Formulations in Nonlinear Mechanics

BACHELOR OF TECHNOLOGY
IN
MECHANICAL ENGINEERING

Submitted by:

Jheel Patel (17BME044)

MECHANICAL ENGINEERING DEPARTMENT
INSTITUTE OF TECHNOLOGY
NIRMA UNIVERSITY OF SCIENCE AND TECHNOLOGY

CERTIFICATE

TO WHOM IT MAY CONCERN

This is to certify that, Mr. Jheel Nikulkumar Patel student of Mechanical engineering, 5th Semester of Nirma University, Institute of Technology, School of Engineering has satisfactorily completed the Mini project report titled Formulations in Nonlinear Mechanics.

Date: 25th November 2019

Guide (s): Prof. Dhaval B. Shah

Head of the Department

Approval Sheet

The Project entitled **Formulations in Nonlinear Mechanics** by **Jheel Nikulkumar Patel (17BME044)** is approved for the course MINI PROJECT I in Mechanical Engineering.

Examiners

_____

_____

_____

Date: _____

Place: _____

ACKNOWLEDGMENT

The success and final outcome of this project required a lot of guidance and assistance from many people and I am extremely privileged to have got this all along the completion of my project. All that I have done is only due to such supervision and assistance and I would not forget to thank them.

Jheel Patel (17BME044)

Date: 25th November 2019
Place: Ahmedabad

ABSTRACT

In science and engineering, before designing some technology a relation between the function (of the technology) and the underlying physics is to be derived from the principles and laws permitted by nature. Often these relations are expressed in terms of mathematical models. Generally, these systems are nonlinear in nature. Nonlinearity has physical significance in nature and it is not only a mathematical definition. Mathematical modeling provides a way to express natural systems and these models are classified accordingly for linear and nonlinear systems. For linear systems analytical solutions can be obtained for special cases but for nonlinear systems it requires very rigorous treatment to obtain analytical solutions even for simplified cases. So, we try to obtain approximate solutions by converting the mathematical models (generally differential equations) into algebraic equations. These algebraic equations can further be linearized in case if the algebraic equations are nonlinear. These algebraic equations are then solved with the help of computers to obtain the required solution. An approximation to the original solution is obtained but it is sufficient and practical as far as engineering domain is concerned. In this project finite element method is employed to solve for nonlinear bending of Euler-Bernoulli beam. The procedure is then analyzed for different cases and iterations.

Key words: Geometric nonlinearity, Nonlinear FEA, Euler-Bernoulli Beams

TABLE OF CONTENTS

LIST OF FIGURES

Page No.

Abbreviation, Notations and Nomenclature

Vector and Matrix Notations

Suppose a structural element as shown below,

$$u_i = u_i$$

$$\Delta_1 = w_a \Delta_2 = \theta_a \Delta_3 = w_b \Delta_4 = \theta_b$$

Where, u, w and θ denotes axial, transverse and angular displacements.
Global vectors:

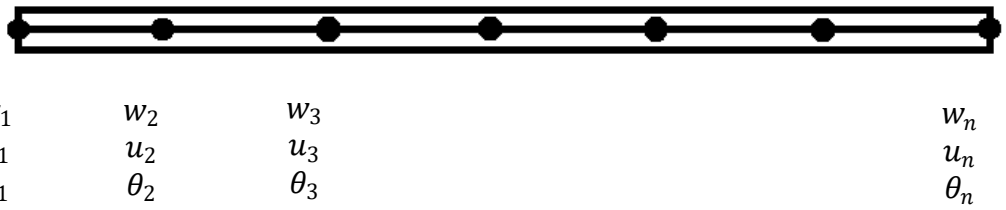


Fig. (a) Nodal displacement notation

$$\Delta = \begin{bmatrix} w_1 \\ \theta_1 \\ w_2 \\ \theta_2 \\ \cdot \\ \cdot \\ \cdot \\ w_n \\ \theta_n \end{bmatrix} \qquad u = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \cdot \\ \cdot \\ \cdot \\ u_n \end{bmatrix}$$

CHAPTER 1

Mathematical Prerequisites

1.1 Interpolation functions

Interpolation technique is used to obtain value of dependent variable at some point using the known/given values of dependent variable near the point of interest.
E.g. Suppose a line element with endpoints x_a and x_b is shown below,

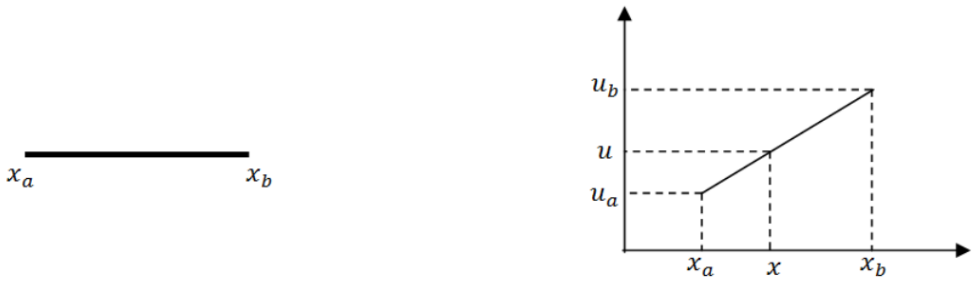


Fig. 1.1 Finite element with displacement plot

Let u_a and u_b be the displacement of the element at endpoints x_a and x_b. Suppose we are interested to find displacement at point x of the element. Using linear interpolation, (Lagrange 2-point interpolation),

$$\frac{x_b - x}{x - x_a} = \frac{u_b - x}{x - u_a} \qquad \ldots (1.1)$$

From above relation unknown u can be found.
Similarly, displacement at any point between/near the endpoints can be obtained using linear interpolation (C^1 continuous).
For quadratic interpolation (C^2 continuous), 3 points are required and so on. 3-point interpolation function is obtained as shown below,

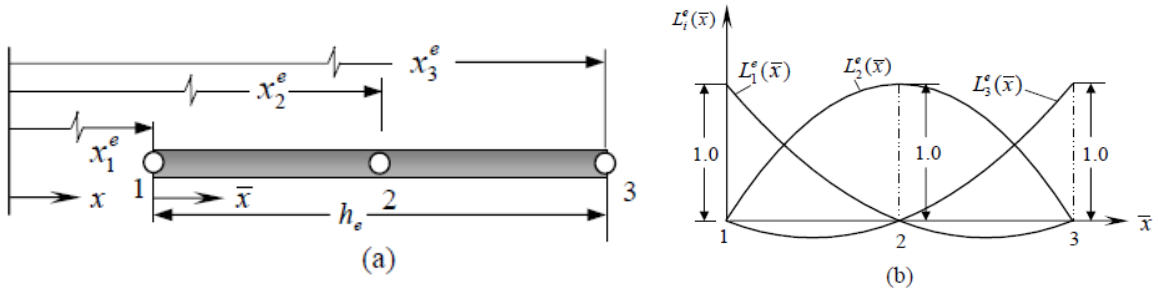


Fig. 1.2 (a) Quadratic finite element, (b) Quadratic Lagrange interpolation functions

$$u(x) = L_a(x)u_a + L_b(x)u_b + L_c(x)u_c \quad \ldots (1.2)$$

$$L_a(x) = \left(\frac{x - x_b}{x_a - x_b}\right)\left(\frac{x - x_c}{x_a - x_c}\right)$$

$$L_b(x) = \left(\frac{x - x_a}{x_c - x_a}\right)\left(\frac{x - x_c}{x_b - x_c}\right)$$

$$L_c(x) = \left(\frac{x - x_a}{x_c - x_a}\right)\left(\frac{x - x_b}{x_c - x_b}\right)$$

Linear and Hermite cubic interpolation will be used based on the type of continuity required.

1.1.1 Linear Lagrange interpolation

C^1 continuous.

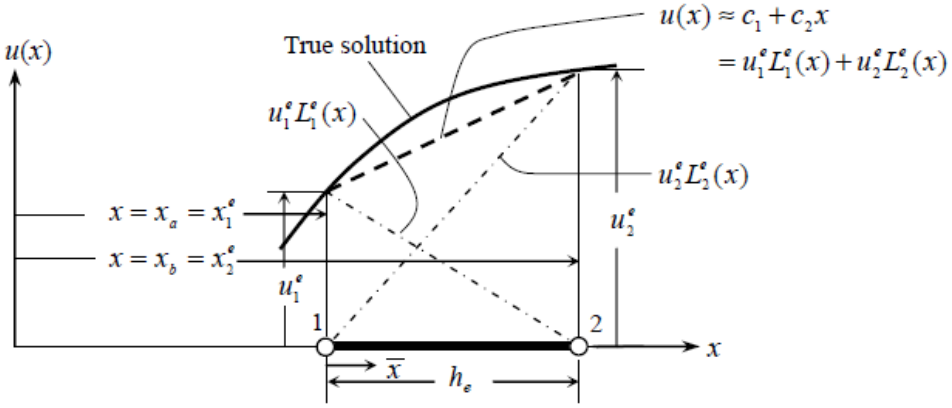


Fig. 1.3 Linear finite element approximation over an element

$$u(x) = L_a(x)u_a + L_b(x)u_b \quad \ldots (1.3)$$

$$L_a(x) = \left(\frac{x_b - x}{x_b - x_a}\right)$$

$$L_b(x) = \left(\frac{x - x_a}{x_b - x_a}\right)$$

1.1.2 Hermite Cubic interpolation

C^3 continuous. 3-degree polynomial, hence, 4 coefficients are required which are determined from the value of dependent variables at the endpoints and the derivatives.

