# Numerical Simulation of Beam Deflection using Finite Difference Methods in Python


**School of Mechanical Sciences**

**Indian Institute of Technology Goa**

# Abstract

This project focuses on the numerical simulation of beam deflection for one-dimensional structural elements using Python. The primary goal was to develop an efficient and accurate computational solver based on the finite difference method (FDM) to analyse deflection behaviour under varying boundary conditions and external loading scenarios. The implementation specifically targets cantilever and simply-supported beam configurations.

To handle non-linearities arising in the beam deflection equations, the Newton-Raphson iterative method was employed. This enabled the solution of nonlinear systems resulting from discretized governing equations, particularly in cases where the linear approximation is insufficient. The numerical model was designed to accommodate both point and uniformly distributed loads, offering a flexible and generalizable approach.

Simulated results were rigorously validated against classical analytical solutions to ensure correctness and minimise numerical error. Particular attention was given to discretization schemes, boundary condition implementation, and convergence behaviour of the iterative solver to maintain computational stability and precision. Data visualization and post-processing were carried out using Matplotlib and Pandas and Data visualisations were created using Plotly, allowing clear representation and analysis of deflection profiles.

Overall, the project demonstrates the effectiveness of combining finite difference methods with iterative solvers like Newton-Raphson for structural analysis and offers a robust framework for the computational exploration of beam behavior under static loading and also compare the behaviour from Linear to nonlinear for the beam.

# Table of Contents


1. **Introduction**
    - 1.1 Background and Motivation
    - 1.2 Objective of the Project
    - 1.3 Overview of the Approach
2. **Numerical Techniques**
    - 2.1 Finite Difference Method (FDM)
        - 2.1.1 Concept and Application in Beam Theory
    - 2.2 Discretization Techniques
        - 2.2.1 Central Difference Approximation
        - 2.2.2 Forward Difference Approximation
        - 2.2.3 Backward Difference Approximation
3. **Initial Approximations and Differential Methods**
    - 3.1 Initial Guess Strategies
    - 3.2 Approaches for Solving Beam Differential Equations
        - 3.2.1 Direct Method
        - 3.2.2 Finite Difference Method - First Order
        - 3.2.3 Finite Difference Method - Second Order
4. **Nonlinear Solution Strategy: Newton-Raphson Method**
    - 4.1 Overview of Newton-Raphson Method
    - 4.2 Justification for Using Newton-Raphson
    - 4.3 Algorithm and Iterative Convergence
5. **Case Study: Cantilever Beam**
    - 5.1 Problem Diagrams for Point Load and Uniform Load
    - 5.2 Boundary Conditions and Their Application
    - 5.3 Code Implementation Overview
        - 5.3.1 Point Load
        - 5.3.2 Point Load – Error Table for Selected EI Values
        - 5.3.3 Comparison: Point Load vs. Distributed Load
        - 5.3.4 Implementation for Multiple Loads (N Cases)
6. **Case Study: Simply-Supported Beam**
    - 6.1 Problem Diagrams for Point Load and Uniform Load
    - 6.2 Boundary Conditions and Their Application
    - 6.3 Code Implementation Overview
        - 6.3.1 Point Load
        - 6.3.2 Comparison: Point Load vs. Distributed Load
        - 6.3.3 Implementation for Multiple Loads (N Cases)
7. **Results and Discussion**
    - 7.1 Cantilever Beam Results
    - 7.2 Simply-Supported Beam Results
    - 7.3 Comparison Between Cantilever Beam and Simply Supported Beam

8. **Conclusion**

9. **Future Work**
   9.1 Extension to Overhanging Beam
   9.2 Development of a GUI/Web Interface for Interactive Input and Visualization

10. **References**

# List of Figures

## 5.1 Cantilever Beam Configuration

**Figure 5.1.1:** Cantilever Beam with Point Load

**Figure 5.1.2:** Cantilever Beam under Uniform Continuous Load

---

## 6.1 Simply-Supported Beam Configuration

**Figure 6.1.1:** Simply-Supported Beam under Point Load

**Figure 6.1.2:** Simply-Supported Beam under Uniform Continuous Loading

---

## 7.1 Cantilever Beam Results

**Figure 7.1.1**: Code 1 – Newton-Raphson Solution for Point Load Case

(a) Deflection comparison: NR (Point Load) vs Euler (Point Load)

(b) Zoomed-in view of deflection comparison

**Figure 7.1.2**: Code 2 – Error Analysis for Point Load

(a) Table: Linear vs Nonlinear deflections at beam tip

(b) Plot: Relative Percentage Difference (%) vs Flexural Rigidity (EI)

(c) Plot: Absolute Difference at free end vs Flexural Rigidity (EI)

**Figure 7.1.3**: Code 3 – Comparison: Point Load vs Distributed Load

(a) Plot: NR and Euler (Point Load & Distributed Load)

(b) Zoomed-in view: NR vs Euler for Distributed Load

(c) Plot: NR (Point Load) vs NR (Distributed Load)

**Figure 7.1.4**: Code 4 – Multiple Load Case Implementation

---

## 7.2 Simply-Supported Beam Results

**Figure 7.2.1**: Code 1 – Mid-Span Point Load

(a) Deflection comparison: NR (Point Load) vs Euler (Point Load)

(b) Zoomed-in view of deflection comparison

**Figure 7.2.2**: Code 2 – Comparison: Point Load vs Distributed Load

(a) Plot: NR and Euler (Point Load & Distributed Load)

(b) Zoomed-in view: NR vs Euler for Distributed Load

(c) Plot: NR (Point Load) vs NR (Distributed Load)

**Figure 7.2.3**: Code 3 – Multiple Load Case Implementation

# Introduction

## 1.1 Background and Motivation

In structural engineering, understanding how beams deform under various loads is a fundamental requirement. Beams are basic elements in bridges, buildings, and many mechanical systems, and analyzing their deflection helps ensure that the structure remains safe and functional. Traditionally, these problems are solved using analytical formulas derived from beam theory, but those are often limited to ideal cases with simple loading and boundary conditions.

With the advancement of computational tools, numerical methods have become an effective way to solve complex structural problems where analytical solutions are either difficult or impossible. Among these methods, the Finite Difference Method (FDM) provides a straightforward and intuitive approach to approximate the deflection of beams by converting differential equations into solvable algebraic equations.

## 1.2 Objective of the Project

The main objective of this project is to simulate the deflection of one-dimensional beams using Python, applying the Finite Difference Method. The focus is on two common boundary conditions: cantilever and simply-supported beams. Both point loads and uniformly distributed loads are considered in this study. Additionally, the Newton-Raphson method is used to solve the nonlinear equations that arise during the discretization process.

This project aims to:

- Understand the beam deflection problem from both physical and mathematical perspectives.
- Implement numerical techniques to solve the governing equations.
- Validate the numerical results with analytical solutions.
- Visualise the deflection behaviour using plotting tools.

## 1.3 Overview of the Approach

The project begins by reviewing the theoretical background of beam bending and the governing differential equations. In classical beam theory, the relation between bending

moment and deflection is linear, but this assumes small slopes. However, in real-life problems involving large deflections or flexible beams, nonlinearity becomes important.

In this project, we used the **nonlinear elastic beam equation** given by:

$$\frac{\frac{d^2v}{dx^2}}{(1+(\frac{dv}{dx})^2)^{(3/2)}} = -\frac{M(x)}{EI}$$

Where:

$v(x)$: deflection at point $x$,

$M(x)$: bending moment,

$EI$: flexural rigidity of the beam.

This equation captures the nonlinear geometric behaviour of the beam and gives more accurate deflection results when slopes are not small. Since this equation is not easy to solve analytically, we discretized it using the Finite Difference Method and solved it numerically.

To handle the nonlinear system, we implemented the Newton-Raphson method, which is an iterative approach commonly used to find roots of nonlinear equations. Using this approach, we simulated the deflection of cantilever and simply-supported beams under both point and distributed loads. Finally, the results were plotted using Matplotlib, and various cases were compared to understand how different parameters affect beam deflection.

# Numerical Techniques

## 2.1 Finite Difference Method (FDM)

The Finite Difference Method (FDM) is a powerful and simple numerical approach for solving differential equations. It replaces derivatives in the equation with approximate expressions using values at discrete points. In beam deflection problems, we often deal with second-order differential equations, which are suitable for FDM-based discretization.

By applying FDM, the continuous beam is divided into a number of equally spaced points (or nodes), and the differential equation is rewritten in terms of values at these nodes. This converts the problem into a system of algebraic equations, which can then be solved using numerical techniques.

## 2.2 Discretization Techniques

To apply FDM, the beam length is discretized into $N$ segments, producing $N$+1 nodes. At each node, we approximate the first and second derivatives using formulas based on neighboring values. These approximations are known as finite difference formulas, and there are three major types: central, forward, and backward difference.

### 2.2.1 Central Difference Approximation

The central difference method is symmetric and generally gives better accuracy compared to forward or backward differences. It uses the values on both sides of a point to estimate the derivative.

For the **second derivative**, the central difference is:

$$f''(x) \approx \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}$$

For the **first derivative**, the approximation is:

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

This method is commonly used at interior points where neighboring values on both sides are available. And this gives the most approximate result than forward and backward approximation.

### 2.2.2 Forward Difference Approximation

The forward difference method is useful near the start of the domain where we may not have points behind the current node. It relies on function values ahead of the current point.

For the **second derivative**:

$$f''(x) \approx \frac{2f(x) - 5f(x+h) + 4f(x+2h) - f(x+3h)}{h^2}$$

For the **first derivative**:

$$f'(x) \approx \frac{-3f(x) + 4f(x+h) - f(x+2h)}{2h}$$

While not as accurate as central difference, this is practical for boundary points.

### 2.2.3 Backward Difference Approximation

The backward difference is used near the end of the beam where points ahead of the node aren't available. It uses values behind the point of interest.

For the **second derivative**:

$$f''(x) \approx \frac{2f(x) - 5f(x-h) + 4f(x-2h) - f(x-3h)}{h^2}$$

For the **first derivative**:

$$f'(x) \approx \frac{3f(x) - 4f(x-h) + f(x-2h)}{2h}$$

These formulas are especially helpful when applying boundary conditions at the end of the beam.

While both forward and backward differences are first-order accurate, the central difference is second-order accurate and preferred when possible.

