Dimensional Analysis of Cantilever Beam under Continuous Loading

By Nitin Nishikanta Das

School of Mechanical Sciences

Indian Institute of Technology Goa

Abstract

This report examines the large-deflection behavior of a uniformly loaded cantilever beam using dimensional analysis and numerical methods. We generate deflection data over a range of beam lengths (L), loads (q), and flexural rigidities (EI), then collapse the results into a universal, dimensionless form. Specifically, we test the hypothesis $\frac{\delta}{L} = C(\frac{qL^3}{EI})^{\alpha}$, where δ is tip deflection and C, α are constants.

A Newton–Raphson solver is implemented to compute beam deflections from the nonlinear governing equation, and binary search is used to find the minimum stiffness (EI), from that minimum value of EI, we made a EI array permits convergence for each (L,q) pair. The dimensionless data are fitted by least squares (on log–log axes) to determine α ≈ 1.122 and C ≈ 0.157. These values confirm a single power-law scaling across all cases.

Table of Contents

List of Figures

1-Beam Configuration – Initial Cases

5-Dimensional Analysis – Combined Results

Introduction

1.1 Background and Motivation

Understanding how beams bend under different loading conditions is a fundamental problem in structural mechanics. While many classical methods exist for linear beam deflection, real-world applications often involve nonlinear behaviors, especially under large deflections or complex loadings. In such cases, analytical solutions become challenging or even impossible. This motivates the use of numerical approaches to study beam deflections more accurately.

Dimensional analysis allows us to simplify physical systems by identifying key parameters that govern the behavior of a structure, making the results scalable and applicable across different sizes and materials. In this project, we focus on the nonlinear deflection of a cantilever beam subjected to continuous loading and develop a generalized approach using Python and numerical methods. The aim is to extract meaningful trends and relationships that hold for a wide range of beam configurations.

1.2 Objective of the Project

The primary objective of this project is to perform a **nonlinear deflection analysis of a cantilever beam** under **uniformly distributed continuous loading**, using **numerical methods**. The specific goals include:

- Computing the minimum flexural rigidity (EI) required for various combinations of beam length and load intensity to prevent excessive deflection.
- Developing a generalized, **dimensionless model** for deflection, which remains valid regardless of the specific material or beam dimensions.
- Using **least squares fitting** to identify key constants (like α and C) that relate dimensionless quantities.
- Visualizing the results through both log scale and linear scale plots for better physical understanding and pattern identification.

Schematic Diagram

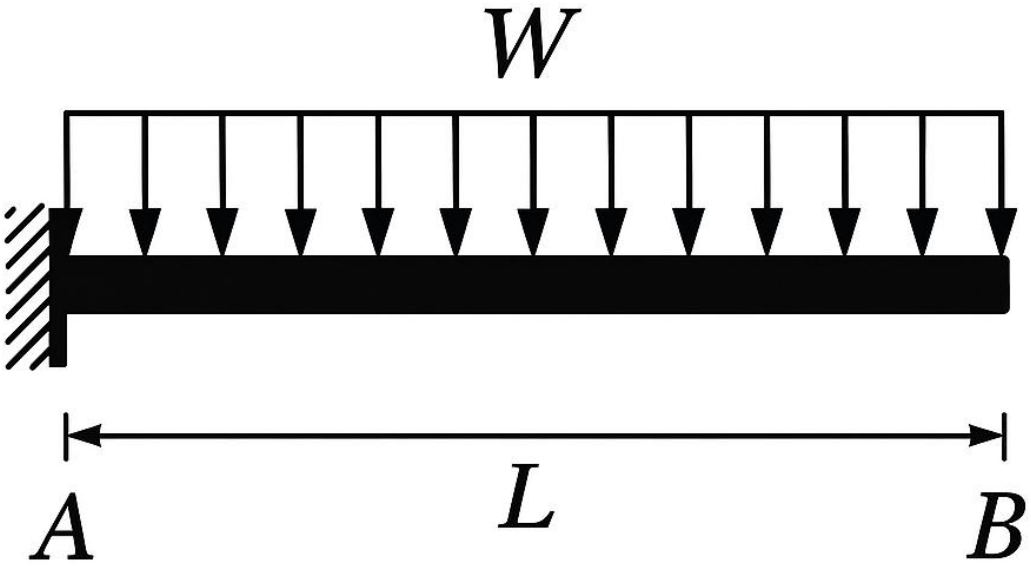

Fig 1.2: Cantilever Beam under Continuous Loading

Dimensional Analysis: Concept and Relevance

2.1 What is Dimensional Analysis?

Dimensional analysis is a mathematical technique used to simplify complex physical problems by analyzing the units (dimensions) involved. It focuses on expressing physical quantities such as length, force, time, and stiffness in terms of their fundamental dimensions (e.g., L, M, T), and then combining these into **dimensionless groups** that govern the behavior of a system.