$$\theta_a = \left(\frac{dw}{dx}\right)_{x=a} \quad \theta_b = \left(\frac{dw}{dx}\right)_{x=b} \qquad \ldots (1.4)$$

$$w(x) = h_{00}(t)w_a + h_{10}(t)\cdot(x_b - x_a)\cdot\theta_a + h_{10}(t)w_b + h_{11}(t)\cdot(x_b - x_a)\cdot\theta_b \qquad \ldots (1.5)$$

$$t = \frac{x - x_a}{x_b - x_a}$$

$$h_{00}(t) = 2t^3 - 3t^2 + 1 \qquad h_{01}(t) = -2t^3 + 3t^2$$

$$h_{10}(t) = t^3 - 2t^2 + 1 \qquad h_{11}(t) = t^3 - t^2$$

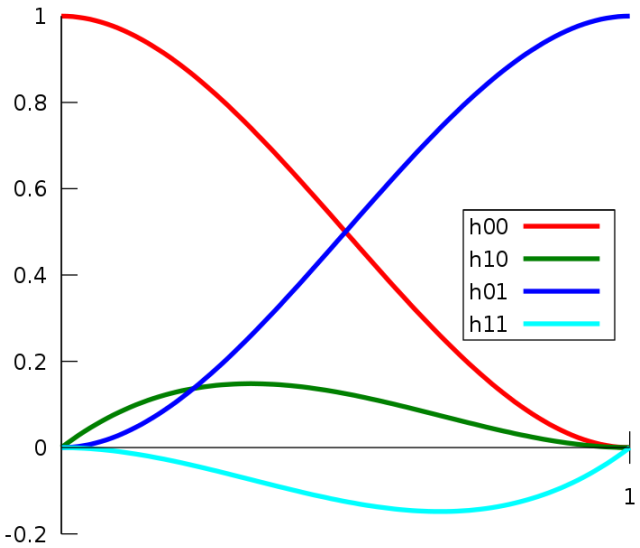


Fig. 1.4 The four Hermite basis functions

1.2 Numerical Integration

Composite trapezoidal rule:

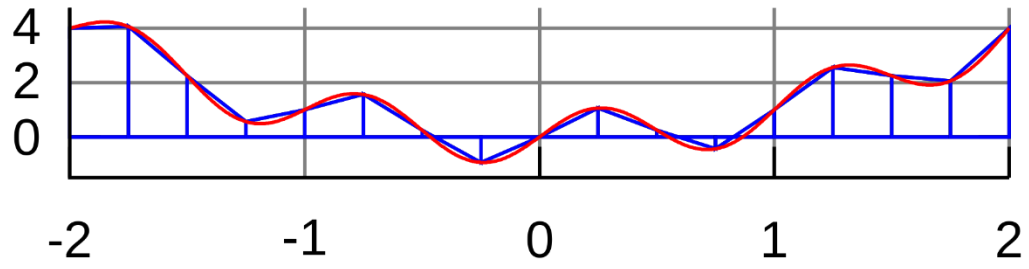


Fig. 1.5 Trapezoidal rule

$$I = \int_a^b f(x)dx \approx \frac{b-a}{n}\left(\frac{f(a)}{2} + \sum_{k=1}^{n-1}\left(f\left(a + k\frac{b-a}{n}\right)\right) + \frac{f(b)}{2}\right)$$

n is the number of trapezoidal elements.
In composite trapezoidal rule, the area under the curve is divided into trapezoidal elements instead of rectangular elements (increasing accuracy).

CHAPTER 2

Linearity and Nonlinearity

Assume a relation expressed as,

$$T(u) = 0 \qquad \ldots (2.1)$$

Where, u is the dependent variable.
Linearity obeys following condition (Linearity Conditions):

$$T(\alpha u_1 + \beta u_2) = \alpha T(u_1) + \beta T(u_2) \qquad \ldots (2.2)$$

Where, α and ß are constants and $u_1$and $u_2$are two solutions for the expression.
If an expression does not satisfy the above condition then it is termed as nonlinear. Now, we'll discuss the nature of nonlinearity in structural mechanics.

2.1 Nonlinearity in Structural Mechanics

Features of Nonlinear Systems [1]:

- **The principle of superposition does not hold.**
  Suppose for example bending of beams. If we consider linear bending limit then the beam deflection is directly proportional to the load, so, if we consider two load cases on two different beams then the combined effect of both the load cases on a single beam is the summation of deflections due to both the load cases applied individually. But this is not the cases in nonlinear bending.
- **Analysis can be carried out for one load case at a time.**
  Due to the same reason as that of non-applicability of the superposition principle. In nonlinear system the sequence in which load cases are applied is important.
- **The history (or sequence) of loading influences the response.**
  Suppose in case of beam, there are two load cases. One load case is applied and deformation occurs. Now the second load case is applied on a deformed beam. In this case the final deformation may be different than would be the case if the sequence of load application is reversed. Hence, the sequence of loading influences the response.
- **The initial state of the system may be important.**
  The reason is same as that made in the previous point. Before application of load cases, initial state of the system due to pre-stress or initial deformation must be considered.

Types of Nonlinearity in Structural Mechanics:

1. **Geometric Nonlinearity**
   Generally, this type of nonlinearity is caused by nonlinear relation between strain and displacement. It is mainly due geometric consideration. Another example is - the change in load configuration due to large deformations.
2. **Material Nonlinearity**
   This type of nonlinearity occurs due to nonlinearity in material properties. For example, nonlinear stress-strain relation.

It is very difficult and mathematically rigorous to find analytical solution for such nonlinear problems. So, we try to find analytical solution. Here one such method called finite element analysis is used to solve nonlinear bending of straight beam.

CHAPTER 3

Finite Element Analysis

Outline of Finite Element Analysis,

a. Mesh Description
b. Kinematic Description: Strain Description
c. Kinetic Description: Stress Tensor and Momentum Equation
d. Finite Element modeling
e. Solving Algebraic equations

3.1 Mesh Description

3.1.1 Dividing Domain into Sub domains (Finite Elements)

In finite element approximation, instead of finding solution which continuously satisfies the governing equations over the whole domain we try to find values of the dependent variable at some discrete points (nodes) in the domain. These nodes divide whole domain into small sub domain called finite elements. From the nodal values, shape functions can be found to obtain values of the dependent variable at other points in the finite elements using interpolation techniques. Using shape functions, weak forms are constructed from which algebraic equations are obtained. After applying initial and boundary conditions, algebraic equations are solved to obtain unknown nodal values.

As shown in the figure, beam geometry is divided into small sub domains.

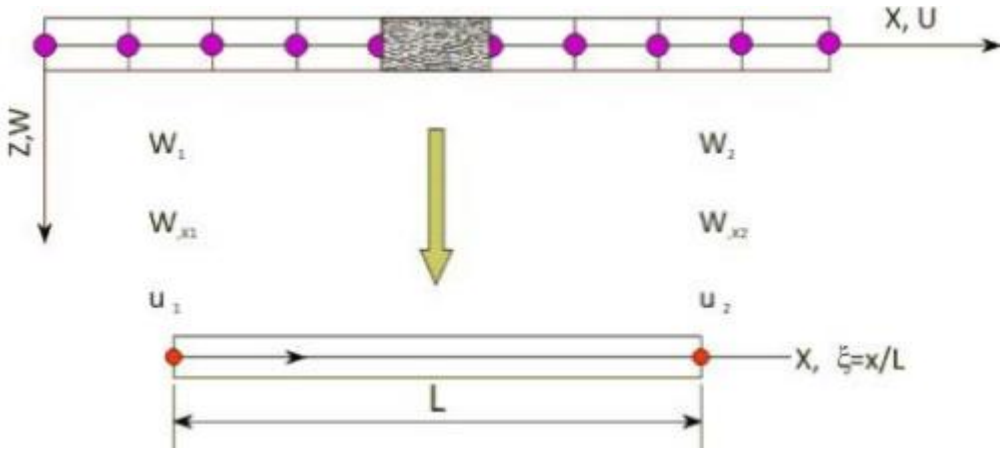


Fig. 3.1 Subdivision of domain into elements

This sub division of geometry into small elements is called meshing. Mesh can be described into two ways; Eulerian mesh description and Lagrangian mesh description.

3.1.2 Lagrangian vs. Eulerian Description [2]

In continuum mechanics, spatial relations can be dealt with either Lagrangian (material) or Eulerian (spatial) description. In Lagrangian description, properties are evaluated following a material particle whereas is Eulerian description properties are evaluated at a specific point in space. In Lagrangian description the spatial point at which properties are evaluated may change with time as we are following the material particle. In Eulerian description the spatial point is same but the material particle occupying that point may change.

Material Coordinates are denoted by **X** and spatial coordinates are denoted by **x**, and initially spatial coordinates are coinciding with material coordinates,

$$\boldsymbol{X} = \boldsymbol{x} \quad at \quad \boldsymbol{t} = 0$$

And,

$$\boldsymbol{x} = \boldsymbol{\varphi}(\boldsymbol{X}, t)$$

is map between spatial coordinates and material points at time t.

Displacement **u** of a material point is,

$$\boldsymbol{u}(\boldsymbol{X}, t) = \boldsymbol{\varphi}(\boldsymbol{X}, t) - \boldsymbol{X} = \boldsymbol{x} - \boldsymbol{X}$$

The Lagrangian coordinates of nodes are time-invariant i.e. the nodes coincide with the material points. Nodal trajectories coincide with the material point trajectories and no material passes between elements. Whereas the Eulerian coordinates of nodes are fixed in space i.e. nodes coincide with spatial points. The nodal trajectories are vertical lines and material points pass across element interfaces.

