SUMIT KUMAR DAS

Doctoral Student, Department of Mechanical Engineering
Indian Institute of Technology Guwahati, Assam - 781039, India

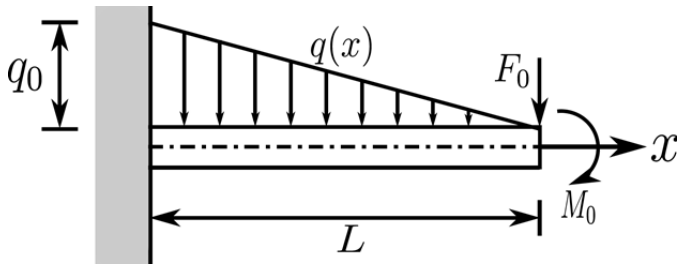
Finite Element Analysis of a Beam in MATLAB

Nov 22, 2021

Have you ever given a thought to the structures of the beam bridge, which you can generally see while travelling to your destination? Then I am sure you are curious about how it bears a massive load at every moment despite the varying nature of the load. The bridge always has only one task, and that is to carry all the burden upon it. The varying nature of load at every moment makes it difficult to analyse the structure. Earlier it wasn't easy to gather the measurement data for analysis of the frame. With the evolution of modern computer technology, it is now easier to study and observe the process. This is not only saving precious time but also the cost born in the whole process. Other structures such as automobile frames, aircraft components, and machine frames also contain beam structures.

You may be thinking about how it is possible to solve these mechanical problems in computers. But thanks to the contributors for developing various numerical methods in solving real-world problems. There are multiple methods to solve these kinds of issues, but the most popular is the finite element method (FEM), which we will use here. In this article, we will understand the underlying mathematics of the problems with the help of a real example. The programming tool we use is MATLAB. Browse through the article to know more.

Problem Statement: A cantilever beam of length 3 m, fixed at one end, is subjected to a uniformly varying load of maximum intensity of 20 kN/m throughout the beam and a concentrated load of 50 kN at the free end. A point moment of 15 kN-m is also acting at the free end of the beam. The Young's modulus and second moment of area of the beam are given as 200 GPa and 30×10^{-6} m^4, respectively.



Solution:
The solution to the above problem has been divided into five segments to give the reader a clear picture of the implementation procedure of the FEM technique.

- Governing differential equation
- Variational form
- Mapping technique
- Elemental calculations
- MATLAB implementation

Governing differential equation

The mathematical model which governs the beam structure under loading is a differential equation of fourth-order. The dependent variable present in the equation represents the transverse deflection of the beam, and the independent variable represents the longitudinal direction of the beam.

$$-\frac{d^2}{dx^2}\left(EI\frac{d^2y(x)}{dx^2}\right)+q(x)=0$$

This equation is also called the **strong form**, and every single point in the domain satisfies the equation. There are various quantities associated with this equation. x is the coordinates in the longitudinal direction of the beam, $y(x)$ is the transverse deflection of the beam due to the loading, $q(x)$ represents the load function acting on the beam. E and I mean the young's modulus of elasticity and the second moment of the beam area, respectively.

Variational form

This form helps us represent the differential form of the governing equation into a discrete form containing a set of algebraic equations, which can be easily solved with simple mathematical manipulations. The principle upon which this process is based is also called the **calculus of variation**.

In this, we consider a virtual displacement $v(x)$ of a point in the beam as a variation of actual displacement $y(x)$. Then, the average integral of the given differential equation along the virtual displacement is set to zero. This new form of the equation is also called the **variational or integral form**.

$$\int_{x_i^e}^{x_j^e} v\left[-\frac{d^2}{dx^2}\left(EI\frac{d^2y}{dx^2}\right)+q\right]dx=0$$

In this equation, the differential equation is integrated over the subdomain $\left[x_i^e, x_j^e\right]$, and v is the variation of y. Solving this equation further using integration by parts, we will get the following expressions.

