Finite Element Analysis
Mechanical Engineering Department
Amrita Vishwa Vidyapeetham Coimbatore

DEFLECTION OF CANTILEVER BEAM

Compiled By:

Vinay R. Veerapur

Engineering Design(CB.EN.P2EDN19010)

Contents

1. Summary

In the following document the modelling of the cantilever beam has been made using the principles of Finite element method. The result has been compared with the analytical result. The element type used is the line element type where the entire beam element has been represented in the form of a line. The 2D line element is given displacement in the Y axis only because it is a beam element. In the end a comparison study has been performed to investigate the change and variation in the results as the mesh size would go on decreasing.

For the above demonstration the **Python 3.0** programming language is used. **About Python 3.0**: It is a multi paradigm, functional, imperative, object oriented, this programming language appeared in the 1990. Typing discipline is duck, dynamic, gradual. Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management(Unlike C++ which would go on declaring memory and would run the memory out.Meaning the Ram would get maxed out and not be managed by the programming language itself.)

The language' core philosophy is summarized in the document

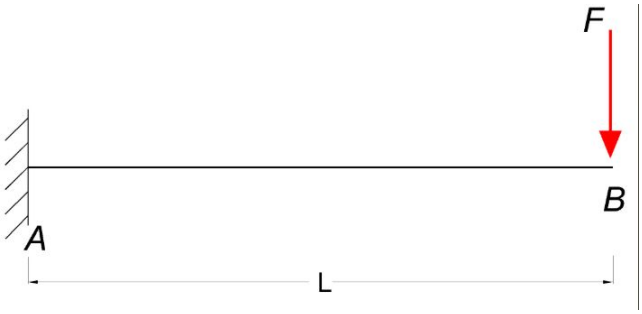
- beautiful is better than ugly
- Explicit is better than implicit
- Simple is better than complex
- Complex is better than complicated
- Readability counts

The book referred for the problem formulation is 'A First Course in the Finite Element Method' by Daryl L Logan

2. Problem definition and background

To Formulate the deflection of beam on the application of a point load using the Finite element formulation in any program of your choice

There are several software available in the market such as the Abaqus, Ansys etc. These are much more simple in application and use. These problems would run on the coding done based on the finite element method, but the coding is hidden and is done in the back ground. The program written here is to present a taste of the same and to give confidence to the reader and the learner and demystify the aura of simulations done on a computer.

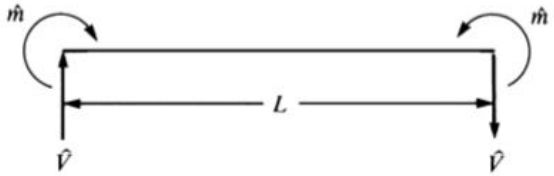


2.1 Literature review

2.1.1 Beam

A beam is a long, slender structural member generally subjected to transverse loading that produces significant bending effects as opposed to twisting or axial effects

This bending deformation is measured as a transverse displacement and a rotation. Hence, the degrees of freedom considered per node are a transverse displacement and a rotation



The Differential equation governing elementary linear elastic beam behavior is the Euler bernoulli beam. The plane cross sections perpendicular to the longitudinal centroidal axis of the beam before bending occurs remaining plane and perpendicular to the longitudinal axis after bending occurs.

Assuming the displacement function to be a 3rd order function we would have the variables as the displacement and slope.

$$N_1 = \frac{1}{L^3}(2\hat{x}^3 - 3\hat{x}^2L + L^3) \qquad N_2 = \frac{1}{L^3}(\hat{x}^3L - 2\hat{x}^2L^2 + \hat{x}L^3)$$

$$N_3 = \frac{1}{L^3}(-2\hat{x}^3 + 3\hat{x}^2L) \qquad N_4 = \frac{1}{L^3}(\hat{x}^3L - \hat{x}^2L^2)$$

we would come up with the above shape functions based on the displacement functions. The displacement functions N1 and N3 are for the displacements and the N2 and N4 are for the slopes.

