Computational Project

Tianzhe HU(th296)
id: 229061856

November 18, 2024

Contents

0 Introduction

In the study of ordinary differential equations (ODEs), solving them numerically is an essential task, especially when the equations cannot be solved analytically. The primary objective of this computational project is to numerically solve a specific class of ODEs using spectral methods, with a focus on Chebyshev polynomials. Spectral methods are particularly effective for solving boundary value problems due to their rapid convergence and high accuracy, especially when using orthogonal polynomials like Chebyshev polynomials.

The differential equation under consideration is defined over the interval [-1, 1] and is complemented with Dirichlet boundary conditions. In this project, we aim to approximate the solution of this ODE by expanding the unknown function as a truncated series of Chebyshev polynomials. This approach relies on the fact that any continuous function on the interval [-1, 1] can be represented as an infinite series of Chebyshev polynomials, making Chebyshev expansion a powerful tool for numerical approximations.

The goal of the project is to develop and implement a series of computational methods that will allow us to compute the Chebyshev coefficients of a given function, solve the ODE numerically by enforcing the boundary conditions, and then compare the numerical solution with the exact solution of the ODE for various test cases. This project is divided into two main parts: the first part focuses on using Chebyshev polynomials as a basis for function representation, and the second applies these techniques to solve the given ODE with boundary conditions.

Through the implementation of Chebyshev-based methods, we will compute and visualize the solutions for different functions and boundary conditions, ultimately demonstrating the effectiveness of spectral methods for solving ODEs numerically.

Part 1: Chebyshev polynomials as a basis

It holds that any continuous function over the interval [-1,1] can be represented as an infinite series of Chebyshev polynomials. This is, for any function $f(x)$, there exist coefficients $\alpha_0, \alpha_1, \cdots$, such that

$$f(x) = \sum_{n=0}^{\infty} \alpha_n T_n(x)$$

The coefficients α_k can be found using the following orthogonality property of the Chebyshev polynomials:

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

1 Q1: Mathematical Proof of α_k

1.1 Question Restatement

Show that

$$\alpha_k = \begin{cases} \frac{2}{\pi}\int_{-1}^{1} \frac{f(x)T_k(x)}{\sqrt{1-x^2}}dx & \text{if } \ k \neq 0 \\ \frac{1}{\pi}\int_{-1}^{1} \frac{f(x)T_k(x)}{\sqrt{1-x^2}}dx & \text{if } \ k = 0 \end{cases}$$

1.2 Proof of the Formula for α_k

Let $T_k(x)$ be the k-th Chebyshev polynomial of the first kind, defined on the interval $[-1,1]$. The function $f(x)$ can be expanded in terms of Chebyshev polynomials as:

$$f(x) = \sum_{n=0}^{\infty} \alpha_n T_n(x)$$

where α_n are the Chebyshev coefficients we seek to find.

As it is known, a key property of Chebyshev polynomials is their orthogonality with respect to the weight function $\frac{1}{\sqrt{1-x^2}}$ over the interval [-1, 1]:

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

To determine α_k, we multiply both sides of $f(x) = \sum_{n=0}^{\infty} \alpha_n T_n(x)$ by $T_k(x)/\sqrt{1-x^2}$ and integrate over [-1,1]:

$$\int_{-1}^{1} \frac{f(x)T_k(x)}{\sqrt{1-x^2}}\,dx = \int_{-1}^{1} \sum_{n=0}^{\infty} \alpha_n \frac{T_n(x)T_k(x)}{\sqrt{1-x^2}}\,dx.$$

Since integration is a linear operator, we can interchange the summation and integration:

$$\int_{-1}^{1} \frac{f(x)T_k(x)}{\sqrt{1-x^2}}\,dx = \sum_{n=0}^{\infty} \alpha_n \int_{-1}^{1} \frac{T_n(x)T_k(x)}{\sqrt{1-x^2}}\,dx.$$

By the orthogonality of Chebyshev polynomials, only the term with n = k survives in this summation:

$$\int_{-1}^{1} \frac{f(x)T_k(x)}{\sqrt{1-x^2}}\,dx = \begin{cases} \alpha_k \cdot \frac{\pi}{2} & \text{if } k \neq 0 \\ \alpha_k \cdot \pi & \text{if } k = 0 \end{cases}.$$

Solving for α_k, we obtain:

$$\alpha_k = \begin{cases} \frac{2}{\pi} \int_{-1}^{1} \frac{f(x)T_k(x)}{\sqrt{1-x^2}}\,dx & \text{if } k \neq 0 \\ \frac{1}{\pi} \int_{-1}^{1} \frac{f(x)T_k(x)}{\sqrt{1-x^2}}\,dx & \text{if } k = 0 \end{cases}$$

2 Q2: Function of Integral

2.1 Question Restatement

Implement a function that takes as an input two function handles $f(x)$ and $g(x)$, and an integer N, and outputs the integral:

$$\frac{2}{\pi} \int_{-1}^{1} \frac{f(x)g(x)}{\sqrt{1-x^2}}dx$$

using trapezoidal integration with N partition points.
Hint: it may be useful to use the change of variable $x = \cos(\theta)$.

2.2 Analysis

To approximate this integral numerically, we use the trapezoidal rule, which estimates the integral as:

$$\int_{a}^{b} f(x)dx \approx \sum_{k=1}^{N} \frac{f(x_{k-1}) + f(x_k)}{2} \Delta x_k.$$

where $\Delta x_k = x_k - x_{k-1}$ and x_k are the partition points of the interval $[a, b]$. Figure 1 illustrates this approach using $f(x) = \cos^2(x)$ on the interval [0, 4]. The red-shaded areas represent the trapezoids approximating the area under the curve.