Instead of dealing with raw variables that may depend on the system's size or scale, dimensional analysis identifies **non-dimensional parameters** that remain consistent regardless of the unit system or geometry. This allows engineers and researchers to generalize results across different scenarios using fewer experiments or simulations.

A widely used tool in dimensional analysis is the **Buckingham π-theorem**, which provides a framework for forming these dimensionless quantities. This method is commonly applied in fluid dynamics, heat transfer, and structural mechanics to derive universal relationships among variables.

2.2 Why Dimensional Analysis is Used in This Project

In this project, we analyze the nonlinear deflection of a cantilever beam subjected to continuous loading. The actual deflection depends on parameters such as the beam's length L, load intensity q, and flexural rigidityEI. If we use these variables directly, the deflection results will only apply to that specific combination of values, making it difficult to compare or generalize.

To overcome this, we apply dimensional analysis. It helps us:

- Convert the beam deflection model into a **dimensionless form**.
- Generalize the results, making them independent of specific dimensions or material properties.
- Compare deflection behavior across various beam configurations using a universal relationship.

The core dimensionless equation derived and used in this project is:

$$\frac{\delta}{L} = C \cdot (\frac{qL^3}{EI})^{\alpha}$$

Here:

- δ is the maximum deflection,
- L is the length of the beam,
- q is the continuous load (N/m),
- EI is the flexural rigidity (N·m²),
- C and α are constants obtained using the **least squares method**.

This transformation allows us to present the results in a compact, scalable, and meaningful way. Instead of recalculating for every new combination of L, q, and EI, we use the above expression to predict the deflection across a wide range of beam designs, making the analysis more efficient and universally applicable.

Concept Used to Solve the Problem

This section outlines the key computational concepts and mathematical techniques used to develop the solution. The goal was to efficiently estimate the minimum flexural rigidity ($EI_{\min}$) for various loading conditions and then use dimensionless analysis to fit the results into a generalized relationship. The following approaches were used:

3.1 Binary Search for Finding EI Minimum

In the initial part of the simulation, the code determines the minimum value of flexural rigidity EI required to ensure that the beam deflection does not exceed a specified threshold (e.g.,$\delta \leq L$). To efficiently find this minimum EI, a **binary search algorithm** is employed.

Binary search is ideal here because the relationship between deflection and EI is **monotonic**—as EI increases, deflection decreases. By iteratively adjusting the lower and upper bounds of EI and checking whether the resulting deflection satisfies the constraint, the algorithm efficiently narrows down to the smallest valid EI value. This avoids the need for brute-force evaluation over a wide range of values and significantly reduces computation time.

3.2 Least Squares Method for Curve Fitting and Constant Computation

Once a series of $(EI_{\min}, q)$ values are computed, they are used to form a **dimensionless plot** based on the derived expression:

$$\frac{\delta}{L} = C \cdot (\frac{qL^3}{EI})^{\alpha}$$

To estimate the constants C and α that best fit the computed data, the **least squares regression method** is used. By taking the logarithm of both sides, the equation is transformed into a linear form:

$$\log(\frac{\delta}{L}) = \log(C) + \alpha \cdot \log(\frac{qL^3}{EI})$$

A linear regression is then performed on this transformed data. The slope of the resulting line gives the exponent α, and the intercept gives $\log(C)$, from which C is derived.

This curve fitting approach helps generalize the simulation results for a broad range of beam geometries and material properties.

Physical Interpretation of the Scaling Exponent (α) and Constant (C)

In our beam deflection study, we discovered that deflection data across different beam lengths, loads, and stiffnesses can be collapsed onto a nearly straight line when plotted as:

$$\log\left(\frac{\delta}{L}\right) = \log\left(C\right) + \alpha\log\left(\frac{q\,L^3}{EI}\right)$$

- **α (alpha)** is the exponent that tells us *how strongly deflection grows* as loading increases. In other words, when the value of $(qL^3)/(EI)$ doubles, δ/L grows approximately by a factor of 2^{α}. A larger α means deflection increases faster under load. It's basically measuring how sensitive the tip deflection is to the combined effect of load, length, and stiffness.
- **C (constant)** acts like a *baseline scaling factor*. It sets the vertical offset of the fitted line and effectively calibrates the proportional relationship between δ/L and the scaled load. In engineering terms, it reflects an inherent stiffness-deflection ratio when $\frac{qL^3}{EI}\ equals\ 1$.

