Pseudo Spectral Methods for Optimal Control - Notes

Abhigyan Roy AE21B002

December 19, 2024

Contents

List of Figures

Introduction

Pseudo Spectral Methods for optimal control are a class of numerical techniques used for solving optimal control problems by approximating the state and control variables using spectral methods.

0.1 Spectral Methods and Pseudospectral Methods

0.1.1 Spectral Methods

Definition: Spectral methods are numerical techniques used for solving differential and integral equations by expanding the solution in terms of globally defined basis functions, such as orthogonal polynomials or trigonometric functions.

Core Idea

The core idea is to represent the solution $u(x)$ as a series:

$$u(x) \approx \sum_{n=0}^{N} a_n \phi_n(x),$$

where $\phi_n(x)$ are the chosen basis functions, and a_n are coefficients determined by projecting the original problem onto the basis functions.

Types of Spectral Methods

- **Fourier Spectral Method**:
  - **Basis Functions**: Trigonometric functions such as sine and cosine.
  - **Application**: Best suited for problems with periodic boundary conditions.
  - **Formulation**:

    $$u(x) \approx \sum_{k=-N/2}^{N/2} \hat{u}_k e^{ikx},$$

    where $\hat{u}_k$ are the Fourier coefficients.
- **Polynomial Spectral Method**: Galerkin and Least Sqaure
  - **Basis Functions**: Orthogonal polynomials, e.g., Chebyshev or Legendre polynomials.
  - **Application**: Suitable for non-periodic boundary conditions.

  - **Formulation**:

$$u(x) \approx \sum_{n=0}^{N} a_n \phi_n(x),$$

    where a_n are coefficients found through projection.

Advantages and Disadvantages

- **Advantages**:
  - High accuracy for smooth problems (exponential convergence).
  - Efficient for global behavior of solutions.
- **Disadvantages**:
  - Not well-suited for non-smooth solutions.
  - More complex for problems with sharp discontinuities.

0.1.2 Pseudospectral Methods

Definition: Pseudospectral methods approximate derivatives and solve differential equations by evaluating the function at selected collocation points, instead of directly manipulating the coefficients in the basis function expansion.

Core Idea

The function $u(x)$ is expanded similarly to spectral methods:

$$u(x) \approx \sum_{n=0}^{N} a_n \phi_n(x).$$

However, the derivatives are computed at discrete points, known as collocation points, leading to a practical discretization.

Collocation Points

Common choices for collocation points include:

- **Chebyshev Nodes**:

$$x_k = \cos\left(\frac{2k-1}{2N}\pi\right), \quad k = 1, 2, \ldots, N.$$

- **Gauss-Lobatto Points**: Nodes used for polynomial interpolation, chosen to enhance numerical stability and minimize errors.

Types of Pseudospectral Methods

- **Fourier Pseudospectral Method**:
  - Utilizes FFT for fast computation.
  - Ideal for periodic problems.
- **Chebyshev Pseudospectral Method**:
  - Uses Chebyshev polynomials and their nodes for derivatives.
  - Suitable for non-periodic problems.

Formulation

For a function $u(x)$ and its derivative $u'(x)$, the derivative at collocation points x_k can be approximated as:

$$u'(x_k) \approx \sum_{j=0}^{N} D_{kj} u(x_j),$$

where D_{kj} is the differentiation matrix based on the basis functions.

Advantages and Disadvantages

- **Advantages**:
  - Flexible for complex boundary conditions.
  - High accuracy, similar to spectral methods, with a practical approach to discretization.
- **Disadvantages**:
  - Computational cost due to dense matrix operations.
  - Sensitive to the choice of collocation points.

0.1.3 Comparison Between Spectral and Pseudospectral Methods

- **Approach to Differentiation**:
  - **Spectral Methods**: Operate in the spectral space by manipulating expansion coefficients.
  - **Pseudospectral Methods**: Operate in the physical space by evaluating derivatives at collocation points.
- **Flexibility**:
  - **Spectral Methods**: Less flexible for complex boundaries.
  - **Pseudospectral Methods**: More adaptable for varied boundary conditions.
- **Implementation**:
  - **Spectral Methods**: Require exact integration for coefficient projection.
  - **Pseudospectral Methods**: Use discrete points and fast algorithms like FFT.

0.2 Introduction

Pseudospectral methods are a class of numerical techniques used primarily for solving differential equations, particularly in the context of optimal control problems and partial differential equations. They combine the strengths of spectral methods and finite difference methods. Spectral methods approximate the solution to a differential equation by expressing it as a sum of global basis functions, typically orthogonal polynomials like Chebyshev polynomials or trigonometric functions. Pseudospectral methods are a specific type of spectral method where the differential equation is enforced at a set of discrete points in the domain, known as collocation points.

The concepts of direct and indirect methods are widely used in optimal control theory and numerical optimization. The two approaches differ fundamentally in the sequence in which they handle the discretization and optimization processes. In direct methods, the problem is first discretized and then optimized.

In this course, we will be using direct methods with a major focus on the discretization part.

0.3 Important Terms

Brief introduction to the important terms we will be using in the discussions.

0.3.1 Optimal Control Problem

Components of an Optimal Control Problem:

1. **State Variables** $x(t)$: These describe the state of the system at any time t. The evolution of these states is typically governed by differential equations.

2. **Control Variables** $u(t)$: These are the inputs or decisions that can be manipulated over time to influence the behavior of the state variables.

3. **Dynamics (State Equations)**:

$$\dot{x}(t) = f(x(t), u(t), t)$$

   This is a set of differential equations describing how the state variables evolve over time as a function of the current state, control inputs, and time.

4. **Cost Function (Objective Function)**:

$$J = \int_{t_0}^{t_f} L(x(t), u(t), t)\, dt + \Phi(x(t_f))$$

   Here, $L(x(t), u(t), t)$ is the running cost (instantaneous cost rate), and $\Phi(x(t_f))$ is the terminal cost at the final time t_f. The goal is to minimize (or maximize) this cost function.

5. **Boundary Conditions**:

$$x(t_0) = x_0 \quad \text{and possibly} \quad x(t_f) = x_f$$

   These specify the initial state (and sometimes the final state) of the system.

6. **Constraints**: These can include state constraints, control constraints, and mixed constraints.

    - **State Constraints**: $g(x(t)) \leq 0$
    - **Control Constraints**: $h(u(t)) \leq 0$
    - **Mixed Constraints**: Involving both state and control variables.

Chapter 1

Pseudospectral Methods

1.1 Least Squares Approach

The **Least Squares approach** is a mathematical method used to find the best-fitting solution to a set of data by minimizing the sum of the squares of the differences between the observed and predicted values. Suppose you have a set of observed data points (x_i, y_i), and you want to fit a model $f(x)$ to these data points. The least squares method minimizes the following sum:

$$S = \sum_{i=1}^{n} (y_i - f(x_i))^2 \tag{1.1}$$

In the least squares method, the approximated function is not guaranteed to be equal to the value of the actual function at the node points.

1.2 Interpolation

Interpolation is a mathematical technique used to estimate unknown values that fall within the range of known data points. It is widely used in numerical analysis, data science, engineering, and other fields where it is necessary to predict values for intermediate points based on a set of discrete data points. Unlike the least squares approach, in interpolation, the modeled function passes through the selected data points.

One common interpolation method is the **Lagrange Interpolation**, which constructs the interpolating polynomial using a linear combination of Lagrange basis polynomials. The interpolating polynomial $P(x)$ is given by:

$$P(x) = \sum_{i=0}^{n} y_i \prod_{\substack{0 \le j \le n \\ j \ne i}} \frac{x - x_j}{x_i - x_j} \tag{1.2}$$

1.3 Class Discussion - MATLAB Plot: Least Squares vs Interpolation

As discussed in class, the interpolation method ensures that the value of the approximated function is exactly the value of the actual function at the selected node points. Figure

1.1 shows the difference between the two approximation methods (Least Squares and Interpolation) via a simple MATLAB plot discussed during the lecture.

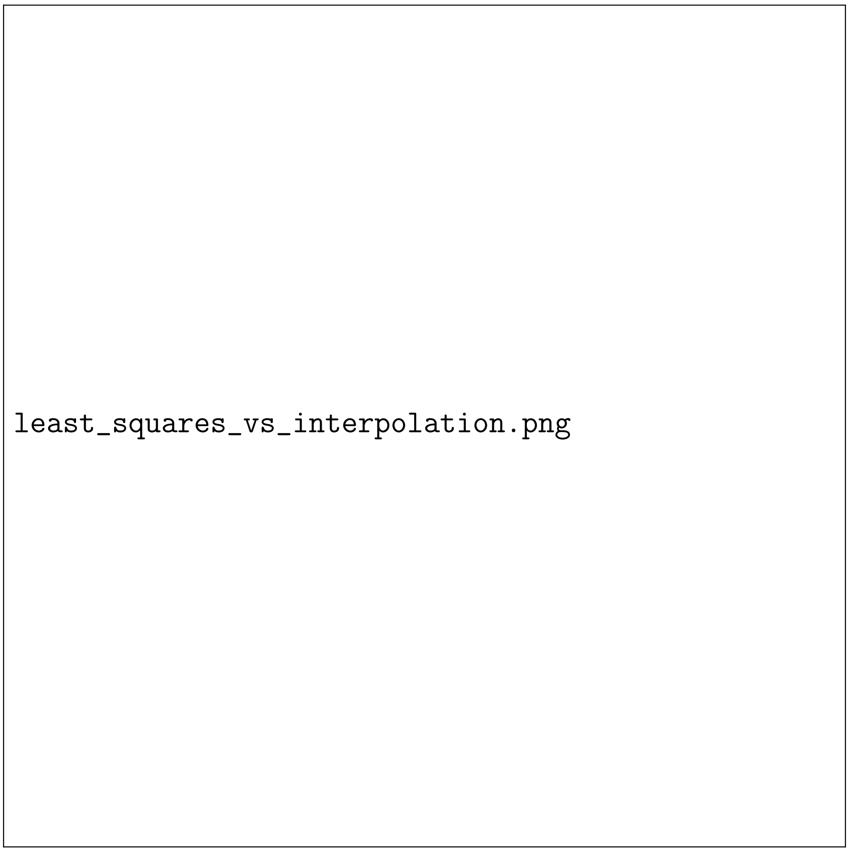


Figure 1.1: Comparison of Least Squares and Interpolation Methods.