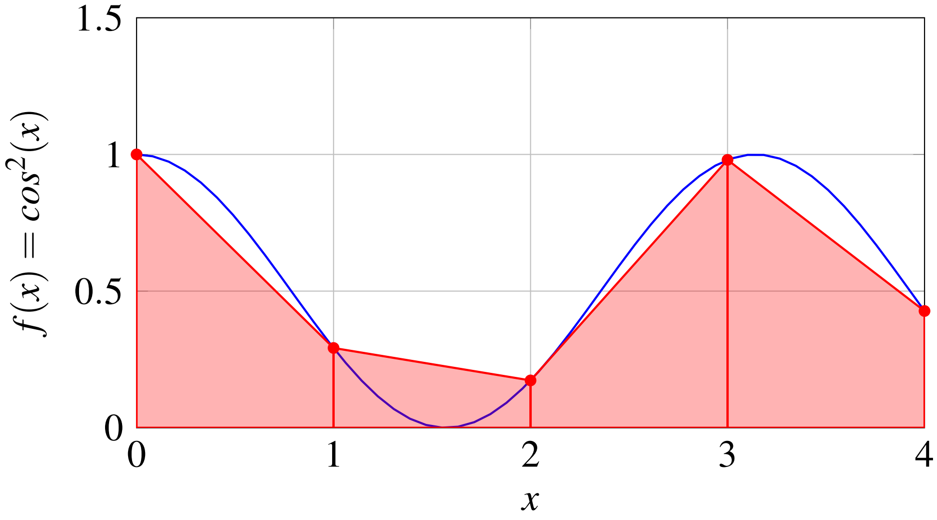


Figure 1: Illustration of the Trapezoidal Rule

Now we want to evaluate the integral:

$$\frac{2}{\pi}\int_{-1}^{1} \frac{f(x)g(x)}{\sqrt{1-x^2}}\,dx$$

We use the change of variable $x = \cos(\theta)$. This transforms the integral from $[-1, 1]$ to $[0, \pi]$:

$$\int_{-1}^{1} \frac{f(x)g(x)}{\sqrt{1-x^2}}\,dx = \int_{0}^{\pi} f(\cos(\theta))g(\cos(\theta))\,d\theta.$$

Then we use the trapezoidal rule to estimate this integral. The following is the MATLAB implementation of the function.

2.3 Solution

```
function integral_value = integral(f, g, N)
    % Input:
    %   f - function handle f(x)
    %   g - function handle g(x)
    %   N - number of partitions for trapezoidal integration
    % Output:
```

```
    %   integral_value
    a = 0;
    b = pi;
    h = (b - a) / N;

    theta = linspace(a, b, N+1);

    % Change of variable: x = cos(theta)
    x = cos(theta);

    % Compute f(x) * g(x)
    fg_values = f(x) .* g(x);

    % Trapezoidal integration
    integral_value = h * (sum(fg_values) - 0.5 *(fg_values(1) + fg_values(end)));

    integral_value = (2 / pi) * integral_value;

end
```

An example usage of this function will be demonstrated in the next section.

3 Q3: Function to Determine Coefficients

3.1 Question Restatement

Using the function of question 2, write a function that for a given function handle f and an integer m, outputs the array $[\alpha_0 \ \alpha_1 \ \cdots \ \alpha_{m-1}]$.

3.2 Analysis

To extend the function from Question 2 and compute the array $[\alpha_0, \alpha_1, \cdots, \alpha_{m-1}]$ for a given function handle f and integer m, we utilize the formula for α_k:

$$\alpha_k = \begin{cases} \frac{2}{\pi}\int_{-1}^{1} \frac{f(x)T_k(x)}{\sqrt{1-x^2}}\,dx & \text{if } k \neq 0 \\ \frac{1}{\pi}\int_{-1}^{1} \frac{f(x)T_k(x)}{\sqrt{1-x^2}}\,dx & \text{if } k = 0 \end{cases}$$

We can implement this by iterating through $k = 0, 1, \cdots, m-1$ and using the numerical integration function from Question 2 to compute each α_k.

3.3 Solution

```
function alpha_array = compute_chebyshev_coefficients(f, m, N)
    % Input:
    %   f - function handle for f(x)
    %   m - the number of Chebyshev coefficients to compute (m terms)
    %   N - number of partitions for trapezoidal integration
    % Output:
    %   alpha_array - [alpha_0, alpha_1, ..., alpha_{m-1}]

    alpha_array = zeros(1, m);

    % Define Chebyshev polynomial function T_k(x)
    chebyshev_poly = @(k, x) cos(k * acos(x));

    for k = 0:(m-1)
        % Define g(x) = T_k(x)
        T_k = @(x) chebyshev_poly(k, x);

        % Compute the integral using Question 2's function
        if k == 0
            % If k = 0, use the coefficient 1/pi
            alpha_array(k + 1) = integral(f, T_k, N) / 2;
        else
            % If k > 0, use 2/pi
            alpha_array(k + 1) = integral(f, T_k, N);
        end
    end
end
```

3.4 Example Usage

```
f = @(x) x.^2;
m = 5;            % the first 5 Chebyshev coefficients
N = 1000;         % Number of partitions

alpha_values = compute_chebyshev_coefficients(f, m, N);
disp(alpha_values);
```

Output

This will display an array with the first m Chebyshev coefficients:

$$[0.5000 \quad 0.0000 \quad 0.5000 \quad 0.0000 \quad -0.0000]$$

This is correct, we will give the math proof:

The Chebyshev coefficients α_k are computed using the following integral formula:

$$\alpha_k = \begin{cases} \frac{2}{\pi}\int_{-1}^{1}\frac{f(x)T_k(x)}{\sqrt{1-x^2}}dx & \text{if } k \neq 0 \\ \frac{1}{\pi}\int_{-1}^{1}\frac{f(x)T_0(x)}{\sqrt{1-x^2}}dx & \text{if } k = 0 \end{cases}$$

For $f(x) = x^2$, we express x^2 in terms of Chebyshev polynomials:

$$x^2 = \frac{T_0(x) + T_2(x)}{2} = \frac{1}{2}T_0(x) + \frac{1}{2}T_2(x)$$

For all other $k \neq 0, 2$, the coefficients are zero due to the orthogonality property of Chebyshev polynomials. Therefore, the non-zero coefficients are:

$$\alpha_0 = \frac{1}{2}, \quad \alpha_2 = \frac{1}{2}$$

and all other $\alpha_k = 0$ for $k \neq 0, 2$.

4 Q4: Calculating Truncated Series

4.1 Question Restatement

Write a function that for a given array $[\alpha_0, \alpha_1, \cdots, \alpha_{m-1}]$, outputs the truncated series

$$\sum_{k=0}^{m-1} \alpha_k T_k(x)$$

where x is a `linspace` array on the interval [-1,1].