# Initial Approximations and Differential Methods

## 3.1 Initial Guess Strategies: random, uniform, zeros

Before applying iterative methods like Newton-Raphson to solve the nonlinear beam deflection equations, we need an initial guess for the deflection profile v(x). This initial guess plays a major role in determining whether the method will converge efficiently and correctly. In our code, we implemented three different strategies to generate initial guesses:

- **Random Guess**:

  Here, each value of v is initialized randomly within a small range depending on the direction of the applied load P. This is done using:

  $$v = \text{np.random.uniform}(0, 0.01 * \text{np.sign}(P), N + 1)$$

  The first value is always set to zero to satisfy the boundary condition at the fixed end. Although random guesses introduce diversity, they might take more iterations to converge or even diverge if not chosen carefully.

- **Uniform Guess:**

  A linearly increasing (or decreasing) guess is made along the beam, based on the load sign:

  $$v = 0.0001 * \text{np.sign}(P) * x[:]$$

  This guess tries to follow the expected deflection shape of the beam, giving the solver a better starting point and usually faster convergence.

- **Zero Guess:**

  This is the simplest option, where the entire deflection array is set to zero initially:

  $$v = \text{np.zeros}(N + 1)$$

  It's useful for small loads or linear cases but may take longer to converge for complex nonlinear conditions.

Each method has its pros and cons, and based on the scenario (load magnitude, EI, number of nodes), one can choose the best fit.

## 3.2 Approaches for Solving Beam Differential Equations

### 3.2.1 Direct Method

- We insert the boundary conditions directly into the nonlinear beam equation and solve the continuous form in one shot, treating derivatives as analytic symbols until the final algebraic rearrangement.
- This keeps the theoretical development clean but becomes algebraically messy for large systems, so it is mainly a teaching benchmark in our project.

### 3.2.2 Finite Difference Method – First Order (FD1)

- First-order forward or backward stencils approximate the slope and curvature near boundaries, delivering $O(\Delta x)$ accuracy while keeping the matrix bands narrow.
- FD1 is straightforward to program, yet the truncation error is relatively high; we therefore reserve it for quick checks or when memory is severely limited.

### 3.2.3 Finite Difference Method – Second Order (FD2)

- Central-difference formulas provide $O(\Delta x^2)$ accuracy for interior nodes, giving noticeably smoother curvature and faster grid convergence than FD1.
- Boundary nodes still need one-sided schemes, but the overall global error remains second order, making FD2 the preferred choice for all production runs in this report.

**Note:**

*Regardless of FD1 or FD2, the boundary conditions are not "added later"; they are discretised along with the differential equation itself so that the matrix rows at the ends enforce the correct support behaviour automatically.*

# Newton-Raphson Method

## 4.1 Introduction to Newton-Raphson Method

The Newton-Raphson (NR) method is a widely used iterative technique for solving nonlinear algebraic equations. In structural mechanics problems like nonlinear beam deflection, the governing equations often become too complex to solve analytically. That's where Newton-Raphson becomes useful—it helps us approximate the root of the equation by starting with an initial guess and improving it step by step.

The general idea of **Newton-Raphson** is:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

This formula works well for single-variable equations. For our beam problem, which involves multiple equations (a system), we apply a matrix-based version using Jacobians.

## 4.3 Newton-Raphson Algorithm (for System of Equations)

To solve the nonlinear beam equation numerically, we implemented the Newton-Raphson method for systems of equations. This section outlines how the algorithm works in our project.

The Newton-Raphson method is used to solve systems of nonlinear equations of the form:

$$F(u) = 0$$

In our project, $u$ represents the vector of beam deflections at various points along the beam, and $F(u)$ represents the nonlinear finite difference equations obtained after discretizing the governing differential equation.

To find a solution / solve this nonlinear system, Newton's method starts by linearly approximating $F$ near a current guess $u\_$ using the first-order Taylor series expansion:

$$F(u) \approx F(u\_) + J(u\_) \cdot (u - u\_)$$

Here, $J$ is the Jacobian matrix of partial derivatives, defined as:

$$J_{i,j} = \frac{\partial F_i}{\partial u_j}$$

Thus, the given nonlinear system can be expressed in an approximate form as:

$$\hat{F}(u) = F(u\_) + J(u\_) \cdot (u - u\_) = 0$$

This is now a linear equation in terms of $u$, which we can solve by the following two-step approach:

First, determine u by solving

$$J \cdot \delta u = -F(u\_)$$

Once $\delta u$ is found, the new guess is updated by:

$$u = u\_ + \delta u$$

### 4.3.1 Incorporating Relaxation

In practice, a relaxation parameter $\omega$ (where $0 < \omega \leq 1$) is sometimes introduced to improve stability and convergence:

$$u = \omega(u\_ + \delta u) + (1 - \omega)u\_ = u\_ + \omega\delta u$$

This weighted update allows partial adjustment toward the new solution in each step and is especially useful when dealing with stiff or sensitive nonlinear systems.

### 4.3.2 Algorithm for Newton's method.

Given the system $F(u) = 0$ and an initial guess $u\_$, The Newton-Raphson algorithm proceeds as:

I. **Compute the Residual**: $R = F(u\_)$

II. **Assemble the Jacobian Matrix**: $J = \frac{\partial F}{\partial u}$

III. **Solve the Linear System**: $J \cdot \delta u = -F(u\_)$

IV. **Update the Solution**:

- Without relaxation: $u = u\_ + \delta u$
- With relaxation: $u = \omega(u\_ + \delta u) + (1 - \omega)u\_ = u\_ + \omega\delta u$
  (In code, we implemented with relaxation factor as it gives more stability to our solution)

V. **Repeat until convergence:**

- If the norm of $\Delta u$ or $R$ is below a defined tolerance, stop.
  (i.e., when $||\delta u||$ or $||F(u)||$ is below a chosen tolerance, in our case, the tolerance is $10^{-6}$)
- Otherwise, repeat from step I

.

# Case Setup for Cantilever Beam

In this section, we describe how the numerical simulation was set up for a cantilever beam under various loading conditions. A cantilever beam is fixed at one end and free at the other, which leads to specific boundary conditions that are incorporated into the numerical scheme. Two types of loads were analyzed: point load and uniform continuous load.

## 5.1 Diagrams for Point Load and Continuous Load

### 5.1.1 Point Load at Free End

A concentrated load $P$ is applied at the free end of the beam (at $x = L$).

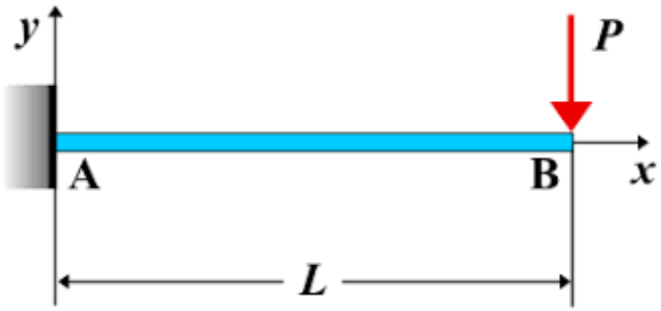


Fig 5.1.1: Cantilever Beam under Point Load

### 5.1.2 Uniform Continuous Load

A constant distributed load W acts over the entire length of the beam.

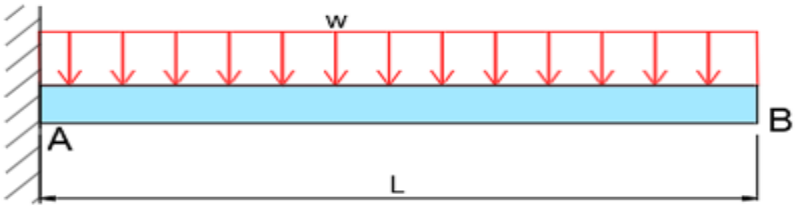


Fig 5.1.2: Cantilever Beam under Uniform Continuous Load

## 5.2 Boundary Conditions (BC) and Explanation

For a cantilever beam, the boundary conditions are based on the fixed support at one end (usually $x = 0$) and a free end at $x = L$. These conditions are:

- At $x$ = 0 (Fixed End):
  - Deflection $v(0) = 0$
  - Slope $\frac{dv}{dx}(0) = 0$

- At $x$ = L (Free End):
  - For Point Load: Bending moment and shear force are used to evaluate end conditions indirectly.
  - For Continuous Load: Conditions are handled via the governing equation, with appropriate numerical handling / Method at the free end.

## 5.3 Code File Order

### 5.3.1 Point Load

For the point load case, we implemented a numerical solver using the Newton-Raphson method to solve the nonlinear beam deflection equation:

$$\frac{\frac{d^2v}{dx^2}}{\left(1+\left(\frac{dv}{dx}\right)^2\right)^{\frac{3}{2}}} = -\frac{M(x)}{EI}$$

This equation arises due to large deformation effects, making the system nonlinear. The beam is fixed at one end and subjected to a downward point load at the free end.

**Step-by-Step Approach in Code:**

**I. Discretization**

We discretized the beam using N intervals over the length L = 1.0. The grid spacing is dx = L / N. A moment distribution M(x) = -P(L-x) was used based on the loading condition.

**II. Initial Guess**

Depending on user input, we initialized the deflection array v as either:

- Random small values
- Linearly increasing (uniform guess)
- All zeros

**III. Residual Function (R)**

We calculated the residual R based on the finite difference form of the nonlinear equation. The beam's boundary conditions are carefully handled in three parts:

- **At Fixed End i.e. x = 0 (i = 0):**

The boundary condition at the fixed end is $v(0) = 0$. For slope $dv/dx$ at this point, we implemented three options:

- **FD2 (second order):**

$$R[0] = \frac{-3v_0 + 4v_1 - v_2}{2dx}$$

- **FD1 (first order):**

$$R[0] = \frac{v_1 - v_0}{dx}$$

- **Direct (second derivative + moment):**

$$R[0] = \frac{2v_0 - 5v_1 + 4v_2 - v_3}{dx^2} + \frac{M[0]}{EI}$$

- **At Free End i.e. x = L (i = N):**

  We used a backward finite difference approximation for curvature and added the nonlinear moment-curvature term:

$$\delta = 3v_N - 4v_{N-1} + v_{N-2}$$

$$R[N] = \frac{2v_N - 5v_{N-1} + 4v_{N-2} - v_{N-3}}{dx^2} + \frac{M[N]}{EI}$$

- **For all interior points (1 ≤ i ≤ N-1):**

  We calculated the central difference for the second derivative and included the nonlinear term:

$$\delta = v_{i+1} - v_{i-1}$$

$$R[i] = \frac{v_{i+1} - 2v_i + v_{i-1}}{dx^2} + \frac{M[i]}{EI}(1 + \frac{\delta^2}{4dx^2})^{3/2}$$

## IV. Jacobian Matrix (J):

We derived the Jacobian matrix entries based on the residual expressions above. This matrix was updated in each Newton-Raphson iteration and used to solve for **delta_v**.