$$-\int_{x_i^e}^{x_j^e} v\frac{d^2}{dx^2}\left(EI\frac{d^2y}{dx^2}\right)dx+\int_{x_i^e}^{x_j^e} vq\,dx=0$$

$$-\left[v\frac{d}{dx}\left(EI\frac{d^2y}{dx^2}\right)\right]_{x_i^e}^{x_j^e}+\int_{x_i^e}^{x_j^e}\frac{dv}{dx}\frac{d}{dx}\left(EI\frac{d^2y}{dx^2}\right)dx+\int_{x_i^e}^{x_j^e} vq\,dx=0$$

$$-\left[v\frac{d}{dx}\left(EI\frac{d^2y}{dx^2}\right)\right]_{x_i^e}^{x_j^e}+\left[\frac{dv}{dx}\left(EI\frac{d^2y}{dx^2}\right)\right]_{x_i^e}^{x_j^e}-\int_{x_i^e}^{x_j^e}\frac{d^2v}{dx^2}\left(EI\frac{d^2y}{dx^2}\right)dx+\int_{x_i^e}^{x_j^e} vq\,dx=0$$

$$-\int_{x_i^e}^{x_j^e} EI\frac{d^2v}{dx^2}\frac{d^2y}{dx^2}\,dx+\int_{x_i^e}^{x_j^e} vq\,dx-\left[v\frac{d}{dx}\left(EI\frac{d^2y}{dx^2}\right)\right]_{x_i^e}^{x_j^e}+\left[\frac{dv}{dx}\left(EI\frac{d^2y}{dx^2}\right)\right]_{x_i^e}^{x_j^e}=0$$

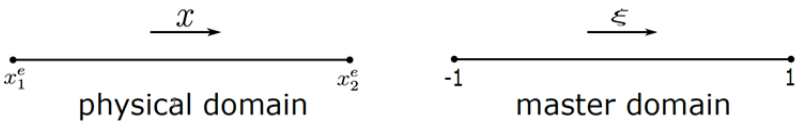
As we can notice that the original differential equation has a fourth-order differential term of the dependent variable y, the function is considered C^3 continuous. But in the variational form, the equation contains the second-order differential term of dependent variable y, resulting in a C^1 continuity requirement of the function. As the continuity requirement reduces from C^3 to C^1 in this process, the calculation time will decrease significantly. This is the reason we call this new equation a **weak form**.

To make the calculations easier, we consider only the integral terms of these equations. And the terms present in the square brackets will be treated separately later during the assembly process.

$$\int_{x_i^e}^{x_j^e} EI\frac{d^2v}{dx^2}\frac{d^2y}{dx^2}\,dx=\int_{x_i^e}^{x_j^e} vq\,dx$$

Mapping technique

At this stage, the integration is difficult to perform in physical space. The physical space is the space where the actual geometry exists. Before integrating, we need to map this space to a new space, called master space, where it is easier to implement the integration. Mapping of coordinates of points is done in the same way as done in bar problem (Ref. Finite Element Analysis of a bar in MATLAB). And here, we use the linear basis functions to map. You can see the operation for the linear mapping below.



Linear mapping of coordinates between the physical and the master domain.

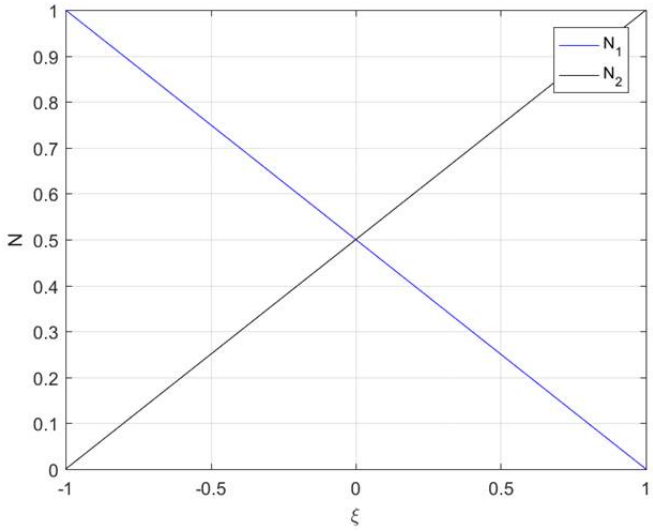
$$x^e(\xi) = \bar{\mathbf{N}}(\xi)\hat{\mathbf{x}}^e$$