1.4 Collocation Method

The collocation method is a technique used to approximate solutions of differential equations by forcing the solution to satisfy the differential equation at a finite number of points called collocation points. For a given set of nodes, the polynomial can be approximated in either of the two ways:

$$P_n(x) = \sum_{i=0}^{n} a_i f(x_i) \tag{1.3}$$

or

$$P_n(x) = \sum_{i=0}^{n} b_i x_i \tag{1.4}$$

Both of these are types of interpolation matrices, and the value of the approximated polynomial is the same as the actual function at the node points.

1.4.1 Vandermonde Matrix

Given a vector of elements $x = [x_1, x_2, \ldots, x_n]$, the Vandermonde matrix V associated with this vector is an $n \times n$ matrix, where each row corresponds to a geometric progression of the elements x_i. Specifically, the matrix is defined as follows:

$$V = \begin{pmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \ldots & x_2^{n-1} \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ 1 & x_n & x_n^2 & \ldots & x_n^{n-1} \end{pmatrix}$$

Each element V_{ij} in the matrix is given by:

$$V_{ij} = x_i^{j-1} \tag{1.5}$$

The determinant of the Vandermonde matrix has a special form and is given by:

$$\det(V) = \prod_{1 \leq i < j \leq n} (x_j - x_i) \tag{1.6}$$

This product is non-zero if and only if all the x_i are distinct, meaning the determinant of the Vandermonde matrix is non-zero (and hence the matrix is invertible) if all the elements in the vector x are distinct. The proof for this will be discussed later.

1.5 Runge Phenomenon

When interpolating a set of points, one common approach is to fit a polynomial that passes through all the given data points. The idea is that the polynomial will closely approximate the underlying function from which the data points are sampled. However, when the function is sampled at equally spaced points and a high-degree polynomial is used, the interpolation error can become quite significant near the endpoints of the interval. This is known as **Runge's Phenomenon**.

1.6 Chebyshev Nodes

Chebyshev nodes are specific points used in numerical interpolation, particularly in polynomial interpolation, to reduce the problems associated with Runge's phenomenon and to achieve more accurate approximations.

Chebyshev nodes are the roots of Chebyshev polynomials of the first kind. For a given interval $[-1, 1]$, the Chebyshev nodes x_i for n interpolation points are defined as:

$$x_i = \cos\left(\frac{2i-1}{2n}\pi\right), \quad i = 1, 2, \ldots, n \tag{1.7}$$

These nodes are spaced more densely near the endpoints of the interval $[-1, 1]$ and more sparsely near the center. When interpolating over a different interval $[a, b]$, you can linearly transform the Chebyshev nodes from $[-1, 1]$ to $[a, b]$ using the transformation:

$$x_i' = \frac{b-a}{2}(x_i + 1) + a$$

1.7 Weierstrass Theorem

The **Weierstrass Approximation Theorem** states that for every continuous function f defined on a closed interval $[a, b]$, and for every $\epsilon > 0$, there exists a polynomial $P(x)$ such that:

$$\|f(x) - P(x)\| < \epsilon \tag{1.8}$$

for all x in $[a, b]$. In other words, any continuous function on a closed interval can be uniformly approximated as closely as desired by a polynomial function. The proof of the Weierstrass Approximation Theorem typically involves constructing a sequence of polynomials that converge uniformly to the given continuous function. One standard approach uses Bernstein polynomials, which are constructed from binomial coefficients and shown to converge uniformly to the function $f(x)$.

1.8 Trapezoidal Method

The **Trapezoidal Method** is used to approximate the area under a curve by dividing the region into a series of trapezoids. The area of each trapezoid is computed and summed to give an approximation of the definite integral.

Consider a function $f(x)$ defined on the interval $[a, b]$. The goal is to approximate the definite integral:

$$\int_a^b f(x)dx \tag{1.9}$$

Using the trapezoidal method, the interval $[a, b]$ is divided into n subintervals of equal width:

$$h = \frac{b-a}{n}$$

The points dividing the interval are:

$$x_0 = a, \quad x_1 = a + h, \quad x_2 = a + 2h, \ldots, x_n = b$$

The trapezoidal rule then approximates the integral as:

$$\int_a^b f(x)dx \approx \frac{h}{2}\left[f(x_0) + 2f(x_1) + 2f(x_2) + \cdots + 2f(x_{n-1}) + f(x_n)\right] \tag{1.10}$$

This formula sums the function values at the endpoints and the midpoints, weighted appropriately, to give a numerical estimate of the integral.

1.9 Trapezoidal Method

1.9.1 Example

The Trapezoidal method provides good accuracy for periodic functions and becomes exact after a certain number of nodes are used for discretization. (The proof for this can be found in the class notes on Moodle.) Figure 1.2 demonstrates the accuracy of the trapezoidal method, as shown by the MATLAB code discussed in class.

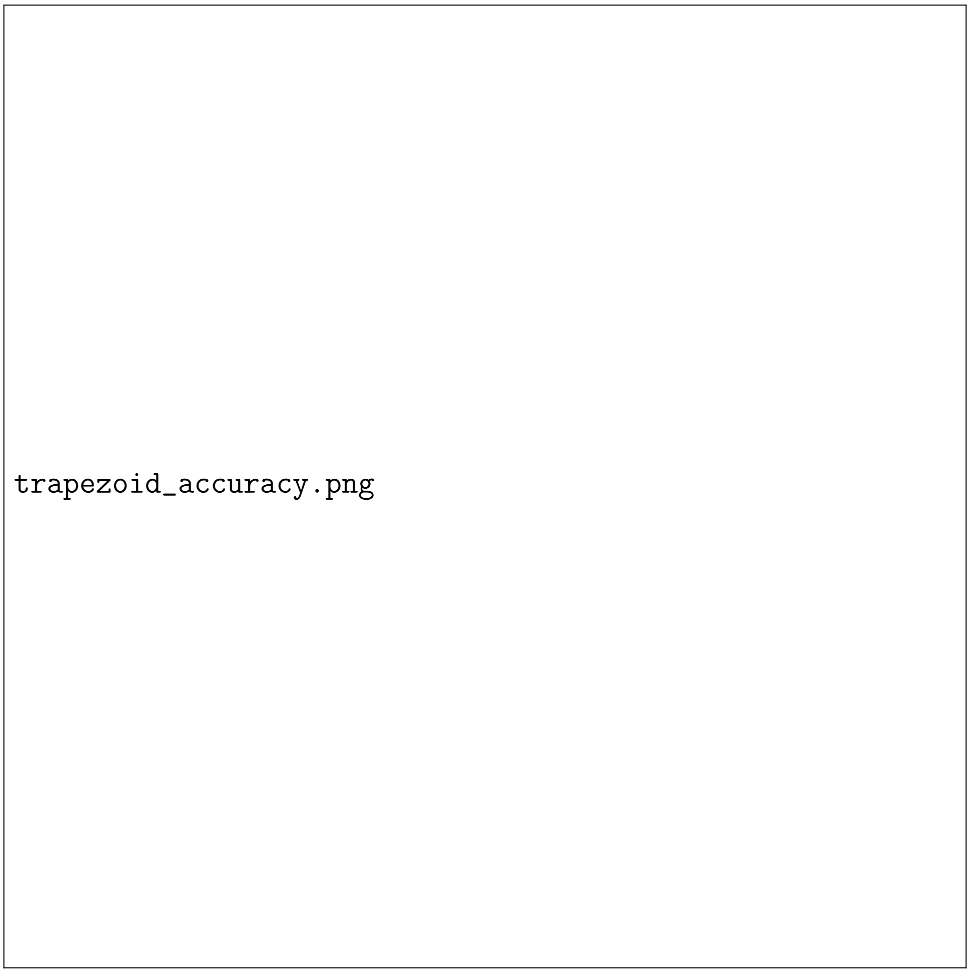


Figure 1.2: Trapezoid method accuracy.

1.10 Central Difference Method

The Central Difference method is a finite difference scheme used to approximate derivatives by expanding around two neighboring points. It is second-order accurate, meaning it approximates derivatives to second-order accuracy in terms of the grid spacing Δx.

1.10.1 Expansion Around $x_{i+1} = x_i + \Delta x$

By expanding the function $f(x)$ around $x_{i+1} = x_i + \Delta x$, we obtain the following Taylor series expansion:

$$f(x_{i+1}) = f(x_i+\Delta x) = f(x_i)+\frac{f'(x_i)}{1!}\Delta x+\frac{f''(x_i)}{2!}(\Delta x)^2+\frac{f'''(x_i)}{3!}(\Delta x)^3+O((\Delta x)^4) \quad (1.11)$$

1.10.2 Expansion Around $x_{i-1} = x_i - \Delta x$

Similarly, expanding the function around $x_{i-1} = x_i - \Delta x$, we get:

$$f(x_{i-1}) = f(x_i - \Delta x) = f(x_i) - \frac{f'(x_i)}{1!}\Delta x + \frac{f''(x_i)}{2!}(\Delta x)^2 - \frac{f'''(x_i)}{3!}(\Delta x)^3 + O((\Delta x)^4) \tag{1.12}$$

1.10.3 Central Difference Formula

Taking the difference between $f(x_{i+1})$ and $f(x_{i-1})$, we get:

$$f(x_{i+1}) - f(x_{i-1}) = 2f'(x_i)\Delta x + \frac{2f'''(x_i)}{6}(\Delta x)^3 + O((\Delta x)^4) \tag{1.13}$$

Approximating over small intervals, the first derivative $f'(x_i)$ is approximated as:

$$f'(x_i) \approx \frac{f(x_{i+1}) - f(x_{i-1})}{2\Delta x} \tag{1.14}$$

Thus, the central difference scheme provides exact results if the highest order of the function is cubic.

1.10.4 Higher-Order Central Difference

By increasing the size of the stencil (i.e., including more neighboring points), we can reduce the approximation error and improve the accuracy of the method. As the stencil size increases, the error tends toward zero for higher-order functions.