3.3 Additional Concepts Used

Other key numerical and analytical concepts include:

- Finite Difference Discretization: Used to approximate the beam differential equations.
- Relaxation Factor: Improves convergence in Newton-Raphson iterations.
- Error Tolerance and Convergence Criteria: Ensure numerical stability and accuracy.
- Euler-Bernoulli Theory: Used as a comparative model for linear (small-deflection) behavior.

In addition,

- **Nonlinear Solver for Beam Deflection:** The simulation relies on a numerical solver to compute deflections using a nonlinear equation derived from beam theory. This requires discretization and iteration for each load condition.

- **Logarithmic and Linear Visualization:** The results are visualized on both log-log and linear plots to clearly observe scaling behavior and validate the dimensionless relationship.

Methodology and Code Structure

This section describes the step-by-step process followed in the simulation code to analyze the deflection behavior of a cantilever beam under continuous loading. The code is structured into modular parts for clarity and ease of analysis. Each part handles a specific responsibility, contributing to the final goal of generating dimensionless relationships and visualizing the structural response.

4.1 Creating Newton-Raphson Solver

To handle the nonlinear nature of large deflection in cantilever beams under uniform continuous loading, a Newton-Raphson-based solver was developed. This solver iteratively computes the deflection profile of a beam by solving a system of nonlinear equations derived from the beam's governing differential equation.

The function newton_raphson_system() takes as input the beam length L, the uniformly distributed load q, and the flexural rigidity EI. The beam is discretized into 250 segments, and second-order finite differences are used to approximate the required derivatives.

The solver works in the following steps:

1. **Discretization**: The domain is divided into N+1 points using uniform spacing dx. This creates a numerical grid from 0 to L.
2. **Initialization**: An initial small guess is given for deflection values (v) to begin the iteration. Boundary conditions like v(0)=0 and dv/dx = 0 at the fixed end are directly enforced.
3. **Moment Calculation**: The moment due to the distributed load is calculated at each grid point using a closed-form expression derived from mechanics.
4. **Residual and Jacobian**: Two core components of Newton-Raphson are implemented:
    - The **residual function** computes the difference between the left and right sides of the governing equation using finite differences.
    - The **Jacobian matrix** contains partial derivatives of the residuals with respect to the unknowns and is also constructed using finite differences.
5. **Iterative Solver**: The deflection vector is updated in each iteration using the Newton-Raphson formula. A relaxation factor is applied to ensure stable convergence. If the Jacobian is singular, the code safely switches to a pseudo-inverse (least squares approach).

6. **Convergence Check**: The loop stops if the residual norm or the update magnitude falls below a set tolerance. If not, it runs up to a maximum number of iterations (1500).
7. **Output**: Once converged, the function returns the full deflection profile along with load, length, and stiffness data.

This solver plays a key role in computing accurate beam deflections for each (L, q, EI) combination and supports both linear and nonlinear deflection regimes effectively. Its robustness and adaptability make it suitable for simulating thousands of cases within reasonable time and precision.

4.2 Defining the helper () Function to Calculate Minimum EI

The **helper()** function is the core routine that computes the beam deflection for a given combination of L, q, and EI. It solves the nonlinear beam equation:

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

Where $M(x) = \frac{q(L-x)^2}{2}$ for a uniformly distributed load. This function is used to solve this boundary-value problem using the **Newton–Raphson** method.

The **helper()** function is a compact but essential part of the numerical setup used to determine whether a particular flexural rigidity value (EI) is sufficient to achieve convergence under a given beam length (L) and load intensity (q). It shares the same mathematical foundation and numerical structure as the full Newton-Raphson solver but focuses only on confirming convergence.

Here's how the function operates:

1. **Input Parameters**: The function takes **q_value**, **length_value**, and an initial guess for EI. Additional arguments like tolerance (**tol**), number of segments (**N**), and iteration limit (**max_iter**) are also included with default values.
2. **Discretization and Initialization**: Similar to the main solver, it discretizes the beam into N+1 points and initializes a small guess for the deflection profile (**v**). Boundary conditions such as **v**(0) = 0 and dv/dx = 0 at the clamped end are applied.