## V. Update Step:

Then solved $J\delta v = -R$, then updated the guess as:

$$v \leftarrow v + \omega \cdot \delta v$$

where $\omega = 0.5$ is the relaxation factor used to ensure numerical stability.

## VI. Comparison:

Once convergence was reached, we compared the nonlinear solution with the analytical linear solution from Euler-Bernoulli theory:

$$v(x) = \frac{Px^2}{6EI}(3L - x)$$

All the Result and Observation are in the section 7.1.1 for the better visual reference.

### 5.3.2 Point Load with EI Variation

In this sub-section, we extended the nonlinear Newton-Raphson solver (used earlier in Section 5.3.1) to study how the beam's deflection at the free end changes with different values of flexural rigidity $EI$. The objective was to analyze and compare the nonlinear and linear (Euler-Bernoulli) solutions for a cantilever beam under a point load, while varying stiffness from very low to very high.

We defined an array of flexural rigidity values:

$$EI_{\text{array}} = [20,30,40,50,100,500,1000,2000]$$

For each value of $EI$, We ran the Newton-Raphson method to compute the nonlinear deflection profile, and also computed the analytical deflection using the Euler-Bernoulli formula. We specifically extracted the tip deflection for both cases and stored them to compare the difference.

The Difference was calculated using:

$$\text{Difference}\% = \left(\frac{v_{\text{nonlinear}} - v_{\text{linear}}}{v_{\text{linear}}}\right) \times 100$$

This comparison was used to evaluate how significant the nonlinear effect is at different stiffness levels for the point load.

These results were stored and presented in the following formats:

**Output Table:**

A table was created to list the deflections at the tip for linear and non-linear and their percentage differences for each EI.

**Output plots are:**

**Plot 1:**

**Title:** *Relative Percentage Difference (%) vs Flexural Rigidity (EI)*

This log-scale plot shows how the relative percentage difference decreases with increasing $EI$. It confirms that:

- For **low** $EI$ , the nonlinear deflection is significantly higher than the linear prediction.
- For **high** $EI$ , both deflections match closely, showing that linear theory is valid when stiffness dominates.

**Plot 2:**

**Title:** *Deflection at the Free End vs Flexural Rigidity (EI)*

This plot compares the actual deflection values of both models:

- It shows **monotonic decay** of deflection with increasing stiffness.
- The nonlinear curve lies above the linear one at low $EI$, again reflecting the additional curvature captured by the nonlinear formulation.

All plots and the complete result table for this section are included in Section 7.1.2 for better visual reference.

### 5.3.3 Comparison: Point Load vs. Distributed Load

In this section, we investigate the difference in nonlinear beam deflection under two fundamental loading conditions — a concentrated point load at the free end, and a uniformly distributed load along the beam. Using a Newton-Raphson-based solver implemented in Python, we solve the nonlinear beam equation for each case and compare the results with classical Euler-Bernoulli theory**.**

**Objective**

The aim is to numerically compute the deflection profiles for a cantilever beam subjected to:

- A point load $P$ at the free end,
- A uniform distributed load $W$ across the span,

and then compare:

a) Newton-Raphson nonlinear results,
b) Euler-Bernoulli linear solutions,

to understand the impact of load type on deflection, and how nonlinearity alters behavior.

**Methodology**

We discretize the beam into $N = 250$ segments and solve the governing nonlinear equation:

$$\frac{d^2v}{dx^2}/(1 + (\frac{dv}{dx})^2)^{3/2} = \frac{M(x)}{EI}$$

using the **Newton-Raphson method**, where:

- $M(x)$ is the bending moment due to either point load or a distributed load,
- $EI$ is the flexural rigidity (set to 2000 N·m²),
- Boundary conditions are $v(0) = 0$ and $dv/dx(0) = 0$

As in section 6.1.1, the point load case is there. So here also, the methodology is the same for calculating deflection for the point load and continuous load. The only difference is the **bending moment $M(x)$.**

Two separate Newton-Raphson loops solve for:

- **Point Load:** $M(x) = -P(L - x)$
- **Distributed Load:** $M(x) = Wx(L - \frac{x}{2}) - \frac{WL^2}{2}$

We explored different initial guess strategies (zeros, uniform, random) and used a relaxation factor $w = 0.5$ to ensure convergence. The code also allows experimenting with three methods for handling the slope boundary condition: FD1 (first-order), FD2 (second-order), and Direct.

**Visualization and Analysis**

The simulation generates interactive plots using Plotly. The results include:

- **Nonlinear (Newton-Raphson)** solutions for both point and distributed loads.
- **Analytical (Euler-Bernoulli)** solutions for comparison.

Below is a qualitative summary of the trends observed in the plot:

- Under the same magnitude of load (P = -100 N / W = -100 N/m):
    - The **distributed load** leads to more uniform deflection along the span, while the **point load** causes higher curvature near the free end.
    - **Maximum deflection** is **larger for point load** than for distributed load.

- The **nonlinear solver** shows deviation from the Euler-Bernoulli solution, especially near the free end where curvature is high.
- For higher deflections (i.e., smaller EI or larger loads), the **nonlinear effects become more significant**, and the Newton-Raphson results diverge noticeably from linear theory.

**Convergence**

The Newton-Raphson method converged successfully in both cases, usually within a few hundred iterations, depending on the initial guess. For certain large values of $N$ or poorly chosen initial guesses, the Jacobian matrix can become ill-conditioned.

All plots and the complete result table for this section are included in Section 7.1.3 for better visual reference.

### 5.3.4 Implementation for Multiple Loads (N Cases)

In the previous sections, we focused on analyzing beam deflection under a single point load or a uniform distributed load. However, real-world structures often experience a combination of multiple point loads and distributed loads acting simultaneously at various positions along the beam. To extend the flexibility and applicability of our simulation, we implemented a generalized version of the Newton-Raphson solver capable of handling multiple loading cases — both point loads and continuous distributed loads — at arbitrary positions and magnitudes.

The key improvement introduced in this section is the **interactive load input system**. Instead of hardcoding a single loading condition, the user can now specify:

- Any number of **point loads**, by providing their respective positions and magnitudes.
- Any number of **continuous distributed loads**, by providing the start and end positions along with the intensity (load per unit length).

How the Generalization Was Achieved:

I. **User Input Functions**:

Two separate functions **get_point_loads()** and **get_continuous_loads()** were created to collect the necessary inputs from the user. These inputs are stored as lists of tuples, which are later used to compute the bending moment at each discretized point.

II. **Modified Moment Equation**:

The **moment_equation()** function was redesigned to support multiple loads. For each x-position on the beam:

- For each point along the beam, we calculate the total bending moment by considering the effect of all point loads that have already been applied before that point. Only the loads acting to the left of the position we're evaluating contribute to the bending moment.
- The farther a load is from the point of interest, the greater its influence on the moment. By checking each load and summing its individual effect, we get the total moment at that point.

III. **General Newton-Raphson Solver Structure**:

The rest of the Newton-Raphson algorithm remains similar, with adjustments in the residual and Jacobian functions to incorporate the updated moment values. The nonlinear curvature term is still retained for large deflection accuracy.

IV. **Visualization Enhancements**:

After computing the deflection under combined loading, results are plotted using **Plotly**, with comparisons to the analytical Euler-Bernoulli solutions for both single-point and distributed loads. This helps visually verify the nonlinear solver's performance under complex loading conditions.

**Why This is Important:**

By introducing support for N loading cases, the script becomes highly versatile. It now acts as a general-purpose nonlinear beam solver, which can handle practical beam configurations encountered in civil, mechanical, and structural engineering scenarios.

All plots and the complete result table for this section are included in Section 7.1.4 for better visual reference.

# Case Setup for Simply Supported Beam

## 6.1 Problem Diagrams for Point Load and Uniform Load

A simply-supported beam is a type of beam that rests freely on two supports—one at each end. It allows vertical movement but prevents vertical displacement at the supports. There is no moment resistance, making it an idealized model for many engineering applications.

In this section, we present schematic diagrams for two commonly studied loading scenarios on a simply-supported beam. These diagrams form the basis for the numerical simulations and help clarify the assumptions and setup used in this study.

We consider two types of loads acting on such beams:

**Case 1: Point Load at the Center**

- A single concentrated load $P$ is applied at the mid-span of the beam.
- The load is symmetric with respect to the beam length. And Both supports carry equal reactions due to symmetry.
- This setup creates the maximum deflection at the center of the beam.

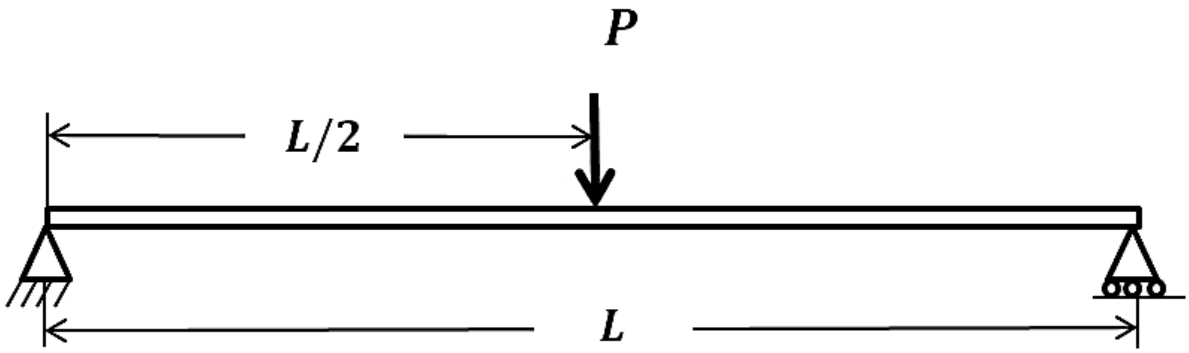


Fig 6.1.1: Simply Supported Beam under Point Load

**Case 2: Uniformly Distributed Load (UDL)**

- The load is spread evenly across the entire length of the beam with an intensity $q$ (force per unit length).
- Each segment of the beam experiences the same load density. Due to symmetry and uniformity, the beam exhibits smooth, symmetric deflection with the maximum occurring at the center.

