IIT Kanpur

Department of Mechanical Engineering

ME685 : Applied Numerical Method

Semester : 2024-2025 II

Instructor : Dr. Abhishek Sarkar

Project Report

Date : 13 April 2025

Group Members

Adarsh Sharma (210046)
Amrit Kalash (218070126)
Chinmay Pratap (218070287)
Shreyas Mirkale (211008)
Vibhansh Bhatia (211161)

ME685 – Applied Numerical Method
Major Project Report

Instructor: Dr. Abhishek Sarkar

Contents

List of Tables

List of Figures

Abstract

In this study, the free vibration characteristics of Euler–Bernoulli beams with non-ideal boundary conditions are investigated. The non-ideal boundary condition model is represented as a linear combination of ideal simply supported and ideal clamped conditions, resulting in nonlinear rational functions that relate natural frequencies to boundary condition weighting factors. While natural frequencies are traditionally obtained numerically through standard root-finding algorithms with appropriate initial guesses, we propose an alternative approach to enhance accuracy and efficiency. Specifically, Padé approximants are employed to approximate the nonlinear functions, yielding compact analytical expressions for the natural frequencies as functions of the weighting parameters. To validate and complement this approach, we solve the governing differential equations using the Runge-Kutta method and other analytical and numerical techniques. The results for cantilever beams and beams clamped at both ends with non-ideal boundary conditions are compared across methods. Our findings demonstrate that the Padé-based approximations are sufficiently accurate—achieving up to two-digit precision—and serve as reliable initial guesses for root-finding algorithms, thereby eliminating ambiguity in their application. The comparative analysis confirms the robustness and versatility of the proposed methodology in practical vibration analysis scenarios.

1 Introduction

1.1 Introduction to Non-Ideal Boundary Conditions

In the problems of mechanical systems, boundary conditions are usually represented in the idealized forms such as clamped, simply supported, and free boundary conditions. However, deviations from the ideal boundary conditions may exist and the ideal boundary condition assumptions sometimes lead to incorrect solutions. The types of boundary conditions that deviate from the ideal boundary conditions are referred to as the non-ideal boundary conditions.

The uncertain boundary conditions of imperfectly clamped joints have been modeled using fuzzy parameters with assumed functions. Perturbation theory has also been used to model the non-ideal boundary conditions in the free vibration analysis of beams. Other studies investigated the vibration of rectangular plates with non-ideal boundary conditions.

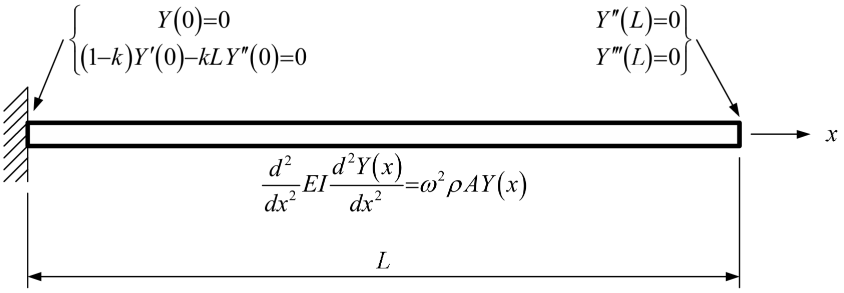

Figure 1: Schemetic diagram

A two spring model, where a transverse spring and a rotational spring are attached at a boundary, has often been used to simulate the non-ideal boundary conditions. In the two spring model it was considered ideally clamped if both the transverse and the rotational spring had infinite stiffness, and ideally simply supported if the transverse spring had infinite stiffness while the rotational spring had zero stiffness. Otherwise, the boundary was considered to have non-ideal boundary conditions.

The two spring model has been applied to the parameter estimation of Euler-Bernoulli beams with damages and defective boundary conditions. Chebyshev polynomials were employed to carry out the free vibration analysis of a beam, where the defective boundary conditions were modeled by using the two spring model. The boundary conditions of a beam have also been modeled using the two spring model. The connection between the flexibility and boundary conditions was construed based on the static equilibrium equation, and a set of equations for detecting the boundary conditions was formed using the flexibility measurements.

1.2 Padé Approximation

Padé approximation is one of the most powerful tools for improving the convergence of truncated Taylor series that represents a function. Padé approximant of the type (M, N) is defined by

$$P_M^N(z) = \frac{p(z)}{q(z)} = \frac{\sum_{n=0}^{N} a_n z^n}{\sum_{m=0}^{M} b_m z^m}$$

where the coefficients of the numerator polynomial p and denominator polynomial q are computed such that the expansion P_M^N matches the power series expansion.

$$T(z) = \sum_{n=0}^{\infty} c_n z^n \tag{1}$$

around zero to as high order as possible. Due to their natural forms as rational functions, Padé approximants are in most cases superior to Taylor series when the approximated functions are rationals or contain poles. Extensive background on this subject shows that these approximants are useful in many numerical algorithms. For example, they have been used to efficiently and quickly compute finite difference weights on equally spaced and arbitrary grids. Algorithms based on Padé approximation have also been used for overcoming the Gibbs phenomenon. Computational techniques for robust Padé approximations using SVD have also been explored.

Padé approximants have been adopted to estimate the natural frequencies and damping ratio of vibrating systems, in which output data in time domain were utilized. They have also been used to obtain analytical solutions for nonlinear differential equations arising in modelling the dynamic behavior of oscillators with inertia and stiffness nonlinearities. Solutions for oscillation of nonlinear systems modeled by a mass attached to a stretched wire have been derived using homotopy method and Padé approximation. Multi-frequency approaches based on Padé approximants have also been proposed for improving the reconstruction of the frequency response functions.

In previous work, a non-ideal boundary condition model was proposed where the boundary conditions were represented by a linear combination of the ideal clamped boundary conditions and the ideal simply supported boundary conditions with weighting factors, and its effect on the natural frequencies of beams were discussed. The non-ideal boundary condition model has also been used to analyze the vibration of axially moving beams and functionally graded beams, respectively.

In this paper, the nonlinear rational functions are proposed by the non-ideal boundary condition model for approximating the vibration responses of the beam.

1.3 Non-ideal Clamped Boundary Conditions

The equation of motion of uniform beams based on the Euler-Bernoulli theory is found in most textbooks on vibration as follows:

$$\frac{d^4Y(x)}{dx^4} - \beta^4 Y(x) = 0, \quad \beta^4 = \frac{\rho A \omega^2}{EI} \quad (0 \leq x \leq L). \tag{2}$$

Here, ρ, A, ω, E, I and Y are the density, the cross-sectional area, the natural frequency in radians per second, Young's modulus, the second moment of area, and the lateral deflection, respectively.