1.11 Example Problem

Consider a block with the following boundary conditions:

$$x(0) = 0, \quad v(0) = 0, \quad x(1) = 1, \quad v(1) = 0$$

The system is governed by the equations:

$$\dot{x} = v, \quad \dot{v} = u$$

The objective is to optimize the following function:

$$J = \int_0^1 U(x)^2 \, dx$$

Using the trapezoidal method, we can approximate the integral as:

$$J = \left(\frac{U_0^2}{2} + U_1^2 + \cdots + U_{n-1}^2 + \frac{U_n^2}{2} \right) \Delta t$$

Two Methods for Non-Linear Equality Constraints:

1. Central Difference Method In the central difference method, the constraint conditions are formed as follows for the interior points:

$$\frac{x_{k+1} - x_{k-1}}{2h} = v_k, \quad \frac{v_{k+1} - v_{k-1}}{2h} = u_k$$

For the boundary points, forward and backward Euler methods are used to satisfy the conditions at the boundaries.

2. Trapezoidal Method In the trapezoidal method, the constraint conditions are written as:

$$x_{k+1} - x_k = \frac{v_{k+1} + v_k}{2}\Delta t, \quad v_{k+1} - v_k = \frac{u_{k+1} + u_k}{2}\Delta t$$

The full implementation of this method can be found in the `nonlincon.m` file, as discussed in class.

To perform the optimization, we use the `fmincon` function from MATLAB. `fmincon` is used for solving constrained nonlinear optimization problems, and it finds the minimum of a scalar function subject to the specified constraints.

1.12 Discussion of MATLAB and Mathematica Presentation in Class

With the same boundary conditions as the previous problem, a related problem discussed in class has 5 unknowns and 4 conditions, making it an optimization problem. This requires the use of similar numerical methods for solution.

1.13 Rolle's Theorem

Rolle's Theorem is a fundamental result in calculus that provides a specific condition under which a function must have a horizontal tangent line (or a derivative equal to zero) at least once within a given interval.

The theorem states that if a continuous function $f(x)$ is differentiable on the open interval (a, b) and satisfies $f(a) = f(b)$, then there exists at least one point $c \in (a, b)$ such that:

$$f'(c) = 0$$

This result is often used in optimization and root-finding algorithms.

1.14 Rolle's Theorem

Rolle's Theorem is a fundamental result in calculus that provides a specific condition under which a function must have a horizontal tangent line (or a derivative equal to zero) at least once within a given interval. It's a special case of the Mean Value Theorem.

Let $f(x)$ be a function that satisfies the following three conditions:

1. **Continuity:** $f(x)$ is continuous on the closed interval $[a, b]$.

2. **Differentiability:** $f(x)$ is differentiable on the open interval (a, b).

3. **Equal Endpoints:** $f(a) = f(b)$.

If these conditions are met, then there exists at least one point $c \in (a, b)$ such that the derivative of $f(x)$ at c is zero:

$$f'(c) = 0$$

1.15 Derivatives of Lagrange Polynomial

Consider a function approximated with a second-order Lagrange polynomial using equally spaced nodes at 0, h, and $2h$. The polynomial can be expressed as:

$$P(x) = y_0 \cdot \ell_0(x) + y_1 \cdot \ell_1(x) + y_2 \cdot \ell_2(x)$$

Where the Lagrange basis polynomials are defined as follows:

$$\ell_0(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)}$$

$$\ell_1(x) = \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)}$$

$$\ell_2(x) = \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)}$$

The derivative of the polynomial $P'(x)$ is given by:

$$P'(x) = y_0 \cdot \frac{2x - (x_1 + x_2)}{(x_0 - x_1)(x_0 - x_2)} + y_1 \cdot \frac{2x - (x_0 + x_2)}{(x_1 - x_0)(x_1 - x_2)} + y_2 \cdot \frac{2x - (x_0 + x_1)}{(x_2 - x_0)(x_2 - x_1)}$$

To represent the derivatives at the node points in matrix format, we can express it as:

$$\begin{bmatrix} P'(x_0) \\ P'(x_1) \\ P'(x_2) \end{bmatrix} = \begin{bmatrix} -\frac{3}{2h} & \frac{2}{h} & -\frac{1}{2h} \\ -\frac{1}{2h} & 0 & \frac{1}{h} \\ \frac{1}{h} & -\frac{2}{h} & \frac{3}{2h} \end{bmatrix} \begin{bmatrix} P(x_0) \\ P(x_1) \\ P(x_2) \end{bmatrix}$$

This shows that the derivatives of the approximated function at the node points are a linear combination of the actual function values at the node points.

1.16 Controlling Error by Altering the Location of Nodes

Claim: When we approximate a function using a polynomial of order n, the error in the representation can be altered by changing the position of the nodes (collocation points).

Proof: Consider the error term defined as:

$$W(t) = f(t) - p_n(f; t) - (t - x_0)(t - x_1) \cdots (t - x_n)K(x)$$

Where $K(x)$ is defined as:

$$K(x) = \frac{f(x) - p_n(f;x)}{(x-x_0)(x-x_1)\cdots(x-x_n)}$$

For an n-th order polynomial, we have $(n+1)$ variables. The function W is zero at:

- $t = x_i$ for all $i = 0, 1, \ldots, n$ (the node points)
- $t = x$

By applying Rolle's Theorem, there exists some ξ in the interval between x_0 and x_n such that:

$$W^{(n+1)}(t) = f^{(n+1)}(t) - (n+1)!K(x)$$

Thus, at point ξ:

$$0 = W^{(n+1)}(\xi) = f^{(n+1)}(\xi) - (n+1)!K(x)$$

From this, we can express $K(x)$ as:

$$K(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi)$$

Consequently, the error $e(x)$ can be written as:

$$e(x) = f(x) - p_n(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi)(x-x_0)(x-x_1)\cdots(x-x_n)$$

This shows that while we do not have direct control over $f^{(n+1)}$ to reduce the error, we can alter the location of the nodes to change the error.

Figure 4 shows the class representation in Mathematica, illustrating that for a function approximated using a second-order polynomial, taking the derivative at the central control point is the same as applying the central difference at that point.

1.17 Difference Between a Polynomial and Its nth Order Interpolation Approximation

The error in approximating a function $f(x)$ using an nth order polynomial $p_n(x)$ can be expressed as:

$$e(x) = f(x) - p_n(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi)(x-x_0)(x-x_1)\cdots(x-x_n)$$

From MATLAB simulations, we found that the optimal node locations for approximating a 2D polynomial with a linear Lagrange polynomial (in the symmetric case) are as follows: - For least RMS error: $x_0 = -\sqrt{\frac{1}{3}}$ and $x_1 = \sqrt{\frac{1}{3}}$ - For least maximum error: $x_0 = -\sqrt{\frac{1}{2}}$ and $x_1 = \sqrt{\frac{1}{2}}$

Figure 5 shows the Mathematica representation of these findings as discussed in class. A similar analysis applies to a 3D polynomial; refer to the MATLAB code for the optimal node points for a third-order polynomial function.

Example Problem

Consider a polynomial function X_{n+1} represented using an nth order polynomial:

$$P_n = \sum_{i=0}^{n} a_i x^i$$

We need to find the leading coefficient of the approximation, a_n. Equally spaced node points are chosen from 0 to 1 (including both endpoints).

Solution: Using the remainder theorem:

$$X_{n+1} - \sum_{i=0}^{n} a_i x^i = A(x-0)(x-\frac{1}{n})(x-\frac{2}{n})\cdots(x-\frac{n-1}{n})(x-1)$$

Here, A comes out to be 1. Comparing the coefficients on the left-hand side and the right-hand side, we find the leading coefficient $a_n = \frac{n+1}{2}$.

1.18 Condition for Vandermonde Matrix Being Invertible

The Vandermonde matrix is always invertible except when two node points are the same. To understand this, we can observe the Vandermonde matrix defined as:

$$V = \begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{pmatrix}$$

If two node points become the same, two rows of the matrix become identical, causing the determinant to be zero, thus making the matrix non-invertible.

Now, consider the Vandermonde matrix with the last row replaced by x instead of x_n:

$$V = \begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x & x^2 & \cdots & x^n \end{pmatrix}$$

If x takes any of the values $x_0, x_1, \ldots, x_{n-1}$, the determinant is zero. Thus, the determinant can be expressed as:

$$\det(V) = A \prod_{k=0}^{n-1} (x_n - x_k)$$

Where A is a coefficient that does not contain any x term. Further observation shows that A is the determinant of a Vandermonde matrix with n terms, applying similar operations to x_n for x_{n-1} now:

$$\det(V) = (x_1 - x_0) \prod_{k=0}^{1} (x_2 - x_k) \prod_{k=0}^{2} (x_3 - x_k) \cdots \prod_{k=0}^{n-1} (x_n - x_k)$$

This form proves that unless two nodes are identical, the determinant of the Vandermonde matrix will exist.

1.19 Lagrange Basis Polynomial

1.19.1 Cramer's Rule

Consider a system of linear equations:

$$Ax = b$$

where: - A is an $n \times n$ matrix of coefficients, - x is a column vector of unknowns, - b is a column vector of constants.

Cramer's Rule states that if $\det(A) \neq 0$, the solution to the system is given by:

$$x_i = \frac{\det(A_i)}{\det(A)} \quad \text{for } i = 1, 2, \ldots, n,$$

where A_i is obtained by replacing the i-th column of A with b.

Proof

For a particular m, consider:

$$x_m = \sum_{j=0}^{n} W(x_j) x_{j,m}$$

where $0 \leq m \leq n$. Note that even though for all m less than n, it appears that the right-hand side will have a higher power of x than m due to the Lagrange polynomial bases, the terms cancel out to match the power on the left-hand side.

The aim is to prove that

$$P_n(x) = \sum_{i=0}^{n} a_i x^i$$

can be represented as

$$P_n(x) = \sum_{i=0}^{n} W_i f(x_i)$$

where the W_i are the Lagrange bases.