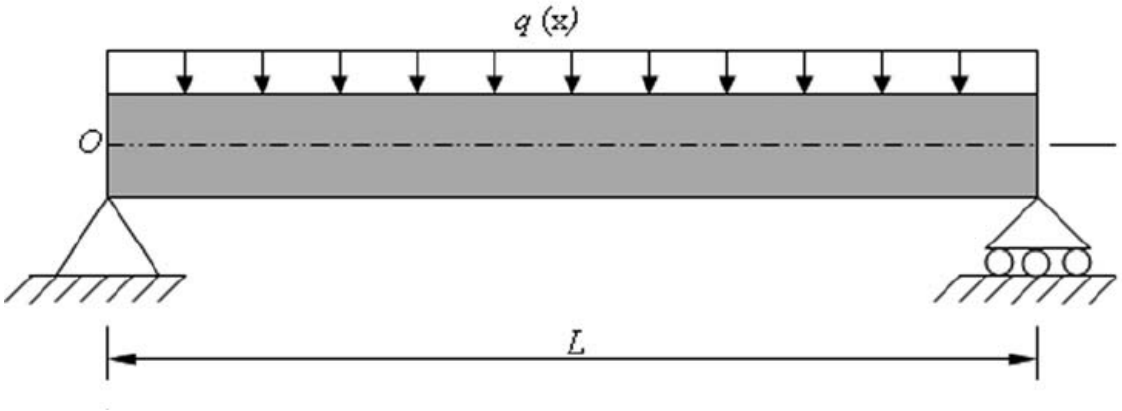


Fig 6.1.2: Simply Supported Beam under Continuous Loading

## 6.2 Boundary Conditions

In this study, we analyze a simply-supported beam, which is a common structural configuration where the beam rests on two supports—one at each end. These supports restrict vertical displacement but allow free rotation. Based on this setup, the following boundary conditions are applied in the simulation:

**Mathematical Boundary Conditions:**

- $v[x = 0] = 0$
- $v[x = L] = 0$
- $\frac{d^2v}{dx^2}[x = 0] = 0$
- $\frac{d^2v}{dx^2}[x = L] = 0$

**Explanation:**

- $v[x = 0] = 0$: The vertical deflection at the **left support** is zero. This is because the beam is **pinned** at this point and cannot move vertically.
- $v[x = L] = 0$: Similarly, the vertical deflection at the **right support** is also zero for the same reason—it is pinned and thus fixed in vertical position.
- $\frac{d^2v}{dx^2}[x = 0] = 0$: The **bending moment** at the left support is zero. This implies the beam is **free to rotate** at this point and does not resist moments.
- $\frac{d^2v}{dx^2}[x = L] = 0$: The bending moment at the right support is also zero, indicating that the beam can **freely rotate** and is not fixed against moment at this end either.

These boundary conditions accurately represent the physical constraints of a simply-supported beam and are crucial for obtaining correct simulation results using finite difference methods.

## 6.3 Code Implementation Overview

The numerical simulation for the simply-supported beam is implemented using the Finite Difference Method (FDM), similar to the cantilever case. However, the key difference lies in the treatment of boundary conditions. In this section, we provide an overview of the code structure, highlighting how the governing equation is discretized and solved.

### 6.3.1 Point Load

**Discretization and Initial Setup**

- The beam is divided into N segments over a length L = 1.0.
- A **point load (P = -100 N)** is applied at the center of the beam.
- The flexural rigidity EI = 2000 N·m² is constant throughout.
- Three types of initial guesses for the deflection array v are supported: "**random**", "**uniform**", and "**zeros**". This allows testing for stability and convergence behavior.

**Boundary Conditions**

As mentioned earlier, the beam is **simply supported**, so the **deflection at both ends is zero**:

$$v(0) = v(L) = 0$$

Additionally, there is **no bending moment at the supports**, implying:

$$\frac{d^2v}{dx^2}(x = 0) = \frac{d^2v}{dx^2}(x = L) = 0$$

These conditions are numerically enforced using **forward and backward finite difference approximations** of the second derivative near the boundaries.

**Moment Distribution**

The internal moment $M(x)$ along the length of the beam is calculated based on standard mechanics of materials theory for a simply supported beam with a central point load:

For $x \leq L/2$

$$M(x) = \frac{P}{2}x$$

For $x > L/2$

$$M(x) = \frac{P}{2}(L - x)$$

This gives a linear variation of moment increasing up to mid-span and symmetrically decreasing beyond it.

**Residual and Jacobian Functions**

The nonlinear governing equation implemented is derived from the curvature-moment relationship:

$$\frac{d^2v}{dx^2} = -\frac{M(x)}{EI}(1 + (\frac{dv}{dx})^2)^{3/2}$$

This equation is discretized using **central difference schemes** for internal points and special **boundary-specific stencils** at the edges.

The residual is evaluated as follows:

- At the **left boundary** $x = 0$:

$$R[0] = \frac{2v_0 - 5v_1 + 4v_2 - v_3}{dx^2}$$

- At the **right boundary** $x = L$:

$$R[N] = \frac{2v_N - 5v_{N-1} + 4v_{N-2} - v_{N-3}}{dx^2}$$

- For **internal nodes** $i \in [1, N-1]$, the residual incorporates the nonlinear term:

$$R[i] = \frac{v_{i+1} - 2v_i + v_{i-1}}{dx^2} + \frac{M(x_i)}{EI}(1 + \frac{1}{4dx^2}(v_{i+1} - v_{i-1})^2)^{3/2}$$

The Jacobian matrix is also constructed analytically, accounting for the nonlinear term's sensitivity to changes in the deflection array $v$.

**Solver and Convergence**

The Newton-Raphson algorithm iteratively updates the solution:

$$v_{\text{new}} = v_{\text{old}} + \omega \cdot \Delta v$$

Here, $\Delta v$ is obtained by solving the linearized system:

$$J(v) \cdot \Delta v = -R(v)$$

If the Jacobian is singular, a pseudo-inverse is applied to compute the least-squares solution. The process continues until the residual or the change in solution drops below a user-defined tolerance.

**Results and Visualization**

Once converged, the final nonlinear deflection $v(x)$ is plotted alongside the classical **Euler-Bernoulli solution** for validation:

- Newton-Raphson (nonlinear)
- Euler-Bernoulli (linear)

An interactive Plotly plot allows for direct visual comparison between both solutions.

All plots and the complete result table for this section are included in Section 7.2.1 for better visual reference.

### 6.3.2 Comparison: Point Load vs. Distributed Load

In this section, the Newton-Raphson method is extended to handle two different types of loading conditions on a simply-supported beam: **a central point load** and a **uniformly distributed load (UDL)**. Since the governing equation remains the same for both cases, the overall structure of the code is similar to the previous implementation. However, the distinction arises in the **moment calculation** for each load type.

**Discretization and Load Setup**

As with the previous case, the beam of length $L = 1$ is discretized into $N + 1$ points using spacing $dx = L/N$. The difference lies in the **nature of the external force**:

- A uniformly distributed load of magnitude $W = -100$ N/m, is applied over the entire length of the beam.
- The initial deflection guess $v_{\text{cont}}$ is constructed based on user input (random, uniform, or zeros). The function sets the first and last values of $v_{\text{cont}}$ to zero, satisfying the boundary condition $v(0) = v(L) = 0$.

**Boundary Conditions**

The boundary conditions remain identical to those already mentioned above in section of 6.3.1:

- The beam is simply supported, so the deflection is zero at both ends.
- The second derivative of the deflection $\frac{d^2v}{dx^2}$, representing the bending moment, is also zero at the boundaries, which is applied using **finite difference approximations** at $x = 0$ and $x = L$

**Moment Distribution for Uniform Load**

- For a Point Load applied at the center of the beam ($x = L/2$), the bending moment at any $x$ is given by:

For $x \leq L/2$

$$M(x) = \frac{P}{2}x$$

For $x > L/2$

$$M(x) = \frac{P}{2}(L - x)$$

- For a **Uniformly Distributed Load (UDL)** over the entire span, the bending moment at any location $x$ follows the standard expression:

$$M(x) = \frac{1}{2}Wx(L - x)$$

This equation is derived from the static analysis of bending moments in a simply-supported beam under uniform load. This gives a parabolic bending moment diagram, peaking at the center.

The program constructs two separate moment arrays – one for each loading type – using these equations, which solve the **Residual and Jacobian Functions.**

**Residual and Jacobian Functions**

- The residual vector *R* and Jacobian matrix *J* remain the same as discussed in Section 6.3.1, and they serve to enforce the nonlinear beam equation:

$$\frac{d^2v}{dx^2} = -\frac{M(x)}{EI} \cdot (1 + (\frac{dv}{dx})^2)^{3/2}$$

- This nonlinear curvature is discretized and implemented using second-order central difference approximations. At the boundaries, forward and backward finite difference schemes are used to maintain the zero-curvature condition:
- At $x = 0$:

$$R_0 = \frac{2v_0 - 5v_1 + 4v_2 - v_3}{dx^2}$$

- At $x = L$:

$$R_N = \frac{2v_N - 5v_{N-1} + 4v_{N-2} - v_{N-3}}{dx^2}$$

**Newton-Raphson Iteration and Convergence**

- The Newton-Raphson algorithm is run independently for both the UDL and point load cases.
- The process terminates when either the residual norm or the solution update norm falls below the convergence threshold.

**Visualization and Comparison**

Once the nonlinear deflections are computed for both loading cases, the solution vectors are plotted alongside their corresponding **Euler-Bernoulli analytical solutions**. The analytical expression for a simply supported beam under UDL is:

$$v_{\text{EB}}(x) = \frac{Wx}{24EI}(L^3 - 2Lx^2 + x^3) \qquad \text{(Continuous Load Case)}$$

And under point load is:

$$\text{For } x \le \frac{L}{2}, \quad v(x) = \frac{Px}{12EI}\left(\frac{3}{4}L^2 - x^2\right)$$

$$\text{For } x > \frac{L}{2}, \quad v(x) = \frac{P(L-x)}{12EI}\left(-x^2 + 2Lx - \frac{1}{4}L^2\right)$$

All plots and the complete result table for this section are included in Section 7.2.2 for better visual reference.