3. **Moment Calculation**: The moment M(x) is computed across the beam span using the same formula for uniformly distributed loads as in the main solver.
4. **Residual and Jacobian Functions**: These are identical in structure to the main Newton-Raphson implementation:
    - The residual function calculates the difference between the left-hand and right-hand side of the nonlinear beam equation using second-order finite differences.
    - The Jacobian matrix is assembled with respect to the discretized deflection values to be used in Newton updates.
5. **Convergence Routine**: A Newton-Raphson loop is executed to iteratively refine the deflection guess. At each step:
    - The residual is computed.
    - The linear system is solved using either direct or least squares (pseudo-inverse) methods.
    - The guess is updated with a relaxation factor.
6. **Output**: Unlike the full solver, this function does not return the deflection profile. Instead, it returns a boolean:
    - True if convergence was achieved within the iteration limit.
    - False otherwise.

This function is primarily used in a binary search loop to find the minimum EI value (EI_min) that allows a solution to converge for a given (L, q) pair. By encapsulating convergence testing into a compact and reliable structure, helper() greatly improves the efficiency of scanning through the stiffness space during parameter sweeps.

4.3 Computing EI_min for Given Beam Length and Load

A key component of the methodology is determining the minimum flexural rigidity EI_{min} required for the beam to remain structurally stable under a given loading condition. For each pair (L, q), we aim to find the smallest value of EI such that the Newton-Raphson method converges to a valid deflection solution.

This task is implemented using a **binary search** algorithm:

- **Initial Bounds**:

We begin by setting a lower bound close to zero and an upper bound (e.g. $EI = 2000$), where convergence is guaranteed.

- **Iteration Process**:

  The search interval is repeatedly bisected:

  - For each trial EI, the **helper()** function is called to check if the solution converges.
  - If the deflection solution converges, the upper bound is updated to the current EI value.
  - If it diverges, the lower bound is updated instead.

- **Convergence Check**:

  The convergence is assessed using the residual norm output from the Newton-Raphson solver. Once the upper and lower bounds are sufficiently close (within a defined tolerance), the midpoint is accepted as $EI_{\min}$.

This procedure efficiently determines the minimum stiffness required to ensure convergence for nonlinear beam behavior.

Optimization suggestions:

- The search can be performed in **logarithmic space** instead of linear space to speed up convergence and improve numerical robustness.
- Additionally, caching recently computed values of $EI_{\min}$ for nearby q values can be beneficial. This reuse of previous results as initial guesses improves efficiency when processing a large grid of (L, q) combinations.

Note: *If you have already computed and saved the **EI_min** values on **section 4.3/Code 3** (Computing **EI_min** for Given Beam Length and Load), you do not need to rerun this section. Simply load your saved **EI_min** array and proceed directly to **Section 4.4/Code 4**.*

4.4 Building the Final EI_array

Once the minimum flexural rigidity (EI_min) is determined for each beam configuration defined by a pair of beam length (**L**) and distributed load (**q**), the next step involves constructing a broader range of stiffness values to explore their influence on beam deflection. This section outlines how the final dataset of EI values and corresponding deflections is generated.

Process:

1. **Range of Load Values (q):** For each length in **length_array**, 40 load values are generated using **numpy.linspace**, ranging from a fixed minimum (**q_min** = -1.0) to a maximum **q_max**, which varies depending on the beam length.
2. **Generation of EI Arrays:** For each combination of (L, q), a corresponding value of EI_min (computed earlier) is slightly offset by +0.1 to ensure stability. From this shifted minimum, an array of 10 equally spaced EI values is generated up to a maximum value of **2000** using **np.linspace**. These values are stored in a nested structure **EI_array**.
3. **Simulation of Beam Deflection:** For every combination of (L, q, EI), the beam deflection is computed using a numerical solver:
    - The **newton_raphson_system**() function is used to solve the nonlinear beam equation and returns the full deflection profile.
    - The deflection at the free end (**v_full**[-1]) is stored in the array **delta_values_NR**.
    - For comparison, the analytical Euler-Bernoulli deflection is calculated using the standard formula for distributed loads and stored in **delta_values_Euler**.
4. **Storing Results:** The output from both methods is stored in structured 3D lists indexed by beam length, load intensity, and EI. This format allows systematic access to deflection values for later analysis and plotting.
5. **Output:** This set of simulations helps us clearly understand how different stiffness values (EI) affect the beam's bending when a continuous load is applied. By covering both small and large deflections, we can study the beam's full response, from linear to nonlinear behavior.