4.2 Analysis

The Chebyshev series is an expansion of a function $f(x)$ in terms of Chebyshev polynomials $T_k(x)$, which are defined as:

$$T_k(x) = \cos(k \cdot \arccos(x))$$

The goal is to compute the truncated Chebyshev series:

$$\sum_{k=0}^{m-1} \alpha_k T_k(x)$$

where α_k are the Chebyshev coefficients, and x is a set of points chosen from the interval $[-1,1]$. In practical terms, this series provides an approximation to $f(x)$, and truncating the series means we only use the first m terms, which can provide a good approximation depending on the function and the value of m.

4.3 Solution

```
function approx_values = chebyshev_truncated_series(alpha, N)
% Input:
% alpha - array of coefficients [alpha_0, alpha_1, ..., alpha_{m-1}]
% N - number of evaluation points
% Output:
% approx_values - evaluated values of the truncated series at points x

    % Determine the number of coefficients
    m = length(alpha);

    x = linspace(-1, 1, N);

    % Initialize the array
    approx_values = zeros(size(x));

    % Loop through the coefficients, compute the truncated series
    for k = 0:(m-1)
        T_k = cos(k * acos(x));

        % Add the contribution of the current term to the series
        approx_values = approx_values + alpha(k + 1) * T_k;
    end
end
```

4.4 Example Usage

We define a set of Chebyshev polynomial coefficients

$$\alpha = [1, 0.5, -0.3, 0.1]$$

and calculate the truncated Chebyshev series approximation at 100 points on the interval $[-1,1]$. The results are then plotted to visualize the approximation, demonstrated in Figure 2.

```
alpha = [1, 0.5, -0.3, 0.1];

% Define number of points for evaluation
```

```
num_points = 100;

% Compute the truncated Chebyshev series
approx_values = chebyshev_truncated_series(alpha, num_points);

% Plot
x = linspace(-1, 1, num_points);
plot(x, approx_values, 'LineWidth', 1.5);
xlabel('x');
ylabel('Approximation');
title('Chebyshev Truncated Series Approximation');
grid on;
```

Output

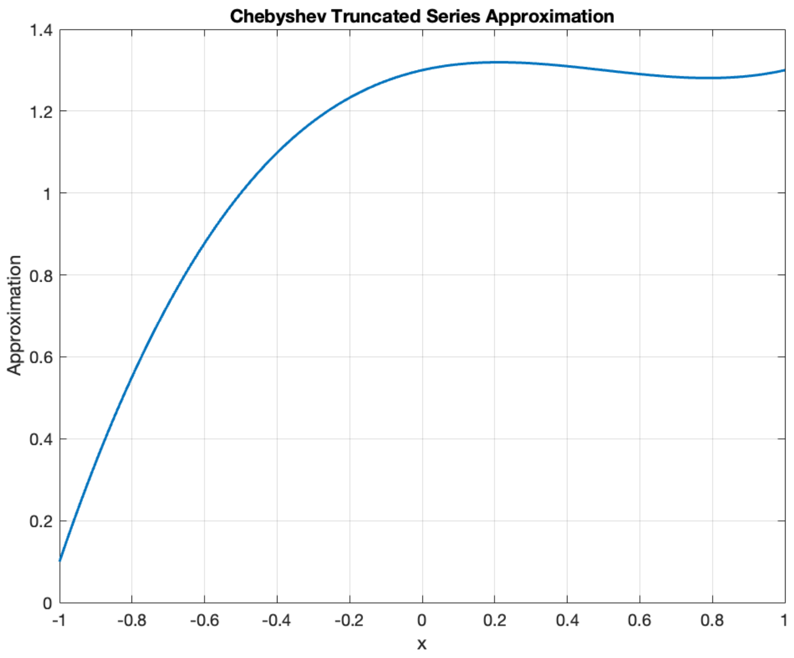


Figure 2: Chebyshev Truncated Series Approximation

5 Q5: Computing Coefficients and Plotting

5.1 Question Restatement

For m large enough, it holds that any given continuous function $f(x)$, defined on the interval [-1,1], can be approximated by its truncated Chebyshev series expansion

$$f(x) \approx \sum_{k=0}^{m-1} \alpha_k T_k(x).$$

Using the function from question 3, compute $[\alpha_0 \alpha_1 \cdots \alpha_{m-1}]$ for the following functions:

- $f(x) = x^2 \sin^3(\pi x)$
- $f(x) = \begin{cases} \frac{x}{2}, \text{ for } x \in [-1,0] \\ 2x, \text{ for } x \in [0,1] \end{cases}$
- $f(x) = \begin{cases} \frac{x}{2}, \text{ for } x \in [-1,0] \\ 2x+1, \text{ for } x \in [0,1] \end{cases}$

and using the function from question 4, plot the function $f(x)$ along with its truncated Chebyshev series for different values of m.

5.2 Analysis

Using the function `compute_chebyshev_coefficients` from Question 3 and `chebyshev_truncated_series` from Question 4, we can now compute the Chebyshev coefficients for the given functions and plot them alongside their approximations.

5.3 Solution

```
% Define three functions
f1 = @(x) x.^2 .* sin(pi * x).^3;
f2 = @(x) (x < 0) .* (x / 2) + (x >= 0) .* (2 * x);
f3 = @(x) (x < 0) .* (x / 2) + (x >= 0) .* (2 * x + 1);

% Set parameters
m_values = [5, 10, 20, 100];
N = 1000;% Number of partitions
x = linspace(-1, 1, 1000); % Points for evaluation

% Compute and plot the functions and their Chebyshev approximations
functions = {f1, f2, f3};

titles = {'f(x) = x^2 sin^3(\pi x)', 'f(x) = x/2 (x < 0), 2x (x >= 0)', ...
'f(x) = x/2 (x < 0), 2x + 1 (x >= 0)'};

for i = 1:3
    % Compute the Chebyshev coefficients for each function
    f = functions{i};

    figure;

    % Plot the original function
    plot(x, f(x), 'LineWidth', 1.5, 'DisplayName', 'Original Function');
    hold on;

    % Plot truncated series approximations for different m
    for m = m_values

        alpha = compute_chebyshev_coefficients(f, m, N);

        approx_values = chebyshev_truncated_series(alpha, N);

        plot(x, approx_values, 'DisplayName', ...
        sprintf('Truncated Series (m = %d)', m));

    end

    % plot property
    title(titles{i});
    xlabel('x');
    ylabel('f(x)');
    legend;
    grid on;
    hold off;

end
```

For each of the three given functions, it computes the first m coefficients for $m = 5, 10, 20, 100$.