Thus,

$$P_n(x) = \sum_{k=0}^{n} a_k (x_k)$$

can be rearranged as follows:

$$P_n(x) = \sum_{k=0}^{n} a_k \sum_{j=0}^{n} W(x_j) x_k^j$$

Rearranging gives:

$$P_n(x) = \sum_{j=0}^{n} W(x_j) \sum_{k=0}^{n} a_k x_j^k$$

Therefore,

$$P_n(x) = \sum_{j=0}^{n} W(x_j) P(x_j)$$

To find the Lagrange basis, we write the system of equations in a matrix format:

$$\begin{pmatrix} 1 & 1 & \cdots & 1 & \\ x_1 & x_2 & \cdots & x_n & \\ x_1^2 & x_2^2 & \cdots & x_n^2 & \\ \vdots & \vdots & \ddots & \vdots & \\ x_{n-1} & 1 & x_{n-1}^2 & \cdots & x_{n-1}^n \end{pmatrix} \begin{pmatrix} W_0 \\ W_1 \\ W_2 \\ \vdots \\ W_n \end{pmatrix} = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

Notably, the Vandermonde matrix V is given by:

$$V^T = \begin{pmatrix} 1 & 1 & \cdots & 1 & \\ x_1 & x_2 & \cdots & x_n & \\ x_1^2 & x_2^2 & \cdots & x_n^2 & \\ \vdots & \vdots & \ddots & \vdots & \\ x_{n-1} & 1 & x_{n-1}^2 & \cdots & x_{n-1}^n \end{pmatrix}$$

Using Cramer's rule, we know that:

$$W_j = \frac{\Delta_j}{\Delta}$$

where Δ is the determinant of the Vandermonde matrix and Δ_j is the determinant of the Vandermonde matrix with the elements x_j replaced with x.

To find the Lagrange polynomials:

$$\Delta = \det(V) = (x_1 - x_0) \prod_{k=1}^{n-1} (x_2 - x_k) \prod_{k=2}^{n-1} (x_3 - x_k) \cdots \prod_{k=n-1}^{n-1} (x_n - x_k)$$

Separating all terms that have an x_j in them:

$$\det(V) = \lambda_j \prod_{k=0}^{j-1} (x_j - x_k) \prod_{k=j+1}^{n} (x_k - x_j)$$

Here, λ_j is the part of the determinant that does not have any x_j terms. Replacing x_j with x:

$$\Delta_j = \det(V) = \lambda_j \prod_{k=0}^{j-1} (x - x_k) \prod_{k=j+1}^{n} (x_k - x)$$

Thus,

$$W(x_j) = \ell_j(x) = \prod_{\substack{0 \le k \le n \\ k \neq j}} \frac{x - x_k}{x_j - x_k}$$

We can conclude that all the Lagrange basis polynomials are of the same order, and for $n+1$ node points over which the polynomial is approximated, the basis function will be of order n. Note that the Lagrange basis polynomial ℓ_j has a value of 1 at the node x_j and zero at all other node points. The polynomial cannot have more zeros than the number of remaining node points, as verified from the degree of the polynomial.

1.19.2 Important Points

Intuitive Understanding of Why the Sum of Weights is One

Every approximation of a function must be able to model a constant function as well. Thus, the sum of weights or the coefficients must equal one. This can be mathematically represented as:

$$P(x) = \sum_{i=0}^{n} W_k(x) f(x_k)$$

For a constant function, say $f(x) = c$:

$$P(x) = c = \sum_{i=0}^{n} W_k(x) c$$

Canceling c on both sides gives:

$$\sum_{i=0}^{n} W_k(x) = 1$$

1.20 Derivatives of Lagrange Polynomial/ Differentiation Matrix

The derivative of a Lagrange polynomial can be computed using specific formulas, which are essential in interpolation problems.

Chapter 2

Runge Phenomena

The Runge phenomenon refers to the large oscillations that can occur when using polynomial interpolation at equidistant points.

2.1 Controlling error by controlling location of nodes

By strategically choosing the locations of interpolation nodes, one can minimize the error in polynomial interpolation.

2.2 Chebyshev Nodes

Chebyshev nodes are specific points used in polynomial interpolation to minimize the interpolation error, particularly to avoid the Runge phenomenon.

Chapter 3

Gauss Quadrature

Chapter 4

Complex Analysis

4.1 Brief Theory Introduction

4.1.1 Limits

A complex function $f(z)$ is said to have a limit L as $z \to z_0$ if:

$$\lim_{z \to z_0} f(z) = L,$$

where L is a complex number. This implies that for every $\epsilon > 0$, there exists a $\delta > 0$ such that $|f(z) - L| < \epsilon$ whenever $0 < |z - z_0| < \delta$.

4.1.2 Derivatives

The derivative of $f(z)$ at $z = z_0$ is given by:

$$f'(z_0) = \lim_{z \to z_0} \frac{f(z) - f(z_0)}{z - z_0},$$

provided the limit exists. A function is holomorphic if it is differentiable everywhere in its domain.

4.1.3 Analytic Functions and Cauchy-Riemann Equations

A function $f(z) = u(x, y) + iv(x, y)$, where $u(x, y)$ and $v(x, y)$ are real-valued functions, is said to be analytic in a domain D if it is differentiable at every point in D. Differentiability in the complex sense implies that the limit:

$$f'(z) = \lim_{\Delta z \to 0} \frac{f(z + \Delta z) - f(z)}{\Delta z}$$

exists and is independent of the direction from which $\Delta z \to 0$.

For $f(z)$ to be analytic, it is necessary and sufficient that:

1. $u(x, y)$ and $v(x, y)$ have continuous first-order partial derivatives, and

2. The Cauchy-Riemann equations are satisfied:

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y}, \quad \frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x}.$$

If these conditions hold, $f(z)$ is both holomorphic (complex differentiable) and harmonic (satisfying Laplace's equation):

$$\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0, \quad \Delta v = \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} = 0.$$

Functions that are analytic throughout the entire complex plane are called *entire* functions. Examples include e^z, $\sin z$, and $\cos z$.

4.1.4 Complex Integration

For a curve Γ parameterized by $z(t) = x(t) + iy(t)$, the integral of $f(z)$ along Γ is:

$$\int_\Gamma f(z)\,dz = \int_a^b f(z(t))z'(t)\,dt.$$

4.1.5 ML Inequality

If $f(z)$ is continuous on a contour Γ, then:

$$\left| \int_\Gamma f(z)\,dz \right| \leq M \cdot L,$$

where M is the maximum of $|f(z)|$ on Γ and L is the length of Γ.

4.1.6 Cauchy Integral Theorem

Simply Connected Domains

The Cauchy Integral Theorem states that if $f(z)$ is analytic in a simply connected domain D and Γ is any closed contour within D, then:

$$\int_\Gamma f(z)\,dz = 0.$$

A domain is ***simply connected*** if any closed contour in the domain can be continuously shrunk to a point without leaving the domain (i.e., the domain has no "holes").

Multiply Connected Domains

For a multiply connected domain, which contains holes or excluded regions, the theorem requires special handling of singularities. Let Γ be a positively oriented closed contour, and let there be n disjoint regions within Γ, each bounded by negatively oriented closed contours $\Gamma_1, \Gamma_2, \ldots, \Gamma_n$. If $f(z)$ is analytic in the multiply connected domain but may have singularities inside Γ, the integral around Γ is related to the sum of integrals around the excluded regions:

$$\int_\Gamma f(z)\,dz = \int_{\Gamma_1} f(z)\,dz + \int_{\Gamma_2} f(z)\,dz + \cdots + \int_{\Gamma_n} f(z)\,dz.$$

If the function $f(z)$ is analytic everywhere except at isolated singularities inside Γ, the integral can be evaluated using the Residue Theorem:

$$\int_\Gamma f(z)\, dz = 2\pi i \sum \text{Residues of } f(z) \text{ inside } \Gamma.$$

This approach allows us to handle more complex domains by effectively "subtracting out" the effects of the excluded regions or singularities.

Practical Example for Multiply Connected Domains

Consider $f(z) = \frac{1}{z}$ in the annular region $1 < |z| < 2$. Let Γ_1 be the circle $|z| = 2$ (positively oriented) and Γ_2 be the circle $|z| = 1$ (negatively oriented). Then:

$$\int_{\Gamma_1} \frac{1}{z}\, dz + \int_{\Gamma_2} \frac{1}{z}\, dz = 2\pi i - 2\pi i = 0.$$

Here, the contribution from each contour cancels, demonstrating the nature of integrals in multiply connected domains.

4.1.7 Cauchy Integral Formula

If $f(z)$ is analytic inside and on a simple closed contour Γ, and z_0 is inside Γ, then:

$$f(z_0) = \frac{1}{2\pi i} \int_\Gamma \frac{f(z)}{z - z_0}\, dz.$$

4.1.8 Taylor and Laurent Series

Taylor Series

If $f(z)$ is analytic in a disk $|z - z_0| < R$, it can be expressed as a Taylor series centered at z_0:

$$f(z) = \sum_{n=0}^{\infty} a_n (z - z_0)^n,$$

where the coefficients a_n are given by:

$$a_n = \frac{1}{n!} \frac{d^n f}{dz^n}(z_0).$$

1. **Radius of Convergence:** The series converges for $|z - z_0| < R$, where R is the distance from z_0 to the nearest singularity.

2. **Application:** Taylor series are commonly used to approximate functions locally around z_0.

Laurent Series

If $f(z)$ is analytic in an annular region $R_1 < |z - z_0| < R_2$, it can be expressed as a Laurent series:

$$f(z) = \sum_{n=-\infty}^{\infty} a_n (z - z_0)^n,$$

where the coefficients a_n are given by:

$$a_n = \frac{1}{2\pi i} \int_\Gamma \frac{f(w)}{(w - z_0)^{n+1}} \, dw,$$

and Γ is a positively oriented contour within the annular region.

1. **Principal Part:** The terms $\sum_{n=-\infty}^{-1} a_n (z - z_0)^n$ describe the behavior near singularities.
2. **Regular Part:** The terms $\sum_{n=0}^{\infty} a_n (z - z_0)^n$ represent the analytic behavior.

Comparison of Taylor and Laurent Series

- Taylor series are valid in simply connected domains without singularities.
- Laurent series are used in regions containing isolated singularities, combining both regular and singular behavior.