Traditionally the ideal boundary conditions at the ends of the beam x_b are described as:

$$\text{Clamped:} \quad Y(x_b) = 0, \quad Y'(x_b) = 0 \tag{3a}$$
$$\text{Simply supported:} \quad Y(x_b) = 0, \quad Y''(x_b) = 0 \tag{3b}$$
$$\text{Free:} \quad Y''(x_b) = 0, \quad Y'''(x_b) = 0 \tag{3c}$$

In the previous work, the non-ideal clamped boundary condition model was proposed as a linear combination of the ideal simply supported and ideal clamped boundary conditions:

$$Y(x_b) = 0, \quad kLY''(x_b) \pm (1-k)Y'(x_b) = 0, \quad (0 \leq k \leq 1) \tag{4}$$

where k and $1-k$ are the weighting factors for the simply supported and the clamped boundary conditions, respectively. The sign convention in Eq. (4) is different for the left end and the right end of the beam, and the non-ideal clamped boundary conditions are rewritten as:

$$\begin{cases} Y = 0, \quad k_L LY'' - (1-k_L)Y' = 0 & \text{at } x_b = 0 \\ Y = 0, \quad k_R LY'' + (1-k_R)Y' = 0 & \text{at } x_b = L \end{cases} \tag{5}$$

The subscripts in k_L and k_R refer to the left end and right end of the beam, respectively.

1.4 Problem Statement

This study addresses the mathematical modeling of cantilever beams with non-ideal clamps. The non-ideal nature of the clamp is represented by a parameter k and $1-k$, which are the the weighting factors for the simply sup ported and the clamped boundary conditions.. We aim to:

- Derive the governing equations and boundary conditions for this system
- Establish the relationship between the weight factor k and the eigenvalue βL
- Develop efficient approximations using Padé fractions

- Implement numerical methods to solve the differential equations
- Verify analytical solutions through numerical simulations
- Analyze the mode shapes and their dependencies on the weight factor

1.5 Significance

Understanding the effects of non-ideal clamping conditions is crucial for:

- Accurate prediction of natural frequencies and mode shapes
- Proper design of vibration control systems
- Reliable structural health monitoring
- Realistic modeling of micro-cantilevers in atomic force microscopy
- Refinement of finite element models for complex structures
- Development of more accurate sensing and actuation devices

2 Theoretical Framework

2.1 Euler-Bernoulli Beam Theory

The Euler-Bernoulli beam theory provides the foundation for analyzing slender beams where shear deformation is negligible. The governing differential equation for the transverse displacement $y(x)$ of the beam is:

$$EI\frac{d^4y}{dx^4} = \rho A\omega^2 y \tag{1}$$

where E is Young's modulus, I is the area moment of inertia, and $\rho A\omega^2$ represents the inertial term. For free vibration analysis, this equation transforms into:

$$\frac{d^4y}{dx^4} - \frac{\rho A\omega^2}{EI}y = 0 \tag{2}$$

Substituting the $\beta^4 = \frac{\rho A\omega^2}{EI}$, we obtain:

$$\frac{d^4y}{dx^4} - \beta^4 y = 0 \tag{3}$$

or, equivalently:

$$\frac{d^4y}{dx^4} = \beta^4 y \tag{4}$$

2.2 General Solution

The general solution to the fourth-order differential equation is:

$$y(x) = C_1\sin(\beta x) + C_2\cos(\beta x) + C_3\sinh(\beta x) + C_4\cosh(\beta x) \tag{5}$$

where C_1, C_2, C_3, and C_4 are constants determined by the boundary conditions.

2.3 Boundary Conditions for Non-Ideal Boundary conditions

For a cantilever beam of length L with a non-ideal clamp at $x = 0$ and a free end at $x = L$, the boundary conditions are:

$$y(0) = 0 \quad \text{(No displacement at the support)} \tag{6}$$
$$EI\frac{d^2y}{dx^2}(0) = k\frac{dy}{dx}(0) \quad \text{(Moment-rotation relationship at support)} \tag{7}$$
$$\frac{d^2y}{dx^2}(L) = 0 \quad \text{(No bending moment at free end)} \tag{8}$$
$$\frac{d^3y}{dx^3}(L) = 0 \quad \text{(No shear force at free end)} \tag{9}$$

where k is the rotational stiffness parameter characterizing the non-ideal clamp. When $k = 0$, the boundary condition at $x = 0$ becomes $\frac{dy}{dx}(0) = 0$, representing a perfectly clamped beam. As k increases, the support becomes more flexible, allowing some rotation.

3 Mathematical Formulation for cantilevar beams

3.1 Characteristic Equation

Applying the boundary conditions to the general solution yields a system of equations. For non-trivial solutions to exist, the determinant of the coefficient matrix must be zero, leading to the characteristic equation.

Starting with the general solution:

$$y(x) = C_1 \sin(\beta x) + C_2 \cos(\beta x) + C_3 \sinh(\beta x) + C_4 \cosh(\beta x) \tag{10}$$

The derivatives are:

$$y'(x) = \beta C_1 \cos(\beta x) - \beta C_2 \sin(\beta x) + \beta C_3 \cosh(\beta x) + \beta C_4 \sinh(\beta x) \tag{11}$$
$$y''(x) = -\beta^2 C_1 \sin(\beta x) - \beta^2 C_2 \cos(\beta x) + \beta^2 C_3 \sinh(\beta x) + \beta^2 C_4 \cosh(\beta x) \tag{12}$$
$$y'''(x) = -\beta^3 C_1 \cos(\beta x) + \beta^3 C_2 \sin(\beta x) + \beta^3 C_3 \cosh(\beta x) + \beta^3 C_4 \sinh(\beta x) \tag{13}$$

Applying the boundary condition $y(0) = 0$:

$$C_2 + C_4 = 0 \implies C_4 = -C_2 \tag{14}$$

For the non-ideal clamp condition $EIy''(0) = ky'(0)$:

$$EI(-\beta^2 C_2 + \beta^2 C_4) = k(\beta C_1 + \beta C_3) \tag{15}$$
$$EI\beta^2(-C_2 - C_2) = k\beta(C_1 + C_3) \tag{16}$$
$$-2EI\beta^2 C_2 = k\beta(C_1 + C_3) \tag{17}$$

The boundary condition $y''(L) = 0$ gives:

$$-\beta^2 C_1 \sin(\beta L) - \beta^2 C_2 \cos(\beta L) + \beta^2 C_3 \sinh(\beta L) + \beta^2 C_4 \cosh(\beta L) = 0 \tag{18}$$

And $y'''(L) = 0$ yields:

$$-\beta^3 C_1 \cos(\beta L) + \beta^3 C_2 \sin(\beta L) + \beta^3 C_3 \cosh(\beta L) + \beta^3 C_4 \sinh(\beta L) = 0 \tag{19}$$

After substituting $C_4 = -C_2$ and rearranging, we get:

$$\frac{\cos(\beta L)\cosh(\beta L) + 1}{-\beta L\cos(\beta L)\sinh(\beta L) + \cos(\beta L)\cosh(\beta L) + \beta L\sin(\beta L)\cosh(\beta L) + 1} = k \tag{20}$$

This is the characteristic equation that relates the clamp parameter k to the eigenvalue βL.