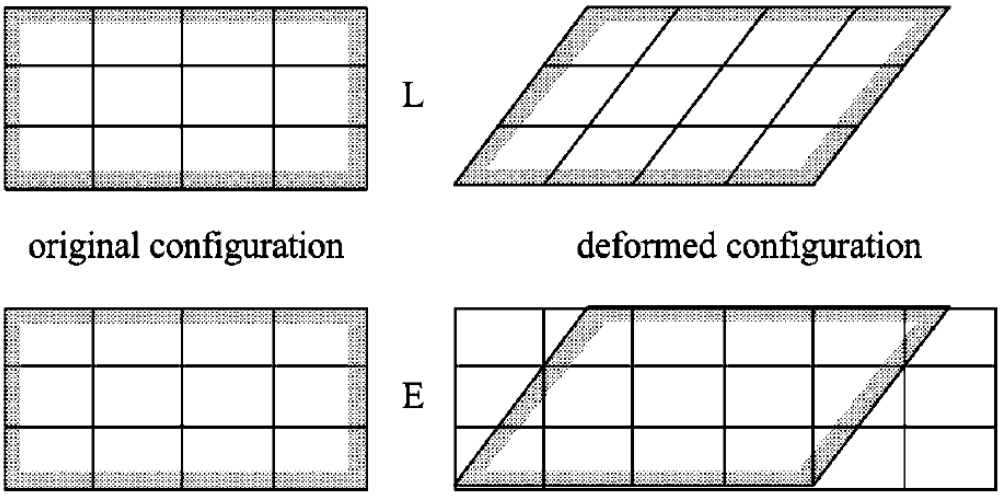


Fig. 3.2 (L) Lagrangian Mesh Description and (E) Eulerian Mesh Description (Ted Belytschko)

3.2 Kinematic Description

3.2.1 Strain Measure [1]

Deformation gradient is defined by,

$$\boldsymbol{F} = \frac{\boldsymbol{\partial \varphi}}{\boldsymbol{\partial X}} = \frac{\boldsymbol{\partial x}}{\boldsymbol{\partial X}}$$

Jacobian is defined as the ratio of an infinitesimal volume in the deformed body to the corresponding segment volume in the undeformed body,

$$\boldsymbol{J} = \frac{\boldsymbol{\partial x} A}{\boldsymbol{\partial X} A_0} = \boldsymbol{F}\frac{A}{A_0}$$

Distance between two material particles separated by infinitesimal displacement $\boldsymbol{dX}$ is,

$$(dS)^2 = d\boldsymbol{X} \cdot d\boldsymbol{X} \quad without\ deformation$$
$$(ds)^2 = d\boldsymbol{X} \cdot (\boldsymbol{F^T} \cdot \boldsymbol{F}) \cdot d\boldsymbol{X} = d\boldsymbol{X} \cdot \boldsymbol{C} \cdot d\boldsymbol{X} \quad after\ deformation$$

Where $\boldsymbol{C}$ is Cauchy-Green deformation tensor,

$$\boldsymbol{C} = \boldsymbol{F}^T \cdot \boldsymbol{F}$$

The change in squared length is,

$$(ds)^2 - (dS)^2 = 2d\boldsymbol{X} \cdot \boldsymbol{\varepsilon} \cdot d\boldsymbol{X}$$

Where, $\boldsymbol{\varepsilon}$ is Green-Lagrange (Green strain tensor for short) strain tensor,

$$\boldsymbol{\varepsilon} = \frac{1}{2}(\boldsymbol{F^T} \cdot \boldsymbol{F} - \boldsymbol{I}) = \frac{1}{2}[(\nabla_0 \boldsymbol{u})^T + (\nabla_0 \boldsymbol{u}) + (\nabla_0 \boldsymbol{u})^T \cdot (\nabla_0 \boldsymbol{u})]$$

$$\nabla_0 \boldsymbol{u} = \frac{\boldsymbol{\partial u}}{\boldsymbol{\partial X}}$$

Therefore, in indicial notation,

$$\varepsilon_{IJ} = \frac{1}{2}\left(\frac{\partial x_m}{\partial X_I}\frac{\partial x_m}{\partial X_J} - \delta_{IJ}\right) = \frac{1}{2}\left(\frac{\partial u_I}{\partial X_J} + \frac{\partial u_J}{\partial X_I} + \frac{\partial u_K}{\partial X_I}\frac{\partial u_K}{\partial X_J}\right) \quad \ldots (3.1)$$

For linear case above equations are approximated to,

$$\varepsilon_{IJ} = \frac{1}{2}\left(\frac{\partial u_I}{\partial X_J} + \frac{\partial u_J}{\partial X_I}\right) \quad \ldots (3.2)$$

3.3 Kinetic Description

3.3.1 Constitutive Equation

Constitutive equations give relation between stress and strain components. For linear elastic and isotropic material,

$$\sigma_{IJ} = E\varepsilon_{IJ} \quad \dots (3.3)$$

Where, *E* is the elastic constant (Young's Modulus).
Note: Here we are going to study geometric nonlinearity only and material nonlinearity will not be considered.

3.3.2 Weak Form and Momentum Equation [3]

Using equilibrium conditions strong form (momentum equation) can be derived and from strong form weak form of the equations can be derived using weighted residual methods or variational methods. Here weak form of equation is derived using principle of virtual work. Without knowing the governing differential equations (momentum equation).
The principle of Virtual work states that if a body is under equilibrium, the total work done by external loads and internal forces through their respective displacements is zero. Virtual displacements are arbitrary. Except at fixed boundaries where virtual displacement is zero.
Over a typical element $\Omega^e(x_a, x_b)$, analytical form of principle of virtual work is,

$$\delta W^e \equiv \delta W_I^e - \delta W_E^e \quad \dots (3.4)$$

δW_I^e is the virtual energy stored in the element e due to work done by stresses in moving through virtual strains $\delta\varepsilon_{ij}$. δW_E^e is the work done by external loads in moving through their respective virtual displacements.
For the beam element e,

$$\delta W_I^e = \int_{V^e} \delta\varepsilon_{ij}\,\sigma_{ij}\,dV \quad \dots (3.5)$$

$$\delta W_E^e = \int_{x_a}^{x_b} q\delta w_0 + \int_{x_a}^{x_b} f\delta u_0 + \int_{x_a}^{x_b} Q_i^e\,\delta\Delta_i^e \quad \dots (3.6)$$

V^e is element volume.
$q(x)$ is distributed transverse loading.
$f(x)$ is distributed axial load.
Q_i^e is generalized nodal force.
$\delta\Delta_i^e$ are virtual generalized nodal displacements.

CHAPTER 4

Euler-Bernoulli Beam Theory

Before plunging further into finite element analysis, we'll now define our problem. A 2 dimensional problem is considered in which nonlinear bending of straight beams is analyzed. Euler-Bernoulli beams are analysed. Assumptions in Euler-Bernoulli beam theory are [1]:
The plane sections perpendicular to the beam axis before deformation remain

- Plane
- Rigid (undeformed)
- Rotate

such that they remain perpendicular to the deformed beam axis. These assumptions imply that transverse strains and Poisson effects are neglected.

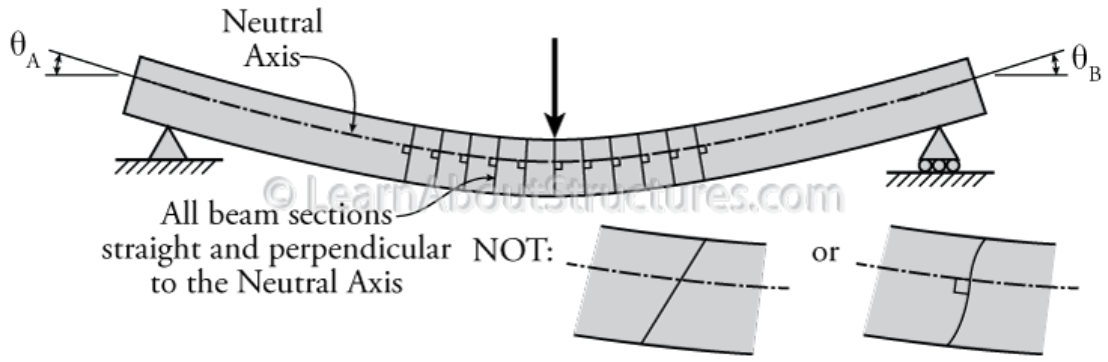


Fig. 4.1 Assumptions in Euler-Bernoulli Beams (learnaboutstructures.com)

Displacement field (in plane Euler-Bernoulli beams) for large rotations with small strains is,

$$u_1 = u_0(x) - z\frac{dw_0}{dx} \quad \ldots (4.1)$$

$$u_2 = 0 \quad \ldots (4.2)$$

$$u_3 = w_0(x) \quad \ldots (4.3)$$

Where, $(u_1,\ u_2,\ u_3)$ are total displacements along the coordinate directions $(x,\ y,\ z)$ and u_0 and w_0 denote the axial and transverse displacements of a point on the neutral axis.

CHAPTER 5

Linear Bending of Straight Beams

5.1 Assumptions

1. Euler-Bernoulli Beams
2. Isotropic material

In case of small transverse bending in straight beams (Euler-Bernoulli Beams), displacements and strains are given b,

$$u_1 = u - z\frac{dw}{dx}, \qquad u_2 = 0, \qquad u_3 = w(x)$$

$$\varepsilon_{xx} = \frac{\partial u_1}{\partial x} = \frac{du}{dx} - z\frac{d^2w}{dx^2} \qquad \ldots (5.1)$$

All other strain components are zero (transverse strains are neglected).u and w are axial and transverse displacements of mean axis.

5.2 Finite Element Modeling

In finite element whole domain is divided into finite sub-domains called finite elements (Here beam is divided into smaller segments. The transverse and axial displacement of mean axis of a given segment can be approximated using Hermite cubic interpolation and Lagrange linear interpolation functions.

$$\mathrm{w}^{e} \approx \sum_{i=1}^{4} \emptyset_i^e w_i^e, \qquad u^e \approx \sum_{i=1}^{2} \varphi_i^e u_i^e \qquad \ldots (5.2)$$

$$w^e = \begin{Bmatrix} w^e \\ \theta^e \\ w^{e+1} \\ \theta^{e+1} \end{Bmatrix}, \qquad u^e = \begin{Bmatrix} u^e \\ u^{e+1} \end{Bmatrix}$$

$$\emptyset^e = transpose\begin{pmatrix} 2t^3 - 3t^2 + 1 \\ (t^3 - 2t^2 + t)(x_{e+1} - x_e) \\ -2t^3 + 3t^2 \\ (t^3 - t^2)(x_{e+1} - x_e) \end{pmatrix}^e \qquad where\ t = \frac{x - x_e}{x_{e+1} - x_e}$$

$$\varphi^e = transpose\begin{pmatrix} \dfrac{x - x_{e+1}}{x_e - x_{e+1}} \\ \dfrac{x - x_e}{x_{e+1} - x_e} \end{pmatrix}$$

$$\varepsilon_{xx} = \sum_{i=1}^{2} \frac{d\varphi_i}{dx} u_i - z \sum_{J=1}^{4} \frac{d^2 \emptyset_J}{dx^2} w_J \qquad \ldots (5.3)$$

$$\delta\varepsilon_{xx} = \sum_{i=1}^{2} \frac{d\varphi_i}{dx} \delta u_i - z \sum_{J=1}^{4} \frac{d^2 \emptyset_J}{dx^2} \delta w_J \qquad \ldots (5.4)$$