$$\frac{dx^e}{d\xi} = \frac{d}{d\xi}\left(\bar{\mathbf{N}}\hat{\mathbf{x}}^e\right) = \frac{d\bar{\mathbf{N}}}{d\xi}\hat{\mathbf{x}}^e = \bar{\mathbf{B}}(\xi)\hat{\mathbf{x}}^e = J(\xi)$$

$$\hat{\mathbf{x}}^e = \begin{pmatrix} x_1^e \\ x_2^e \end{pmatrix}$$

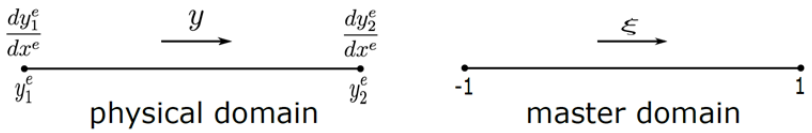
$$\bar{\mathbf{N}} = \begin{bmatrix} \bar{N}_1 & \bar{N}_2 \end{bmatrix} = \begin{bmatrix} (1-\xi)/2 & (1+\xi)/2 \end{bmatrix}$$

$$\bar{\mathbf{B}} = \begin{bmatrix} \frac{d\bar{N}_1}{d\xi} & \frac{d\bar{N}_2}{d\xi} \end{bmatrix} = \begin{bmatrix} -1/2 & 1/2 \end{bmatrix}$$



Linear shape functions.

But in the beam problem, mapping a vector containing displacement and rotation of a node is done differently, unlike the bar problem. This is because the beam is associated with two pieces of information: displacement and rotation of a node. Here we use Hermite basis functions, which are cubic interpolating polynomials. Please refer to the figure and operations, which maps a physical space to master space.



Mapping of displacement and rotation between the physical and the master domain.

$$y^e(\xi) = N_1(\xi)y_1^e + N_2(\xi)\frac{dy_1^e}{d\xi} + N_3(\xi)y_2^e + N_4(\xi)\frac{dy_2^e}{d\xi}$$

$$\frac{dy^e}{d\xi} = \frac{\frac{dy^e}{dx^e}}{\frac{d\xi}{dx^e}} = J\frac{dy^e}{dx^e}$$

$$y^e(\xi) = N_1y_1^e + N_2J\frac{dy_1^e}{dx^e} + N_3y_2^e + N_4J\frac{dy_2^e}{dx^e}$$

$$y^e(\xi) = \mathbf{N}(\xi)\hat{\mathbf{y}}^e$$

$$\hat{\mathbf{y}}^e = \begin{pmatrix} y_1^e \\ \frac{dy_1^e}{dx^e} \\ y_2^e \\ \frac{dy_2^e}{dx^e} \end{pmatrix}$$

$$\mathbf{N} = \begin{bmatrix} N_1 & N_2J & N_3 & N_4J \end{bmatrix}$$

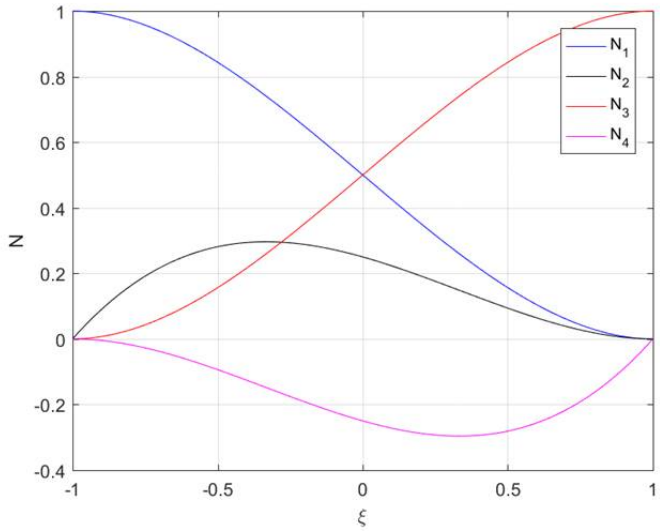
$$N_1 = \frac{1}{4}(1-\xi)^2(2+\xi)$$
$$N_2 = \frac{1}{4}(1-\xi)^2(1+\xi)$$
$$N_3 = \frac{1}{4}(1+\xi)^2(2-\xi)$$
$$N_4 = \frac{1}{4}(1+\xi)^2(\xi-1)$$