### 6.3.3 Implementation for Multiple Loads (N Cases)

In this section, we extend the previous simulation framework to handle **multiple types of loads** applied simultaneously to a simply supported beam. Unlike the earlier sections that only dealt with **a single point load** or **a single uniform distributed load**, this part of the project allows the user to input **any number of point loads** and **continuous loads** defined over arbitrary intervals. The implementation solves the resulting **nonlinear beam deflection** using the **Newton-Raphson method**.

**Objective**

The primary objective of this section is to simulate the **nonlinear bending behavior** of a simply-supported beam subjected to a **combination of multiple loading conditions**, including both **point loads** and **uniformly distributed loads** applied at arbitrary locations along the span. This goes beyond the earlier scenarios where we dealt with only a single load type, making the analysis more realistic and applicable to practical engineering problems. The aim is to develop a **flexible and interactive Python-based solver** using the **Newton-Raphson method** that can handle these generalized load inputs, compute the resulting deflection profile, and **compare the nonlinear results** with traditional **Euler-Bernoulli analytical solutions** for special cases.

**Load Handling Strategy**

The code dynamically prompts the user to enter:

- Number of point loads
- For each point load:
    - **Position** (in meters, from the left end)
    - **Magnitude** (in Newtons)
- Number of continuous (uniform) loads
- For each continuous load:
    - **Start position** and **end position** (in meters)
    - **Intensity** (in N/m)

These inputs are then used to construct a combined **moment distribution** along the beam using the principle of superposition.

**Bending Moment Calculation**

For each grid point $x_i$ along the beam, the net bending moment $M(x_i)$ is computed by summing contributions from:

- All **point loads** using piecewise moment equations
- All **continuous loads** by evaluating the effect over the relevant intervals

This moment is then passed to the Newton-Raphson solver, where it is used in the **nonlinear differential equation** governing beam deflection.

**Governing Equation (Nonlinear Form)**

The solver solves the following nonlinear deflection equation for a simply-supported beam:

$$\frac{d^2v}{dx^2} + \frac{M(x)}{EI}(1 + (\frac{dv}{dx})^2)^{3/2} = 0$$

**Code Features**

- User-friendly inputs for multiple load types
- Flexible handling of arbitrary positions and spans
- Residual and Jacobian are updated iteratively
- Relaxation factor w used for convergence control
- Fallback to least-squares solution if Jacobian is singular
- Automatic plotting using Plotly for:
    - Newton-Raphson numerical solution
    - Euler-Bernoulli (analytical) solution for point and distributed loads

**Sample Plot**

After solving the beam deflection, the code generates an interactive plot that compares:

- The numerical solution (from Newton-Raphson)
- The Euler-Bernoulli analytical solution for **mid-span point load and uniform load over the entire beam.**

This helps visually understand the deviation between linear and nonlinear behavior when multiple loads are involved.

**Observations**

- The Newton-Raphson method converges successfully for multiple load cases when initial guesses are chosen properly (zeros or uniform).

- The nonlinear solution shows slight deviation from analytical solutions, especially in regions where **loads overlap**.
- If grid size N is too large (e.g., > 350), the Jacobian may become **singular**, indicating numerical instability.

All plots and the complete result table for this section are included in Section 8.2.3 for better visual reference.

# Results and Discussion

## Figure 7.1.1: Newton-Raphson Solution for Point Load Case – Observations

**Input Parameters Summary**

- **Discretization**: N = 250 segments
- **Beam Length**: L = 1 m
- **Flexural Rigidity**: EI = 2000 N·m²
- **Applied Load**: P = -100.0 N (downward point load at free end)
- **Relaxation Factor**: w = 0.5
- **Initial Guess**: Uniform distribution
- **Boundary Condition Method**: FD2 (second-order finite difference)

**(a) Deflection Comparison: Newton-Raphson vs Euler-Bernoulli**

- **Overall Deflection Behavior:** Both Newton-Raphson (nonlinear) and Euler-Bernoulli (linear) solutions show the characteristic cantilever beam deflection pattern, with maximum deflection occurring at the free end (x = 1.0).
- **Solution Convergence:** The Newton-Raphson method successfully converged with the given parameters, demonstrating the effectiveness of the FD2 boundary condition treatment and uniform initial guess strategy.
- **Nonlinear vs Linear Comparison:** The two solutions appear to follow very similar paths along the beam length, suggesting that for the given load magnitude and beam stiffness, nonlinear effects are relatively small but still present.
- **Deflection Magnitude:** The maximum deflection at the free end appears to be approximately -0.016 to -0.017 m, indicating a relatively stiff beam response under the applied 100 N load.
- **Beam Curvature Pattern:** Both solutions show smooth, continuous deflection curves with increasing curvature toward the free end, which is expected for cantilever beams under point loading**.**

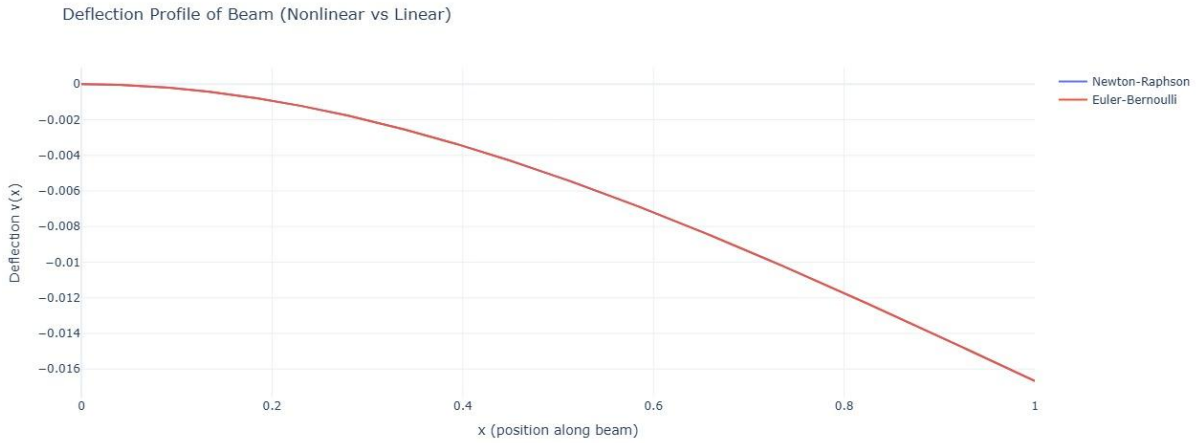


Figure 7.1.1 (a): Deflection Profile for Point Load Only

---

**(b) Zoomed-in View of Deflection Comparison**

- **Nonlinear Effects Visibility**: The zoomed view reveals subtle but measurable differences between the Newton-Raphson and Euler-Bernoulli solutions, particularly in the region near the free end where deflections are largest.
- **Solution Accuracy**: The close agreement between both methods validates the numerical implementation of the Newton-Raphson solver and confirms that the discretization with N = 250 provides sufficient resolution.
- **Convergence Quality**: The smooth nature of the Newton-Raphson solution curve indicates that the relaxation factor of w = 0.5 provided stable convergence without oscillations or numerical instabilities.
- **Physical Interpretation**: The small deviation between linear and nonlinear solutions suggests that geometric nonlinearity becomes more significant as deflections increase, even though the overall effect remains modest for this particular load case.
- **Numerical Precision**: The consistent spacing between the two curves demonstrates that the finite difference discretization successfully captures the nonlinear beam behavior with good accuracy.

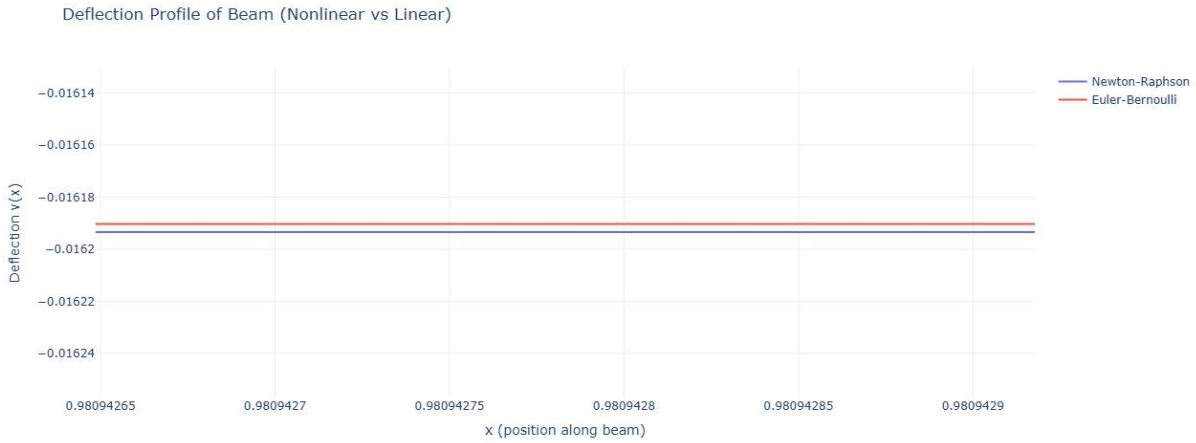


Figure 7.1.1 (b): Zoomed-in View of Deflection Profile for Point Load Only

## Figure 7.1.2: Error Analysis for Point Load

### Input Parameters Summary

- **Discretization**: N = 250 segments
- **Beam Length**: L = 1 m
- **Flexural Rigidity**: EI varies between [20, 30, 40, 50, 100, 500, 1000, 2000]
- **Applied Load**: P = -100.0 N (downward point load at free end)
- **Relaxation Factor**: w = 0.5
- **Initial Guess**: Uniform distribution
- **Boundary Condition Method**: FD2 (second-order finite difference)

### (a) – Linear vs Non-Linear Tip Deflection (Table)

- Deflection magnitudes drop steadily as EI increases, matching the classic cantilever formula $v_{tip}(x) = \frac{PL^2}{3EI}$ that predicts an inverse $1/EI$ trend.
- The percentage error peaks at EI ≈ 50 N·m² (~156 %), then declines towards zero for rigid beams (EI ≥ 1000 N·m²), indicating that geometric non-linearity is most critical only within a mid-range stiffness band.