$$\sigma_{xx} = E\varepsilon_{xx} \qquad \ldots (5.5)$$

$$\delta E_{int}^e = \int_{V^e} \sigma_{xx} \delta\varepsilon_{xx} dV \qquad \ldots (5.6)$$

$$\begin{aligned} \delta E_{int}^e = \int_{V^e} \Bigg(& E \sum_{i=1}^{2} \frac{d\varphi_i}{dx} u_i \sum_{j=1}^{2} \frac{d\varphi_j}{dx} \delta u_j - zE \sum_{i=1}^{2} \frac{d\varphi_i}{dx} u_i \sum_{J=1}^{4} \frac{d^2 \emptyset_J}{dx^2} \delta w_J \\ & - zE \sum_{J=1}^{4} \frac{d^2 \emptyset_J}{dx^2} w_J \sum_{i=1}^{2} \frac{d\varphi_i}{dx} \delta u_i + z^2 E \sum_{I=1}^{4} \frac{d^2 \emptyset_I}{dx^2} w_I \sum_{J=1}^{4} \frac{d^2 \emptyset_J}{dx^2} \delta w_J \Bigg) dV \\ = \int_{x_e}^{x_{e+1}} \Bigg(& A_{xx} \sum_{i=1}^{2} \frac{d\varphi_i}{dx} u_i \sum_{j=1}^{2} \frac{d\varphi_j}{dx} \delta u_j - B_{xx} \sum_{i=1}^{2} \frac{d\varphi_i}{dx} u_i \sum_{J=1}^{4} \frac{d^2 \emptyset_J}{dx^2} \delta w_J \\ & - B_{xx} \sum_{J=1}^{4} \frac{d^2 \emptyset_J}{dx^2} w_J \sum_{i=1}^{2} \frac{d\varphi_i}{dx} \delta u_i + D_{xx} \sum_{I=1}^{4} \frac{d^2 \emptyset_I}{dx^2} w_I \sum_{J=1}^{4} \frac{d^2 \emptyset_J}{dx^2} \delta w_J \Bigg) dV \\ = \int_{x_e}^{x_{e+1}} \Bigg(& A_{xx} \sum_{i=1}^{2} \frac{d\varphi_i}{dx} u_i \sum_{j=1}^{2} \frac{d\varphi_j}{dx} \delta u_j - B_{xx} \sum_{i=1}^{2} \frac{d\varphi_i}{dx} u_i \sum_{J=1}^{4} \frac{d^2 \emptyset_J}{dx^2} \delta w_J \\ & - B_{xx} \sum_{J=1}^{4} \frac{d^2 \emptyset_J}{dx^2} w_J \sum_{i=1}^{2} \frac{d\varphi_i}{dx} \delta u_i + D_{xx} \sum_{I=1}^{4} \frac{d^2 \emptyset_I}{dx^2} w_I \sum_{J=1}^{4} \frac{d^2 \emptyset_J}{dx^2} \delta w_J \Bigg) dx \end{aligned}$$

$$A_{xx} = \int_{A^e} E dA$$

$$B_{xx} = \int_{A^e} E z dA \qquad first\ moment\ of\ area$$

$$D_{xx} = \int_{A^e} Ez^2 dA \qquad second\ moment\ of\ area$$

As first moment of area about mean line is zero,

$$B_{xx} = 0$$

Hence,

$$\delta E^e_{int} = \int_{x_e}^{x_{e+1}} \left(A_{xx} \sum_{i=1}^{2} \frac{d\varphi_i}{dx} u_i \sum_{j=1}^{2} \frac{d\varphi_j}{dx} \delta u_j + D_{xx} \sum_{I=1}^{4} \frac{d^2 \emptyset_I}{dx^2} w_I \sum_{J=1}^{4} \frac{d^2 \emptyset_J}{dx^2} \delta w_J \right) dx \qquad \ldots (5.7)$$

$$\delta E^e_{ext,f} = \sum_{i}^{2} \int_{x_a}^{x_b} f_i \delta u_i \varphi_i dx \qquad \ldots (5.8)$$

$$\delta E^e_{ext,q} = \sum_{i}^{4} \int_{x_a}^{x_b} q_i \delta w_i \emptyset_i dx \qquad \ldots (5.9)$$

$$\delta E^e_{ext,Q} = \sum_{I}^{4} Q_I \delta w_I + \sum_{i}^{2} Q_i \delta u_i \qquad \ldots (5.10)$$

$$(5.7) - (5.8) - (5.9) - (5.10) = 0$$

Collecting terms with common virtual displacements and equating to zeros gives 2 decoupled systems of linear algebraic equations -one system for transverse and moment components and second for axial components. As shown below,

$$\sum_{J}^{4} K(\emptyset)^e_{IJ} w^e_J - \int_{x_a}^{x_b} f_I \emptyset_I dx - Q_I = 0 \qquad \ldots (5.11)$$

$$K(\emptyset)^e_{IJ} = D_{xx} \int_{x_a}^{x_b} \frac{d^2 \emptyset_I}{dx^2} \frac{d^2 \emptyset_J}{dx^2} dx$$

$$\sum_{j}^{2} K(\varphi)^e_{ij} u^e_j - \int_{x_a}^{x_b} q_i \varphi_i dx - Q_i = 0 \qquad \ldots (5.12)$$

$$K(\varphi)^e_{ij} = A_{xx} \int_{x_a}^{x_b} \frac{d\varphi_i}{dx} \frac{d\varphi_j}{dx} dx$$

$K(\varphi)^e and\ K(\emptyset)^e$ are element stiffness matrices which are combined into global stiffness matrices $K(\varphi)\ and\ K(\emptyset)$. In overall sense,

$$K(\varphi)w - f - Q_w = 0 \quad and \quad K(\emptyset)u - q - Q_u = 0 \qquad \ldots (5.13)$$

Where,

$$w = \begin{Bmatrix} w_1 \\ \theta_1 \\ . \\ . \\ w_{n+1} \\ \theta_{n+1} \end{Bmatrix} \qquad overall\ transverse\ and\ angular\ displacement\ vector\ of\ all\ nodes$$

$$u = \begin{Bmatrix} u_1 \\ u_2 \\ . \\ . \\ . \\ u_{n+1} \end{Bmatrix} \qquad overall\ axial\ displacement\ vector\ of\ all\ nodes$$

$$f = \begin{Bmatrix} f_1 \\ m_1 \\ . \\ . \\ f_{n+1} \\ m_{n+1} \end{Bmatrix} \qquad nodal\ equivalent\ of\ transverse\ force\ and\ moment\ acting\ on\ elements$$

$$q = \begin{Bmatrix} q_1 \\ q_2 \\ . \\ . \\ . \\ q_{n+1} \end{Bmatrix} \qquad nodal\ equivalent\ of\ axial\ forces\ acting\ on\ elements$$

$$Q_w = \begin{Bmatrix} Q_{w1} \\ Q_{\theta 1} \\ . \\ . \\ Q_{w|n+1} \\ Q_{\theta|n+1} \end{Bmatrix} \qquad transverse\ and\ moment\ reaction\ forces\ of\ elements\ on\ nodes$$

$$Q_u = \begin{Bmatrix} Q_{u1} \\ Q_{u2} \\ . \\ . \\ . \\ Q_{u|n+1} \end{Bmatrix} \qquad axial\ reaction\ forces\ of\ elements\ on\ nodes$$

Where, n is the number of elements.

5.3 Boundary Conditions

The fixed supports are described by setting displacements of nodes at supports equal to zero. Then from the overall equations the equations corresponding to fixed nodes are removed (to remove the triviality).
Then the equations are solved to obtain the unknown nodal displacements,

$$w = K(\varphi)^{-1}(f + Q_w) \quad and \quad u = K(\emptyset)^{-1}(q + Q_u) \qquad \ldots (5.14)$$

after removing the equations prescribing boundary conditions.

CHAPTER 6

Nonlinear Bending of Straight Beams

6.1 Strain-Displacement relations

From the nonlinear strain-displacement relations (Green strain tensor), it gives,

$$\varepsilon_{11} = \varepsilon_{xx} = \frac{du_0}{dx} - z\frac{d^2w_0}{dx^2} + \frac{1}{2}\left(\frac{dw_0}{dx}\right)^2$$

$$= \left[\frac{du_0}{dx} + \frac{1}{2}\left(\frac{dw_0}{dx}\right)^2\right] - z\frac{d^2w_0}{dw^2}$$

$$= \varepsilon_{xx}^0 + z\varepsilon_{xx}^1$$

$$\varepsilon_{xx}^0 = \frac{du_0}{dx} + \frac{1}{2}\left(\frac{dw_0}{dx}\right)^2 \quad ; \qquad \varepsilon_{xx}^1 = -\frac{d^2w_0}{dw^2} \qquad \ldots (6.1)$$

And all other strain components are zero.

6.2 Finite Element Modeling

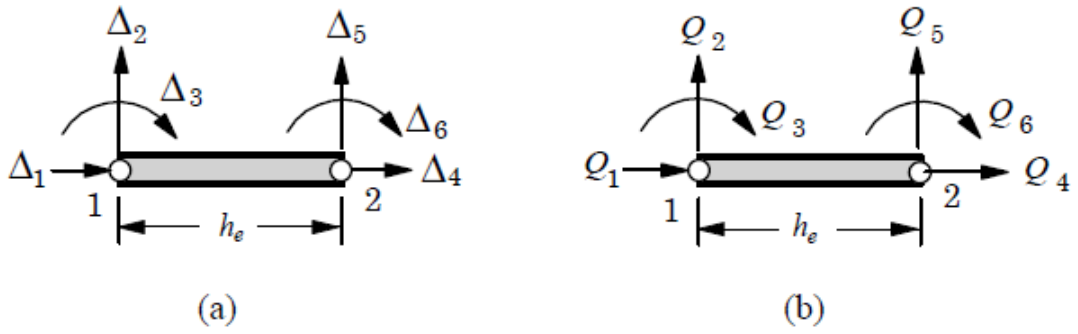


Fig. 6.1 Finite element displacement (a) and reaction (b) notations

$$\Delta_1^e = u_0(x_a), \quad \Delta_2^e = w_0(x_a), \quad \Delta_3^e = \left(-\frac{dw_0}{dx}\right)_{x_a} = \theta(x_a)$$

$$\Delta_4^e = u_0(x_b), \quad \Delta_5^e = w_0(x_b), \quad \Delta_6^e = \left(-\frac{dw_0}{dx}\right)_{x_b} = \theta(x_b)$$

$$Q_1^e = -N_{xx}(x_a), \quad Q_2^e = -\left(\frac{dw_0}{dx}N_{xx} + \frac{dM_{xx}}{dx}\right)_{x_a}$$

$$Q_4^e = N_{xx}(x_b), \quad Q_5^e = \left(\frac{dw_0}{dx} N_{xx} + \frac{dM_{xx}}{dx}\right)_{x_b}$$

$$Q_3^e = -M_{xx}(x_a), \quad Q_6^e = M_{xx}(x_b)$$

Following the same procedure as done in the case of linear bending we obtain following equations by considering nonlinear strain displacement relations (6.1).