Example: Expansion Around Singularities

For $f(z) = \frac{1}{z(z-1)}$:

1. **At $z_0 = 0$:** The Laurent series in $|z| < 1$ is:

$$f(z) = \frac{1}{z} - 1 - z - z^2 - \cdots .$$

2. **At $z_0 = 1$:** The Laurent series in $|z - 1| < 1$ is:

$$f(z) = \frac{1}{z - 1} + \frac{1}{z - 1} \sum_{n=1}^{\infty} z^{-n}.$$

4.1.9 Residue Theorem

The Residue Theorem provides an efficient way to evaluate complex integrals. If $f(z)$ is analytic in a domain D except for isolated singularities $z_1, z_2, \ldots, z_n$ inside a closed contour Γ, then:

$$\int_\Gamma f(z) \, dz = 2\pi i \sum_{k=1}^{n} \text{Res}(f, z_k),$$

where $\text{Res}(f, z_k)$ is the residue at z_k.

Residue at Isolated Singularities

1. **Simple Pole at z_0:** For a simple pole at z_0,

$$\text{Res}(f, z_0) = \lim_{z \to z_0} (z - z_0) f(z).$$

2. **Pole of Order m at z_0:** For a pole of order m,

$$\text{Res}(f, z_0) = \frac{1}{(m-1)!} \lim_{z \to z_0} \frac{d^{m-1}}{dz^{m-1}} \left[(z - z_0)^m f(z) \right].$$

Applications of Residue Theorem

1. **Computing Contour Integrals:** Evaluate $\int_\Gamma \frac{e^z}{z(z-1)}\, dz$, where Γ encloses $z = 0$ and $z = 1$:

$$\text{Res}\left(\frac{e^z}{z(z-1)}, z = 0 \right) = 1,$$
$$\text{Res}\left(\frac{e^z}{z(z-1)}, z = 1 \right) = e.$$

Then:

$$\int_\Gamma \frac{e^z}{z(z-1)}\, dz = 2\pi i \left(1 + e\right).$$

2. **Evaluating Real Integrals:** The integral $\int_{-\infty}^{\infty} \frac{\cos(x)}{x^2+1}\, dx$ can be converted to a contour integral using:

$$f(z) = \frac{e^{iz}}{z^2+1}.$$

Using residues at $z = \pm i$, we find the value of the integral.

4.2 Usage in PseudoSpecteal Methods

$$f(x) - P_n(x) = \frac{1}{2\pi i} \oint_C \phi(x, z) dz$$

Assume that the poles of ϕ are x and x_k, where k = 0, 1, ... , n

Chapter 5

Weistrass Theorem

The Weistress approximation theorem states that any continuous function defined on a closed interval can be uniformly approximated by polynomial functions to any degree of accuracy.

Weierstrass Theorem (Bernstein polynomial)

$$\exists\ P_n(x) \text{ such that } |f(x) - P_n(x)| < \epsilon$$

$$f(x),\ x \in [0,1]$$

$$|f(x) - f(x_0)| < \epsilon$$

$$x \in [x_0 - \delta, x_0 + \delta] \quad \text{closed interval (can go to } \infty \text{ at endpoints)}$$

$$(a+b)^n = \sum_{k=0}^{n} \binom{n}{k} a^k b^{n-k}$$

$$(a + (1-x))^n = \sum_{k=0}^{n} \binom{n}{k} x^k (1-x)^{n-k} = 1$$

Bernstein polynomial terms: $B_k(x)$

$$B_n(f,x) = \sum_{k=0}^{n} f\left(\frac{k}{n}\right) b_k(x)$$

$$\Rightarrow B_n(f) \to f$$

$$|B_n(f,x) - f(x)| = \left| \sum_{k=0}^{n} \left(f\left(\frac{k}{n}\right) - f(x) \right) b_k(x) \right|$$

Trying to find upper bound for this

$$|x - \frac{k}{n}| \leq \delta$$

$$f(x) \leq M \quad \text{bounded and continuous in interval taken}$$

$$|f(x) - f(x_0)| \leq 2M$$

Further Analysis on Bernstein Polynomials

$$\left|\sum_{k=0}^{n}\left(f\left(\frac{k}{n}\right)-f(x)\right)b_k(x)\right|=\sum_{k=0}^{n}\left|f\left(\frac{k}{n}\right)-f(x)\right|b_k(x)$$

$$|B_n(f,x)-f(x)|\leq\sum_{|x-\frac{k}{n}|\leq\delta}\left|f\left(\frac{k}{n}\right)-f(x)\right|b_k(x)+\sum_{|x-\frac{k}{n}|>\delta}\left|f\left(\frac{k}{n}\right)-f(x)\right|b_k(x)$$

$$\leq\sum_{|x-\frac{k}{n}|\leq\delta}\frac{\epsilon}{2}b_k(x)+\sum_{|x-\frac{k}{n}|>\delta}\frac{2M}{n\delta^2}b_k(x)$$

$$\leq\sum_{|x-\frac{k}{n}|\leq\delta}\frac{\epsilon}{2}b_k(x)+\sum_{|x-\frac{k}{n}|>\delta}\frac{2M}{n\delta^2}b_k(x)$$

Using bounds and conditions, we proceed as follows:

$$=\sum_{|x-\frac{k}{n}|\leq\delta}\frac{\epsilon}{2}b_k(x)+\frac{2M}{n\delta^2}\sum_{|x-\frac{k}{n}|>\delta}b_k(x)$$

Expanding on this:

$$=\frac{\epsilon}{2}\sum b_k(x)+\frac{2M}{n\delta^2}\sum\left(\frac{k}{n}-x\right)^2b_k(x)$$

Define parameters for probability and expectation:

$$p=nx,\quad\sigma=\sqrt{nx(1-x)},\quad E[(k-\mu)^2]=\sigma^2=E[k^2]-E[k]^2$$

$$E[k]=\sum k\cdot p(k),\quad\sigma^2=E[(k-\mu)^2]$$

Where:

$$E[(k-\mu)^2]=E[k^2]-E[k]^2\quad\text{with}\quad\mu=nx$$

$$B_n(f,x)=\sum_{k=0}^{n}f\left(\frac{k}{n}\right)\binom{n}{k}x^k(1-x)^{n-k}$$

$$|B_n(f,x)-f(x)|=\left|\sum_{k=0}^{n}\left(f\left(\frac{k}{n}\right)-f(x)\right)\binom{n}{k}x^k(1-x)^{n-k}\right|$$

$$\leq\sum\left|f\left(\frac{k}{n}\right)-f(x)\right|\binom{n}{k}x^k(1-x)^{n-k}$$

$$=\sum_{|x-\frac{k}{n}|\leq\delta}\left|f\left(\frac{k}{n}\right)-f(x)\right|\binom{n}{k}x^k(1-x)^{n-k}+\sum_{|x-\frac{k}{n}|>\delta}\left|f\left(\frac{k}{n}\right)-f(x)\right|\binom{n}{k}x^k(1-x)^{n-k}$$

Define these two regions as:

$$A = \sum_{|x-\frac{k}{n}|\leq\delta} \left| f\left(\frac{k}{n}\right) - f(x) \right| \binom{n}{k} x^k(1-x)^{n-k} \leq \frac{\epsilon}{2}$$

$$B = \sum_{|x-\frac{k}{n}|>\delta} \left| f\left(\frac{k}{n}\right) - f(x) \right| \binom{n}{k} x^k(1-x)^{n-k}$$

For B:

$$\leq \sum_{k\in B} \frac{2M}{n^2\delta^2} \binom{n}{k} x^k(1-x)^{n-k}$$

Using the probability statistics:

$$\leq \frac{\epsilon}{2} + \frac{2M}{n\delta^2}$$

We adjust so that:

$$\frac{M}{2n\delta} = \frac{\epsilon}{2} \Rightarrow \epsilon$$

Thus:

$$|B_n(f,x) - f(x)| \leq \epsilon$$

Hence, this justifies that interpolation is valid.

$$\int_0^1 h(t)\,dt$$

And constraints:

$$\mathbf{Dx} = A$$

$$\int_0^T h(t)\,dt \approx \int_0^T \sum_{k=0}^{n} h(t_k)L_k(t)\,dt = \sum_{k=0}^{n} h(t_k) \underbrace{\int_0^T L_k(t)\,dt}_{w_k}$$

$$h \in P_n$$

can this also be reversed order to make it exact, not approximating the function.

we're approximating the integral to a higher

5.1 Bernstein Polynomials and Their Applications

Definition: Bernstein polynomials are a particular set of polynomials that are used to approximate continuous functions on a closed interval $[0, 1]$. They are fundamental in approximation theory and are used to prove the Weierstrass approximation theorem, which states that any continuous function can be uniformly approximated by polynomials.

5.1.1 Mathematical Definition

Given a function $f(x)$ defined on the interval $[0, 1]$, the Bernstein polynomial $B_n(f, x)$ of degree n is defined as:

$$B_n(f, x) = \sum_{k=0}^{n} f\left(\frac{k}{n}\right)\binom{n}{k}x^k(1-x)^{n-k},$$

where $\binom{n}{k}$ is the binomial coefficient.

5.1.2 Properties

- **Pointwise Convergence**: The sequence $B_n(f, x)$ converges to $f(x)$ as $n \to \infty$ for any continuous function $f(x)$ on $[0, 1]$.
- **Preservation of Monotonicity**: If $f(x)$ is a monotonically increasing or decreasing function, then so is $B_n(f, x)$.
- **Preservation of Convexity**: If $f(x)$ is convex, then $B_n(f, x)$ is also convex for sufficiently large n.