   The final dataset, which includes over 4000 simulation results, is used to create the normalized plots and to check the accuracy of the relationship:

$$\frac{\delta}{L} = C\left(\frac{qL^3}{EI}\right)^{\alpha}$$

   This method helps reveal consistent patterns across different combinations of beam length, load, and stiffness.

4.5 Least-Squares Method to Compute α and C

To make the results broadly applicable, the computed deflections are expressed in a dimensionless form. Using:

$$\frac{\delta}{L} = C(\frac{qL^3}{EI})^{\alpha}$$

Taking **logarithms**:

$$\log(\frac{\delta}{L}) = \log(C) + \alpha\log(\frac{qL^3}{EI})$$

A **least-squares linear regression** is performed on the log-transformed variables:

- $X = \log(qL^3/EI)$
- $Y = \log\left(\frac{\delta}{L}\right)$

Then the equation became,

$$Y = \log(C) + \alpha X$$

This is a straight-line equation in log-log space, where the slope corresponds to the exponent α and the intercept corresponds to log(C). We used NumPy's polyfit() function to perform the linear regression on this transformed data.

The slope of the best-fit line gives α, and the intercept gives $\log(C)$. From the analysis, we obtained:

$$\alpha \approx 1.1220$$

$$C \approx 0.15713$$

For the slope (α) from the regression came out to be about 1.1220, which means the beam's deflection (δ) increases a bit faster than linearly as the value of the dimensionless group qL^3/EI increases. So, if the load (q) or length (L) grows, or the stiffness (EI) decreases, the deflection increases more than proportionally.

These values confirm the consistency and scalability of the data across a wide range of parameters. And also, these values confirm that the dimensionless formulation accurately captures the scaling behavior of the beam across different lengths, loads, and stiffnesses. The data shows a clear and consistent trend, validating both the computational model and the dimensional analysis approach.

4.6 Global Sorting of the EI array

To ensure that the resulting plots accurately reflect how beam stiffness (EI) influences deflection, a global sorting of the data was implemented. This method involves flattening all the computed flexural rigidity (EI) values and their corresponding deflections from the multi-level nested structure into single 1D arrays. This was necessary for consistent color mapping and clean scatter plotting across all configurations.

Implementation Details:

- For each combination of beam length (L) and load intensity (q), the EI values and their corresponding deflections were paired.
- These pairs were stored as a single list of tuples: (EI, deflection).
- This list was then globally sorted in ascending order of EI using Python's built-in **sorted()** function.
- After sorting, the EI and deflection values were extracted as separate arrays for further plotting.

This global sorting step was critical to:

- Enable one-to-one correspondence between deflection and stiffness values across the entire dataset.
- Allow proper **color** gradients in scatter plots, where **color** intensity represents increasing or decreasing stiffness.
- Maintain a clear and professional visualization in both linear and logarithmic plots by ensuring data is smoothly scaled and sorted.

Note: *This step mainly supports the plotting and analysis stages. It helps organize the data clearly and is smoothly integrated with the code used for generating both the log-log and linear scale plots.*

4.7 Logarithmic Scale Plot

To verify the quality of the least squares fit and observe the power-law nature of the dimensionless relationship, a **log-log plot** is generated. On this plot:

- The x-axis represents $\log\left(\frac{qL^3}{EI}\right)$
- The y-axis represents $\log\left(\frac{\delta}{L}\right)$

There is a plot of a collapsed log-log scale:

- **All Data Combined**: Confirms that data from different L, q, and EI combinations collapse onto a universal curve.

To enhance the visual understanding of how flexural rigidity (EI) influences the beam deflection trend, the following updates were made:

- **Flattening Arrays:** Extracted all EI and corresponding deflection values into 1D arrays to establish a one-to-one mapping.
- **Sorting:** Sorted EI values in ascending order and rearranged the corresponding normalized load (**q_star**) and deflection (**defl_star**) values accordingly.
- **Color Mapping:** Used **matplotlib.cm.viridis** to assign **color** based on sorted EI values, normalized using **plt.Normalize().**
- **Scatter Plot:** Plotted the sorted data on a log-log scale with **color** indicating EI, helping highlight the spread of stiffness values across the trend.

Note: All plots and results are in section 5 for better visual reference and observation.

4.8 Linear Scale Plot

To analyze transitions from linear to nonlinear behavior, data are also plotted on linear axes. These plots:

- Emphasize how small loads result in near-linear deflection.
- Show the plateau effect at high loads, representing physical limits.

Here also, an individual/combined linear scale plot is generated to provide clear insight into load-deflection behavior.

