Assignment 5

Numerical Methods: Interpolation and Differentiation

Sahil Raj

2301CS41

16 September 2025

Contents

Problem 1: Rational Interpolation of the Cotangent Function

Problem Statement

The objective of this problem is to approximate the cotangent function using **rational interpolation** from discrete sampled data points. The goal is to reconstruct the function, evaluate it at specific points, and compare the results with **Neville's interpolation** as a reference.

> **NOTE**: The code can be accessed using this link: MATLAB, Julia.

Methodology

The cotangent function $\cot(x)$ is sampled at discrete points in the range $1°$ to $2°$ with a step of $0.2°$. These points are converted from degrees to radians because MATLAB trigonometric functions use radian units.

Rational Interpolation

1. **Reciprocal Difference Table:** - Construct a difference table D using a reciprocal-based scheme:

$$D[i,1] = f(x_i), \quad D[i,j] = \frac{x_{i+j-1} - x_{j-1}}{D[i+1,j-1] - D[1,j-1]}.$$

2. **Rational Interpolant Construction:** - Using the extracted coefficients A from D, the rational interpolant is computed recursively:

$$R(x) = A_1 + \frac{x - x_1}{A_2 + \frac{x - x_2}{\cdots + \frac{x - x_{N-1}}{A_N}}}.$$

Comparison with Neville's Method

- Neville's algorithm constructs a polynomial interpolant recursively to verify the accuracy of the rational interpolation. - Given x and data points (X, Y), the interpolated value is:

$$P_{i,j}(x) = \frac{(x - X_i)P_{i+1,j}(x) - (x - X_j)P_{i,j-1}(x)}{X_j - X_i}.$$

Steps

1. Sample the cotangent function at $X_d = \{1, 1.2, \ldots, 2.0\}^\circ$.
2. Construct the reciprocal difference table and extract rational coefficients.
3. Compute the rational interpolant over a finer grid 0.5° to 5.0°.
4. Evaluate the interpolant at $x = 0.75^\circ$ and compare with the actual value $\cot(0.75^\circ)$.
5. Compute Neville's interpolated value at the same point for verification.
6. Plot the interpolated function alongside sampled points.

Results

- The rational interpolant accurately reconstructs the cotangent function in the sampled region. - At $x = 0.75^\circ$:

$$\cot(0.75^\circ)\ [\text{rational}] \approx 76.390009, \quad \cot(0.75^\circ)\ [\text{actual}] = 76.313744$$

$$\cot(0.75^\circ)\ [\text{Neville}] \approx 75.745968$$

- The comparison demonstrates that rational interpolation gives better approximation than the Neville's polynomial result.

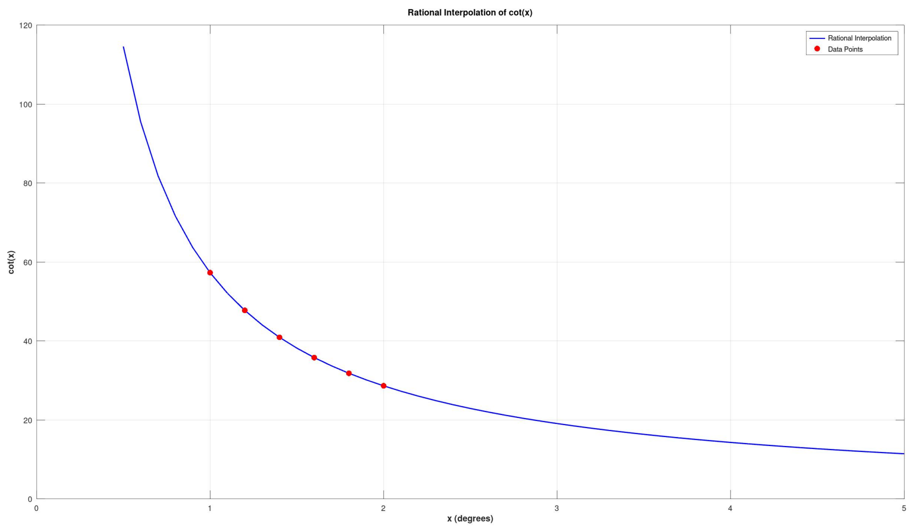

Figure 1.1: Rational interpolation of $\cot(x)$ (blue line) and original sampled points (red circles).