$$\frac{d^2y^e}{d\xi^2} = \frac{d}{d\xi}\left(J\frac{dy^e}{dx^e}\right) = J\frac{d}{dx^e}\left(\frac{dy^e}{dx^e}\right)\frac{dx^e}{d\xi} = J^2\frac{d^2y^e}{dx^{e2}}$$

$$\frac{d^2y^e}{dx^{e2}} = \frac{1}{J^2}\frac{d^2y^e}{d\xi^2} = \frac{1}{J^2}\frac{d^2(\mathbf{N}\hat{\mathbf{y}}^e)}{d\xi^2} = \frac{1}{J^2}\frac{d^2\mathbf{N}}{d\xi^2}\hat{\mathbf{y}}^e$$

$$\mathbf{B}(\xi) = \frac{1}{J^2}\frac{d^2\mathbf{N}}{d\xi^2}$$

$$\frac{d^2y^e}{dx^{e2}} = \mathbf{B}(\xi)\hat{\mathbf{y}}^e$$



Hermite shape functions.

A similar procedure will do the mapping for the variation term also.

$$v^e(\xi) = \mathbf{N}(\xi)\hat{\mathbf{v}}^e$$

$$\frac{d^2v^e}{dx^{e2}} = \mathbf{B}(\xi)\hat{\mathbf{v}}^e$$

Elemental calculations

As we understood the basis of transforming the physical space into master space, we can represent the weak form in the elemental form to solve it locally. Solving locally, we mean to solve the equations element-wise.

$$\int_{x_i^e}^{x_j^e} EI\frac{d^2v^e}{dx^{e2}}\frac{d^2y^e}{dx^{e2}}\,dx^e = \int_{x_i^e}^{x_j^e} v^e q\,dx^e$$

Here the superscript e states that the integration is taken element-wise. Then we can define the variables in terms of master space variables in the domain $[-1,1]$. Please refer to the following operation.

$$\int_{-1}^{1} EI(\mathbf{B}\hat{\mathbf{v}}^{\mathbf{e}})^{\mathrm{T}}(\mathbf{B}\hat{\mathbf{y}}^{\mathbf{e}})J\ d\xi = \int_{-1}^{1} (\mathbf{N}\hat{\mathbf{v}}^{e})^{\mathrm{T}} qJ\ d\xi$$

$$[\hat{\mathbf{v}}^{e}]^{\mathrm{T}} \left[\int_{-1}^{1} EIJ\mathbf{B}^{\mathrm{T}}\mathbf{B}\ d\xi\right] [\hat{\mathbf{u}}^{e}] = [\hat{\mathbf{v}}^{e}]^{\mathrm{T}} \left[\int_{-1}^{1} qJ\mathbf{N}^{\mathrm{T}}\ d\xi\right]$$

$$\left[\int_{-1}^{1} EIJ\mathbf{B}^{\mathrm{T}}\mathbf{B}\ d\xi\right] [\hat{\mathbf{u}}^{e}] = \left[\int_{-1}^{1} qJ\mathbf{N}^{\mathrm{T}}\ d\xi\right]$$

$$[\mathbf{k}^{e}]\,[\hat{\mathbf{u}}^{e}] = [\mathbf{F}^{e}]$$

As you can notice, the original differential equation is converted to the discrete form. We can easily solve them to get the results. Here $\mathbf{k}^{e}$ represents the element stiffness matrix, $\mathbf{F}^{e}$ represents the element load vector, and $\hat{\mathbf{u}}^{e}$ represents the vector containing the displacement and rotation of nodes of an element.

Now, using the available numerical integration technique, we can find the integration. For better results, we use the Gauss quadrature integration scheme. In the Gauss quadrature scheme, we can represent the integration in terms of the weighted sum of the integrand.

$$\mathbf{k}^{e} = \int_{-1}^{1} EIJ\mathbf{B}^{\mathrm{T}}\mathbf{B}\ d\xi = \left[EIJ\mathbf{B}^{\mathrm{T}}\mathbf{B}\right]_{\xi_i} w_i$$

$$\mathbf{F}^{e} = \int_{-1}^{1} qJ\mathbf{N}^{\mathrm{T}}\ d\xi = \left[qJ\mathbf{N}^{\mathrm{T}}\right]_{\xi_i} w_i$$

Here, ξ_i is called the Gauss point, and w_i is called the corresponding weight.