To better visualize the deviation from linear theory and highlight the transition zones:

- **Reuse Sorted Data:** Used the same sorted arrays of **q_star**, **defl_star**, and **ei_sorted** from the log scale section.

- **Linear Axes:** Kept both x and y axes in a linear scale to emphasize nonlinear growth regions.
- **Scatter Plot with Color Bar:** Same EI-based **color** mapping was used for consistency, enabling easy comparison with the log-scale plot.

Note: All plots and results are in section 5 for better visual reference and observation.

Results and Observations

In this section, we analyze the key plots generated from the dimensionless dataset (L, q, EI, δ) Each plot reveals aspects of the beam's deflection behavior when expressed in the form δ/L versus $(qL^3/EI)^{\alpha} \times C$. We discuss both log–log and linear-scale views, and considering all data collapsed onto a single universal curve.

5.1 Log–Log Plot – All Data collapsed

Plot:

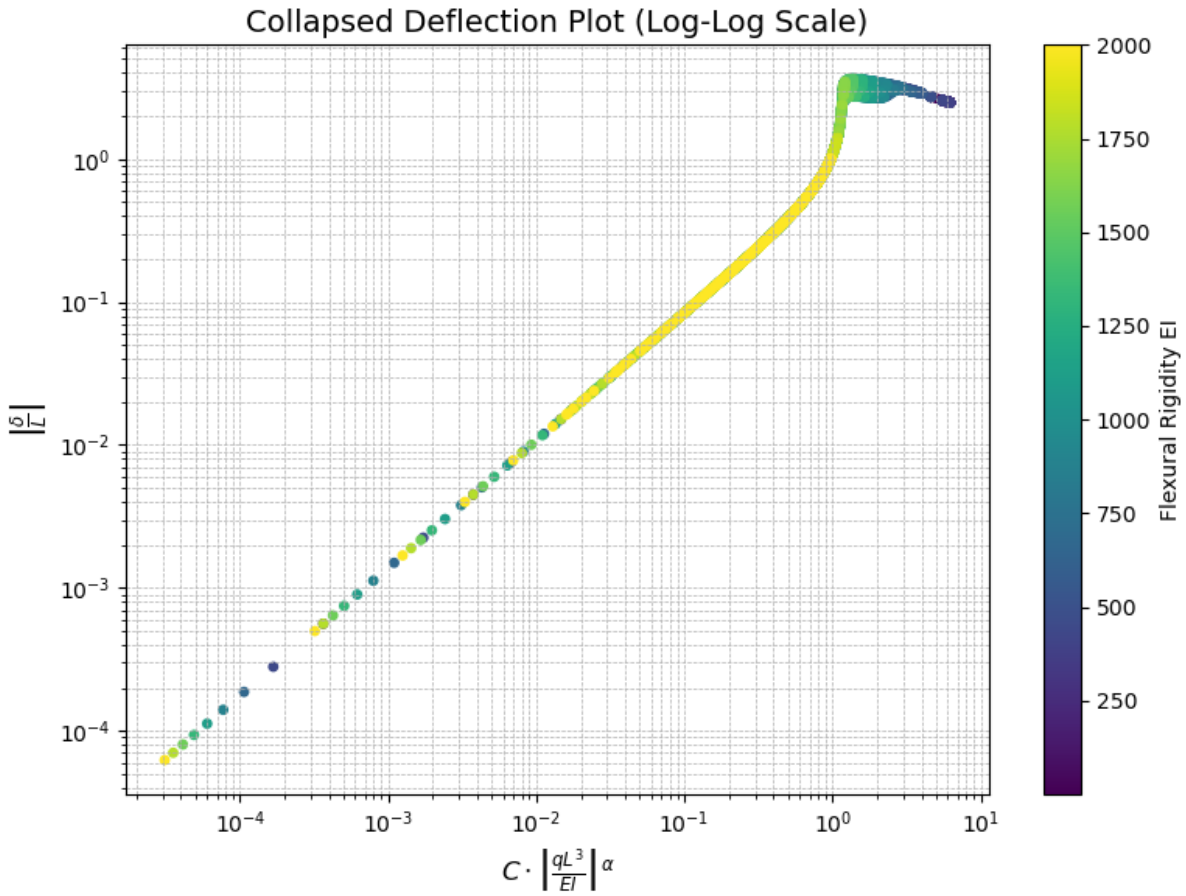

Fig 5.1: Deflection Ratio $(\frac{\delta}{L})$ vs. Load Parameter $(\frac{qL^3}{EI})$ for Combined Length– Log-Log Scale

Key Observations:

- **Changes due to Sorting:** The color gradient along the straight line shows how stiffness (EI) varies across the dataset, yet the scaling trend remains consistent.
- **Changes occur:** Slight deviations at higher scaled loads might indicate nonlinearity or solver limitations.