Output

From the following graphics we can know that for larger m, the approximation gets closer to the actual function. However, it is important to note that when m becomes too large, the approximation might begin to overfit the function, leading to oscillations or an increase in error due to the phenomenon known as ”Runge’s phenomenon.” This occurs because higher-order Chebyshev polynomials can exhibit large variations, especially for functions with sharp changes or discontinuities.

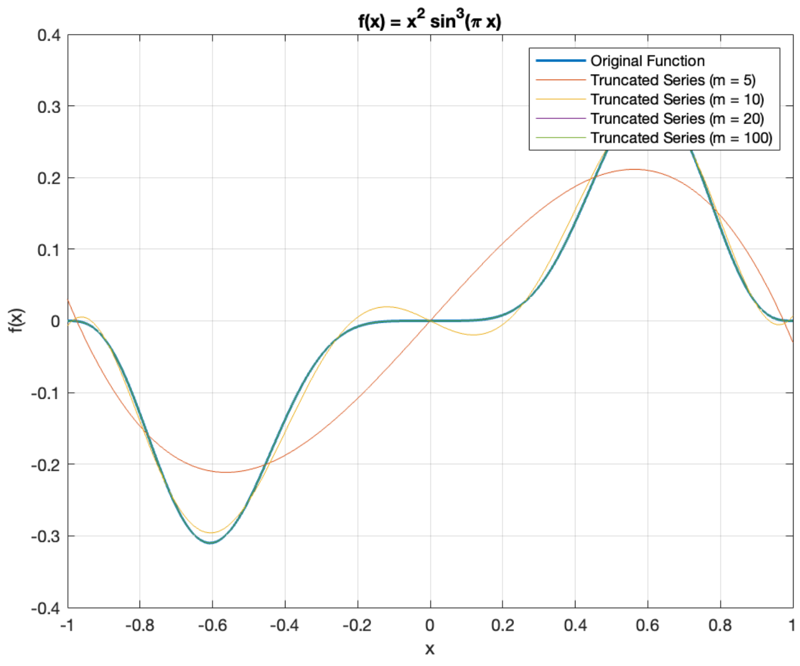


Figure 3: $f(x) = x^2 sin^3(\pi x)$

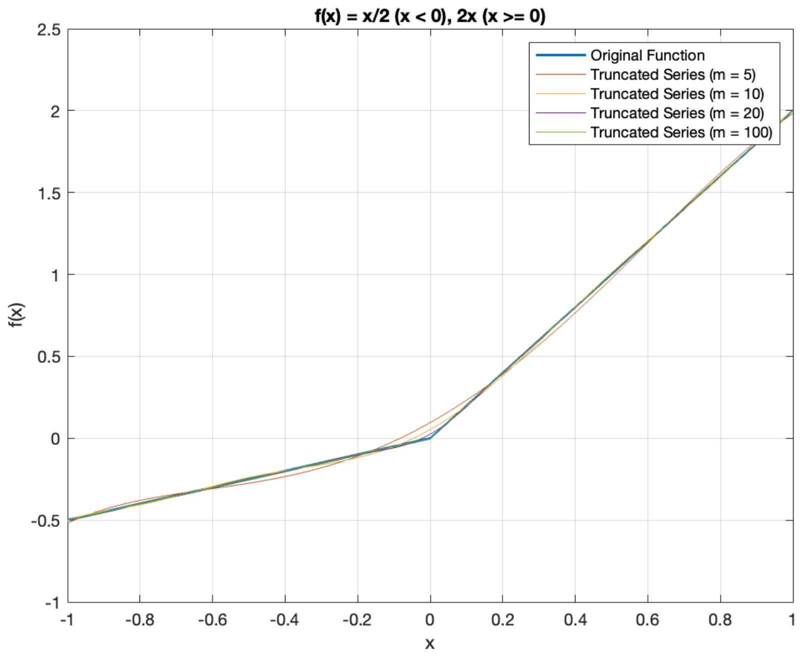


Figure 4: $f(x) = x/2(x < 0), 2x(x \geq 0)$

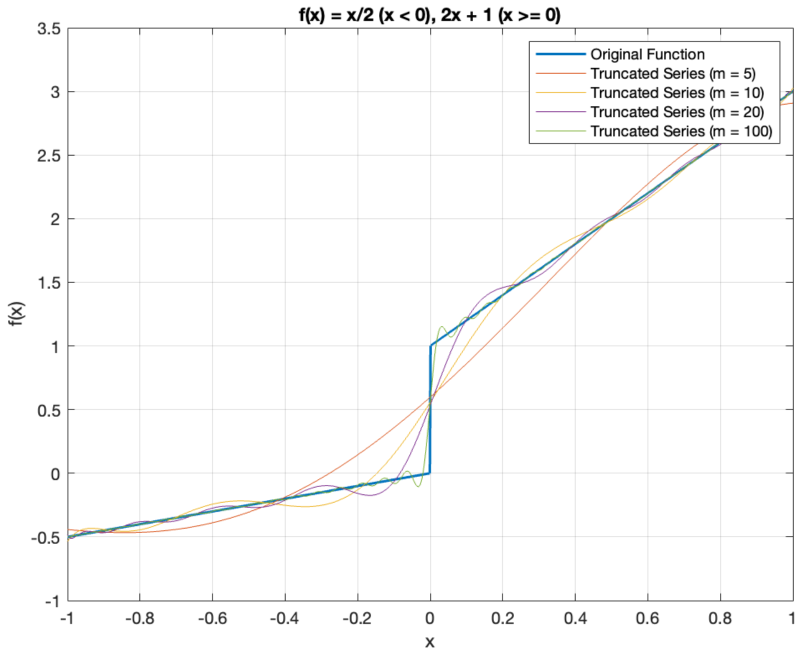


Figure 5: $f(x) = x/2(x < 0), 2x+1(x \geq 0)$

Part 2: Solving O.D.E

To solve the ordinary differential equation (1), one can approximate u by its truncated Chebyshev expansion

$$u(x) \approx \sum_{k=0}^{m-1} \alpha_k T_k(x).$$

Thus, one can represent the function u, by its Chebyshev coefficients. Hence, the goal is to form a system of equations for determining the coefficients $\alpha_0, \alpha_1, \cdots \alpha_{m-1}$.