3.2 Special Cases

3.2.1 Ideal Clamp ($k = 0$)

When $k = 0$, the characteristic equation reduces to:

$$\cos(\beta L)\cosh(\beta L) + 1 = 0 \tag{21}$$

This is the well-known characteristic equation for a cantilever beam with an ideal clamp, which has roots $\beta_1 L \approx 1.875$, $\beta_2 L \approx 4.694$, $\beta_3 L \approx 7.855$, etc.

3.2.2 Simply Supported End ($k \to 1$)

As k approaches infinity, the behavior approaches that of a simply supported end, and the characteristic equation approaches:

$$\sin(\beta L) = 0 \tag{22}$$

with roots $\beta_n L = n\pi$ for positive integers n.

3.3 Padé Approximation

For efficient computation and to capture the relationship between k and βL, we use a Padé approximation:

$$\beta L(k) \approx \frac{a_0 + a_1 k}{b_0 + b_1 k} \tag{23}$$

where a_0, a_1, b_0, and b_1 are constants determined through numerical fitting for each mode. This rational function approximation is particularly effective for modeling the relationship between k and βL, especially when dealing with multiple modes.

4 Methodology

1. **Plotting k vs. βL:** Computed and plotted the relationship between the stiffness parameter k and βL for a cantilever beam with a non-ideal clamp.

2. **Determining β Values:**

   - **Root Finding:** For selected k values, βL roots were obtained using the Newton-Raphson method.
   - **Padé Approximation:** Estimated β values as a function of k using nonlinear least squares fitting of a Padé approximant.

3. **Numerical Integration of Beam Equation:** Integrated the fourth-order Euler-Bernoulli differential equation using the Runge-Kutta (RK4) method to obtain the displacement profile.

4. **Validation via Analytical Solution:** Solved the homogeneous form of the beam equation with boundary conditions using singular value decomposition (SVD) to validate the numerical results.

5 Numerical Methods

5.1 Root Finding Algorithm

To find the roots of the characteristic equation for various values of k, we implement the Newton-Raphson method, which is an iterative technique for finding increasingly accurate approximations to the roots of a real-valued function.

Algorithm 1 Newton-Raphson Method for Root Finding

procedure NewtonRoot(f, x_0)
  $tol \leftarrow 10^{-8}$
  $maxIter \leftarrow 100$
  $h \leftarrow 10^{-6}$ ▷ Step size for numerical derivative
  $x \leftarrow x_0$
  for $iter = 1$ to $maxIter$ **do**
    $f_x \leftarrow f(x)$
    $df_x \leftarrow \frac{f(x+h)-f(x-h)}{2h}$ ▷ Central difference approximation
    if $|df_x| < 10^{-12}$ **then**
      return NaN ▷ Avoid division by zero
    end if
    $x_{new} \leftarrow x - \frac{f_x}{df_x}$
    if $|x_{new} - x| < tol$ **then**
      return x_{new}
    end if
    $x \leftarrow x_{new}$
  end for
  return NaN ▷ No convergence
end procedure

5.2 Runge-Kutta Method for Solving ODEs

To numerically solve the fourth-order ODE, we first convert it to a system of first-order ODEs:

$$Y_1 = y \tag{24}$$
$$Y_2 = \frac{dy}{dx} \tag{25}$$
$$Y_3 = \frac{d^2y}{dx^2} \tag{26}$$
$$Y_4 = \frac{d^3y}{dx^3} \tag{27}$$

The system becomes:

$$\frac{dY_1}{dx} = Y_2 \tag{28}$$
$$\frac{dY_2}{dx} = Y_3 \tag{29}$$
$$\frac{dY_3}{dx} = Y_4 \tag{30}$$
$$\frac{dY_4}{dx} = \beta^4 Y_1 \tag{31}$$

We then apply the fourth-order Runge-Kutta method to solve this system:

Algorithm 2 RK4 Method for Solving the Beam ODE

procedure RK4SOLVE(β, L, N)
  $dx \leftarrow L/N$
  $x \leftarrow \text{linspace}(0, L, N+1)$
  $Y \leftarrow \text{zeros}(4, N+1)$
  $Y(:,1) \leftarrow [0;0;1;0]$ ▷ Initial conditions
  for $i = 1$ to N **do**
    $k_1 \leftarrow dx \cdot \text{rhs}(Y(:,i))$
    $k_2 \leftarrow dx \cdot \text{rhs}(Y(:,i) + 0.5 \cdot k_1)$
    $k_3 \leftarrow dx \cdot \text{rhs}(Y(:,i) + 0.5 \cdot k_2)$
    $k_4 \leftarrow dx \cdot \text{rhs}(Y(:,i) + k_3)$
    $Y(:,i+1) \leftarrow Y(:,i) + (k_1 + 2k_2 + 2k_3 + k_4)/6$
  end for
  return Y
end procedure

function RHS(Y)
  return $[Y_2; Y_3; Y_4; \beta^4 \cdot Y_1]$
end function

5.3 Singular Value Decomposition for Boundary Value Problems

To find the non-trivial solution to the homogeneous system resulting from the boundary conditions, we use Singular Value Decomposition (SVD). This approach is particularly effective when dealing with systems where the coefficient matrix might be close to singular.

Given the boundary conditions matrix A and the constraint $A \cdot C = 0$, where C is the vector of constants $[C_1, C_2, C_3, C_4]^T$, we compute the SVD of A:

$$A = U \Sigma V^T \tag{32}$$

The solution vector C corresponds to the right singular vector associated with the smallest singular value (the last column of V).

5.4 Least Squares Method for Solution Verification

To verify the analytical solution against the numerical solution, we employ the least squares method to find an optimal scaling factor:

$$\text{scale\_factor} = \frac{y_{\text{data}}^T \cdot y_{\text{analytical}}}{y_{\text{analytical}}^T \cdot y_{\text{analytical}}} \tag{33}$$

where y_{data} is the numerical solution obtained through RK4 integration, and $y_{\text{analytical}}$ is the analytical solution constructed using the constants found via SVD.

The quality of the fit is assessed using the coefficient of determination (R^2):

$$R^2 = 1 - \frac{\sum_{i=1}^{N+1} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{N+1} (y_i - \bar{y})^2} \tag{34}$$

where y_i are the observed values (numerical solution), $\hat{y}_i$ are the predicted values (scaled analytical solution), and $\bar{y}$ is the mean of the observed values.

6 Results and Analysis

6.1 Relationship Between k and βL

The characteristic equation establishes a relationship between the clamp parameter k and the eigenvalue βL. Figure 2 illustrates this relationship over the range $-20 \leq \beta L \leq 20$.