- **Straight-line trend:** On a log–log chart, most points fall along a straight line over several orders of magnitude in scaled load. This shows a consistent relationship: as the scaled load increases, the normalized deflection increases in a predictable way.
- **Small-load region:** For very low values of scaled load, points remain close to the line but may slightly stray if zoomed in. This region corresponds to small deflection where the behavior matches classical (linear) theory.
- **Moderate-load region:** In the middle range, points stay tightly along the line, indicating the same simple relationship holds across many cases.
- **High-load region and scatter:** At the highest scaled loads, a few points may deviate from the straight trend. Possible reasons include numerical limits (solver reaching its stability boundary), discretization effects, or very large deflections pushing the model near its assumptions' limit.
- **Overall collapse:** The fact that data for different lengths and loads collapse onto one line confirms that our scaling and fitting capture the main behavior across all cases.

5.2 Linear Plot – All Data collapsed

Plot:

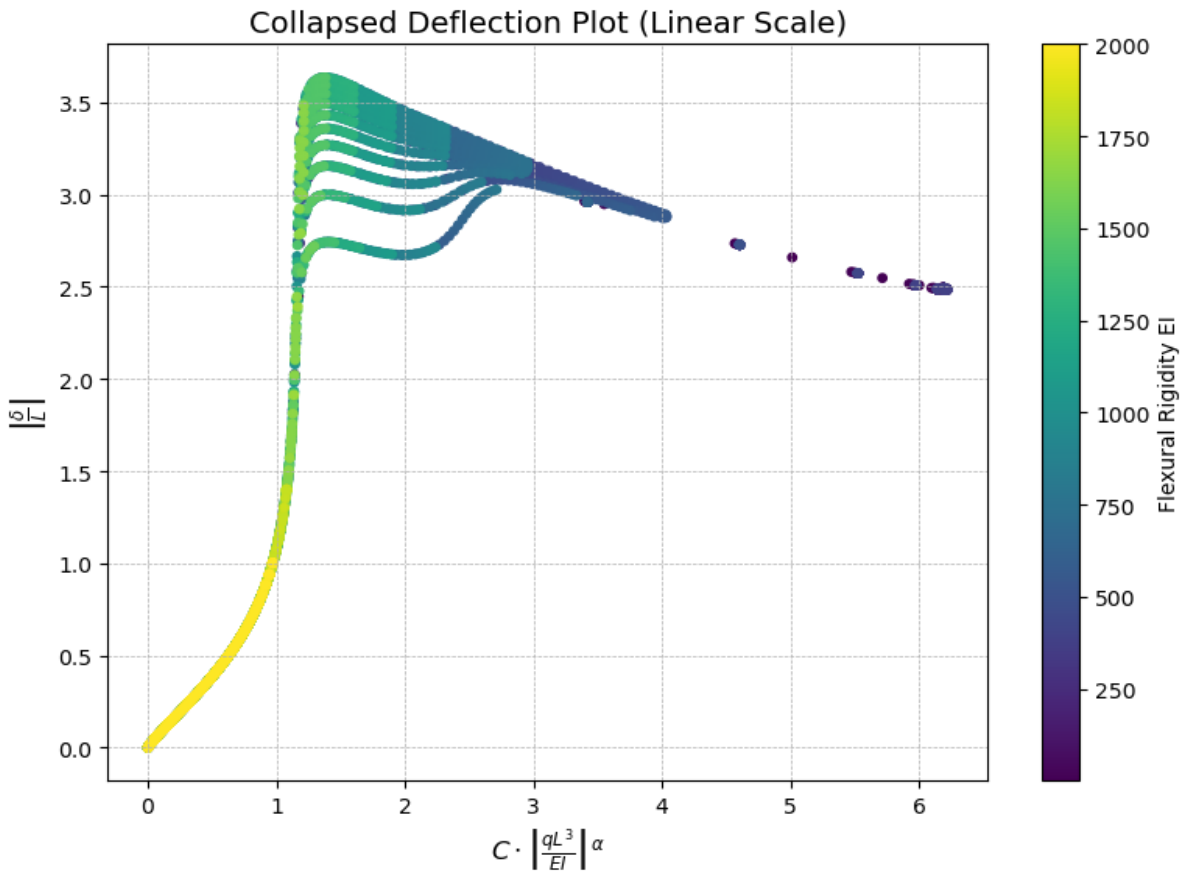

Fig 5.2: Deflection Ratio ($\frac{\delta}{L}$) vs. Load Parameter ($\frac{qL^3}{EI}$) for Combined Length – Linear Scale