$$\int_{x_a}^{x_b} A_{xx} \frac{d\delta u_0}{dx} \left\{\frac{du}{dx} + \frac{1}{2}\left(\frac{dw_0}{dx}\right)^2\right\} dx - \int_{x_a}^{x_b} f(x)\delta u_0 dx - Q_1 \delta u_0(x_a) - Q_4 \delta u_0(x_b) = 0 \quad \ldots (6.2)$$

$$\int_{x_a}^{x_b} \left[A_{xx} \frac{d\delta w_0}{dx} \frac{dw_0}{dx} \left\{\frac{du_0}{dx} + \frac{1}{2}\left(\frac{dw_0}{dx}\right)^2\right\} + D_{xx} \frac{d\delta^2 w_0}{dx^2} \frac{d^2 w_0}{dx^2}\right] dx - \int_{x_a}^{x_b} q\delta w_0 dx - Q_2 \delta w_0(x_a) - Q_3 \delta\theta(x_a) - Q_5 \delta w_0(x_b) - Q_6 \delta\theta(x_b) = 0 \quad \ldots (6.3)$$

$$A_{xx} = \int_A E dA \; D_{xx} = \int_A z^2 dA$$

Here x_a and x_b are global coordinates of the endpoints of the element. And A is the area of C/S. Taking Linear Lagrange and Hermite polynomial as approximation to unknowns $u(x)$ and $w(x)$,

$$w_0(x) \approx \sum_{j=1}^{4} \Delta_j' \emptyset_j(x), \qquad u_0(x) \approx \sum_{i=1}^{2} u_i \psi_i(x) \quad \ldots (6.4)$$

$$\Delta_1' = w_0(x_a) \Delta_2' = \theta(x_a) \Delta_3' = w_0(x_b) \Delta_4' = \theta(x_b)$$

$\psi_i(x)$ are linear Lagrange interpolation functions and $\emptyset_j(x)$ are Hermite cubic interpolation functions.

Substituting above approximations in (6.2) and (6.3) and $\delta u_0(x) = \sum_{i=1}^{2} \delta u_i \psi_i(x)$ and $\delta w_0(x) = \sum_{j=1}^{4} \delta \Delta_j' \emptyset_j(x)$, weight functions, we obtain following set of system of nonlinear algebraic equations,

$$\sum_{j=1}^{2} K_{ij}^{11} u_j + \sum_{J=1}^{4} K_{iJ}^{12} \Delta_J' - F_i' = 0 \qquad i = 1, 2 \quad \ldots (6.5)$$

$$\sum_{j=1}^{2} K_{Ij}^{21} u_j + \sum_{J=1}^{4} K_{IJ}^{22} \Delta_J' - F_I' = 0 \qquad I = 1, 2, 3, 4 \quad \ldots (6.6)$$

Where stiffness matrices K are obtained as shown below,

$$K_{ij}^{11} = \int_{x_a}^{x_b} A_{xx} \frac{d\psi_i}{dx} \frac{d\psi_j}{dx} dx \; K_{ij}^{12} = \frac{1}{2} \int_{x_a}^{x_b} A_{xx} \frac{dw_0}{dx} \frac{d\psi_i}{dx} \frac{d\emptyset_J}{dx} dx$$

$$K_{Ij}^{21} = \int_{x_a}^{x_b} A_{xx} \frac{dw_0}{dx} \frac{d\psi_j}{dx} \frac{d\emptyset_I}{dx} dx \; K_{Ij}^{21} = 2K_{jI}^{12}$$

$$K_{IJ}^{22} = \int_{x_a}^{x_b} D_{xx} \frac{d^2\emptyset_I}{dx^2} \frac{d^2\emptyset_J}{dx^2} dx + \frac{1}{2} \int_{x_a}^{x_b} A_{xx} \left(\frac{dw_0}{dx}\right)^2 \frac{d\emptyset_I}{dx} \frac{d\emptyset_J}{dx} dx \qquad \ldots (6.7)$$

$$F_i^{'} = \int_{x_a}^{x_b} f\psi_i dx + \widehat{Q_i} F_I^2 = \int_{x_a}^{x_b} f\emptyset_I dx + \overline{Q_I}$$

For $i, j = 1, 2$ and $I, J = 1, 2, 3, 4$, where $\widehat{Q_1} = Q_1$, $\widehat{Q_2} = Q_4$, $\overline{Q_1} = Q_2$, $\overline{Q_2} = Q_3$, $\overline{Q_3} = Q_5$, $\overline{Q_4} = Q_6$.

Above equations are obtained after following a linearization procedure. The stiffness matrices are function ofw_0. Hence, the sets of algebraic equations (6.5) and (6.6) are nonlinear.

In matrix form equations can be written as,

$$\begin{bmatrix} [K^{11}] & [K^{12}] \\ [K^{21}] & [K^{22}] \end{bmatrix} \begin{Bmatrix} \{\Delta^1\} \\ \{\Delta^2\} \end{Bmatrix} = \begin{Bmatrix} \{F^1\} \\ \{F^2\} \end{Bmatrix}$$

$$\Delta_i^1 = u_i \Delta_J^1 = \Delta_J^{'}$$

Iterative methods are used for the solution of such nonlinear equations. One such method is Direct iteration and Newton-Raphson method. Newton-Raphson method is employed as it shows better convergence.

6.3 Newton-Raphson procedure

$$[T(\{\Delta\}^{r-1})]\{\delta\Delta\}^r = -\{R(\{\Delta\}^{r-1})\} = \{F\} - ([K]\{\Delta\})^{r-1} \qquad \ldots (6.8)$$

Where r denote the iteration index.$T(\{\Delta\}^{r-1})$ is the tangent matrix which is given by,

$$[T] \equiv \left(\frac{\partial\{R\}}{\partial\{\Delta\}}\right)^{r-1}, or \quad T_{ij}^e \equiv \left(\frac{\partial R_i^e}{\partial \Delta_j^e}\right)^{r-1} \qquad \ldots (6.9)$$

Here e denotes the element number. The solution at rth iteration is given by,

$$\{\Delta\}^r = \{\Delta\}^{r-1} + \{\delta\Delta\} \qquad \ldots (6.10)$$

The components of tangent matrix are,

$$[T^{11}] = [K^{11}][T^{21}] = [K^{21}]$$

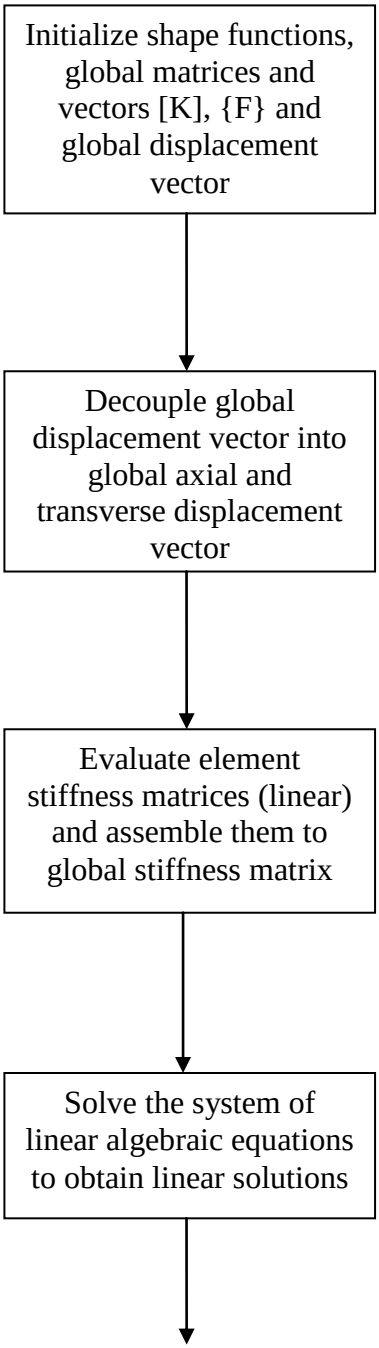
$$T_{iJ}^{12} = K_{Ji}^{21}$$

$$T_{IJ}^{22} = K_{IJ}^{22} + \int_{x_a}^{x_b} A_{xx} \left(\frac{du_0}{dx} + \frac{dw_0}{dx} \frac{dw_0}{dx} \right) \frac{d\emptyset_I}{dx} \frac{d\emptyset_J}{dx} dx \qquad \ldots (6.11)$$