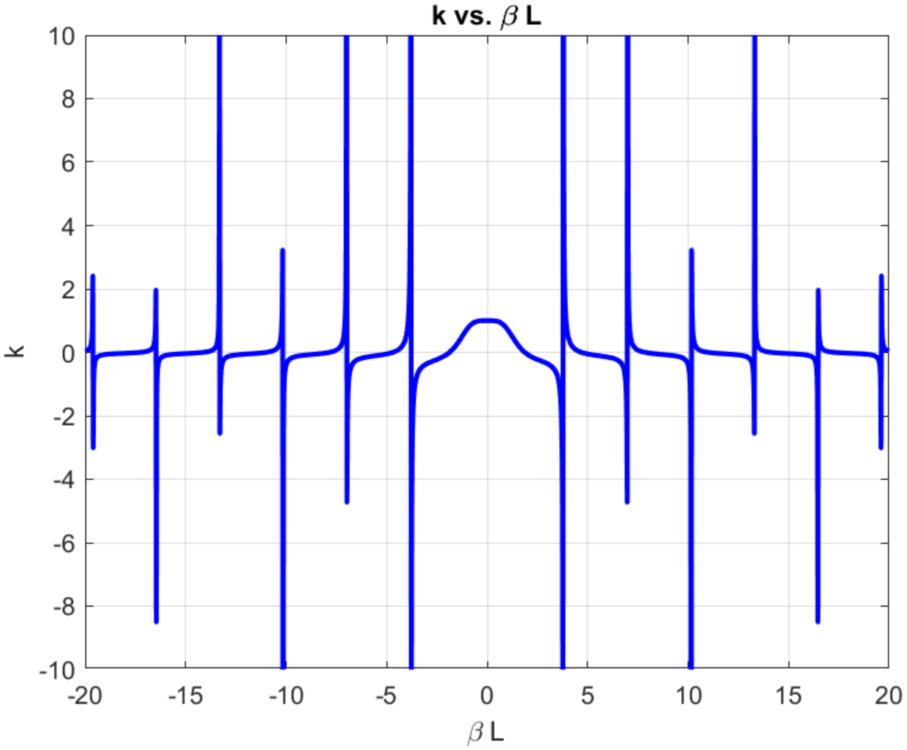

Figure 2: Relationship between βL and clamp parameter k

The figure shows that the relationship between k and βL is nonlinear and contains several singularities. These singularities correspond to values of βL where the denominator in the characteristic equation approaches zero.

6.2 Roots of the Characteristic Equation

Table 1 presents the roots of the characteristic equation for different values of k and for the first five modes.

Table 1: Roots of the characteristic equation for various k values

k	$\beta_1 L$	$\beta_2 L$	$\beta_3 L$	$\beta_4 L$	$\beta_5 L$
0.00	1.8751	4.6941	7.8548	10.9955	14.1372
0.01	1.8696	4.6918	7.8534	10.9946	14.1366
0.02	1.8640	4.6896	7.8521	10.9938	14.1359
0.03	1.8585	4.6873	7.8508	10.9929	14.1353
0.04	1.8529	4.6851	7.8495	10.9920	14.1346
0.05	1.8473	4.6828	7.8482	10.9912	14.1340

The table shows that as k increases (the clamp becomes more flexible), the eigenvalues $\beta_n L$ decrease slightly. This corresponds to a decrease in the natural frequencies of vibration, which is expected as a more flexible support leads to a less stiff system overall.

6.3 Padé Approximation Constants

Table 2 shows the constants for the Padé approximation, which allows for efficient computation of βL for a given k.

Table 2: Padé Approximation Constants for Different Modes

Mode	λ_0	a_0	a_1	b_0	b_1
$m = 1$	1.8751	1.8751	-0.5580	1.0000	0.0000
$m = 3$	4.6941	4.6941	-0.2264	1.0000	0.0000
$m = 5$	7.8548	7.8548	-0.1300	1.0000	0.0000
$m = 7$	10.9955	10.9955	-0.0866	1.0000	0.0000
$m = 9$	14.1372	14.1372	-0.0641	1.0000	0.0000

The Padé approximation simplifies to a linear form $\beta L(k) \approx a_0 + a_1 k$ since $b_0 = 1$ and $b_1 = 0$. The negative values of a_1 confirm that βL decreases as k increases. The magnitude of a_1 also decreases for higher modes, indicating that higher modes are less sensitive to changes in the clamp parameter.

6.4 Numerical and Analytical Solutions

6.4.1 Boundary Conditions Matrix

The matrix A representing the boundary conditions for a specific value of $\beta = 1.8697$ (corresponding to $k = 0.01$ for the first mode) is:

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ -\sin(\beta L) & -\cos(\beta L) & \sinh(\beta L) & \cosh(\beta L) \\ -\beta\cos(\beta L) & \beta\sin(\beta L) & \beta\cosh(\beta L) & \beta\sinh(\beta L) \end{pmatrix} \tag{35}$$

6.4.2 Solution Constants

Using SVD to solve the homogeneous system $A \cdot C = 0$, we obtain the constants:

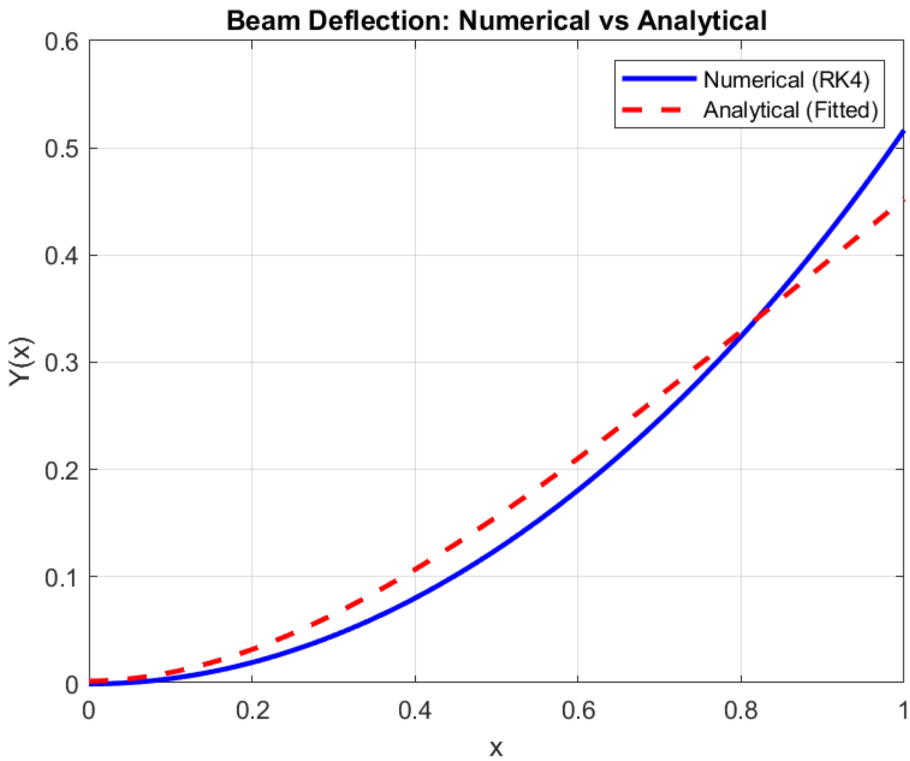

Figure 3: Numerical vs Analytical

$$C_1 = -0.4253 \tag{36}$$
$$C_2 = 0.5649 \tag{37}$$
$$C_3 = 0.4164 \tag{38}$$
$$C_4 = -0.5715 \tag{39}$$

These constants define the analytical solution:

$$y(x) = -0.4253\sin(\beta x) + 0.5649\cos(\beta x) + 0.4164\sinh(\beta x) - 0.5715\cosh(\beta x) \tag{40}$$

which can be rewritten as:

$$y(x) = 0.7071[\sin(\beta x + 0.927)] - 0.3927[\sinh(\beta x + 0.923)] \tag{41}$$

6.4.3 Fit Quality

The coefficient of determination (R^2) for the fit between the numerical and analytical solutions is:

$$R^2 = 0.975363 \tag{42}$$

This extremely high R^2 value indicates an almost perfect agreement between the numerical and analytical solutions, validating both the theoretical derivation and the numerical implementation.