MATLAB implementation

In this programming tool, first, we store the given information of the problem, like geometric and material information.

```
% Information given
%=========================================================================
L   = 3;        % Length of the beam
E   = 200e+9;   % Young's modulus of the beam
I   = 30e-6;    % Second moment of area of the beam
q0  = 20000;    % Maximum intensity of the uniformly varying load in the beam
F0  = 50000;    % Transverse load at the right end of the beam
M0  = 15000;    % Concentrated moment at the right end of the beam

```

As we already discussed the integration procedure in the master domain, we need some information about the integration scheme we use. Here we use the Gauss quadrature scheme, which contains the information about the Gauss points and weights. We give this information in the MATLAB framework.

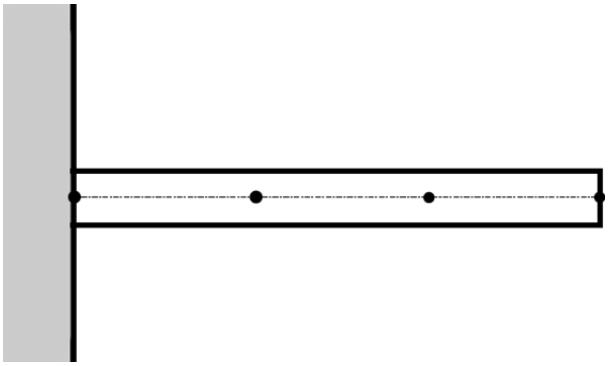
```
10
11      % Data for three-point Gauss Quadrature
12      %=======================================================
13 -    xi   = [-sqrt(3/5), 0, sqrt(3/5)];     % Gauss points
14 -    w    = [5/9, 8/9, 5/9];                % Weights
15
```

Now to find the integrands, we consider the functions. After the functions are defined, we will use the function calling technique to call the function every time we want to find the values at any Gauss point. There are two integrands in the integral equation, so two functions are defined. Please note that these are separate scrip files from the main file in MATLAB.

```
1      function ke = stiffness(xi,xvec,E,I)
2
3 -    B_bar    = [-1/2, 1/2];
4 -    J        = B_bar*xvec;
5      % N1=(1-xi)^2*(2+xi)/4;
6      % N2=(1-xi)^2*(xi+1)/4;
7      % N3=(1+xi)^2*(2-xi)/4;
8      % N4=(1+xi)^2*(xi-1)/4;
9      % N=[N1, N2*J, N3, N4*J];
10 -    ddN1     = 3*xi/2;
11 -    ddN2     = (3*xi-1)/2;
12 -    ddN3     = -3*xi/2;
13 -    ddN4     = (3*xi+1)/2;
14 -    B        = [ddN1, J*ddN2, ddN3, J*ddN4]/J^2;
15
16 -    ke       = (E*I*J)*(B'*B);
```

```
1      function Fe = loadvec(xi,xvec,L,q0)
2
3 -    N_bar    = [(1-xi)/2, (1+xi)/2];
4 -    B_bar    = [-1/2, 1/2];
5 -    J        = B_bar*xvec;
6 -    xe       = N_bar*xvec;
7 -    q        = q0*(1-xe/L);
8 -    N1       = (1-xi)^2*(2+xi)/4;
9 -    N2       = (1-xi)^2*(xi+1)/4;
10 -    N3       = (1+xi)^2*(2-xi)/4;
11 -    N4       = (1+xi)^2*(xi-1)/4;
12 -    N        = [N1, N2*J, N3, N4*J];
13
14 -    Fe       = (q*J)*N';
```

After that, we calculate the element stiffness matrix and element load vector for each element by calling the functions mentioned above. As you can see, the domain is discretised into three elements, and each element has two nodes according to the linear mapping of coordinates. This says we will have three matrices and three vectors.