CHAPTER 7

Implementation in MATLAB

7.1 Flowchart



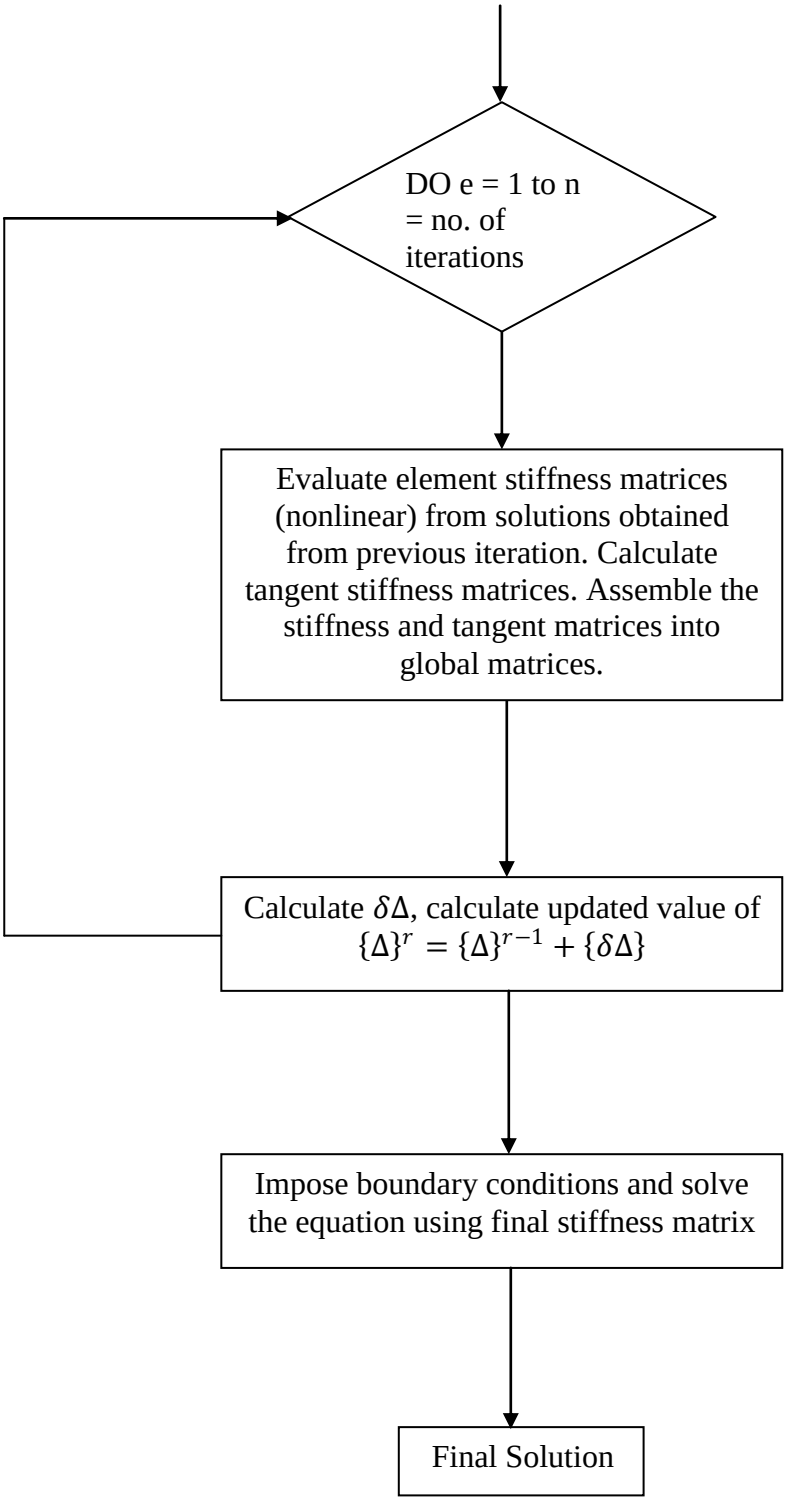


Fig. 7.1 Flow of MATLAB Program

7.2 MATLAB Code

```
n=0;
syms x;
syms qu;
syms qw;
syms qm;

prompt="Length = ";
l=input(prompt);
prompt="Height of CS = ";
bb=input(prompt);
prompt="Width of CS = ";
hb=input(prompt);
Izz=(hb*bb^3)/12;

Axx=hb*bb;
Bxx=0;
Dxx=(hb*bb^3)/12;

prompt="Mesh (Size Description=1, Nodes Description=2) = ";
ansc=input(prompt);
if ansc==1
    prompt="Mesh Size = ";
    h=input(prompt);
    n=cast((l/h),'uint32');
    xa=linspace(0, l, n+1);
    d=zeros(3*(n+1), 1);
    dR=zeros(3*(n+1), 1);
    Q=sym(zeros(3*(n+1), 1));
    u=zeros(n+1, 1);
    w=zeros(2*(n+1), 1);
end
if ansc==2
    prompt="Nodes Array = ";
    xa=input(prompt);
    sz=size(xa);
    n=sz(2);
    dR=zeros(3*(n+1), 1);
    Q=sym(zeros(3*(n+1), 1));
    u=zeros(n+1, 1);
    w=zeros(2*(n+1), 1);
end
prompt="Material properties (Constant=1, Different for different elements=2) =
";
asnc=input(prompt);
if ansc==1
    prompt="Elastic Constant = ";
    E=input(prompt)*ones(n, 1);
end
if ansc==2
    promp="Elastic Constants array = ";
    E=input(prompt);
end

%Shape Functions -------------------------
```

```
L=sym(zeros(n, 4));
for e=1:n
    h=xa(e+1)-xa(e);
    L(e, 1)=2*((x-xa(e))/h)^3-3*((x-xa(e))/h)^2+1;
    L(e, 2)=(((x-xa(e))/h)^3-2*((x-xa(e))/h)^2+((x-xa(e))/h))*(xa(e+1)-xa(e));
    L(e, 3)=-2*((x-xa(e))/h)^3+3*((x-xa(e))/h)^2;
    L(e, 4)=(((x-xa(e))/h)^3-((x-xa(e))/h)^2)*(xa(e+1)-xa(e));
end
Lu=sym(zeros(n, 2));
for e=1:n
    h=xa(e+1)-xa(e);
    Lu(e, 1)=(x-xa(e+1))/(-h);
    Lu(e, 2)=(x-xa(e))/h;
end

d=zeros(3*(n+1), 1);

prompt="No. of fixtures = ";
f_n=input(prompt);
fix=zeros(3*(n+1), 1);
fix_u=zeros(n+1, 1);
fix_w=zeros(2*(n+1), 1);
fix_t=zeros(n+1, 1);
fix_w_d=0;
fix_u_d=0;
fix_d=0;
for i=1:f_n
    fix_p=zeros(3*(n+1), 1);
    reactions=zeros(3*(n+1), 1);
    ansc = input(['Type of fixture # ' i ' (pin=1, hinged=2, roller=3) = ']);
if ansc==1
        n_n=input(['Node no. of fixture # ' i ' = ']);
        fix_t=1;
        d(3*n_n-2)=0;
        d(3*n_n-1)=0;
        d(3*n_n)=0;
        Q(3*n_n-2)=qw;
        Q(3*n_n-1)=qm;
        Q(3*n_n)=qu;
        fix(3*n_n-2)=1;
        fix(3*n_n-1)=1;
        fix(3*n_n)=1;
        fix_u(n_n)=1;
        fix_w(2*n_n-1)=1;
        fix_w(2*n_n)=1;
        fix_u_d=[fix_u_d, n_n];
        fix_w_d=[fix_w_d, 2*n_n-1, 2*n_n];
        fix_d=[fix_d, 3*n_n-2, 3*n_n-1, 3*n_n];
end
if ansc==2
        n_n=input(['Node no. of fixture # ' i ' = ']);
        fix_t=2;
        d(3*n_n-2)=0;
        d(3*n_n)=0;
        Q(3*n_n-2)=qw;
        Q(3*n_n)=qu;
```

```
        fix(3*n_n-2)=1;
        fix(3*n_n)=1;
        fix_u(n_n)=1;
        fix_w(2*n_n-1)=1;
        fix_u_d=[fix_u_d, n_n];
        fix_w_d=[fix_w_d, 2*n_n-1];
        fix_d=[fix_d, 3*n_n-2, 3*n_n];
end
if ansc==3
        n_n=input(['Node no. of fixture # ' i ' = ']);
        fix_t=3;
        d(3*n_n-2)=0;
        Q(3*n_n-2)=qw;
        fix(3*n_n-2)=1;
        fix_w(2*n_n-1)=1;
        fix_w_d=[fix_w_d, 2*n_n-1];
        fix_d=[fix_d, 3*n_n-2];
end
end
fix_d(:, 1)=[];
fix_u_d(:, 1)=[];
fix_w_d(:, 1)=[];

l_n=input('No. of load points = ');
load_p=zeros(2*(n+1),1);
for i=1:l_n
    n_n=input(['Node no. of load # ' i ' = ']);
    l_value=input('Transverse Load value = ');
    Q(3*n_n-2)=l_value;
    l_value=input('Moment value = ');
    Q(3*n_n-1)=l_value;
    l_value=input('Axial Load value = ');
    Q(3*n_n)=l_value;
end

f=str2sym(input('Distributed Transverse Load function in x: ','s'));
Fw=sym(zeros(2*(n+1), 1));
bf=input('Force per unit volume (Body force in y-dir) = ');
bf_linear=bf/(hb*bb);
for e=1:n
    h=xa(e+1)-xa(e);
if f~=0
        fun=L(e, 1);
        Fw(2*e-1)=Fw(2*e-1)+int(f*fun, xa(e), xa(e+1));
        fun=L(e, 2);
        Fw(2*e)=Fw(2*e)+int(f*fun, xa(e), xa(e+1));
        fun=L(e, 3);
        Fw(2*e+1)=Fw(2*e+1)+int(f*fun, xa(e), xa(e+1));
        fun=L(e, 4);
        Fw(2*e+2)=Fw(2*e+2)+int(f*fun, xa(e), xa(e+1));
end
if bf_linear~=0
        fun=L(e, 1);
        Fw(2*e-1)=Fw(2*e-1)+int(bf_linear*fun, xa(e), xa(e+1));
        fun=L(e, 2);
        Fw(2*e)=Fw(2*e)+int(bf_linear*fun, xa(e), xa(e+1));
```

```
        fun=L(e, 3);
        Fw(2*e+1)=Fw(2*e+1)+int(bf_linear*fun, xa(e), xa(e+1));
        fun=L(e, 4);
        Fw(2*e+2)=Fw(2*e+2)+int(bf_linear*fun, xa(e), xa(e+1));
end
end

f=str2sym(input('Distributed Axial Load function in x: ','s'));
bf=input('Force per unit volume (Body force in x-dir) = ');
bf_linear=bf/(hb*bb);
Fu=sym(zeros(n+1, 1));
for e=1:n
    h=xa(e+1)-xa(e);
if f~=0
        fun=Lu(e, 1);
        Fu(e)=Fu(e)+int(f*fun, xa(e), xa(e+1));
        fun=Lu(e, 2);
        Fu(e+1)=Fu(e+1)+int(f*fun, xa(e), xa(e+1));
end
if bf_linear~=0
        fun=Lu(e, 1);
        Fu(e)=Fu(e)+int(bf_linear*fun, xa(e), xa(e+1));
        fun=Lu(e, 2);
        Fu(e+1)=Fu(e+1)+int(bf_linear*fun, xa(e), xa(e+1));
end
end

[K_G_U, K_G_W]=K_Calc(L, Lu, n, Axx, Dxx, xa, E);
K_G_U(fix_u_d, :)=[];
K_G_U(:, fix_u_d)=[];
K_G_W(fix_w_d, :)=[];
K_G_W(:, fix_w_d)=[];

del_d=0;
for i=1:n+1
    del_d=[del_d 3*i];
end
del_d(:, 1)=[];
Qw=Q;
Qw(del_d, :)=[];

del_d=0;
for i=1:n+1
    del_d=[del_d 3*i-2 3*i-1];
end
del_d(:, 1)=[];
Qu=Q;
Qu(del_d, :)=[];

Qu_M=Qu;
Qw_M=Qw;
Qu_M(fix_u_d, :)=[];
Qw_M(fix_w_d, :)=[];
Fu_M=Fu;
Fw_M=Fw;
Fu_M(fix_u_d, :)=[];
```

```
Fw_M(fix_w_d, :)=[];


U=inv(K_G_U)*(Qu_M+Fu_M);
W=inv(K_G_W)*(Qw_M+Fw_M);

j=1;
for i=1:(n+1)
if fix_u(i)==0
        u(i)=U(j);
        j=j+1;
end
end

j=1;
for i=1:2*(n+1)
if fix_w(i)==0
        w(i)=W(j);
        j=j+1;
end
end

for i=1:(n+1)
    d(3*i-2)=w(2*i-1);
    d(3*i-1)=w(2*i);
    d(3*i)=u(i);
end
d_linear=d;
d_pre=d;

F=sym(zeros(3*(n+1),1));
for i=1:(n+1)
    F(3*i-2)=Fw(2*i-1);