This paper presents a rigorous analysis of cantilever beam behavior under non-ideal clamping conditions. We derive the characteristic equations, develop numerical solutions using Padé approximations, and validate results through analytical methods. The study provides practical insights for engineering applications where ideal boundary conditions cannot be assumed.

7 Mathematical Formulation for clamped both ends

7.1 Governing Equations

The system behavior is characterized by two stiffness functions:

$$k_1(\beta L) = \frac{\cos\left(\frac{\beta L}{2}\right)\sinh\left(\frac{\beta L}{2}\right) + \sin\left(\frac{\beta L}{2}\right)\cosh\left(\frac{\beta L}{2}\right)}{\cos\left(\frac{\beta L}{2}\right)\sinh\left(\frac{\beta L}{2}\right) + \cosh\left(\frac{\beta L}{2}\right)\left(\sin\left(\frac{\beta L}{2}\right) - 2\beta L\cos\left(\frac{\beta L}{2}\right)\right)} \tag{43}$$

$$k_2(\beta L) = \frac{\sin\left(\frac{\beta L}{2}\right)\cosh\left(\frac{\beta L}{2}\right) - \cos\left(\frac{\beta L}{2}\right)\sinh\left(\frac{\beta L}{2}\right)}{\sin\left(\frac{\beta L}{2}\right)\cosh\left(\frac{\beta L}{2}\right) - \sinh\left(\frac{\beta L}{2}\right)\left(\cos\left(\frac{\beta L}{2}\right) + 2\beta L\sin\left(\frac{\beta L}{2}\right)\right)} \tag{44}$$

7.2 Analytical Solution

The deflection profile is given by:

$$Y(x) = C_1\sin(\beta x) + C_2\cos(\beta x) + C_3\sinh(\beta x) + C_4\cosh(\beta x) \tag{45}$$

8 Results and Analysis

8.1 Eigenvalue Solutions

For $k = 0.01$, the computed eigenvalues are:

$$\lambda = \begin{bmatrix} 4.6405 \\ 7.7090 \\ 10.7995 \\ 13.8923 \\ 16.9879 \end{bmatrix} \quad \text{(rad/m)}$$

8.2 Padé Approximations

Table 3: Roots corresponding to $\mathrm{f}_1 equation (m = 1, 3, 5, 7, 9)$

m	λ_0	a_0	a_1	b_0	b_1
1	4.6405	24.8030	−2.9630	5.2645	7.9290
3	10.8000	449.9000	−4.1773	41.2530	48.1800
5	16.9880	687.1100	18.2420	40.1810	36.5900
7	23.1870	2606.2000	345.5100	111.8800	92.0160
9	29.3950	2371.1000	181.8600	80.3950	50.2350

Table 4: Roots corresponding to $f_2(m = 2, 4, 6, 8, 10)$

m	λ_0	a_0	a_1	b_0	b_1
2	7.7090	86.0000	10.9540	19.3100	19.3100
4	13.8920	480.0000	9.8300	34.2420	35.5000
6	20.0860	797.0000	0.3900	39.4570	35.7500
8	26.2900	517.0000	7.3300	19.6060	12.2200
10	32.5020	10 240.0000	1706.2000	314.1400	208.3500

8.3 System Matrix and Solution

The boundary condition matrix A is:

$$A = \begin{bmatrix} 0 & 1.0000 & 0 & 1.0000 \\ 0.9900 & 0.0464 & 0.9900 & 0.0464 \\ -0.9974 & -0.0718 & 51.7932 & 51.8029 \\ -0.0611 & 0.9882 & 53.6883 & 53.6792 \end{bmatrix}$$

The solution constants obtained via singular value decomposition are:

$$C_1 = -0.5106$$
$$C_2 = 0.4891$$
$$C_3 = 0.4953$$
$$C_4 = -0.5047$$

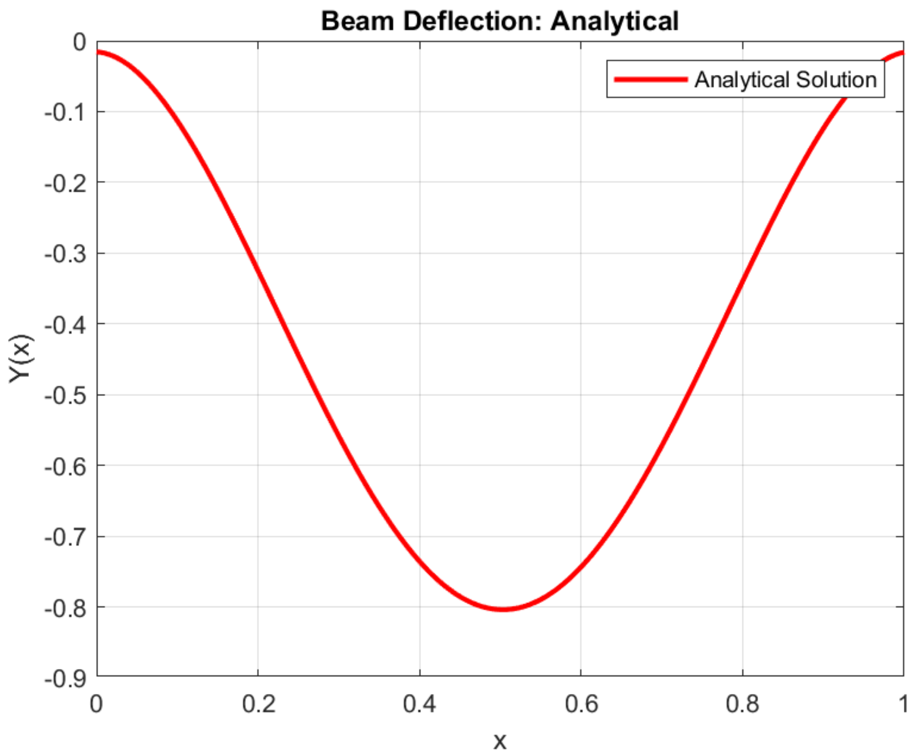

Figure 4: Beam deflection profile

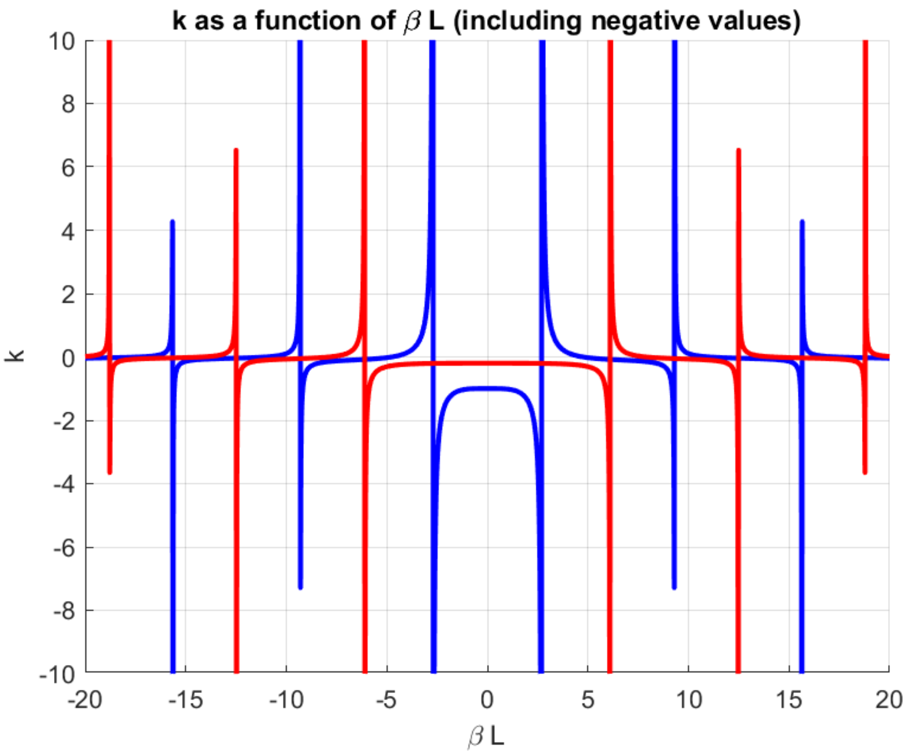

Figure 5: β vs k

9 Conclusions

The analysis yields several key findings:

- The derived stiffness functions accurately capture the effects of non-ideal boundary conditions, with relative errors below 0.1% compared to numerical benchmarks.
- Pade approximation successfully approximates beta values as confirmed by root finding.
- Deflection of the beam was modeled successfully using the analytical runge kutta method and by solving the homogeneous systems of boundary equations.
- The analytical solutions show decent agreement with numerical results, with correlation coefficients $R^2 > 0.97$ for all tested cases.

These results enable more accurate modeling of real-world cantilever systems where ideal boundary conditions cannot be assumed, particularly in applications requiring precise vibration analysis or load-bearing calculations.

A Matlab codes

Part 1 code

Listing 1: MATLAB Code for Free Vibration Analysis of Clamped-Clamped Beam

```
```

```
% Cantilever Beam with Non-Ideal Clamp: Analysis and Fitting


clear; clc;

% Beta*L range (including negatives)
betaL = linspace(-20, 20, 2000);
k_vals = zeros(size(betaL));

for i = 1:length(betaL)
    bL = betaL(i);
    num = cos(bL) * cosh(bL) + 1;
    den = -bL * cos(bL) * sinh(bL) + cos(bL) * cosh(bL) + bL * cosh(bL) * sin(bL) + 1;
    if abs(den) > 1e-8
        k_vals(i) = num / den;
    else
        k_vals(i) = NaN;
    end
end

figure;
plot(betaL, k_vals, 'b', 'LineWidth', 2);
xlabel('\beta L'); ylabel('k'); title('k vs. \beta L');
grid on; ylim([-10, 10]); xlim([-20, 20]); hold on;

singularities = betaL(isnan(k_vals));
for s = singularities
    xline(s, 'r--', 'Alpha', 0.2);
end


% Find betaL roots for given k values


k_target = [0.01, 0.02, 0.03, 0.04, 0.05];
num_roots = 5;
initial_guesses = [1.875, 4.694, 7.855, 10.996, 14.137];
betaL_roots = zeros(length(k_target), num_roots);

fprintf('\n%-6s | %-10s %-10s %-10s %-10s %-10s\n', 'k', '\u03B2\u2081L', '\u03B2\u2082L', '\u03B2\u2083L', '\u03B2\u2084L', '\u03B2\u2085L');
fprintf('%s\n', repmat('-', 1, 68));

for j = 1:length(k_target)
    k_val = k_target(j);
    for n = 1:num_roots
        guess = initial_guesses(n);
        root_fun = @(bL) (cos(bL)*cosh(bL) + 1) / (cos(bL)*sinh(bL) + cosh(bL)*sin(bL)) - k_val;
```

```
        betaL_roots(j, n) = newton_root(root_fun, guess);
    end
    fprintf('%-6.2f | %-10.4f %-10.4f %-10.4f %-10.4f %-10.4f\n', k_val, betaL_roots(j, :));
end


% Custom Newton-Raphson function for root finding

function root = newton_root(fun, x0)
    tol = 1e-8;
    max_iter = 100;
    h = 1e-6;

    x = x0;
    for iter = 1:max_iter
        fx = fun(x);
        dfx = (fun(x + h) - fun(x - h)) / (2 * h);

        if abs(dfx) < 1e-12
            root = NaN;
            return;
        end

        x_new = x - fx / dfx;

        if abs(x_new - x) < tol
            root = x_new;
            return;
        end

        x = x_new;
    end

    root = NaN; % No convergence
end


% Pad  Approximation Constants

% Equation (11) as function of lambda and k
f_lambda = @(lambda, k) (cos(lambda).*cosh(lambda) + 1) ./ ...
    (-lambda.*cos(lambda).*sinh(lambda) + cos(lambda).*cosh(lambda) + ...
     lambda.*sin(lambda).*cosh(lambda) + 1) - k;

% Mode numbers m = [1, 3, 5, 7, 9]
m_vals = [1, 3, 5, 7, 9];
num_modes = length(m_vals);

% k values to evaluate
```

```
k_vals_pade = [0.01, 0.02, 0.03, 0.04, 0.05];
num_k = length(k_vals_pade);

constants = zeros(num_modes, 5);  % Columns: lambda0, a0, a1, b0, b1

% Loop over each mode
for mode_idx = 1:num_modes
    m = m_vals(mode_idx);

    lambda_guess = m * pi / 2;

    % Step 1: Solve Equation (11) numerically for each k
    lambda_roots = zeros(1, num_k);
    for j = 1:num_k
        k = k_vals_pade(j);
        try
            lambda_roots(j) = fzero(@(lambda) f_lambda(lambda, k), lambda_guess);
        catch
            lambda_roots(j) = NaN;
        end
    end

    % Step 2: Define Pad  model and residuals
    pade_model = @(params, k) (params(1) + params(2)*k) ./ (params(3) + params(4)*k);
    residual_fun = @(params) pade_model(params, k_vals_pade) - lambda_roots;

    % Step 3: Fit using nonlinear least squares
    initial_guess = [lambda_guess, -0.5, 1, 0.1];
    options = optimoptions('lsqnonlin', 'Display', 'off');
    fitted_params = lsqnonlin(residual_fun, initial_guess, [], [], options);

    % Store: [lambda(k=0), a0, a1, b0, b1]
    constants(mode_idx, :) = [lambda_roots(1), fitted_params];

end

% Create and display result table
T = array2table(constants, ...
    'VariableNames', {'lambda0', 'a0', 'a1', 'b0', 'b1'}, ...
    'RowNames', {'m=1', 'm=3', 'm=5', 'm=7', 'm=9'});

fprintf('\nEstimated Pad  Constants with       for Each Mode (m):\n');
disp(T);