- At EI = 20 N·m², the nonlinear analysis gives roughly 29 % less tip deflection than the linear formula because the beam's large curvature adds extra geometric stiffness, so it resists bending more as it deforms.

| EI (Nm^2) | Deflection (linear) | Deflection (nonlinear) | % Difference |
|---|---|---|---|
| 20 | -1.666667 | -1.179111 | -29.253325 |
| 30 | -1.111111 | -1.385655 | 24.708923 |
| 40 | -0.833333 | -1.634777 | 96.173294 |
| 50 | -0.666667 | -1.708163 | 156.224478 |
| 100 | -0.333333 | -0.367194 | 10.158329 |
| 500 | -0.066667 | -0.066895 | 0.343235 |
| 1000 | -0.033333 | -0.033361 | 0.084049 |
| 2000 | -0.016667 | -0.016670 | 0.019456 |

Figure 7.1.2 (a): Table for variation in EI (Flexural Rigidity)

---

**(b) Plot: Relative Percentage Difference (%) vs Flexural Rigidity (EI)**

- **Peak Nonlinear Effect**: The relative percentage difference reaches its maximum (~156%) around EI = 50 N·m², indicating that geometric nonlinearity has the strongest influence within this mid-range stiffness region.
- **Rapid Decay with Increasing Stiffness**: As EI increases beyond 100 N·m², the percentage difference drops sharply toward zero on the semi-log plot, demonstrating that nonlinear effects become negligible for stiffer beams.
- **Convergence to Linear Theory**: For EI ≥ 500 N·m², the difference remains below 1%, confirming that the classical Euler-Bernoulli linear theory provides adequate accuracy for conventional structural applications with high flexural rigidity.

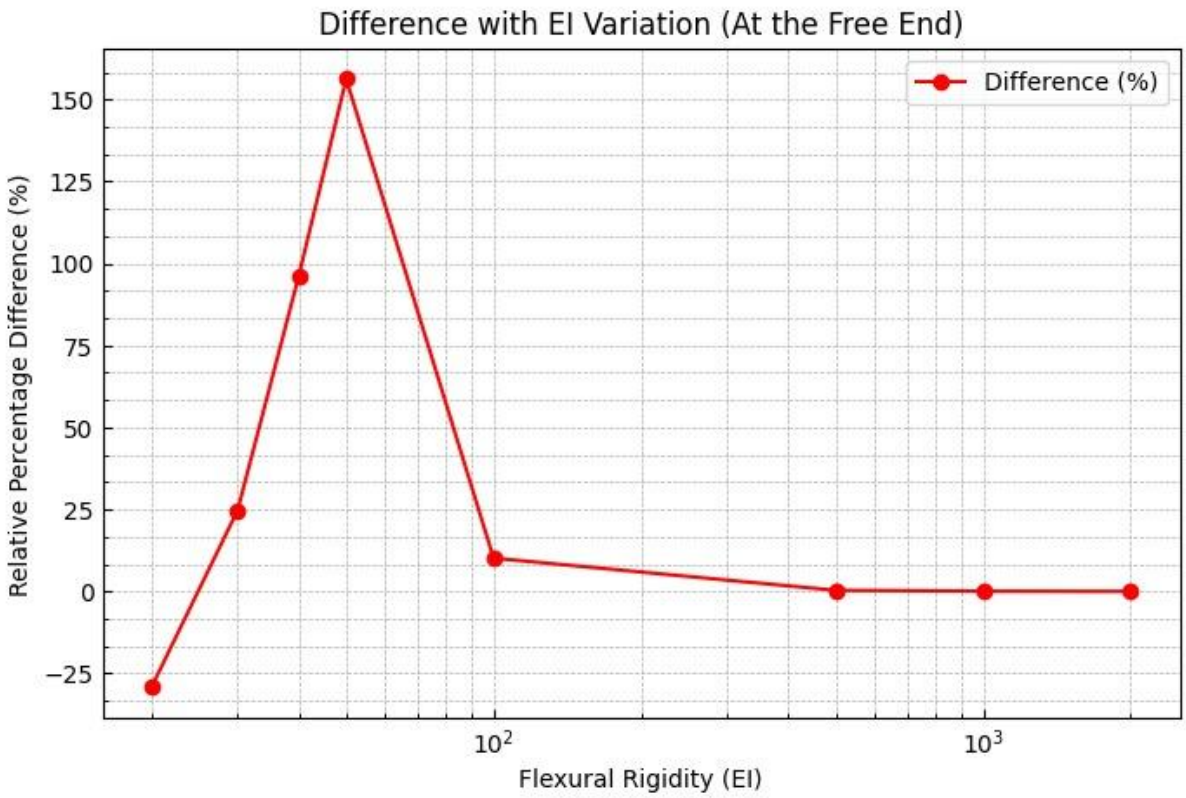


Figure 7.1.2 (b): Relative Percentage Difference (%) vs Flexural Rigidity (EI)

---

**Figure 7.1.2 (c) – Absolute Tip-Deflection Difference vs Flexural Rigidity (EI)**

- The absolute gap between nonlinear (Newton–Raphson) and linear (Euler-Bernoulli) tip deflections drops steeply as EI rises, mirroring the 1/EI dependence of cantilever deflection predicted by classical beam theory.
- For flexible beams with EI < 100 N·m², the mismatch remains in the centimetre range, showing that geometric nonlinearity must be considered to avoid under- or over-estimating real deformations.
- Once EI exceeds roughly 1000 N·m², the curve flattens close to zero, confirming that linear Euler-Bernoulli theory is sufficiently accurate for stiff members where deflections stay small relative to the span.

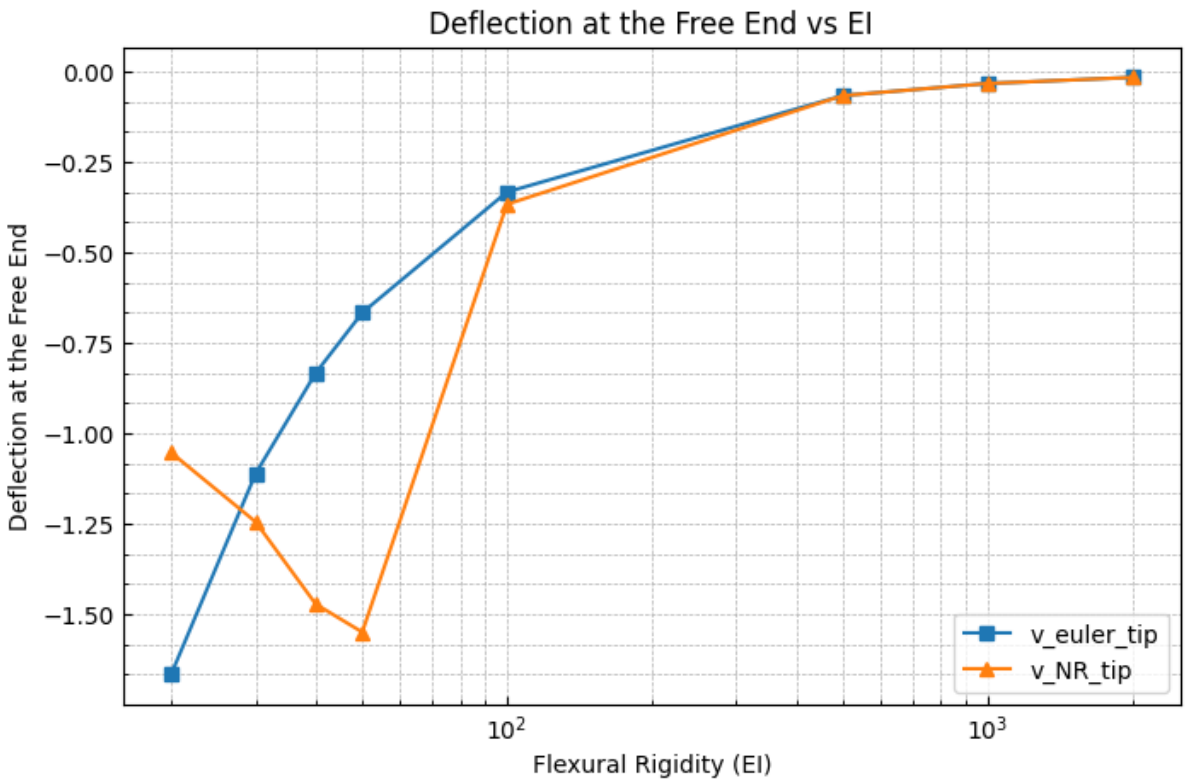


Figure 7.1.2 (c): Deflection at the Free End vs Flexural Rigidity (EI):

## Figure 7.1.3 – Comparison: Point Load vs Distributed Load

**Input Parameters Used for the Output Plot**

- **Discretization**: N = 250 segments
- **Beam Length**: L = 1 m
- **Flexural Rigidity**: EI = 2000 N·m²
- **Applied Loads**: P = -100.0 N (downward point load at free end) and W = -100.0 N/m (uniformly distributed load)
- **Relaxation Factor**: w = 0.5
- **Initial Guess**: Uniform distribution
- **Boundary Condition Method**: FD2 (second-order finite difference)

---

**(a) Plot: NR and Euler (Point Load & Distributed Load)**

- All four curves display the characteristic downward cantilever deflection profile, starting from zero at the fixed end and reaching maximum values at the free end, confirming proper implementation of boundary conditions.
- The distributed load cases (both NR and Euler) consistently show smaller deflections compared to their point load counterparts throughout the beam span, demonstrating how load distribution affects the overall structural response.
- The Newton-Raphson nonlinear curves almost perfectly overlap with their corresponding Euler-Bernoulli linear solutions, indicating that geometric nonlinearity effects are minimal at this stiffness level (EI = 2000 N·m²).