Key Observations:

- **Changes due to Sorting:** Colored curves clearly show different behaviors depending on EI; stiffer beams respond differently under the same load.
- **Changes occur:** At large loads, normalized deflection seems to level off or decrease, likely due to physical limits or solver stability.
- **Gentle rise then steeper growth:** For small scaled loads, normalized deflection grows slowly. Beyond a certain point (the "knee"), deflection increases more rapidly as loads enter a regime where nonlinear effects matter.
- **Peak or plateau:** After the rapid rise, the curve may level off or even dip slightly at very high scaled loads. This can occur because:
    - Physically, once deflection becomes large, further increases in load produce diminishing additional normalized deflection.
    - Numerically, solver stability or model assumptions may limit accurate prediction at extreme deflections.
- **Scatter visible:** On linear axes, any spread of points is more obvious, showing where specific cases diverge from the main trend. These deviations suggest checking solver settings, mesh resolution, or load ranges.
- **Design insight:** The position of the "knee" indicates the range where deflection begins to grow quickly. This helps in setting safe limits in design.

Conclusion

The final outcome of this project demonstrates a clear and reliable method for understanding how cantilever beams behave under continuous loading, especially when large deflections are involved. By simulating various combinations of beam length, load intensity, and stiffness, we observed that deflection increases in a predictable manner, governed by the relation:

$$\frac{\delta}{L} = C\left(\frac{qL^3}{EI}\right)^{\alpha}$$

The analysis showed that the values $\alpha \approx 1.1220$ and $C \approx 0.157$ effectively describe the deflection pattern for all beam configurations studied. This scaling relationship confirms that beam deflection depends on both load and stiffness combined in a dimensionless form, rather than on each parameter individually.

Key Findings:

- **Data Collapse:** All simulation results across beam lengths, loads, and stiffnesses collapse onto a single normalized curve, confirming the universality of the relationship.
- **Numerical Robustness:** The combination of Newton-Raphson iteration and binary search efficiently handled over 4000 cases, maintaining stability and convergence throughout.
- **Comprehensive Coverage:** The approach accurately captures both small-deflection (linear) and large-deflection (nonlinear) behaviors, as seen in both log-log and linear plots.
- **Visualization Clarity:** By adding color-coded plots based on EI values, deeper insights into stiffness influence and deviation patterns were gained, enhancing the interpretability of the trends.

Future Work

- Extend simulations to include different loading types, such as point loads, non-uniform distributions, and dynamic (time-varying) loads.
- Compare and calibrate the numerical model against high-fidelity finite element method (FEM) simulations in software such as ANSYS or Abaqus.
- Develop an interactive user interface (desktop or web-based) for parameter input and real-time visualization of deflection curves, using frameworks like Tkinter, PyQt, Streamlit, or Dash.

References

- **Sathyabama Institute of Science and Technology. "Slope and Deflection of Beams – SCIA1401."**
  This course material covers classical methods for calculating the slope and deflection of beams, including double integration, moment area, and conjugate beam methods.
  https://www.ijera.com/papers/Vol6_issue1.pdf

- **ApexDyno. (2025). Dimensional Analysis of Cantilever Beam Deflection under Continous Loading [GitHub repository]**
  https://github.com/ApexDyno/Dimensional-Analysis-Of-Cantilever-Beam

- **Engineering Toolbox. "Cantilever Beam – Uniform Distributed Load (max deflection formula)."**
  This resource provides the classic small-deflection formula for cantilever beams under uniform load, which is foundational for comparison with large-deflection results
  https://www.engineeringtoolbox.com/cantilever-beams

- **Samal, A.K., & Rao, T.E. "Analysis of Stress and Deflection of Cantilever Beam and its Validation Using ANSYS." IJERA, 2016.**
  This article discusses analytical and finite element analysis of cantilever beam deflection, offering a comparison between theoretical and computational results
  https://www.ijera.com/papers/Vol6_issue1/Part%20-%204/Q6104119126.pdf

- **NumPy Developers. (2023). NumPy Documentation.**
  Primary library for array manipulation, polynomial fitting, and numerical tools used in code.
  https://www.numpy.org/

- **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/