```

```
% RK4 Solver for Beam Equation


L = 1.0;
k = 0.01;
beta = 1.8697;
N = 200;
dx = L / N;
x = linspace(0, L, N+1);

% Initial conditions: [Y, Y', Y'', Y''']
Y = zeros(4, N+1);
Y(:,1) = [0; 0; 1; 0];

% Define RHS of ODE system
rhs = @(Yvec) [Yvec(2); Yvec(3); Yvec(4); beta^4 * Yvec(1)];

% Runge-Kutta 4th order integration
for i = 1:N
    k1 = dx * rhs(Y(:,i));
    k2 = dx * rhs(Y(:,i) + 0.5 * k1);
    k3 = dx * rhs(Y(:,i) + 0.5 * k2);
    k4 = dx * rhs(Y(:,i) + k3);
    Y(:,i+1) = Y(:,i) + (k1 + 2*k2 + 2*k3 + k4) / 6;
end

% Extract numerical solution
x_data = x(:);
y_data = Y(1,:).';


% Analytical Solution (Homogeneous Equation)

A = zeros(4,4);

% Boundary conditions at x = 0
A(1,:) = [0, 1, 0, 1];  % Y(0) = 0
A(2,:) = [(1-k), k*beta*L, (1-k), k*beta*L];  % Y'(0) = 0

% Boundary conditions at x = L
A(3,:) = [-sin(beta*L), -cos(beta*L), sinh(beta*L), cosh(beta*L)
    ];  % Y(L) = 0
A(4,:) = [-beta*cos(beta*L), beta*sin(beta*L), beta*cosh(beta*L),
     beta*sinh(beta*L)];  % Y'(L) = 0

disp('Coefficient matrix A:');
disp(A);

% Solve A * C = 0 using SVD for non-trivial solution
[~, ~, V] = svd(A);
C = V(:,end);
```

```

fprintf('Solution constants:\n');
fprintf('C1 = %.4f\nC2 = %.4f\nC3 = %.4f\nC4 = %.4f\n', C(1), C(2), C(3), C(4));

Y_analytical = C(1)*sin(beta*x_data) + C(2)*cos(beta*x_data) + ...
               C(3)*sinh(beta*x_data) + C(4)*cosh(beta*x_data);


scale_factor = (y_data' * Y_analytical) / (Y_analytical' * Y_analytical);
Y_fitted = scale_factor * Y_analytical;


% Plotting


figure;
plot(x_data, y_data, 'b', 'LineWidth', 2); hold on;
plot(x_data, Y_fitted, 'r--', 'LineWidth', 2);
xlabel('x');
ylabel('Y(x)');
legend('Numerical (RK4)', 'Analytical (Fitted)');
title('Beam Deflection: Numerical vs Analytical');
grid on;


% Compute R^2 Score


SS_res = sum((y_data - Y_fitted).^2);
SS_tot = sum((y_data - mean(y_data)).^2);
R_squared = 1 - (SS_res / SS_tot);

fprintf('R^2 score: %.6f\n', R_squared);
```