5.1.3 Weierstrass Approximation Theorem

The Weierstrass approximation theorem states that for any continuous function $f(x)$ defined on a closed interval $[0, 1]$ and for any $\epsilon > 0$, there exists a polynomial $P_n(x)$ such that:

$$|f(x) - P_n(x)| < \epsilon \quad \text{for all } x \in [0, 1].$$

Using Bernstein polynomials, this theorem can be demonstrated by showing that:

$$\lim_{n\to\infty} B_n(f, x) = f(x).$$

5.1.4 Derivation and Analysis

To show the convergence:

$$|B_n(f, x) - f(x)| = \left|\sum_{k=0}^{n}\left(f\left(\frac{k}{n}\right) - f(x)\right)\binom{n}{k}x^k(1-x)^{n-k}\right|.$$

For small $\delta > 0$, we can split the summation into two parts:

$$\sum_{|x-\frac{k}{n}|\le\delta}\left|f\left(\frac{k}{n}\right) - f(x)\right|\binom{n}{k}x^k(1-x)^{n-k} + \sum_{|x-\frac{k}{n}|>\delta}\left|f\left(\frac{k}{n}\right) - f(x)\right|\binom{n}{k}x^k(1-x)^{n-k}.$$

Upper Bound Analysis:

$$|B_n(f, x) - f(x)| \le \sum_{|x-\frac{k}{n}|\le\delta}\frac{\epsilon}{2}\binom{n}{k}x^k(1-x)^{n-k} + \sum_{|x-\frac{k}{n}|>\delta}\frac{2M}{n\delta^2}\binom{n}{k}x^k(1-x)^{n-k}.$$

5.1.5 Example: Bernstein Polynomials for $f(x) = x^2$

For $f(x) = x^2$, the Bernstein polynomial of degree n is:

$$B_n(x^2, x) = \sum_{k=0}^{n} \left(\frac{k}{n}\right)^2 \binom{n}{k} x^k (1-x)^{n-k}.$$

5.2 Applications of Bernstein Polynomials

- **Approximation Theory**: Bernstein polynomials are used to approximate continuous functions due to their convergence properties.
- **Computer Graphics**: They are employed in Bézier curves, which are essential in vector graphics and computer-aided geometric design.
- **Numerical Analysis**: Bernstein polynomials help in numerical integration and solving differential equations.

Chapter 6

Gauss Jacobi Integration

6.1 Formulation

We have an m^{th} order function $P_m(x)$ with an n^{th} order approximation using collocation, $q_n(x)$, where $m > n$.

$$P_m(x) \approx q_n(x) = \Sigma_{k=0}^{n} L_k(x) P_m(x_k)$$

where x_k are the node/collocation points, $k \in I_n = \{0, 1, ..., n\}$

Given this information, we can write the error between the approximation and the actual function as the following

$$\underbrace{P_m(x) - q_n(x)}_{(m^{th} order)} = \underbrace{[\Pi_{k \in I_n}(x - x_k)]}_{(n+1)^{th} order} \underbrace{\{R_{m-n-1}(x)\}}_{(m-n-1)^{th} order}$$

$$P_m(x) = q_n(x) + \Pi_{k \in I_n}(x - x_k) R_{m-n-1}(x)$$

Now, we integrate both sides

$$\int_{-1}^{1} P_m(x) w(x) dx = \int_{-1}^{1} \Sigma_{k=0}^{n} L_k(x) P_m(x_k) dx + \int_{-1}^{1} \Pi_{k \in I_n}(x - x_k) R_{m-n-1}(x) dx$$

For convenience, we take the weight function to be unity (w(x) = 1, i.e Legendre polynomials as base?) and the limits of integral from -1 to 1.
If we choose x_k as the roots of some basis polynomial, here of Legendre polynomial $\phi_{n+1}(x)$, then the second term goes to zero by using the orthogonality principle.

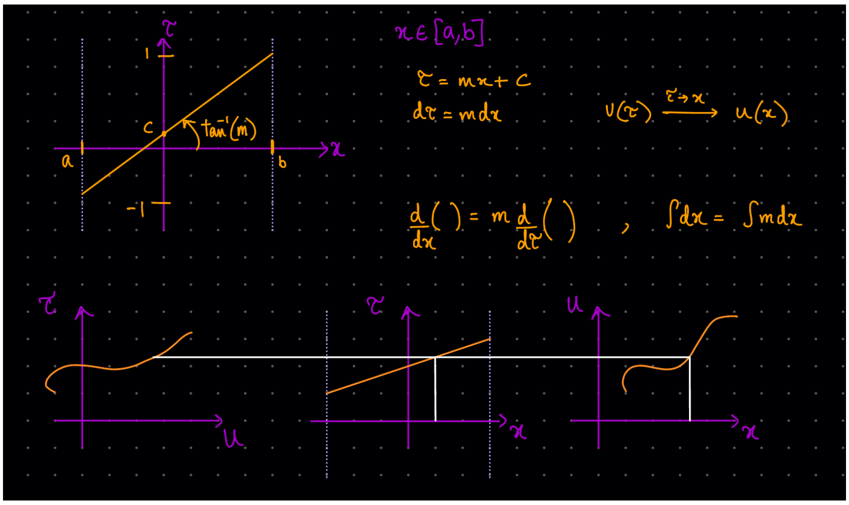
$$\int_{-1}^{1} P_m(x) w(x) dx = \int_{-1}^{1} L_k(x) P_m(x_k) dx + \int_{-1}^{1} \Pi_{k \in I_n}(x - x_k) R_{m-n-1}(x) dx$$

$$\int_{-1}^{1} P_m(x) w(x) dx = \Sigma_{k=0}^{n} P_m(x_k) \underbrace{\int_{-1}^{1} L_k(x) w(x) dx}_{w_k} \tag{6.1}$$

Thus, this is the final relation for Gauss Jacobi integration with w_k as the weight for summation for each node point.

Conditions/Prerequisites

- The function P(x) $\in P_n(x)$, where $P_n(x)$ are the set of polynomials that can be represented as as a combination of $\phi_k(x)$, $k \in I_n$
- m - n - 1 $\leq$ n $\implies$ m $\leq$ 2n + 1
- Transformation due to limits of integral being from -1 to 1 to domain limits



f(x) = P(x) $\in P_n(x) \rightarrow$ Lagrange polynomial representation is exact for 'n+1' node points for n^{th} order

Chapter 7

Chebyshev Polynomials

7.1 Definition of Chebyshev Polynomials

The Chebyshev polynomials $T_n(x)$ are defined recursively and can be expressed explicitly as:

$$T_n(x) = \cos(n \cos^{-}(x)), \quad \text{for } x \in [-1, 1].$$

7.2 Examples of the First Few Chebyshev Polynomials

$$\begin{aligned} T_0(x) &= 1, \\ T_1(x) &= x, \\ T_2(x) &= 2x^2 - 1, \\ T_3(x) &= 4x^3 - 3x, \\ T_4(x) &= 8x^4 - 8x^2 + 1. \end{aligned}$$

7.3 Orthogonality Property

The Chebyshev polynomials $T_n(x)$ satisfy the orthogonality condition:

$$\int_{-1}^{1} \frac{T_m(x)T_n(x)}{\sqrt{1-x^2}}\, dx = \begin{cases} 0, & \text{if } m \neq n, \\ \pi, & \text{if } m = n = 0, \\ \frac{\pi}{2}, & \text{if } m = n \neq 0. \end{cases}$$

7.4 Nodes of Chebyshev Polynomials

The nodes x_k (roots) of the Chebyshev polynomial $T_n(x)$ are given by:

$$x_k = \cos\left(\frac{2k+1}{2n}\pi\right), \quad k = 0, 1, 2, \ldots, n.$$

7.5 Derivative of Chebyshev Polynomials

The derivative $T'_{n+1}(x)$ can be computed as:

$$T'_{n+1}(x) = nU_n(x),$$

where $U_n(x)$ is the Chebyshev polynomial of the second kind.

7.6 Weights for Numerical Integration

The weights for numerical integration using Chebyshev polynomials are:

$$w_k = \frac{\pi}{n+1}, \quad \text{for } k = 0, 1, 2, \ldots, n.$$

where k are the node points

7.7 Recurrence Relation

The Chebyshev polynomials $T_n(x)$ can be generated using the recurrence relation:

$$T_{n+1}(x) + T_{n-1}(x) = 2xT_n(x) \quad \text{for } n \geq 2$$

with initial conditions:

$$T_0(x) = 1, \quad T_1(x) = x.$$

7.8 Chebyshev-Gauss-Lobatto Nodes

Definition: Chebyshev-Gauss-Lobatto nodes are specific collocation points used for interpolation and numerical methods involving Chebyshev polynomials. These nodes are particularly effective in minimizing the Runge phenomenon and ensuring numerical stability in polynomial interpolations.

7.8.1 Mathematical Formulation

The Chebyshev-Gauss-Lobatto nodes are defined in the interval $[-1, 1]$ as:

$$x_k = \cos\left(\frac{k\pi}{N}\right), \quad k = 0, 1, 2, \ldots, N,$$

where N is the total number of nodes.

7.8.2 Properties

- **Boundary Inclusion**: The nodes include the endpoints $x_0 = -1$ and $x_N = 1$, which is particularly useful for enforcing boundary conditions in numerical methods.
- **Clustering at Endpoints**: The distribution of nodes clusters more densely near the endpoints of the interval, which helps in reducing errors in polynomial approximations.

7.8.3 Applications

- **Spectral and Pseudospectral Methods**: Chebyshev-Gauss-Lobatto nodes are commonly used for constructing differentiation matrices in pseudospectral methods.

- **Interpolation and Approximation**: These nodes are employed in polynomial interpolation to achieve better accuracy and stability.

7.8.4 Example

For $N = 4$, the Chebyshev-Gauss-Lobatto nodes are:

$$x_0 = \cos\left(\frac{0\pi}{4}\right) = 1, \quad x_1 = \cos\left(\frac{1\pi}{4}\right) = \frac{\sqrt{2}}{2}, \quad x_2 = \cos\left(\frac{2\pi}{4}\right) = 0,$$

$$x_3 = \cos\left(\frac{3\pi}{4}\right) = -\frac{\sqrt{2}}{2}, \quad x_4 = \cos\left(\frac{4\pi}{4}\right) = -1.$$

7.8.5 Differentiation Matrix

In pseudospectral methods, the differentiation matrix D is constructed using Chebyshev-Gauss-Lobatto nodes. The matrix D approximates the derivative of a function sampled at these nodes and is defined by:

$$D_{kj} = \begin{cases} \frac{2N^2+1}{6}, & \text{for } k = j = 0, \\ \frac{-c_k}{c_j}\frac{(-1)^{k+j}}{x_k - x_j}, & \text{for } k \neq j, \\ -\frac{x_k}{2(1-x_k^2)}, & \text{for } 1 \leq k = j \leq N-1, \\ -\frac{2N^2+1}{6}, & \text{for } k = j = N, \end{cases}$$

where:

$$c_k = \begin{cases} 2, & \text{for } k = 0 \text{ or } k = N, \\ 1, & \text{otherwise.} \end{cases}$$

7.8.6 Orthogonality and Weights

Chebyshev polynomials of the first kind $T_n(x)$ are orthogonal with respect to the weight function $w(x) = \frac{1}{\sqrt{1-x^2}}$ over $[-1, 1]$:

$$\int_{-1}^{1} T_m(x)T_n(x)w(x)\,dx = \begin{cases} 0, & \text{for } m \neq n, \\ \pi, & \text{for } m = n = 0, \\ \frac{\pi}{2}, & \text{for } m = n \neq 0. \end{cases}$$

The quadrature weights w_k associated with Chebyshev-Gauss-Lobatto nodes are given by:

$$w_k = \frac{\pi}{N}\left(1 - x_k^2\right).$$

7.9 Conclusion

Chebyshev-Gauss-Lobatto nodes play a vital role in numerical approximation methods due to their favorable distribution and properties that facilitate accurate polynomial interpolation and stable numerical differentiation.