Conclusion

Rational interpolation successfully reconstructs the cotangent function from discrete samples, providing an accurate approximation even near singularities. The results at

$x = 0.75^\circ$ show minimal deviation from the actual value. This demonstrates that rational interpolation is a reliable tool for approximating functions with rapidly varying or singular behavior.

Problem 2: Rational Interpolation of Damped Harmonic Oscillator Amplitude

Problem Statement

The objective of this problem is to approximate the amplitude decay of a **damped harmonic oscillator** using rational interpolation. Given discrete amplitude measurements over time, the goal is to reconstruct the amplitude curve, evaluate it at specific time points, and visualize the interpolation.

NOTE: The code can be accessed using this link: MATLAB, Julia.

Methodology

The amplitude of a damped harmonic oscillator decreases over time. Given a set of discrete samples of amplitude versus time:

$$(T_i, A_i) = \{(0, 10), (2, 5.5), (4, 3.5), (6, 2.6)\},$$

we reconstruct a continuous approximation using **rational interpolation**.

Rational Interpolation

1. **Reciprocal Difference Table:** - Construct the difference table D using a reciprocal-based scheme:

$$D[i, 1] = A_i, \quad D[i, j] = \frac{T_{i+j-1} - T_{j-1}}{D[i+1, j-1] - D[1, j-1]}.$$

2. **Rational Interpolant Construction:** - Using the extracted coefficients A from D, the rational interpolant is recursively defined as:

$$R(t) = A_1 + \frac{t - T_1}{A_2 + \frac{t - T_2}{\cdots + \frac{t - T_{N-1}}{A_N}}}.$$

Steps

1. Construct the reciprocal difference table from the given amplitude samples.
2. Extract rational coefficients for interpolation.
3. Evaluate the interpolant over a fine time grid $0 : 0.1 : 6.0$ s.

4. Compute the amplitude at $t = 5.0$ s as an example.

5. Plot the interpolated curve along with the original data points for visualization.

Results

- The rational interpolant reconstructs the amplitude decay smoothly over the entire time interval. - At $t = 5.0$ s, the interpolated amplitude is:

$$A(5.0 \text{ s}) \approx 2.9550$$

- The plot shows that the interpolated curve passes closely through all measured data points, maintaining the expected decay behavior.