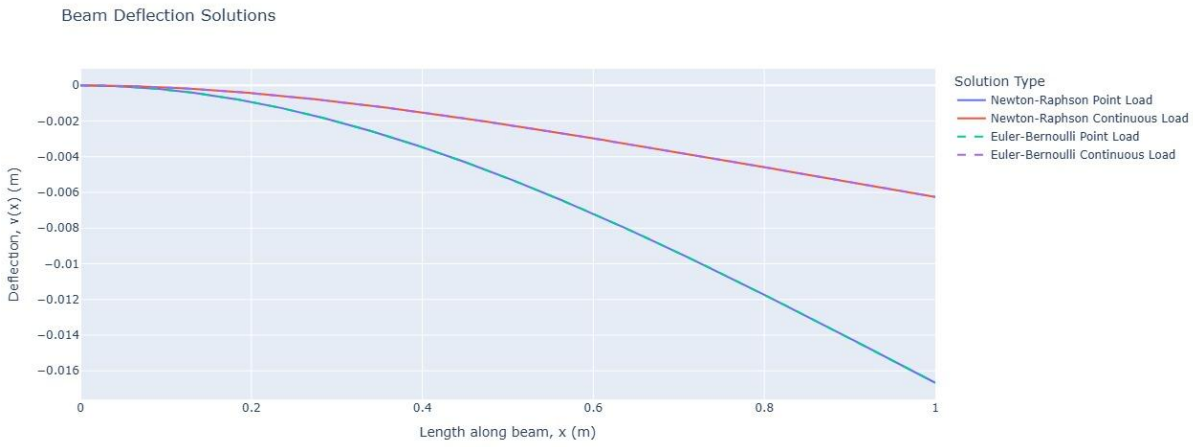


Figure 7.1.3 (a): Plot for NR (Point & Continuous Load) and Euler (Point & Continuous Load)

---

**(b) Zoomed-in View: NR vs Euler for Distributed Load**

- The zoomed comparison reveals excellent agreement between the Newton-Raphson and Euler-Bernoulli solutions for the distributed load case, with any differences being barely visible even at this magnified scale.
- The smooth convergence of both curves demonstrates the numerical stability of the FD2 boundary condition implementation and validates the choice of relaxation factor (w = 0.5) for this loading scenario.
- The uniformly distributed load produces a gentler curvature profile compared to point loading, resulting in more predictable linear behavior that closely matches classical beam theory expectations.


Figure 7.1.3 (b): Zoomed image of NR (Continuous Load) and Euler (Continuous Load)

---

**(c) Plot: NR (Point Load) vs NR (Distributed Load)**

- The Newton-Raphson point load solution shows significantly larger tip deflection compared to the distributed load case, highlighting how concentrated loading creates more severe deformation than equivalent distributed loading.
- Both nonlinear solutions exhibit smooth, physically realistic deflection curves without numerical oscillations, confirming the robustness of the Newton-Raphson implementation for different loading conditions.
- The comparison clearly demonstrates that for the same total force magnitude (-100 N vs -100 N/m over 1 m span), the load distribution pattern has a substantial impact on maximum deflection and overall beam response.

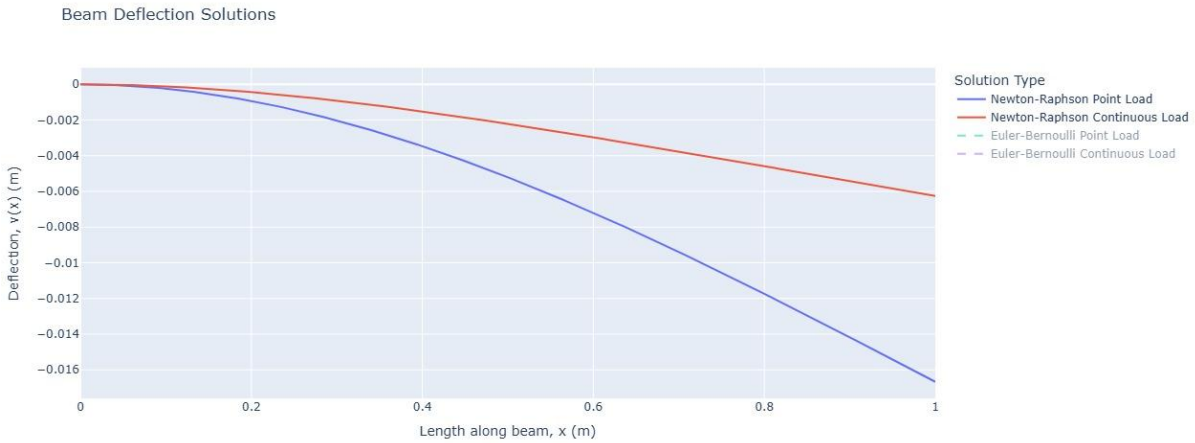


Figure 7.1.3 (c): Plot for NR (Point Load) and NR (Continuous Load)

**(d) Code 4 – Multiple Load Case Implementation for Cantilever Beam**

- **What the code does:** The Code lets you combine any number of point loads and uniformly distributed loads (UDLs) on the same beam in a single run.It combines point and uniformly distributed loads through superposition, then employs a Newton-Raphson solver to produce numeric results and on-demand deflection plots for the chosen load mix.
- **Why the plots are not available:** No sample figure is included because the program is designed for interactive use; user can enter load magnitudes, positions, and stiffness values, run the script, and it generates plots tailored to those chosen inputs, encouraging you to explore different loading scenarios on your own

## Figure 7.2.1 – Mid-Span Point-Load on a Simply-Supported Beam

**Input Parameters Summary**

- **Discretization**: N = 250 segments
- **Beam Length**: L = 1 m
- **Flexural Rigidity**: EI = 2000 N·m²
- **Applied Load**: P = −100 N (downward point load at mid-span)
- **Relaxation Factor**: w = 0.5 for Newton–Raphson iterations
- **Initial Guess**: Uniform
- **Boundary Condition Method**: FD2 (second-order finite difference)

---

**(a) Deflection Comparison – NR (Point Load) vs Euler (Point Load)**

- Both curves exhibit the classic symmetric dip of a simply-supported beam, with the largest downward deflection occurring exactly at the mid-span $(x = L/2)$. This matching mid-span peak shows that the supports have been modelled correctly, confirming that the simply-supported boundary conditions were enforced as intended.

- The Newton–Raphson (non-linear) solution sits almost exactly on top of the Euler-Bernoulli (linear) line, showing that geometric non-linearity is negligible for this stiff beam and modest load.

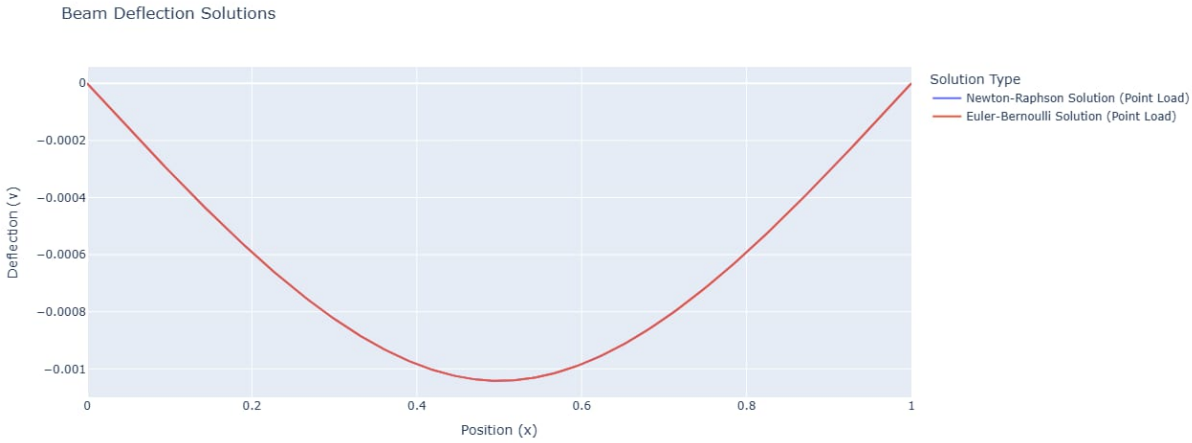


Figure 7.2.1 (a): Plot for NR (Point Load) and Euler (Point Load)

---

**(b) Zoomed-In View of Mid-Span Region**

- Even under magnification, the two solutions differ by only fractions of a millimetre, confirming excellent agreement and stable NR convergence with $w$=0.5.
- The near-horizontal overlap indicates that the second-order FD2 Condition provides sufficient accuracy without grid refinement for N=250.
- Because the beam's flexural rigidity (EI) is high, the standard small-deflection (linear) equations already give design-level accuracy. The non-linear solution is therefore run only as a check to verify that the linear prediction is valid, rather than as the primary design result.

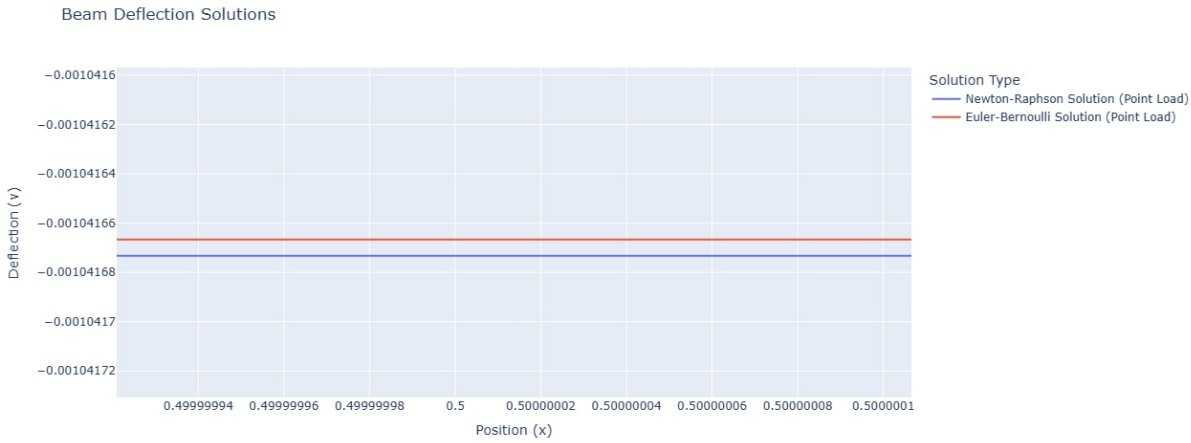


Figure 7.2.1 (a): Zoomed image of Plot for NR (Point Load) and Euler (Point Load)

## Figure 7.2.2 – Comparison: Point Load vs Distributed Load

### Input Parameters Summary

- **Discretization**: N = 250 segments
- **Beam Length**: L = 1 m
- **Flexural Rigidity**: EI = 2000 N·m²
- **Applied Loads**: −100 N point load at mid-span and −100 N/m uniformly distributed load (UDL)
- **Relaxation Factor**: w = 0.5
- **Initial Guess**: Uniform
- **Boundary Condition Method**: FD2 (second-order finite difference)

---

### (a) Plot – NR and Euler (Point & Distributed Load)

- Each curve displays the familiar, mirror-image dip typical of a simply-supported beam, with maximum deflection at mid-span, confirming that the support conditions are modelled correctly.
- The UDL pair lies above the point-load pair because spreading the load reduces local bending moments, leading to smaller peak curvature and deflection for the same total force.