Discretization of the beam into three elements with four nodes.

```
16
17      % Elemental stiffness matrices and force vectors
18      %=========================================================================
19
20      % Element1
21 -    xvec1   = [0, L/3]';        % Nodal coordinates of element1
22 -    ke1     = stiffness(xi(1),xvec1,E,I)*w(1)   + stiffness(xi(2),xvec1,E,I)*w(2)   + stiffness(xi(3),x
23 -    Fe1     = loadvec(xi(1),xvec1,L,q0)*w(1)    + loadvec(xi(2),xvec1,L,q0)*w(2)    + loadvec(xi(3),xv
24
25      % Element2
26 -    xvec2   = [L/3, 2*L/3]';    % Nodal coordinates of element2
27 -    ke2     = stiffness(xi(1),xvec2,E,I)*w(1)   + stiffness(xi(2),xvec2,E,I)*w(2)   + stiffness(xi(3),x
28 -    Fe2     = loadvec(xi(1),xvec2,L,q0)*w(1)    + loadvec(xi(2),xvec2,L,q0)*w(2)    + loadvec(xi(3),xv
29
30      % Element3
31 -    xvec3   = [2*L/3, L]';      % Nodal coordinates of element3
32 -    ke3     = stiffness(xi(1),xvec3,E,I)*w(1)   + stiffness(xi(2),xvec3,E,I)*w(2)   + stiffness(xi(3),x
33 -    Fe3     = loadvec(xi(1),xvec3,L,q0)*w(1)    + loadvec(xi(2),xvec3,L,q0)*w(2)    + loadvec(xi(3),xv
34
```

These elements are now assembled to get the global stiffness matrix and global load vector. These quantities store the primary numerical information about the whole beam. As each node is associated with two information, namely, displacement and rotation, the four nodes of the entire domain demand the eight information or eight degrees of freedom. Hence the matrix will have eight rows and eight columns.

```
35
36      % Assembly
37      %=========================================================================
38 -    kg              = zeros(8,8);           % Global stiffness matrix
39 -    Fg              = zeros(8,1);           % Global load vector
40 -    kg(1:4,1:4)     = ke1;                  % Storing ke1 in kg
41 -    kg(3:6,3:6)     = kg(3:6,3:6) + ke2;    % Storing ke2 in kg
42 -    kg(5:8,5:8)     = kg(5:8,5:8) + ke3;    % Storing ke3 in kg
43 -    Fg(1:4)         = Fe1;                  % Storing Fe1 in Fg
44 -    Fg(3:6)         = Fg(3:6)   + Fe2;      % Storing Fe2 in Fg
45 -    Fg(5:8)         = Fg(5:8)   + Fe3;      % Storing Fe3 in Fg
46 -    Fg(7)           = Fg(7)     + F0;       % Storing point load F0 in Fg
47 -    Fg(8)           = Fg(8)     + M0;       % Storing point moment M0 in Fg
48
```

The matrix will be singular at this stage, and the system doesn't have a unique solution. So, boundary conditions are imposed to solve the problems. Information about the boundary conditions can be found in Finite Element Analysis of a bar in MATLAB. In this problem, displacement and rotation (Dirichlet boundary condition) at the fixed node is zero and loads (Neumann boundary condition) at the free end are known. At this stage, we apply the

Dirichlet boundary condition because the Neumann boundary condition is already taken care of during the assembly step (refer to the MATLAB code in the assembly process).

```
49
50      % Imposition of boundary conditions
51      %=========================================================================
52 -    u       = zeros(8,1);                                   % Nodal vector containing transeverse defle
53 -    u(1)    = 0;    u(2)    = 0;                            % No displacement and rotation at the fixed
54 -    k_red   = kg(3:8,3:8);                                  % Eliminating first as well as second rows
55 -    F_red   = Fg(3:8) - kg(3:8,1)*u(1) - kg(3:8,2)*u(2);    % Eliminating first as well as second rows
56
```

Now the size of the global stiffness matrix and global load vector is reduced. We can solve the equation to find the nodal displacements and rotations at the other nodes. These solutions will be utilised to calculate the reaction forces and moments at the nodes.

```
57
58      % Solutions
59      %=========================================================================
60 -    u_red   = k_red\F_red;  % Solution of reduced equation
61 -    u(3:8)  = u_red;        % Storing u_red in the original nodal vector u
62
63
64      % Post-processing
65      %=========================================================================
66 -    F_reac = kg*u; % Reaction forces and moments at the nodes
67
```

As we obtained the displacement and rotation at nodes, we can use them to find the displacement and rotation at any point in the domain using interpolation theory.

