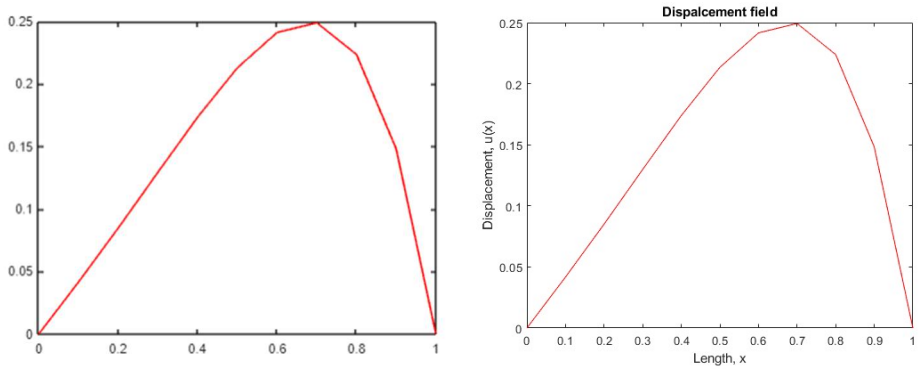


Figure 2: Case2: (a) symbolic integration (b) numerical integration

- Post processing of the results using the super convergent recovery method yields a almost smooth variation of strain field.

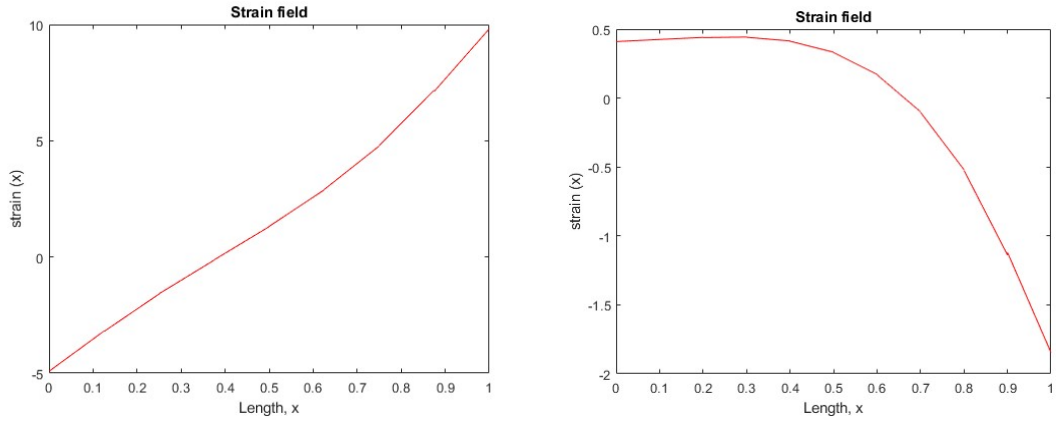


Figure 3: strain fields in case 1 and case 2 respectively