Chapter 8

Boundary Value Problem

Boundary Value Problem (BVP) Example from Boyd - Collocation Method

$$P(x) = \sum a_k x^k = \sum L_k(\hat{x}) f(x_k)$$

Where:

- x_k: Find a_k
- L_k: Find L_k

Boundary conditions:

$$u(0) = 0, \quad u(1) = 1$$

Thus:

$$u = a_0 + a_1 x + a_2 x^2$$

We approximate the solution to a polynomial in the interval $[0, 1] \to [-1, 1]$.
Given equation:

$$(1 + u)\frac{d^2u}{dx^2} - \frac{du}{dx} = 0$$

At midpoint $t = \frac{1}{2}$:

$$a_2 = - \quad \text{(solve for } a_2\text{)}$$

Galerkin Method

$$\frac{\partial}{\partial a_k} \int R^2 \, dx$$

$$\Rightarrow \int R \frac{\partial R}{\partial a_k} = 0$$

Where:

$$J = \int R^2 \, w \, dx$$

J: Residual norm (Least Squares) with different weights w.

Note: Collocation method is easier for nonlinear problems.

BVP Transformation and Numerical Sine in Class

The BVP transforms the problem domain to $[0, 1]$ and then uses appropriate weights to do the Least Squares (LSQ) approximation.

$$\begin{bmatrix} u(x_0) \\ u(x_1) \\ \vdots \\ u(x_N) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$[D][a] = \sum D_{ij}u_j - \text{residual}$$

Numerical Sine in Class

Differential equation:

$$(1+u)\frac{d^2u}{dx^2} + \left(\frac{du}{dx}\right)^2 = 0$$

Boundary conditions:

$$u(0) = 0, \quad u(1) = 1$$

Time t relation:

$$t = c \quad \Rightarrow \quad \text{explicit approximation to solution}$$

$$P(x) = \sum L_k(x)f(x_k)$$

Transformation:

$$t = \frac{t-1}{c-1} = -1, \quad t = 1 \to \frac{2(c-t)}{t}$$

Jacobian determinant:

$$\det J = 2dx$$

Mapping for $u(t = 0) = 0$ and $u(t = 1) = 1$:

$$L_k(x_j) \Rightarrow [B] \Rightarrow L(x)$$

Matrix representation:

$$\begin{pmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{pmatrix} = \begin{pmatrix} u(t) & \text{differential operator} \end{pmatrix}$$

Question:

Which one is it?

Explanation of the Numerical Approach Using Chebyshev Collocation

To solve the nonlinear Boundary Value Problem (BVP) numerically, we use the Chebyshev collocation method. This approach leverages the properties of Chebyshev nodes for better numerical stability and accuracy. The process involves:

1. Chebyshev Nodes Calculation

We compute the Chebyshev nodes x_{cheby} in the domain $[-1, 1]$. These nodes are derived using:

$$x_{\text{cheby}} = \cos\left(\frac{\pi(2k-1)}{2n}\right), \quad k = 1, \ldots, n$$

These nodes are then transformed to the desired interval $[0, 1]$ using:

$$x_{\text{transformed}} = \frac{a+b}{2} + \frac{b-a}{2} \cdot x_{\text{cheby}}$$

2. Differentiation Matrix

The Chebyshev differentiation matrix D is obtained for these nodes. This matrix allows for the computation of derivatives at Chebyshev nodes.

3. Analytical Solution for Comparison

An analytical solution $u(x) = \sqrt{3x+1} - 1$ is defined and evaluated at the transformed Chebyshev nodes to compare against the numerical solution.

4. Initial Guess

We use the analytical solution to create an initial guess u_{initial} for the solver. Boundary conditions are applied to the initial guess:

$$u(0) = 1, \quad u(1) = 0$$

5. Solving the Nonlinear System

We solve the BVP using MATLAB's `fsolve` function, which minimizes the residuals defined by:

$$R = (1+u)\frac{d^2u}{dx^2} + \left(\frac{du}{dx}\right)^2$$

The function `residual_with_boundary` enforces the boundary conditions by fixing $u(0)$ and $u(1)$.

6. Plotting the Results

The numerical solution $u(x)$ is plotted alongside the analytical solution for comparison, illustrating the accuracy of the numerical method.

Detailed Code Walkthrough

Chebyshev Node Transformation

The Chebyshev nodes x_{cheby} are transformed to the interval $[0, 1]$ for this problem domain.

Initial Guess Setup

The initial guess u_{initial} is adjusted to match the boundary conditions $u(0) = 1$ and $u(1) = 0$.

Residual Function

The function `residual_with_boundary` defines the nonlinear residual R while ensuring the boundary conditions are met by setting:

$$\text{res}(1) = u(0) - 1, \quad \text{res}(n) = u(1) - 0$$

Solver Configuration

The nonlinear system is solved using `fsolve` with options set to display iteration steps for convergence monitoring.

Explanation of Boundary Conditions Enforcement

The boundary conditions $u(0) = 1$ and $u(1) = 0$ are crucial for ensuring that the numerical solution adheres to the physical requirements of the problem. These conditions are imposed directly within the residual function, keeping the first and last elements of u fixed to enforce $u(0)$ and $u(1)$.

Chapter 9

Fourier Series

The Fourier series of a general function, f(x) can be written as

$$S_n(x) = a_0 + \sum_{k=1}^{\infty} \left(a_k \cos\left(\frac{2\pi kx}{T}\right) + b_k \sin\left(\frac{2\pi kx}{T}\right)\right) \tag{9.1}$$

where, the coefficients are

$$a_0 = \frac{1}{T}\int_0^T f(x)\,dx \tag{9.2}$$

$$a_k = \frac{2}{T}\int_0^T f(x)\cos\left(\frac{2\pi kx}{T}\right)\,dx \quad \text{for } k \geq 1 \tag{9.3}$$

$$b_k = \frac{2}{T}\int_0^T f(x)\sin\left(\frac{2\pi kx}{T}\right)\,dx \quad \text{for } k \geq 1 \tag{9.4}$$

It can also be written in complex form as

$$f(x) \sum_{n=-\infty}^{\infty} c_n e^{ikx} \tag{9.5}$$

where

$$c_n = \frac{2}{T}\int_0^T f(x)e^{-ikx}\,dx \tag{9.6}$$

but we are considering a periodic function f(x) with x ∈ [0, T]. So we can essentially take a part of a general function and consider that to be the period, hence we can do this analysis on any general function.

. . . .

9.1 Fourier Series Convergence

- What happens to $\lim_{k\to\infty} a_k$ and $\lim_{k\to\infty} b_k$? Do they both tend to zero?
- Does $\lim_{n\to\infty} S_n(x)$ converge to $f(x)$?

9.1.1 Numerical Proof

We can first try to answer these questions numerically using two methods for comparison, Fast Fourier Transform (FFT) and Numerical integration with the trapezoidal rule (trapz).

1. **Initialization:**
   - Define the number of sample points $N = 20$ and the time vector x spanning $[0, 2\pi]$.
2. **Performing FFT:**
   - Compute $F = \text{fft}(\text{func})$, where $\text{func} = \sin(x)$.
   - Extract magnitudes $|F|$ and phases $\angle F$ to find the Fourier coefficients:

$$\begin{aligned} a_k &= \frac{2 \cdot \text{Re}(F)}{N}, \quad k = 1, 2, \ldots, \frac{N}{2} \\ b_k &= -\frac{2 \cdot \text{Im}(F)}{N} \end{aligned}$$

3. **Numerical Integration (Trapz):**
   - Calculate a_k and b_k using:

$$\begin{aligned} a_k &= \frac{2}{T} \int_0^T f(x) \cos(2kx)\, dx \\ b_k &= \frac{2}{T} \int_0^T f(x) \sin(kx)\, dx \end{aligned}$$

   - The integrals are approximated with MATLAB's `trapz` function.
4. **Reconstruction:**
   - Reconstruct $f(x)$ from the Fourier series coefficients using:9

$$f(x) \approx \frac{a_0}{2} + \sum_{k=1}^{n} [a_k \cos(kx) + b_k \sin(kx)]$$

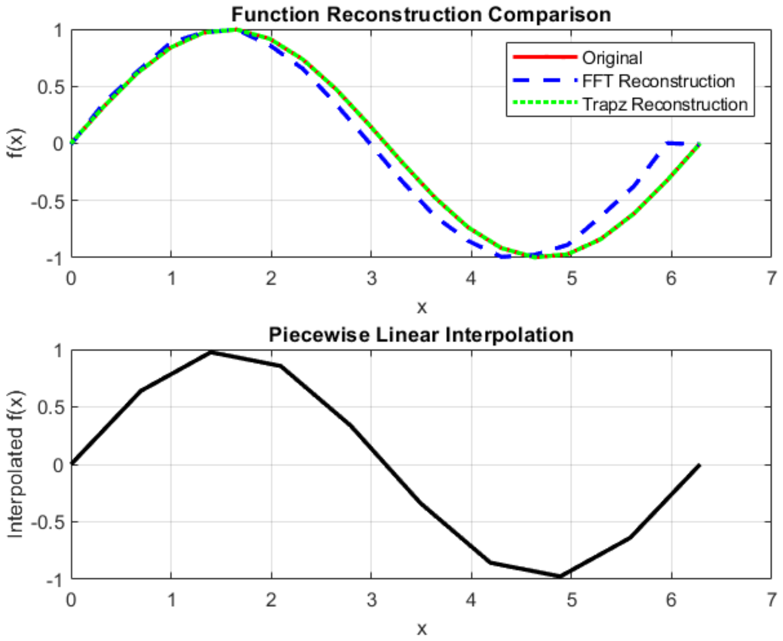
5. **Interpolation:**
   - Use `interp1` for piecewise linear interpolation to create a denser set of points for visualization.