Upon defining the stress strain relationships and the elemental stiffness matrix and the equations we would end up the elemental equations relating the deflections and the slopes with the applied forces and moment is

$$\begin{Bmatrix} \hat{f}_{1y} \\ \hat{m}_1 \\ \hat{f}_{2y} \\ \hat{m}_2 \end{Bmatrix} = \frac{EI}{L^3} \begin{bmatrix} 12 & 6L & -12 & 6L \\ 6L & 4L^2 & -6L & 2L^2 \\ -12 & -6L & 12 & -6L \\ 6L & 2L^2 & -6L & 4L^2 \end{bmatrix} \begin{Bmatrix} \hat{d}_{1y} \\ \hat{\phi}_1 \\ \hat{d}_{2y} \\ \hat{\phi}_2 \end{Bmatrix}$$

where the square matrix is the stiffness matrix.The Stiffness matrix is assembled in the global domain. this global domain matrix is solved.

2.2 Reference solution

The reference solution that is taken for this particular case would be the deflection of cantilever beam analytical formula. This formula would be the comparison for the deflection of the beam. The formula is as follows:

$$deflection = \frac{WL^3}{3EI}$$

Where the W would be the point load, L would be the lendth of the entire span, the EI is the flexural rigidity. The reference solution of each test case is taken as the analytical formula the reason being 1) experimental data for the each case would be difficult to obtain and generalization would become very difficult. 2) It is easy compare the result as the calculation is done on the system and not needed to enter manually for each iterations. The above stated reason may be trivial but the subtle details would matter for detailed study.

3. Programing

3.1 Python Program

```
#GITHUB https://github.com/Vinay5SVeerapur/Finite-element-analysis
#That is, keep the units consistent in the input variables
print("DEFLECTION OF CANTILEVER BEAM")

#Beam length
Length=float(input("Enter the length of Beam: "))

#Larger side of the cross section
Breadth=float(input("Enter the Breadth of the cross section:"))

#Smaller side of cross section
Width=float(input("Enter the width of the cross section:"))

#cross section area
Area=Breadth*Width

#Youngs modulus input
E=float(input("Enter the Young's Modulus:"))

#The discritization
nodes=int(input("Enter the Mesh nodes Number:(How many nodes you want)"))

import numpy as np

#The stiffness matrix
Main=np.zeros((nodes*2,nodes*2))

#Force matrix
Force=np.zeros((nodes*2-2,1))

#Elemental length
L=Length/(nodes-1)

#Enter the Point load at the end node
pointload=float(input("Enter the Load N at the end:"))

#payload initialized to the Global Matrix
```

```
Force[(nodes*2)-2-2][0]=pointload
b=np.zeros((nodes*2,1))
matrix=np.zeros((4,4))

#Moment of Inertia
I=(Breadth*(Width**3))/12

#Global Matrix assemblage using elemental matrix
Add=np.zeros((nodes*2,nodes*2))
inverse=np.zeros((nodes*2-2,nodes*2-2))
for i in range(nodes*2):
    for j in range(nodes*2):
        if i==j and i%2==0 and j%2==0 and i!=nodes*2-2 and j!=nodes*2-2:
            Add[i][j]=((E*I)/L**3)*12
            Add[i][j+1]=((E*I)/L**3)*6*L
            Add[i][j+2]=((E*I)/L**3)*-12
            Add[i][j+3]=((E*I)/L**3)*6*L
            Add[i+1][j]=((E*I)/L**3)*6*L
            Add[i+1][j+1]=((E*I)/L**3)*4*L**2
            Add[i+1][j+2]=((E*I)/L**3)*-6*L
            Add[i+1][j+3]=((E*I)/L**3)*2*L**2
            Add[i+2][j]=((E*I)/L**3)*-12
            Add[i+2][j+1]=((E*I)/L**3)*-6*L
            Add[i+2][j+2]=((E*I)/L**3)*12
            Add[i+2][j+3]=((E*I)/L**3)*-6*L
            Add[i+3][j]=((E*I)/L**3)*6*L
            Add[i+3][j+1]=((E*I)/L**3)*2*L**2
            Add[i+3][j+2]=((E*I)/L**3)*-6*L
            Add[i+3][j+3]=((E*I)/L**3)*4*L**2
            Main=Main+Add
            Add.fill(0)

#importing the inverse function for matrix
from numpy.linalg import inv
#Linear Equation solution function importing
from scipy import linalg
for i in range(nodes*2):
    for j in range(nodes*2):
        if i >1 and j>1:
            inverse[i-2][j-2]=Main[i][j]        #Inverse of the matrix
#Finding the variable matrix by dot product
solution=np.dot(inv(inverse),Force)

#will display the deflection of last node
print("FEM Calculation:",solution[(nodes*2)-1-2-1][0])

#deflection by formula (WL^3/3EI)
print("Analytical formula:",(pointload*Length**3)/(3*E*I))
```

```
print("Error% : ",100*(((pointload*Length**3)/(3*E*I))
-(solution[(nodes*2)-1-2-1][0]))/((pointload*Length**3)/(3*E*I)))
```