6 Q6: Constructing D for Mapping α to β

6.1 Question Restatement

The Chebyshev polynomials satisfy the following relation

$$2T_k(x) = \frac{1}{k+1}\frac{dT_{k+1}(x)}{dx} - \frac{1}{k-1}\frac{dT_{k-1}(x)}{dx}.$$

Using this relation, it is possible to find coefficients β_k such that

$$\sum_{k=0}^{m-1} \alpha_k \frac{dT_k(x)}{dx} \approx \sum_{k=0}^{m-1} \beta_k T_k(x).$$

Write a function that takes as input the integer m, and outputs a matrix D that maps
the coefficients $\alpha_0, \alpha_1, \cdots \alpha_{m-1}$ to the coefficients $\beta_0, \beta_1, \cdots \beta_{m-1}$.

6.2 Analysis

The Chebyshev polynomials $T_k(x)$ satisfy the following recurrence relation:

$$2T_k(x) = \frac{1}{k+1}\frac{dT_{k+1}(x)}{dx} - \frac{1}{k-1}\frac{dT_{k-1}(x)}{dx}.$$

This relation allows us to express the derivatives of Chebyshev polynomials in terms of other Chebyshev polynomials. The goal is to find the coefficients β_k such that the following approximation holds:

$$\sum_{k=0}^{m-1} \alpha_k \frac{dT_k(x)}{dx} \approx \sum_{k=0}^{m-1} \beta_k T_k(x).$$

In other words, we want to map the derivatives of the Chebyshev polynomials (represented by the coefficients $\alpha_0, \alpha_1, \cdots, \alpha_{m-1}$) to the Chebyshev polynomials themselves (represented by the coefficients $\beta_0, \beta_1, \cdots, \beta_{m-1}$).

Using the recurrence relation, we can construct a matrix D that maps the coefficients α_k to the coefficients β_k. The matrix D is defined by the following rules:

- The elements of the matrix are filled according to the recurrence relation and the structure of the Chebyshev polynomials.
- The first row of the matrix is divided by 2 to account for the specific form of the recurrence for $k = 0$.

The matrix D is then used to convert the vector of coefficients $\alpha = [\alpha_0, \alpha_1, \ldots, \alpha_{m-1}]$ into the vector of coefficients $\beta = [\beta_0, \beta_1, \ldots, \beta_{m-1}]$.

The following function constructs the matrix D for a given integer m.

6.3 Solution

```
function D = construct_D_matrix(m)

D = zeros(m);

% Loop over the rows
for i = 1:m
    for j = i+1:2:m
        % Set values based on the formula of induction
        D(i, j) = 2 * (j - 1);
    end
end

% Adjust the first row of the matrix, dividing it by 2
D(1,:) = D(1,:) / 2;

end
```

Explanation of the Code

The function `construct_D_matrix` takes the integer m as input and outputs the matrix D that maps the coefficients α_k to the coefficients β_k. Here's how the function works:

- The matrix D is initialized as an $m \times m$ zero matrix.
- The `for` loops iterate over the rows and columns of the matrix. The matrix is populated based on the recurrence relation for the derivatives of Chebyshev polynomials.
- The values of $D(i, j)$ are set according to the formula $2(j-1)$ for appropriate indices i and j.
- The first row of the matrix is adjusted by dividing it by 2 to account for the special case when $k = 0$.

This matrix D can be used to perform the transformation from the derivative coefficients α_k to the Chebyshev polynomial coefficients β_k.

6.4 Example Usage

Take $m = 5$ as an example:

```
>> construct_D_matrix(5)

ans =

     0     1     0     3     0
     0     0     4     0     8
     0     0     0     6     0
     0     0     0     0     8
     0     0     0     0     0
```

7 Q7: Constructing the Matrix BC for Boundary Conditions

7.1 Question Restatement

To enforce the boundary conditions the coefficients $\alpha_0, \alpha_1, \cdots \alpha_{m-1}$ should satisfy

$$\sum_{k=0}^{m-1} \alpha_k T_k(-1) = l$$

and

$$\sum_{k=0}^{m-1} \alpha_k T_k(1) = r.$$

Build a matrix BC of size $2 \times m$ that represents these relations, i.e.

$$BC \cdot \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{m-1} \end{pmatrix} = \begin{pmatrix} l \\ r \end{pmatrix}.$$

7.2 Analysis

The goal is to construct a matrix BC of size $2 \times m$ that maps the coefficient vector $\alpha = [\alpha_0, \alpha_1, \ldots, \alpha_{m-1}]$ to the boundary values l and r. This results in the system:

$$BC \cdot \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_{m-1} \end{pmatrix} = \begin{pmatrix} l \\ r \end{pmatrix}.$$

The matrix BC is initialized as a $2 \times m$ zero matrix. Each row corresponds to the evaluation of the Chebyshev polynomials at the boundary points $x = -1$ and $x = 1$.

For the first row, we need to evaluate $T_k(-1)$. Using the property of Chebyshev polynomials, we know that:

$$T_k(-1) = (-1)^k \quad \forall\, k = 0, 1, 2, \ldots, m-1.$$

This alternates between 1 and -1 depending on whether k is even or odd.

For the second row, we evaluate $T_k(1)$. Since:

$$T_k(1) = 1 \quad \forall\, k = 0, 1, 2, \ldots, m-1$$

each entry in the second row is 1.

Thus, the matrix BC is constructed as follows:

$$BC = \begin{pmatrix} (-1)^0 & (-1)^1 & (-1)^2 & \cdots & (-1)^{m-1} \\ 1 & 1 & 1 & \cdots & 1 \end{pmatrix}.$$

7.3 Solution

```
function BC = construct_BC_matrix(m)

% Initialize the BC matrix with zeros
BC = zeros(2, m);

% First row, T_k(-1) = (-1)^k, alternating 1 and -1
for k = 0:(m-1)
    BC(1, k + 1) = (-1)^k;
end

% Second row for T_k(1)
BC(2, :) = 1;

end
```

7.4 Example Usage

```
>> BC = construct_BC_matrix(5)

BC =

     1    -1     1    -1     1
     1     1     1     1     1
```

8 Q8: Function to Solve O.D.E.

8.1 Question Restatement

Build a function that given a function handler f, the real values l and r, and an integer m, find a numerical solution to

$$\frac{d^2u(x)}{dx^2} + u(x) = f(x)$$

Note that the matrix associated to the second derivative corresponds to D^2.