- Newton–Raphson traces overlap the Euler-Bernoulli lines almost perfectly, indicating that geometric non-linearity is insignificant at this stiffness level and load magnitude.

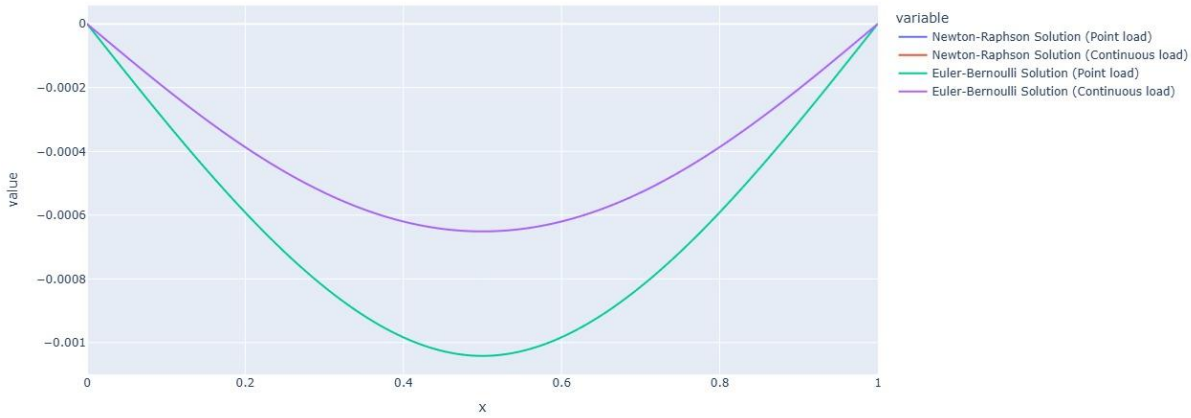


Figure 7.2.2 (a): Plot for NR (Point & Continuous Load) and Euler (Point & Continuous Load)

---

**(b) Zoomed-In View – NR vs Euler for Distributed Load**

- Even under magnification, the two distributed-load curves are nearly indistinguishable, demonstrating that linear theory gives design-level accuracy when EI is high.
- The small gap at the beam's centre (less than 1 mm) simply shows that the nonlinear solver predicts a slightly larger deflection when the beam bends more sharply. Because this extra deflection is tiny, it has no meaningful impact on routine design checks and can usually be ignored.
- Smooth convergence of the NR solution confirms that the chosen relaxation factor and FD2 condition provide a stable numerical scheme for this loading scenario.

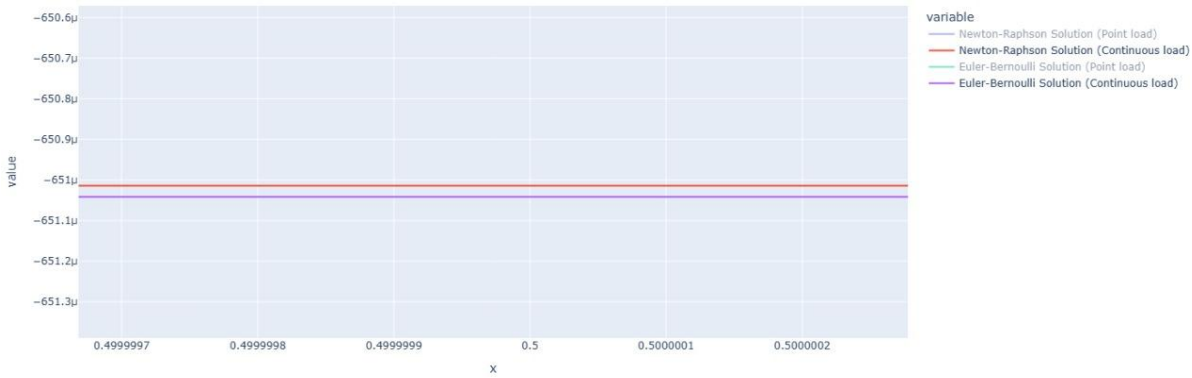

Figure 7.2.2 (b): Zoomed image of NR (Continuous Load) and Euler (Continuous Load)

---

**(b) Plot – NR (Point Load) vs NR (Distributed Load)**

- The point-load solution shows a noticeably larger mid-span deflection than the distributed-load case, underscoring how concentrated forces create higher local bending moments.
- For the same total applied force, load distribution governs maximum deflection more than geometric non-linearity at this stiffness, making the UDL configuration preferable when limiting deflection is critical.

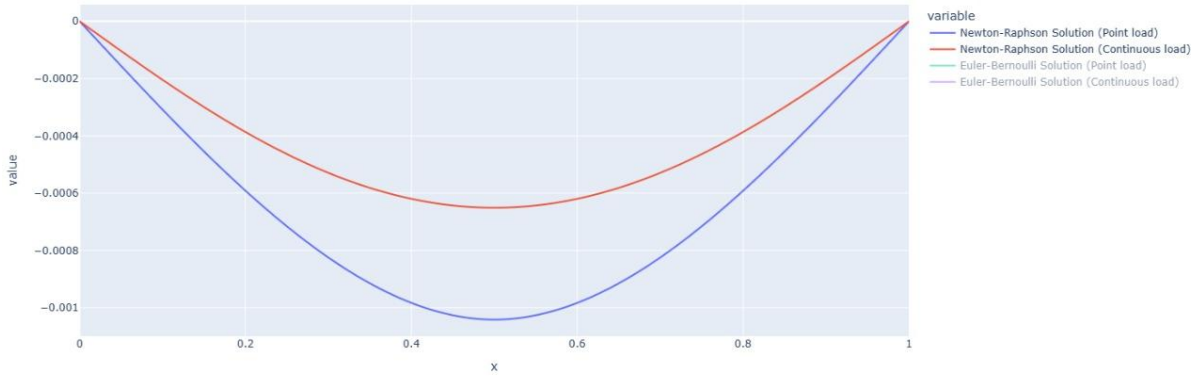


Figure 7.2.2 (c): Plot for NR (Point Load) and NR (Continuous Load)

---

**Figure 7.1.4 – Code 4: Multiple-Load Case Implementation for Simply Supported Beam**

- **What the code does:** The script lets you combine any number of point loads and uniformly-distributed loads, adds their bending-moment effects through the superposition principle, and then solves the nonlinear beam equation with a Newton–Raphson loop to give the full deflection profile in both numeric and graphical form.
- **Why no plots are shown here:** No sample figure is included because the program is designed for interactive use; you enter your own load magnitudes, positions, and stiffness values, run the script, and it generates plots tailored to those chosen inputs, encouraging you to explore different loading scenarios on your own

# Conclusion

In this project, we developed a numerical simulation for analyzing the deflection of beams under various loading conditions using the finite difference method in Python. Starting with the classical Euler-Bernoulli beam theory, we implemented both linear and nonlinear formulations and validated the nonlinear deflection using a Newton-Raphson iterative solver. We analyzed two types of beams—cantilever and simply supported—under point loads, uniformly distributed loads, and multiple load conditions.

The results show that the nonlinear approach yields more accurate deflection predictions, especially for large deflections or low flexural rigidity values. We also compared our numerical results with analytical solutions wherever applicable, observing good agreement and highlighting the regions where linear theory starts to deviate.

This work helped in deepening the understanding of how beam deflections behave under different scenarios and provided hands-on experience with solving nonlinear differential equations numerically.

# Future Work

## 9.1 Extension to Overhanging Beam

An immediate extension of this study would be to model **overhanging beams**, which are partially supported beams with sections extending beyond the supports. This would require adapting boundary conditions in the finite difference setup and analyzing the effect of loads applied on the overhanging part. It would add complexity but also broaden the scope and real-world applicability of the solver.

## 9.2 Development of a GUI/Web Interface for Interactive Input and Visualization

Currently, the simulation is executed through code with hardcoded values. A user-friendly **Graphical User Interface (GUI)** or a **web application** can be developed where users can input beam parameters, load conditions, and choose between linear/nonlinear methods. The outputs—such as deflection curves, comparison plots, and error analysis—can be visualized interactively, making this tool more accessible to students and engineers alike.

# References


- **Finite Difference Method for Solving Differential Equations** – beam-bending example

  PDF: https://nm.mathforcollege.com/mws/gen/08ode/mws_gen_ode_spe_finitedif.pdf

- **Comparative Analysis of the Deflections of Two Beams Using the Finite Difference Method**

  **(SSRN preprint):** https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3801459

- **Newton-Raphson Method for Non-Linear Finite Element Analysis (LearnFEA article):** https://learnfea.com/newton-raphson-method-for-nonlinear-fea/

- **Beam Deflection: Definition, Formula, and Examples (SkyCiv tutorial):** https://skyciv.com/docs/tutorials/beam-tutorials/what-is-deflection/

- **Plotly Python Library** – Getting Started Guide: https://plotly.com/python/getting-started/

- **Plotly Technologies Inc., *Collaborative Data Science* white paper (official citation guidance):** https://plotly.com/python/static/whitepaper.pdf

- pandas Documentation and Citation (with Zenodo DOI):

- **MathWorks, Inc. (2024). MATLAB (R2024a) [Software]. Natick, MA, USA:** The MathWorks, Inc. Available at https://www.mathworks.com

- **Perplexity AI. (2025). Perplexity – AI-powered search engine.**
  Perplexity AI is an advanced conversational search engine that synthesizes web results and was used as a research tool in this project
  https://www.perplexity.ai/

- **OpenAI. (2025). ChatGPT.**
  ChatGPT is a generative AI chatbot developed by OpenAI, utilized in this project for drafting, summarizing, and code assistance
  https://chat.openai.com/