3.2 Solution

```
DEFLECTION OF CANTILEVER BEAM
Enter the length of Beam: 1000
Enter the Breadth of the cross section:15
Enter the width of the cross section:50
Enter the Young's Modulus:200000
Enter the Mesh nodes Number:(How many nodes you want)100
Enter the Load N at the end:300
FEM Calculation: 3.1999999826041665
Analytical formula: 3.2
Error% :  5.436198019515004e-07
```

The above text is the output of the program, the numbers input is in MM (millimeter) and the loads are all provided in the Newtons. The Young's modulus is all according to the standard inputs. (Just like in Abaqus). The 'FEM Calculation:' is the solution that is calculated by the matrix inverse after the Gaussian elimination and the 'Analytical Formula' is the deflection calculated by the formula.

In the above example we have the length of 1meter and a cross section of 50mm by 15 mm and a load of 30 kg or 300N is applied along the 50 mm dimension, the deflection that we have got is 3.199mm and the deflection that we get from the analytical formula is 3.2mm. The error percentage is negligible. The number of nodes chosen is 100 meaning the discritization is having 99 elements.

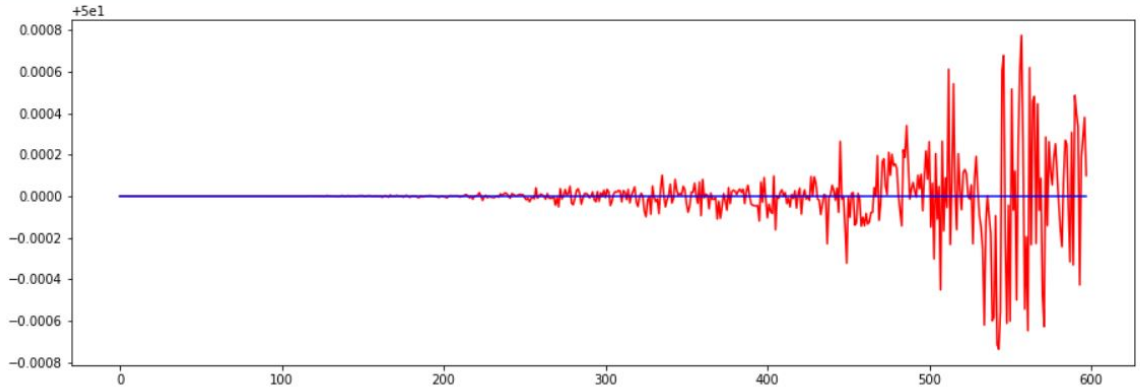
Repository Github for the program https://github.com/Vinay5SVeerapur/Finite-element-analysis

4. NODES vs Deflection/Displacement Study

Since the program has been set up. An additional study has been done out of curiosity. The experiment is to study how the deflection would vary as the nodes would be increased. Whether the accuracy of the data go on increasing? The above were the questions that were an inspiration to do the study.