8.2 Analysis

Given the Chebyshev coefficients α_k, we approximate the solution $u(x)$ as:

$$u(x) \approx \sum_{k=0}^{m-1} \alpha_k T_k(x).$$

We utilize the Chebyshev polynomials $T_k(x) = \cos(k \cdot \arccos(x))$ to evaluate the solution over a set of points x in the interval $[-1,1]$. Create a loop that iteratively computes:

$$y_{\text{vals}} = \sum_{k=0}^{m-1} \alpha_k T_k(x_{\text{vals}}).$$

To evaluate the second derivative $\frac{d^2u(x)}{dx^2}$, we use the matrix D to map coefficients of $u(x)$ to its derivatives. Let D be the differentiation matrix such that:

$$D2 = D \cdot D,$$

where $D2$ is the second differentiation matrix. Using $D2$, the coefficients for $\frac{d^2u(x)}{dx^2}$ are given by:

$$u''_{\text{coeff}} = D2 \cdot \alpha.$$

The second derivative is then reconstructed as:

$$u''(x) \approx \sum_{k=0}^{m-1} u''_{\text{coeff}}[k] \cdot T_k(x).$$

Then the left-hand side of the differential equation is computed as:

$$\text{LHS} = \frac{d^2u(x)}{dx^2} + u(x)$$

which we evaluate at the points x_{vals}.

Finally, we compare the LHS values with the given function $f(x)$. The difference is plotted to visualize the error or deviation:

$$\text{Error} = \text{LHS} - f(x).$$

This comparison provides insight into how well the numerical solution satisfies the differential equation over the interval $[-1,1]$.

Example usages of the following MATLAB function will be demonstrated in Q9 and Q10 sections.

8.3 Solution

```
function u = solveODE(f, l, r, m)
    I = eye(m);

    % Construct the second derivative matrix D^2
    D = construct_D_matrix(m);
    D2 = D * D;

    % Create matrix A = (D^2 + I)
    A = D2 + I;

    % Construct the boundary condition matrix
    BC = construct_BC_matrix(m);

    % Modify A to incorporate boundary conditions
    A = [A; BC];

    % Compute the Chebyshev coefficients of f(x)
    f_coeff = compute_chebyshev_coefficients(f, m, 60);

    % Create the right-hand side vector b
    b = f_coeff';

    % Apply boundary conditions
    b = [b; l; r];

    % Solve the linear system for the coefficients of u(x)
    u = A \ b;

    % Plot the solution and compare with f(x)
    plot_solution_and_lhs(u, f, m);
end
```

9 Q9: Verification of $u(x)$ with $f(x) = 1$

9.1 Question Restatement

If $l = r = 0$ and $f(x) = 1$, the solution can be shown to be $u(x) = 1 - cos(x)/cos(1)$. Compare your numerical solution this the exact solution for different values of m.

9.2 Analysis

In this task we set $l = r = 0$, and $f(x) = 1$. Using the function implemented in Q8, we obtain the numerical solution $u(x)$. Additionally, we compare this numerical solution to the exact solution $1 - \cos(x)/\cos(1)$ by evaluating the error for different values of m. The comparison results are visualized through plots, illustrating the accuracy and convergence of the numerical method as $m$varies. The error of numerical solution and exact solution is calculated by `norm( , 2)`.

9.3 Solution

```
clc
clear
close all;
l = 0;
r = 0;
f = @(x) ones(size(x));

% Set the exact u(x)
exact_solution = @(x) 1 - cos(x) / cos(1);

% Set different m
m_values = [ 3, 5, 10, 50];

```

```
% Take 100 points on [-1, 1]
x_vals = linspace(-1, 1, 100);
exact_vals = exact_solution(x_vals);

% Test with different m
for m = m_values

    % Implement 'solveODE'
    u = solveODE(f, l, r, m);

    y = zeros(size(x_vals));
    for k = 0:(m-1)
        T_k = cos(k * acos(x_vals));
        y = y + u(k + 1) * T_k;
    end

    % Calculate the error
    error = norm(y - exact_vals, 2);
    disp(['Error for m = ', num2str(m), ': ', num2str(error)]);
    fprintf('\n')

    % Plot to compare numerical solution to exact solution
    figure;
    subplot(2, 1, 1);
    hold on;
    plot(x_vals, y, '-', 'LineWidth', 1.2);
    plot(x_vals, exact_vals, '.', 'LineWidth', 2);
    legend('Numerical Solution of u(x)', 'Exact Solution of u(x)');
    title(['Comparison of Numerical and Exact Solutions ...
        (m = ', num2str(m), ')']);
    xlabel('x');
    ylabel('u(x)');
    grid on;
    hold off;

    % plot the error
    subplot(2, 1, 2);
    plot(x_vals, y - exact_vals, 'k-', 'LineWidth', 2);
    title(['Error between Numerical and Exact Solutions ...
        (m = ', num2str(m), ')']);
    xlabel('x');
    ylabel('Error');
    grid on;
end
```