Visualization

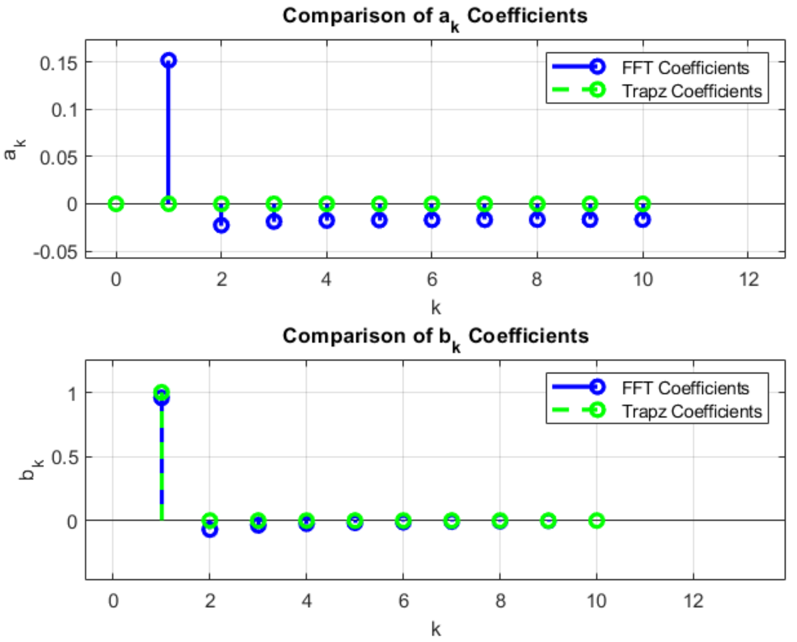
The code generates plots to:

- Compare the original function with its FFT-based and numerical integration-based reconstructions.

- Show a piecewise linear interpolation of $f(x)$.



- Display the Fourier coefficients a_k and b_k obtained from both methods.



. . . .

A similar piecewise linear interpolation can be done manually but it is a very tedious process. For example, take the function sin^2x with 3 points including ends (done in my notes). We have the coefficients $b_k = 0$ and

$$a_k = \begin{cases} \frac{-4}{\pi}k^2 & \text{k is odd} \\ 0 & \text{k is even} \end{cases}$$

So, here $S_n(x) = \Sigma_{k=0}^{n} \, a_k \; cos(kx)$ with calculated a_k as shown.

$|f(x) - sin^2(x)| = \frac{1}{(n+1)^2} + \frac{1}{(n+1)^2} + \dots$

(there's some proof about how the summation is is bounded or something, idk; then its extended to piecewise linear interpolation for general function, just writing out in therms of variabes basically)

9.1.2 Analytical Proof

The sine and cosine functions are orthogonal over the interval $[0, 2\pi]$. This property is fundamental in Fourier series analysis. The orthogonality conditions are defined as follows:

- **Orthogonality of Sine Functions:**

$$\int_0^{2\pi} \sin(mx)\sin(nx)\,dx = \begin{cases} 0, & \text{for } m \neq n \\ \pi, & \text{for } m = n \neq 0 \end{cases}$$

- **Orthogonality of Cosine Functions:**

$$\int_0^{2\pi} \cos(mx)\cos(nx)\,dx = \begin{cases} 0, & \text{for } m \neq n \\ \pi, & \text{for } m = n \neq 0 \\ 2\pi, & \text{for } m = n = 0 \end{cases}$$

- **Orthogonality Between Sine and Cosine:**

$$\int_0^{2\pi} \sin(mx)\cos(nx)\,dx = 0, \quad \text{for any } m, n.$$

These properties mean that when integrating the product of two different sine or cosine functions over a full period $[0, 2\pi]$, the result is zero. This orthogonality is crucial for expressing periodic functions as a sum of sine and cosine terms in a Fourier series.

proof that Sn(x) goes to zero if n tends to infinity, Sn(x) tends to f(x),

9.2 Collocation using Fourier Series as Basis

$S_n(x)$ is the approximation for the function f(x) using its Fourier series representation. The collocation points are x_k. The Fourier series decomposes a periodic function into a sum of sine and cosine terms as

$$S_n(x) = \alpha_0 + \sum_{m=1}^{\infty} \left(\alpha_m \cos\left(\frac{2\pi m x}{T}\right) + \beta_m \sin\left(\frac{2\pi m x}{T}\right) \right)$$

and

$$S_n(x_k) = f(x_k)$$

with $x_k = \frac{Tk}{2n}$ for k = 0, 1, 2, ... 2n+1

Using relations 9 and 9, where dx = $\frac{T}{2n+1}$

$$\alpha_m = \frac{2}{2n+1} \sum_{k=1}^{2n+1} f(x_k) \cos\left(\frac{2\pi m x}{T}\right)$$

$$\beta_m = \frac{2}{2n+1} \sum_{k=1}^{2n+1} f(x_k) \sin\left(\frac{2\pi m x}{T}\right)$$

Example: Fourier Series Approximation of (Numerical)

We approximate a given piecewise function $f(x)$ using its Fourier series representation. A periodic function $f(x)$ defined over the interval $[0, 2\pi]$ can be approximated by a Fourier series.

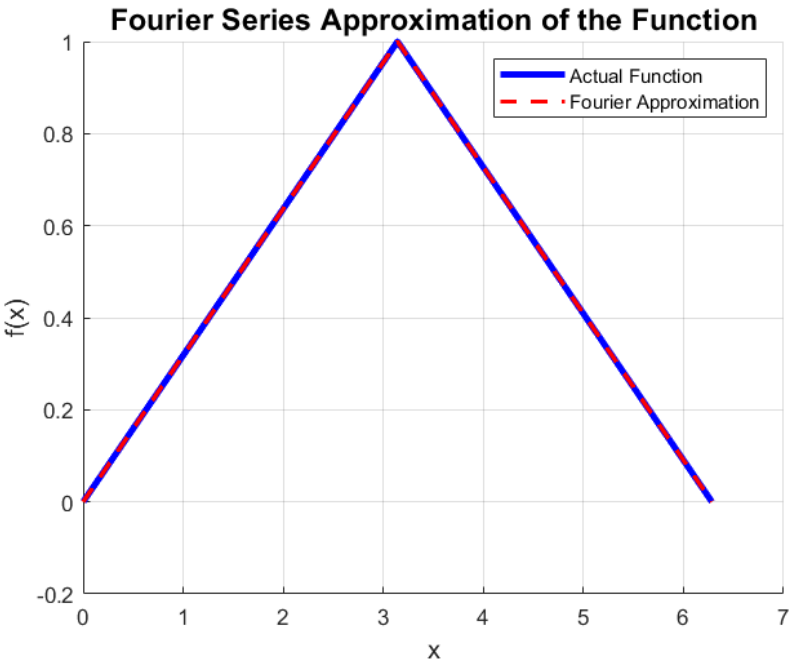
- **Initialization and Sample Points:**
  - The number of terms in the series n is set.
  - The vector x represents the points sampled from the interval $[0, 2\pi]$, specifically $2n + 1$ points for better accuracy.
- **Definition of the Function:**
  - The piecewise function is defined as:

$$f(x) = \begin{cases} \frac{x}{\pi}, & \text{for } 0 \le x < \pi, \\ 2 - \frac{x}{\pi}, & \text{for } \pi \le x \le 2\pi. \end{cases}$$

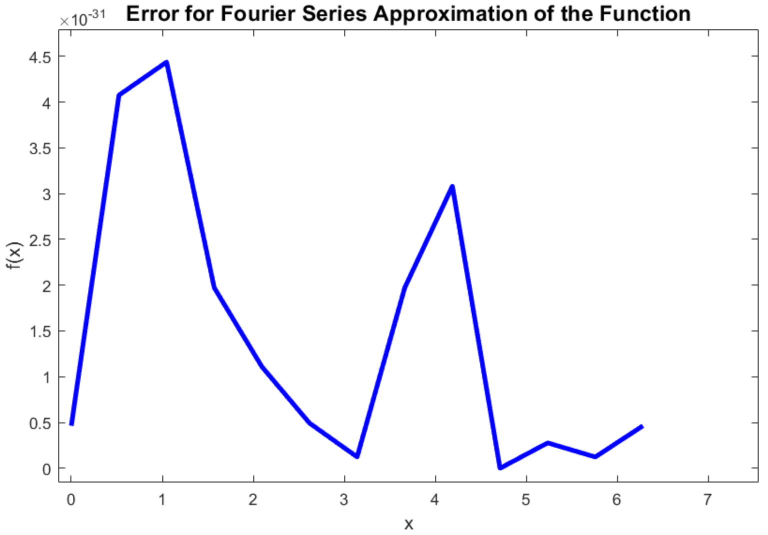
  - This is represented as two segments, `fx1` for $[0, \pi]$ and `fx2` for $[\pi, 2\pi]$.
- **Calculation of Coefficients:** The Fourier coefficients α_m and β_m using numerical integration (via the `trapz` function) as shown above for a specific piecewise function defined over the interval $[0, 2\pi]$.
- **Function Reconstruction:**
  - The function $S_n(x)$ is reconstructed using the calculated α_m and β_m coefficients.
  - The approximation $S_n(x)$ is then compared to the original piecewise function $f(x)$ by calculating the squared error $e(x)$.

Plotting and Analysis

- The first plot displays the original function $f(x)$ and its Fourier series approximation $S_n(x)$.



- The second plot visualizes the squared error $e(x)$ across the interval, which helps to assess the quality of the approximation.



This numerical approach highlights the convergence behavior of the Fourier series in approximating piecewise continuous functions.