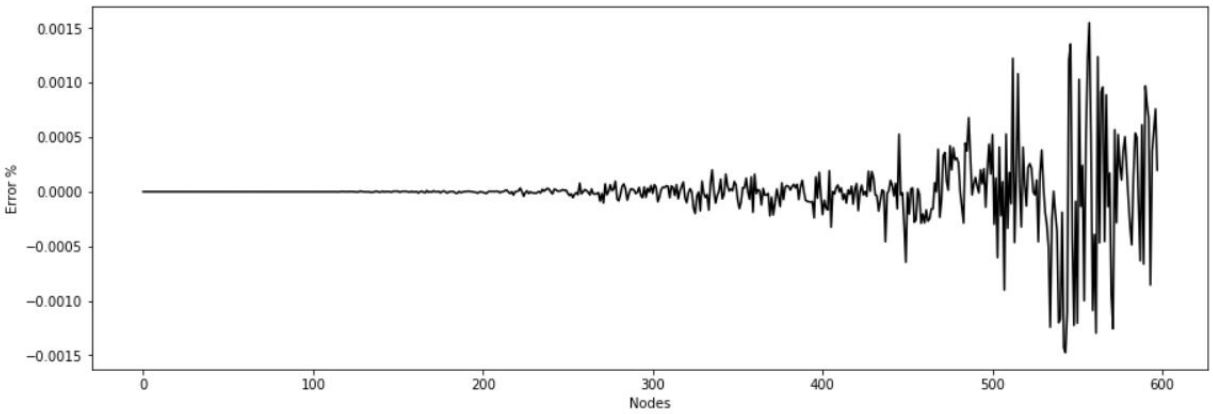
The same program has been modified in such a way that the input data like the length, E and breadth etc would remain the same just the nodes would go on increasing. The solution obtained for each of the configuration is stored and plotted against the reference value that is the analytical constant formula.

```
DEFLECTION OF CANTILEVER BEAM
Enter the length of Beam: 2000
Enter the Breadth of the cross section:20
Enter the width of the cross section:20
Enter the Young's Modulus:200000
Enter the Load N at the end:50
Calculating: 100%|████████████████████████████████████████████████████| 598/598 [14:32<00:00,  1.46s/it]
```



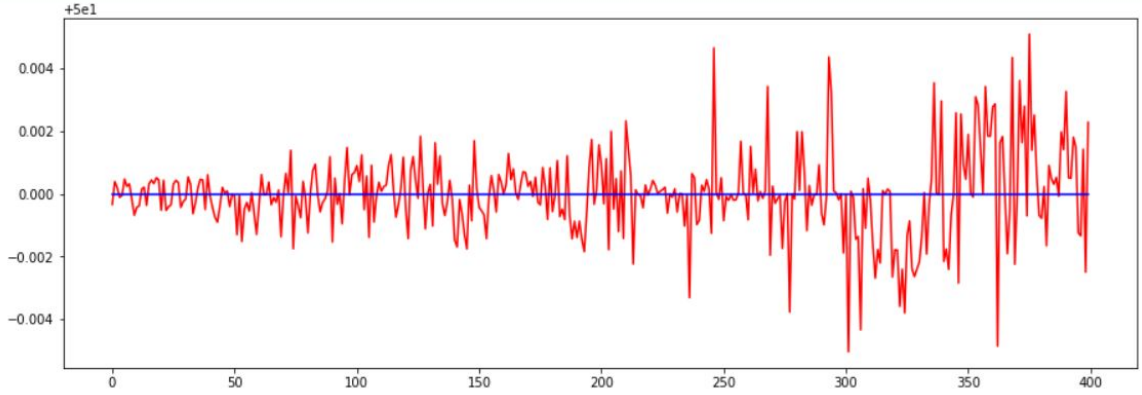
The above is the output of the modified program. where it is plotting the deflection vs the nodes in the red line. And the Blue line is the constant analytical formula of the considered formulation. In the above study a 2 meter beam with 20x20mm cross section, Steel material and 5 Kg load was considered. The calculations for the deflections have been done for 2 nodes to 600 nodes. The entire calculation needed around 14 mins of computation time.

The expectation was that as the nodes would go on increasing the solution would stabilize around the analytical solution that is the blue line, but after around 200 nodes the solution is moving up and down from the analytical solution. Even though the movement looks large but the value by which it is changing is negligible.