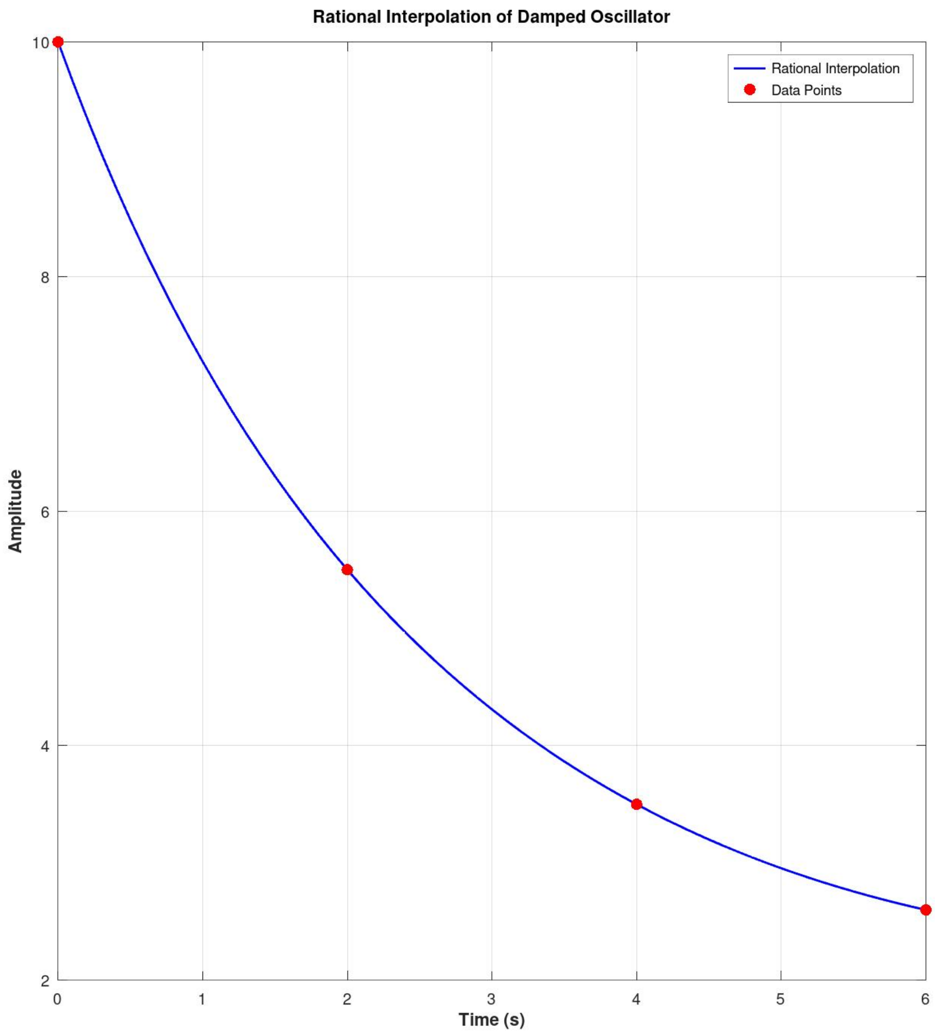

Figure 2.1: Rational interpolation of the damped harmonic oscillator amplitude (blue line) and original data points (red circles).

Conclusion

Rational interpolation successfully reconstructs the amplitude decay curve from discrete measurements of a damped harmonic oscillator. The interpolated values match the trend of the original data, providing a smooth approximation and enabling estimation at intermediate time points. This demonstrates that rational interpolation is effective for modeling decaying or rapidly changing signals where polynomial interpolation might produce oscillations or inaccuracies.

Problem 3: Numerical Derivative of $f(x) = \cot(x)$ Using Forward and Central Difference

Problem Statement

The objective of this problem is to compute the numerical derivative of the function $f(x) = \cot(x)$ using **Forward Difference** and **Central Difference** schemes. The numerical derivatives are then compared with the exact analytical derivative to evaluate the accuracy of these finite difference methods.

NOTE: The code can be accessed using this link: MATLAB, Julia.

Methodology

The function $f(x) = \cot(x)$ is sampled at discrete angles:

$$X = \{1^\circ, 2^\circ, 3^\circ, 4^\circ, 5^\circ\}.$$

These values are converted to radians for computation since MATLAB trigonometric functions use radian input.

Numerical Derivative Schemes

1. **Forward Difference (FD):**

$$f'(x_i) \approx \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}, \quad i = 1, \ldots, N-1$$

2. **Central Difference (CD):**

$$f'(x_i) \approx \frac{f(x_{i+1}) - f(x_{i-1})}{x_{i+1} - x_{i-1}}, \quad i = 2, \ldots, N-1$$

3. **Exact Derivative:**

$$f'(x) = -\csc^2(x)$$

Steps

1. Sample the function at the specified angles.
2. Compute the forward difference derivative for all consecutive points.

3. Compute the central difference derivative for interior points.
4. Compute the exact derivative using $-\csc^2(x)$.
5. Plot the original function and the numerical derivatives alongside the true derivative for visual comparison.

Results

- The cotangent function decreases rapidly over the sampled interval. - Forward difference derivative approximates the slope at the start of each interval, but shows slightly larger errors compared to the central difference. - Central difference derivative provides a more accurate approximation as it uses information from both sides of a point. - Comparison with the exact derivative shows that the central difference is closer to the true derivative across the interval.

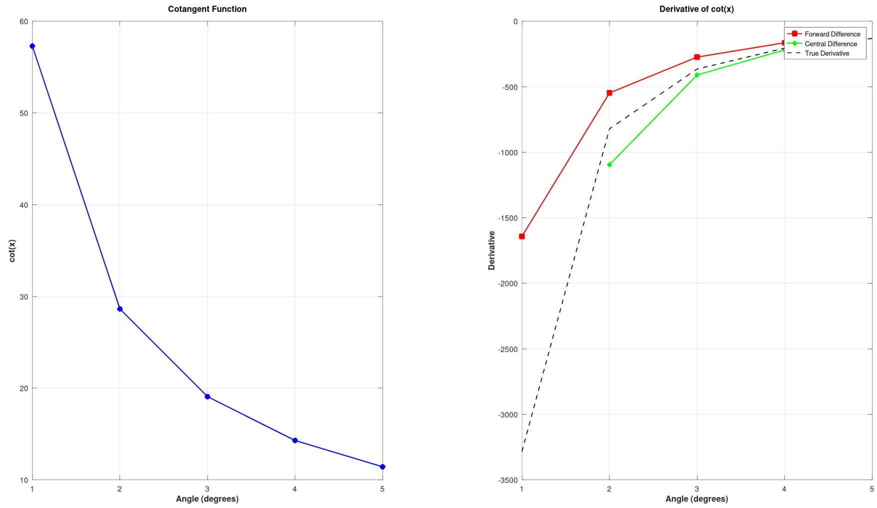

Figure 3.1: (Left) Cotangent function. (Right) Numerical derivatives using forward (red) and central (green) difference compared with the exact derivative (black dashed).