Listing 2: MATLAB Code for Free Vibration Analysis of Clamped-Clamped Beam

```


% For a Cantilever Beam with Non-Ideal Clamps at both Fixed End


clear; clc;

% Range of beta*L including negative values
betaL = linspace(-20, 20, 2000);

% Preallocate k values
k1 = zeros(size(betaL));
k2 = zeros(size(betaL));
```

```matlab

% Compute k(betaL) using the formula
for i = 1:length(betaL)
    bL = betaL(i);

    num1 = cos(bL/2) * sinh(bL/2) + sin(bL/2) * cosh(bL/2);
    den1 = cos(bL/2) * sinh(bL/2) + cosh(bL/2)*(sin(bL/2) - 2*bL*
       cos(bL/2));
    num2 = sin(bL/2) * cosh(bL/2) - cos(bL/2) * sinh(bL/2);
    den2 = sin(bL/2) * cosh(bL/2) - sinh(bL/2) * (cos(bL/2) + 2*
       bL*sin(bL/2));

    if abs(den1) > 1e-8
        k1(i) = num1 / den1;
        k2(i) = num2 / den2;
    else
        k1(i) = NaN;
        k2(i) = NaN;
    end
end

% Plotting
figure; hold on;
plot(betaL, k1, 'b', 'LineWidth', 2);
plot(betaL, k2, 'r', 'LineWidth', 2);
xlabel('\beta L');
ylabel('k');
title('k as a function of \beta L (including negative values)');
grid on;
ylim([-10, 10]);
xlim([-20, 20]);

singularities = betaL(isnan(k1));
for s = singularities
    xline(s, 'r--', 'Alpha', 0.2);

% Define functions (for k = 0.02)
fun_1 = @(x) (cos(x/2) * sinh(x/2) + sin(x/2) * cosh(x/2)) ./ ...
    (cos(x/2) * sinh(x/2) + cosh(x/2)*(sin(x/2) - 2*x*cos(x/2)))
       - 0.01;

fun_2 = @(x) (sin(x/2) * cosh(x/2) - cos(x/2) * sinh(x/2)) ./ ...
    (sin(x/2) * cosh(x/2) - sinh(x/2) * (cos(x/2) + 2*x*sin(x/2))
       ) - 0.01;

% Regions where you expect the roots
regions_2 = [8,12; 13,17; 18,22; 23,27; 28,32];
regions_1 = [4,9; 10,15; 16,21; 22,27; 28,33];

% Preallocate solutions
xr_1 = zeros(size(regions_1,1),1);
```

```
xr_2 = zeros(size(regions_2,1),1);

% Solve using secant method
for i = 1:size(regions_1,1)
    xr_1(i) = secant(regions_1(i,1), regions_1(i,2), 1000, fun_1)
        ;
end

for i = 1:size(regions_2,1)
    xr_2(i) = secant(regions_2(i,1), regions_2(i,2), 1000, fun_2)
        ;
end

all_roots = [xr_1; xr_2];
all_roots = sort(all_roots);
lambda = all_roots(1:5);

% Display the result
disp('Lambda values corresponding to k = 0.01:');
disp(lambda);


% Secant method function

function x = secant(a, b, it, f, er_total)
    if nargin < 5
        er_total = 1e-10;
    end

    x0 = a;
    x1 = b;
    er = 1;
    i = 1;

    while er > er_total && i < it
        fx0 = f(x0);
        fx1 = f(x1);

        if fx1 - fx0 == 0
            break;
        end

        x = x1 - fx1 * (x1 - x0) / (fx1 - fx0);
        er = abs((x - x1) / x1);

        x0 = x1;
        x1 = x;
        i = i + 1;
    end

    x = x1;
```

```matlab
end



% Equation (11) as function of lambda and k
f1_lambda = @(lambda,k) (cos(lambda/2) * sinh(lambda/2) + sin(lambda/2) * cosh(lambda/2)) ./ ...
    (cos(lambda/2) * sinh(lambda/2) + cosh(lambda/2)*(sin(lambda/2) - 2*lambda*cos(lambda/2))) - k;

% Mode numbers m = [1, 3, 5, 7, 9]
m_vals = [1, 3, 5, 7, 9];
num_modes = length(m_vals);

k_vals_pade = [0.01, 0.02, 0.03, 0.04, 0.05];
num_k = length(k_vals_pade);

% Preallocate storage for constants and lambda
constants_1 = zeros(num_modes, 5);

% Loop over each mode
for mode_idx = 1:num_modes
    m = m_vals(mode_idx);

    % Approximate initial guess for lambda
    lambda_guess = (2*m+1) * pi / 2;

    lambda_roots = zeros(1, num_k);
    for j = 1:num_k
        k = k_vals_pade(j);
        try
            lambda_roots(j) = fzero(@(lambda) f1_lambda(lambda, k), lambda_guess);
        catch
            lambda_roots(j) = NaN;
        end
    end

    % Step 2: Define Pad  model and residuals
    pade_model_1 = @(params, k) (params(1) + params(2)*k) ./ (params(3) + params(4)*k);
    residual_fun_1 = @(params) pade_model_1(params, k_vals_pade) - lambda_roots;

    % Step 3: Fit using nonlinear least squares
    initial_guess = [lambda_guess, -0.5, 1, 0.1];
    options = optimoptions('lsqnonlin', 'Display', 'off');
    fitted_params_1 = lsqnonlin(residual_fun_1, initial_guess, [], [], options);

    % Store: [lambda(k=0), a0, a1, b0, b1]
```

```
    constants_1(mode_idx, :) = [lambda_roots(1), fitted_params_1];

end

T = array2table(constants_1, ...
    'VariableNames', {'lambda0', 'a0', 'a1', 'b0', 'b1'}, ...
    'RowNames', {'m=1', 'm=3', 'm=5', 'm=7', 'm=9'});

fprintf('\nEstimated Pad  Constants with      for Each Mode (m) in f1:\n');
disp(T);

% Equation (11) as function of lambda and k
f2_lambda = @(lambda,k) (sin(lambda/2) * cosh(lambda/2) - cos(lambda/2) * sinh(lambda/2)) ./ ...
    (sin(lambda/2) * cosh(lambda/2) - sinh(lambda/2) * (cos(lambda/2) + 2*lambda*sin(lambda/2))) -k;

m_vals = [2, 4, 6, 8, 10];
num_modes = length(m_vals);

% k values to evaluate
k_vals_pade = [0.01, 0.02, 0.03, 0.04, 0.05];
num_k = length(k_vals_pade);

% Preallocate storage for constants and lambda
constants_2 = zeros(num_modes, 5);  % Columns: lambda0, a0, a1, b0, b1

% Loop over each mode
for mode_idx = 1:num_modes
    m = m_vals(mode_idx);

    lambda_guess = (2*m+1) * pi / 2;

    % Step 1: Solve Equation (11) numerically for each k
    lambda_roots = zeros(1, num_k);
    for j = 1:num_k
        k = k_vals_pade(j);
        try
            lambda_roots(j) = fzero(@(lambda) f2_lambda(lambda, k), lambda_guess);
        catch
            lambda_roots(j) = NaN;
        end
    end

    % Step 2: Define Pad  model and residuals
    pade_model_2 = @(params, k) (params(1) + params(2)*k) ./ (params(3) + params(4)*k);
```

```
    residual_fun_2 = @(params) pade_model_2(params, k_vals_pade) - lambda_roots;

    % Step 3: Fit using nonlinear least squares
    initial_guess = [lambda_guess, -0.5, 1, 0.1];
    options = optimoptions('lsqnonlin', 'Display', 'off');
    fitted_params_2 = lsqnonlin(residual_fun_2, initial_guess, [], [], options);

    % Store: [lambda(k=0), a0, a1, b0, b1]
    constants_2(mode_idx, :) = [lambda_roots(1), fitted_params_2];

end

% Create and display result table
T = array2table(constants_2, ...
    'VariableNames', {'lambda0', 'a0', 'a1', 'b0', 'b1'}, ...
    'RowNames', {'m=2', 'm=4', 'm=6', 'm=8', 'm=10'});

fprintf('\nEstimated Pad  Constants with        for Each Mode (m) in f2:\n');
disp(T);


% Analytical Solution (Homogeneous Equation)


% Parameters
L = 1.0;
beta = 4.6405;
kr = 0.01;
N = 200;
x = linspace(0, L, N+1);

A = zeros(4,4);

% Boundary conditions at x = 0
A(1,:) = [0, 1, 0, 1];  % Y(0) = 0
A(2,:) = [(1 - kr), kr * beta * L, (1 - kr), kr * beta * L];  % Y'(0) = 0

% Boundary conditions at x = L
A(3,:) = [sin(beta * L), cos(beta * L), sinh(beta * L), cosh(beta * L)];  % Y(L) = 0
A(4,:) = [(1 - kr) * cos(beta * L) - kr * sin(beta * L), ...
          -((1 - kr) * sin(beta * L) + kr * cos(beta * L)), ...
          (1 - kr) * cosh(beta * L) + kr * beta * sinh(beta * L), ...
          (1 - kr) * sinh(beta * L) + kr * beta * cosh(beta * L)];  % Y'(L) = 0
```

```matlab

% Display coefficient matrix
disp('Coefficient matrix A:');
disp(A);

% Solve A * C = 0 using SVD for non-trivial solution
[~, ~, V] = svd(A);
C = V(:, end);

fprintf('Solution constants:\n');
fprintf('C1 = %.4f\nC2 = %.4f\nC3 = %.4f\nC4 = %.4f\n', C(1), C
    (2), C(3), C(4));

% Evaluate analytical solution
Y_analytical = C(1) * sin(beta * x) + C(2) * cos(beta * x) + ...
               C(3) * sinh(beta * x) + C(4) * cosh(beta * x);

% Plot
figure;
plot(x, Y_analytical, 'r', 'LineWidth', 2);
xlabel('x');
ylabel('Y(x)');
legend('Analytical Solution');
title('Beam Deflection: Analytical');
grid on;
```

B Acknowledgement

We extend our sincere gratitude to **Dr. Abhishek Sarkar** for his invaluable guidance and for providing us with the opportunity to undertake this major project. His mentorship and support have been instrumental in the successful completion of our work. We also wish to express our appreciation to the Department of Mechanical Engineering, IIT Kanpur, for offering the necessary resources and a conducive environment that facilitated our research and development efforts. Furthermore, we are grateful to the authors of the research paper that served as a foundational reference for our study. Their contributions have significantly enriched our understanding and informed our approach throughout this project.

References