In-order to get more clarity the Error percentage vs the nodes is shown, just so that we don't get misled into believing wrong conclusions. In the above graph we can clearly see that the maximum oscillation is between 0.0015 to -0.0015 Percentages. which is not much. This is still permissible as it is less than 5 percentage. Since we are seeing that the error is going on increasing as we are increasing the nodes can increase the nodes and see how it changes the error.

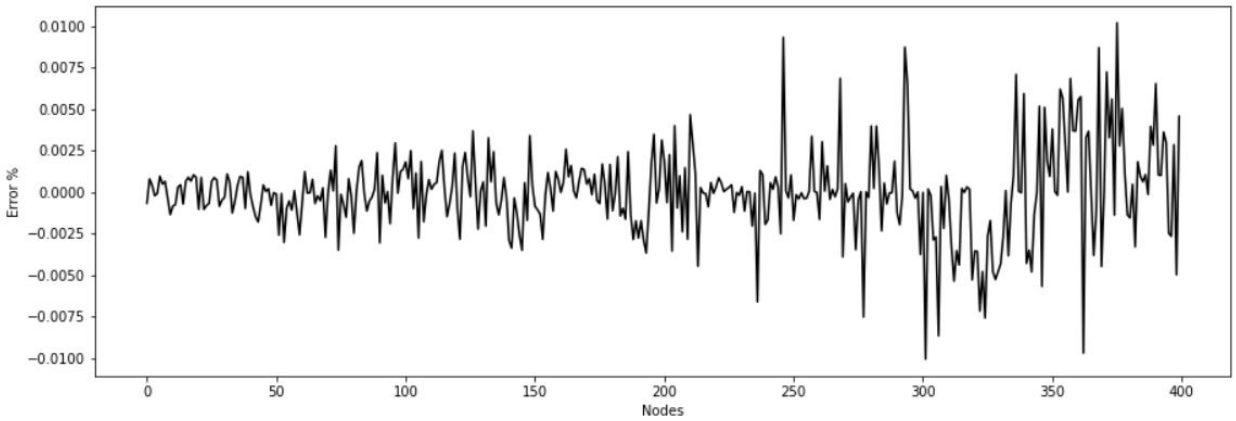
```
DEFLECTION OF CANTILEVER BEAM
Enter the length of Beam: 2000
Enter the Breadth of the cross section:20
Enter the width of the cross section:20
Enter the Young's Modulus:200000
Enter the Load N at the end:50
Calculating: 100%|████████████████████████████████████████████████| 400/400 [1:28:10<00:00, 13.23s/it]
```



The above graph shows the variation of the deflection solution with respect to nodes (600 to 1000). Where the zeroth data corresponds to the data for 400 nodes and the 400th data on the graph is for the 1000 nodes.The total computation time was 1hr 30mins. The time per iteration is 13.21 seconds . All these statistics are collected using the TQDM module of python library.So, as the nodes goes on increasing the deviation to the analytical value goes on increasing. Which is counter intuitive.

Upon careful analysis of the above graph it is visible that at some of the locations the value is near the analytical value. The main question is , why is there a spike and sudden deviations at some of the node numbers to the analytical data. For example at the nodes between 900 nodes to the 950 nodes the data just dips below the

blue line. And between the 950 to 1000 range there is a huge fluctuation in the data around the Blue line or the analytical line.



This is the error percentage to the analytical value. Here it is clearly visible that the max and min range for the error has changed to 0.01 to -0.01 percentage compared to the 0.0015 percentage. This change is an appreciable change. It is very interesting to note that the best result is got when the element is just 1 where there is less error.

5. Results

The goal of the project was to determine the deflection of the cantilever beam and comparing the results to the analytical value, The program to calculate the deflection was made in python. A special investigative study was also carried out. Interesting observation was observed. Solution instead of getting closer and closer to the analytical value the solution is oscillating about the analytical value. The results were observed for 2 to 1000 nodes of discritization.

6. Acknowledgements

I would like to thank Prof Dr Ajith Ramesh for the support and the guidance to complete this modelling. The infrastructure support at Amrita helped me to complete this simulations.