9.4 Output

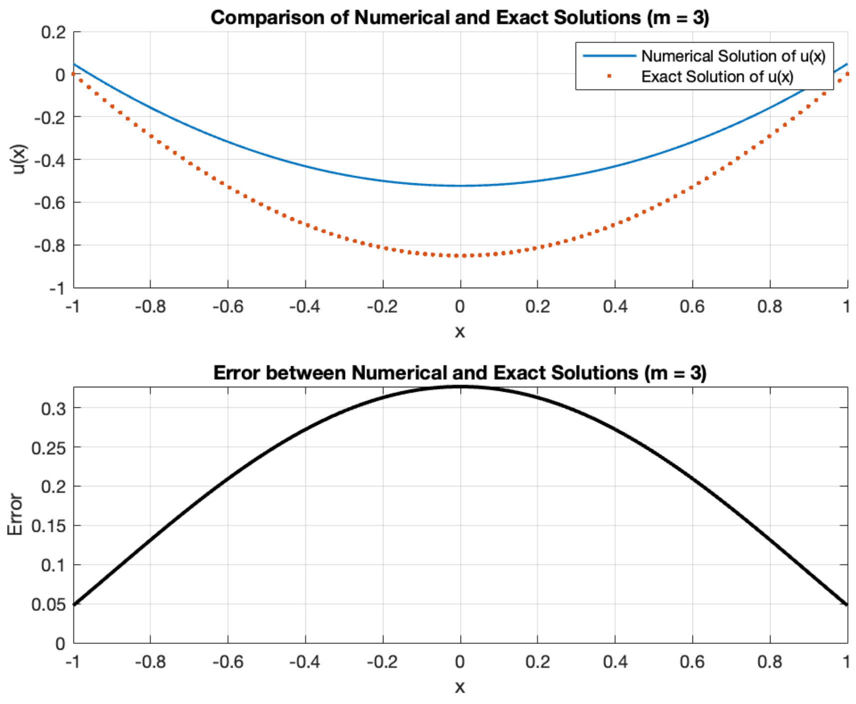


Figure 6: m = 3, error: 2.3905

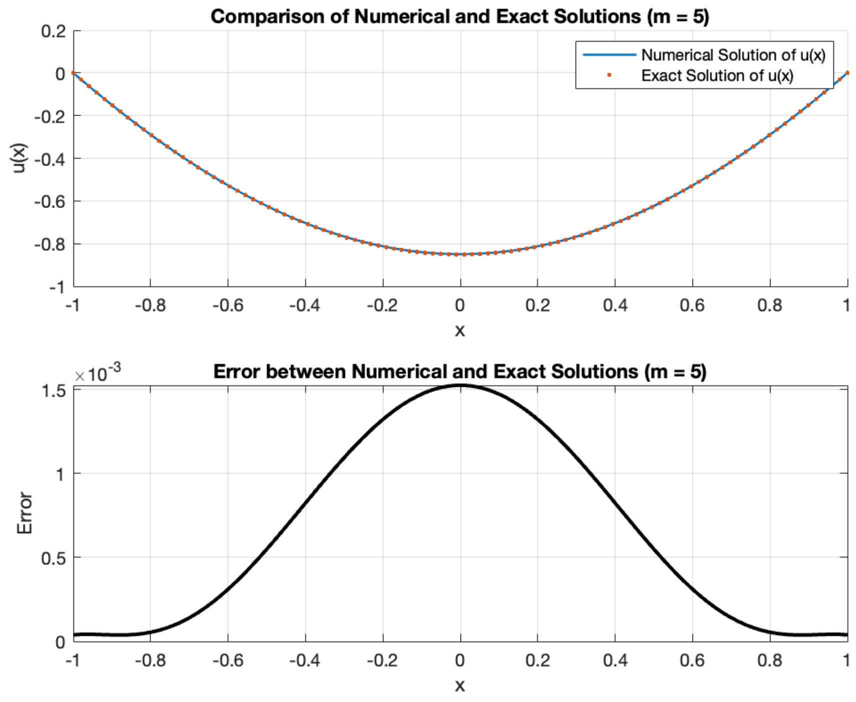


Figure 7: m = 5, error: 0.0085502

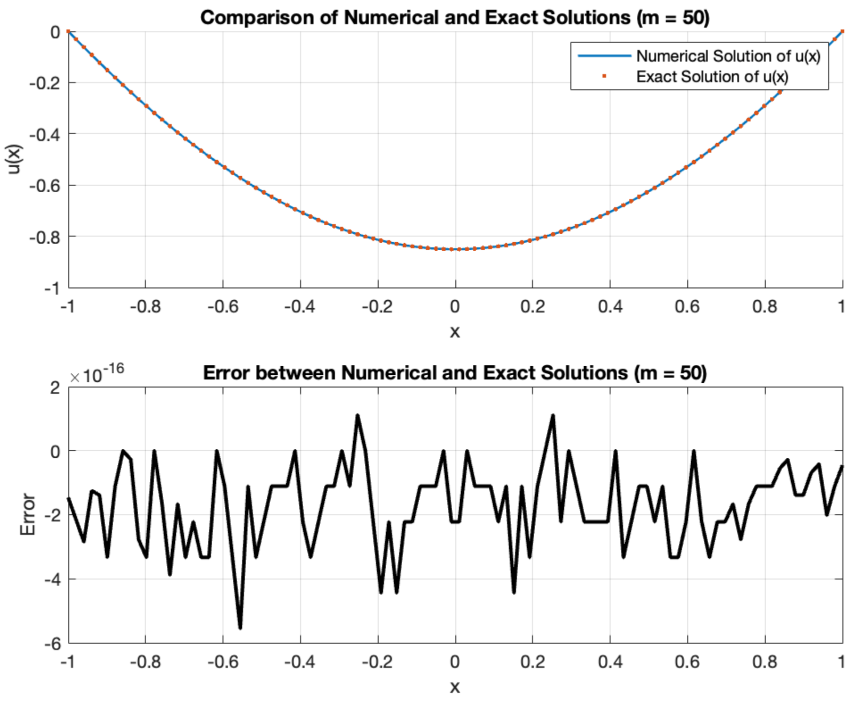


Figure 8: m = 50, error: 2.1127e-15

As m increases, the numerical solution closely approximates the exact solution, with the error significantly decreasing. For instance, when $m = 3$, the error is 2.3905, but this reduces to nearly zero (2.1127×10^{-15}) by $m = 50$. This demonstrates that using more Chebyshev nodes enhances the solution's accuracy.

10 Q10: Test with Different $f(x)$, m and l

10.1 Question Restatement

Show your numerical solution to $\frac{d^2u(x)}{dx^2} + u(x) = f(x)$ for different $f(x)$ and different l and m

10.2 Example Usage

We consider $r = 0$ and test with the following two functions:

- $f(x) = \sin(x)$
- $f(x) = e^x$

For each function, we set l to be -1, 0, and 1; and choose m values as 1, 3, 6, and 20.

Since the exact solution $u(x)$ is unknown, we visualize our results by plotting the left-hand side (LHS) of the equation, defined as $\frac{d^2u}{dx^2} + u(x)$, alongside $f(x)$. This comparison allows us to assess the accuracy of our numerical solutions.

10.3 Output

To illustrate the differences in error, we created two tables displaying the error metrics under various values of m and l.