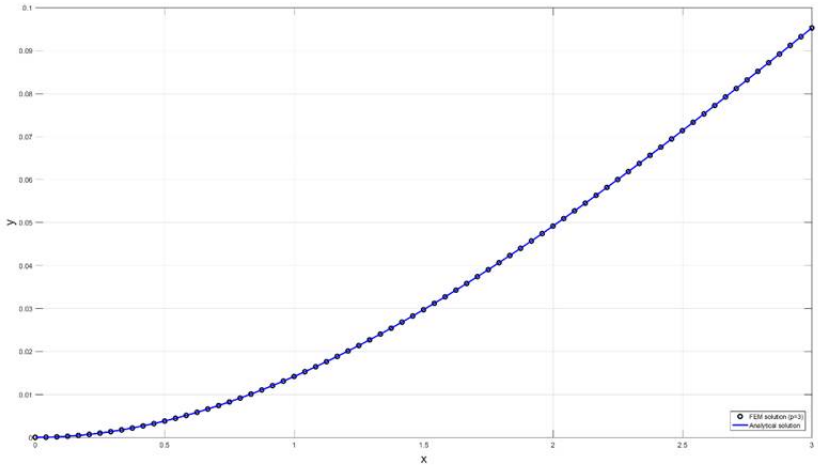
```
68
69      % FEM solution using quadratic basis functions
70      %=============================================
71 -    x   = unique([xvec1; xvec2; xvec3]);
72 -    xn  = [];
73 -    un  = [];
74 -    for i = 1:3
75 -        xe      = x(i:i+1);
76 -        ue      = u((2*i-1):2*(i+1));
77 -        xi      = linspace(-1,1,25)';
78 -        N_bar   = [(1-xi)/2, (1+xi)/2];
79 -        B_bar   = [-1/2, 1/2];
80 -        J       = B_bar*xe;
81 -        N1      = (1-xi).^2.*(2+xi)/4;
82 -        N2      = (1-xi).^2.*(xi+1)/4;
83 -        N3      = (1+xi).^2.*(2-xi)/4;
84 -        N4      = (1+xi).^2.*(xi-1)/4;
85 -        N       = [N1, N2*J, N3, N4*J];
86 -        xn      = [xn; N_bar*xe];
87 -        un      = [un; N*ue];
88 -    end
89
```

Analytical values of displacement and rotation are also calculated for the basis of comparison with the FEM solution. The expression to calculate the analytical solution is given in the below code.

```
90
91      % Analytical solution
92      %=========================================================================
93 -    xa      = linspace(x(1),x(4),75)';
94 -    xbar    = xa/L;
95 -    ua      = (q0*L^4/120/E/I)*(10*xbar.^2-10*xbar.^3+5*xbar.^4-xbar.^5) ...
96              + (F0*L^3/6/E/I)*(3*xbar.^2-xbar.^3) + (M0*L^2/2/E/I)*(xbar.^2);
97
```

As we obtained the numerical as well as the analytical results, we can compare them in a plot. The code to plot a comparison graph between FEM solution and analytical solution is presented. From the figure, we can observe the exactness of the numerical results.

```
98
99      % Comparison plot
100     %=========================================================================
101 -    plot(xn,un,'ko',xa,ua,'b-','linewidth',2);
102 -    xlabel('x','fontsize',18);
103 -    ylabel('y','fontsize',18);
104 -    legend('FEM solution (p=3)','Analytical solution','location','southeast');
105 -    grid on;
106 -    set(gcf, 'position', get(0,'Screensize'));
107 -    saveas(figure(1),'comparison','png');
108
```



I hope you enjoyed reading the article. If you are interested to learn about the bar analysis using FEM, you are referred to Finite Element Analysis of a Bar in MATLAB. Find more articles in the blog section.

Thanks for reading... 😊