    F(3*i-1)=Fw(2*i);
    F(3*i)=Fu(i);
end

for i=1:(n+1)
    Q(3*i-2)=Qw(2*i-1);
    Q(3*i-1)=Qw(2*i);
    Q(3*i)=Qu(i);
end

i_n=input('No. of iterations = ');
i_n_a=zeros(i_n, 1);
x_i=linspace(1, i_n, 1);
for q=1:i_n
    [K_G_R, T_G_R]=K_Calc_Nonlinear(d, L, Lu, n, Axx, Dxx, xa, E);
    K_G_M=K_G_R;
    T_G_M=T_G_R;
    Q_M=Q;
    Q_M(fix_d, :)=[];
    F_M=F;
    F_M(fix_d, :)=[];
    dRM=sym(zeros(size(F_M)));
```

```
    K_G_M(fix_d, :)=[];
    K_G_M(:, fix_d)=[];
    T_G_M(fix_d, :)=[];
    T_G_M(:, fix_d)=[];
    d_M=inv(K_G_M)*(Q_M+F_M);
    R=F_M+Q_M-K_G_M*d_M;
    dRM=inv(T_G_M)*R;
    j=1;
for i=1:3*(n+1)
if fix(i)==0
            d(i)=double(d_M(j));
            j=j+1;
end
end
    j=1;
for i=1:3*(n+1)
if fix(i)==0
            dR(i)=double(dRM(j));
            j=j+1;
end
end
    prompt="Iteration";
    disp(prompt)
    i_n_a(q)=rms(d-d_pre);
    rms(d-d_pre)
    d_pre=d;
    d=d+dR;
end

for i=1:3*(n+1)
if fix(i)==1
        Q(i)=K_G_R(i, :)*vpa(subs(d))-F(i);
end
end

del_d=0;
for i=1:n+1
    del_d=[del_d 3*i];
end
del_d(:, 1)=[];
wp=d;
wp(del_d, :)=[];

del_d=0;
for i=1:n+1
    del_d=[del_d 3*i-2 3*i-1];
end
del_d(:, 1)=[];
up=d;
up(del_d, :)=[];

w0=sym(zeros(n+1));
w1=sym(zeros(n+1));
j=0;
for i=1:2:2*(n+1)
    j=j+1;
```

```
    w0(j)=w(i);
end
j=0;
for i=2:2:2*(n+1)
    j=j+1;
    w1(j)=w(i);
end
plotfun=sym(zeros(n, 1));
for i=1:n
    plotfun(i)=L(i, :)*vpa(wp(2*i-1:2*i-1+3));
end


%functions ---------------------------------
function [K_u, K_w]=K_Calc(L, Lu, n, Axx, Dxx, xa, E)

    Ku=zeros(n, 2, 2);
    Kw=zeros(n, 4, 4);

for e=1:n
for i=1:2
for j=1:2
                fun=E(e)*Axx*diff(Lu(e, i))*diff(Lu(e, j));
                Ku(e, i, j)=int(fun, xa(e), xa(e+1));
end
end
end
for e=1:n
for i=1:4
for j=1:4
                fun=E(e)*Dxx*diff(diff(L(e, i)))*diff(diff(L(e, j)));
                Kw(e, i, j)=int(fun, xa(e), xa(e+1));
end
end
end

    K_G_u=zeros(n+1, n+1);

for e=1:n
for i=1:2
for j=1:2
                K_G_u(e-1+i, e-1+j)=K_G_u(e-1+i, e-1+j)+Ku(e, i, j);
end
end
end

    K_G_w=zeros(2*(n+1), 2*(n+1));

for e=1:n
for i=1:4
for j=1:4
                K_G_w(2*e+i-2, 2*e+j-2)=K_G_w(2*e+i-2, 2*e+j-2)+Kw(e, i, j);
end
end
end
```

```
    K_u=K_G_u;
    K_w=K_G_w;

end

function [K_Global, T_Global]=K_Calc_Nonlinear(d, L, Lu, n, Axx, Dxx, xa, E)
    Kw1=zeros(n, 2, 4);
    Kw2=zeros(n, 4, 4);
    Ku1=zeros(n, 2, 2);
    Ku2=zeros(n, 4, 2);
    w_it=sym(zeros(n, 1));
    u_it=sym(zeros(n, 1));

    del_d=0;
for i=1:n+1
        del_d=[del_d 3*i];
end
    del_d(:, 1)=[];
    w=d;
    w(del_d, :)=[];
for e=1:n
        w_it(e)=L(e, :)*w((2*e-1):(2*(e+1)));
end

    del_d=0;
for i=1:n+1
        del_d=[del_d 3*i-2 3*i-1];
end
    del_d(:, 1)=[];
    u=d;
    u(del_d, :)=[];
for e=1:n
        u_it(e)=Lu(e, :)*transpose([u(e), u(e+1)]);
end

for e=1:n
for i=1:4
for j=1:2
                fun=E(e)*Axx*diff(w_it(e))*diff(L(e, i))*diff(Lu(e, j));
                Ku2(e, i, j)=int(fun, xa(e), xa(e+1));
end
end
end
for e=1:n
for i=1:2
for j=1:2
                fun=E(e)*Axx*diff(Lu(e, i))*diff(Lu(e, j));
                Ku1(e, i, j)=int(fun, xa(e), xa(e+1));
end
end
end
for e=1:n
for i=1:2
for j=1:4
                fun=E(e)*Axx*diff(w_it(e))*diff(Lu(e, i))*diff(L(e, j));
```

```
                Kw1(e, i, j)=(1/2)*int(fun, xa(e), xa(e+1));
end
end
end
for e=1:n
for i=1:4
for j=1:4
                fun1=E(e)*Dxx*diff(diff(L(e, i)))*diff(diff(L(e, j)));
                fun2=E(e)*Axx*((diff(w_it(e)))^2)*diff(L(e, i))*diff(L(e, i));
                Kw2(e, i, j)=int(fun1, xa(e), xa(e+1))+(1/2)*int(fun2, xa(e),
xa(e+1));
end
end
end

    K_G=zeros(3*(n+1), 3*(n+1));
for e=1:n
for i=1:2
for j=1:2
                K_G(3*e-3+3*i, 3*e-3+3*j)=K_G(3*e-3+3*i, 3*e-3+3*j)+Ku1(e, i,
j);
                K_G(3*e-3+3*i, 3*e-3+3*j-2)=K_G(3*e-3+3*i, 3*e-3+3*j-2)+Kw1(e,
i, 2*j-1);
                K_G(3*e-3+3*i, 3*e-3+3*j-1)=K_G(3*e-3+3*i, 3*e-3+3*j-1)+Kw1(e,
i, 2*j);
end
end
for i=1:2
for j=1:2
                K_G(3*e-3+i, 3*e-3+3*j)=K_G(3*e-3+i, 3*e-3+3*j)+Ku2(e, i, j);
                K_G(3*e-3+i, 3*e-3+3*j-2)=K_G(3*e-3+i, 3*e-3+3*j-2)+Kw2(e, i,
2*j-1);
                K_G(3*e-3+i, 3*e-3+3*j-1)=K_G(3*e-3+i, 3*e-3+3*j-1)+Kw2(e, i,
2*j);
end
end
for i=4:5
for j=1:2
                K_G(3*e-3+i, 3*e-3+3*j)=K_G(3*e-3+i, 3*e-3+3*j)+Ku2(e, i-1,
j);
                K_G(3*e-3+i, 3*e-3+3*j-2)=K_G(3*e-3+i, 3*e-3+3*j-2)+Kw2(e, i-
1, 2*j-1);
                K_G(3*e-3+i, 3*e-3+3*j-1)=K_G(3*e-3+i, 3*e-3+3*j-1)+Kw2(e, i-
1, 2*j);
end
end
end
    Tu1=Ku1;
    Tu2=Ku2;
    Tw1=2*Kw1;
    Tw2=zeros(n, 4, 4);
for e=1:n
for i=1:4
for j=1:4
                fun=Axx*(diff(u_it(e))+(diff(w_it(e)))^2)*diff(L(e,
i))*diff(L(e, j));
                Tw2(e, i, j)=Kw2(e, i, j)+int(fun, xa(e), xa(e+1));
```

```
end
end
end
    T_G=zeros(3*(n+1), 3*(n+1));
for e=1:n
for i=1:2
for j=1:2
                T_G(3*e-3+3*i, 3*e-3+3*j)=T_G(3*e-3+3*i, 3*e-3+3*j)+Tu1(e, i,
j);
                T_G(3*e-3+3*i, 3*e-3+3*j-2)=T_G(3*e-3+3*i, 3*e-3+3*j-2)+Tw1(e,
i, 2*j-1);
                T_G(3*e-3+3*i, 3*e-3+3*j-1)=T_G(3*e-3+3*i, 3*e-3+3*j-1)+Tw1(e,
i, 2*j);
end
end
for i=1:2
for j=1:2
                T_G(3*e-3+i, 3*e-3+3*j)=T_G(3*e-3+i, 3*e-3+3*j)+Tu2(e, i, j);
                T_G(3*e-3+i, 3*e-3+3*j-2)=T_G(3*e-3+i, 3*e-3+3*j-2)+Tw2(e, i,
2*j-1);
                T_G(3*e-3+i, 3*e-3+3*j-1)=T_G(3*e-3+i, 3*e-3+3*j-1)+Tw2(e, i,
2*j);
end
end
for i=4:5
for j=1:2
                T_G(3*e-3+i, 3*e-3+3*j)=T_G(3*e-3+i, 3*e-3+3*j)+Tu2(e, i-1,
j);
                T_G(3*e-3+i, 3*e-3+3*j-2)=T_G(3*e-3+i, 3*e-3+3*j-2)+Tw2(e, i-
1, 2*j-1);
                T_G(3*e-3+i, 3*e-3+3*j-1)=T_G(3*e-3+i, 3*e-3+3*j-1)+Tw2(e, i-
1, 2*j);
end
end
end
    T_Global=T_G;
    K_Global=K_G;
end
```