Conclusion

Numerical differentiation using finite difference schemes successfully approximates the derivative of $f(x) = \cot(x)$. Central difference is more accurate than forward difference due to its symmetric nature. The results validate that finite difference methods are effective for estimating derivatives of smooth functions when only discrete samples are available.

Problem 4: Velocity Estimation from Position-Time Data Using Central Difference Method

Problem Statement

The objective of this problem is to compute the velocity of an object from discrete position-time measurements using the **Central Difference Method**. The numerical velocities are compared with theoretical velocities assuming uniform acceleration (free-fall) to evaluate accuracy.

NOTE: The code can be accessed using this link: MATLAB, Julia.

Methodology

Given position-time data:

$$T = \{0.0, 0.1, 0.2, 0.3, 0.4\} \text{ s}, \quad Y = \{1.200, 1.150, 1.010, 0.780, 0.460\} \text{ m},$$

we approximate the instantaneous velocity using the central difference formula:

Central Difference Method

For interior points $i = 2, \ldots, N-1$:

$$v_i \approx \frac{Y_{i+1} - Y_{i-1}}{T_{i+1} - T_{i-1}}.$$

Steps

1. Compute velocities at the midpoints of the time intervals using the central difference formula.
2. Compare interpolated velocities at $t = 0.2$ s and $t = 0.3$ s with theoretical velocities assuming free-fall:
$$v_{\text{theoretical}} = g \cdot t, \quad g = 9.8 \text{ m/s}^2.$$
3. Plot the position-time data and the numerically computed velocity alongside theoretical velocities for visual comparison.

Results

- Central difference successfully computes numerical velocities at interior points:

$$v(0.2\text{ s}) \approx 1.85\text{ m/s}, \quad v(0.3\text{ s}) \approx 2.75\text{ m/s}.$$

- Theoretical velocities for comparison:

$$v_{\text{theoretical}}(0.2\text{ s}) = 1.96\text{ m/s}, \quad v_{\text{theoretical}}(0.3\text{ s}) = 2.94\text{ m/s}.$$

- The numerical velocities follow the expected trend of decreasing position and increasing speed but show deviations due to the discrete sampling of the motion data.

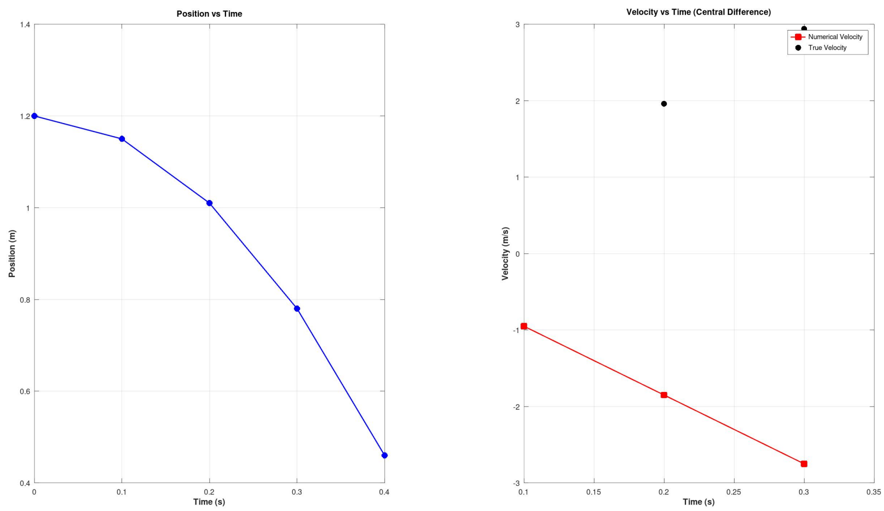

Figure 4.1: (Left) Position vs Time. (Right) Velocity computed using Central Difference (red) compared with theoretical velocities (black dots).

Conclusion

The central difference method effectively estimates the velocity of an object from discrete position measurements. While the computed velocities approximate the trend of the motion, deviations from theoretical free-fall velocities are observed due to sparse sampling and potential measurement noise. This method is suitable for estimating derivatives when only discrete data is available.