CHAPTER 8

Analysis of the Model

8.1 Defining the problem

After implementing the procedure in MATLAB it is tested for a test model as shown below. Results are then compared with linear approximation and the deviations are observed from nonlinear case.

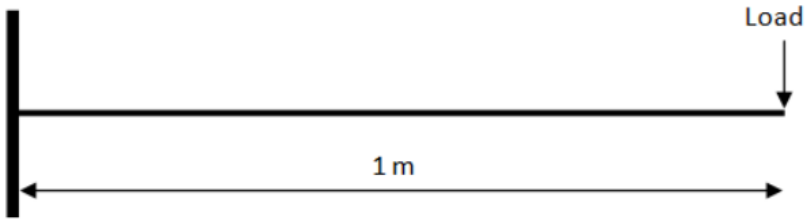


Fig. 8.1 Test model diagram (C/S: 0.05*0.05 m^2)

8.2 Results

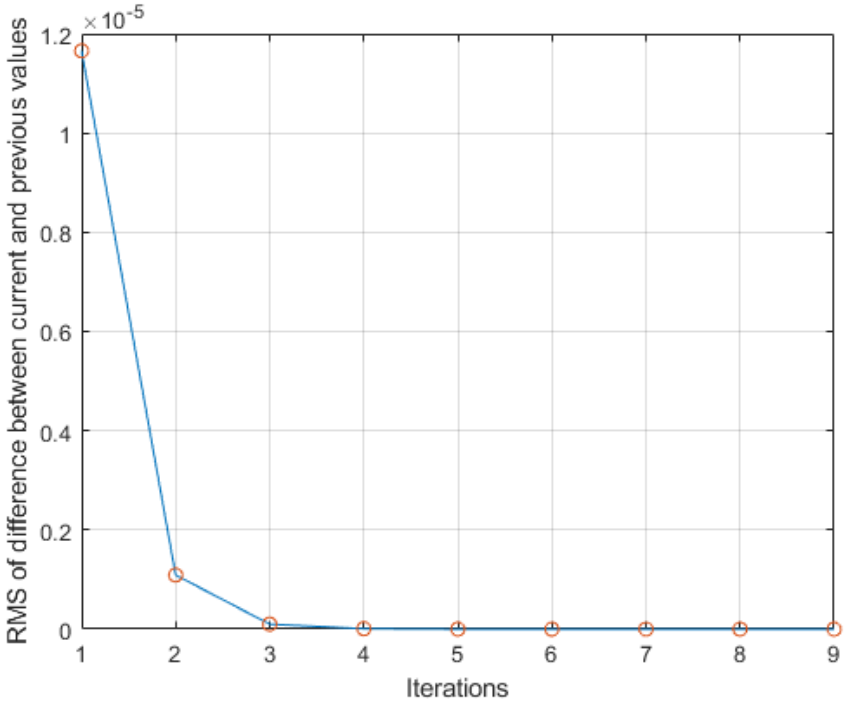


Fig. 8.2 Convergence (No. of elements = 10) (Load = 100 N)

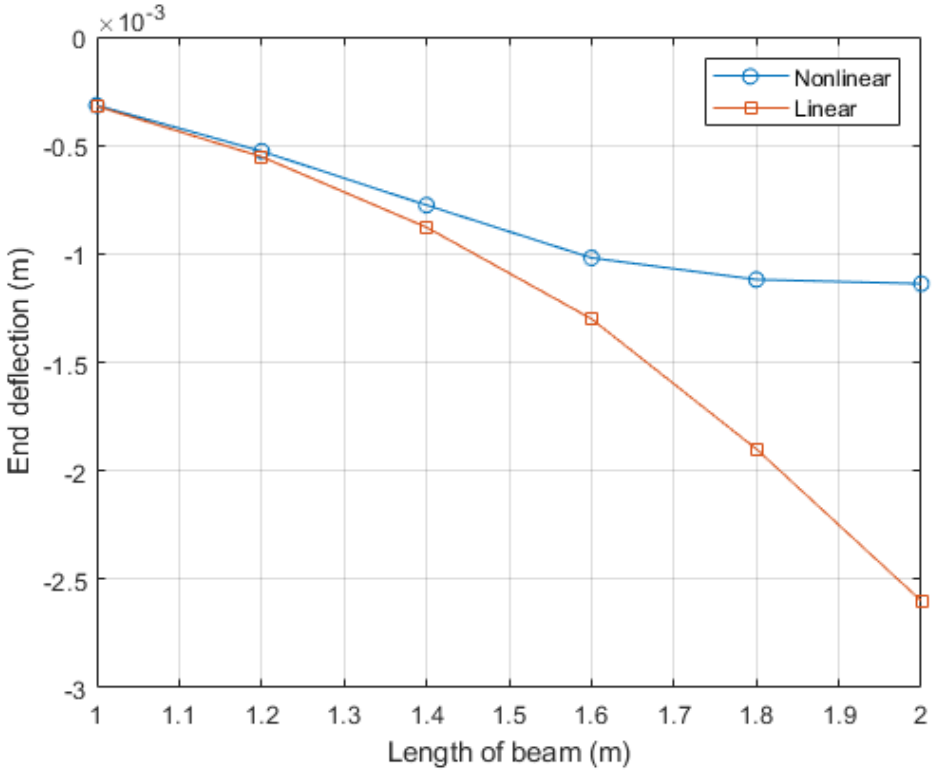


Fig. 8.3 Comparison of Nonlinear and Linear deflection with length of beam (Load = 100 N)

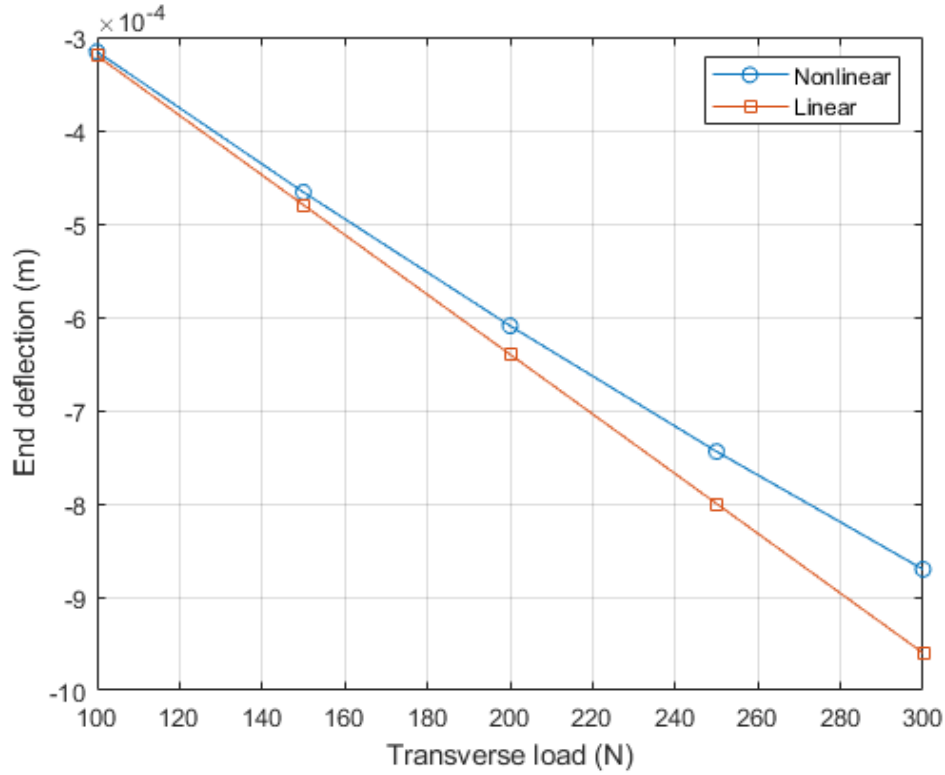


Fig. 8.4 Comparison of Nonlinear and Linear deflection with load (Length = 1 m)

8.2 Conclusion

In this project a method called finite element analysis was used to solve for nonlinear bending of Euler-Bernoulli Beams. The results were compared with linear results. It was found that for higher loads nonlinear results showed lower deflection. This implies that in nonlinear case the stiffness matrices have higher values than in case of linear approximation. Also with increasing beam length nonlinear beams showed deviation from linear case in that deflection was lower in nonlinear case. Hence, it can be inferred that for lower values of load and deflection linear results can be used for practical purposes but when load and deflection increases above some limit nonlinearity must be considered,.

References

[1] J. N. Reddy, An Introduction to Nonlinear Finite Element Analysis, 2004
[2] Ted Belytschko, Wing Kam Liu, Brian Moran, Khalil Elkhodary, Nonlinear Finite Elements for Continua and Structures [2 ed.], 2004
[3] O. C. Zienkiewicz, R. L. Taylor, The finite element method [Volume 1], 2000