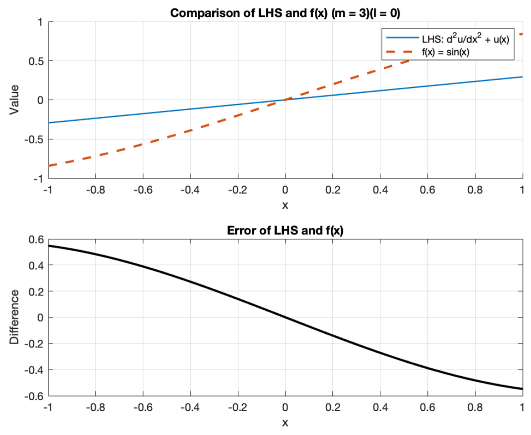
Error	$m = 1$	$m = 3$	$m = 6$	$m = 20$
$l = -1$	19.6016	6.5152	0.10285	2.3271e-14
$l = 0$	16.5216	11.1729	0.034318	2.5905e-14
$l = 1$	19.6106	17.7223	0.10709	3.2096e-14

Table 1: The error of $f(x) = \sin(x)$ under different m and l

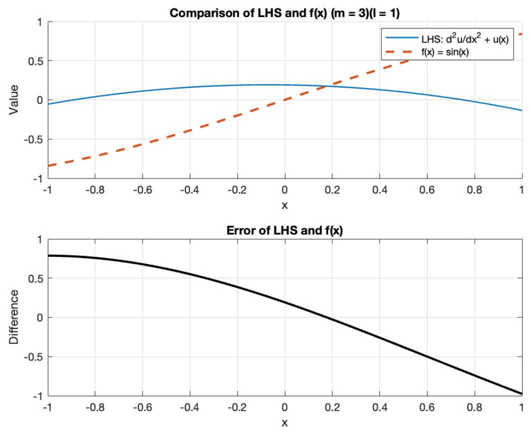
Error	$m = 1$	$m = 3$	$m = 6$	$m = 20$
$l = -1$	40.1873	10.5873	0.31067	6.4903e-14
$l = 0$	31.6426	13.8142	0.21201	6.2287e-14
$l = 1$	24.6981	19.4019	0.11434	5.2047e-14

Table 2: The error of $f(x) = e^x$ under different m and l

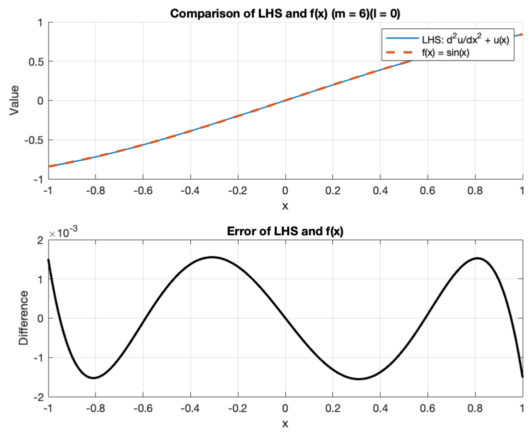
Additionally, we present the graphs of $f(x) = \sin(x)$ and $f(x) = \exp(x)$ in Figures 9 and 10, respectively. These graphs are plotted for $m = 3, 6, 20$ and $l = 0, 1$, and will be shown over the next two pages.



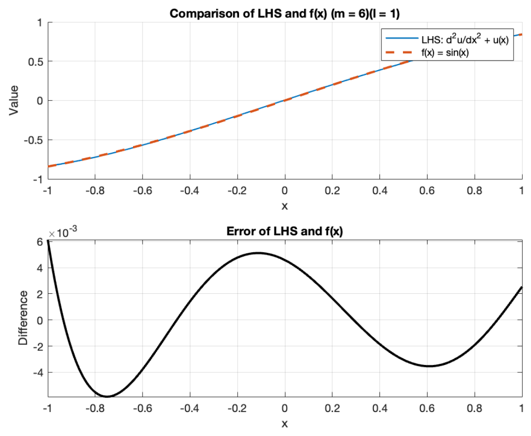
(a) $m = 3$
$l = 0$



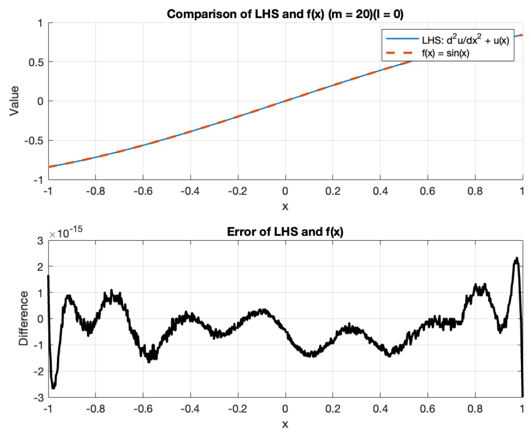
(b) $m = 3$
$l = 1$



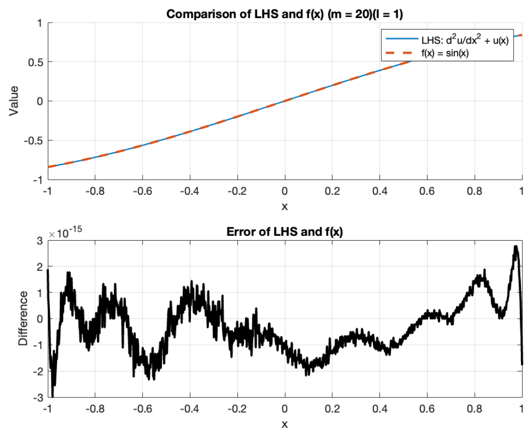
(c) $m = 6$
$l = 0$



(d) $m = 6$
$l = 1$

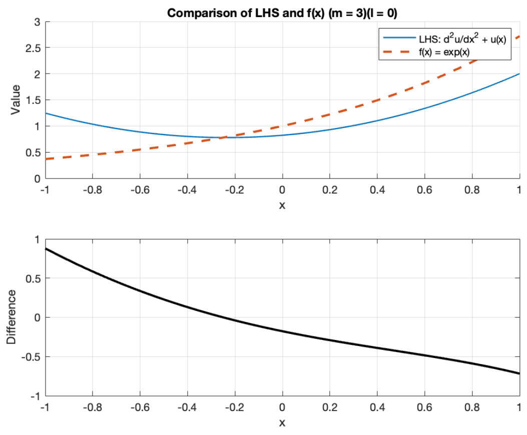


(e) $m = 20$
$l = 0$

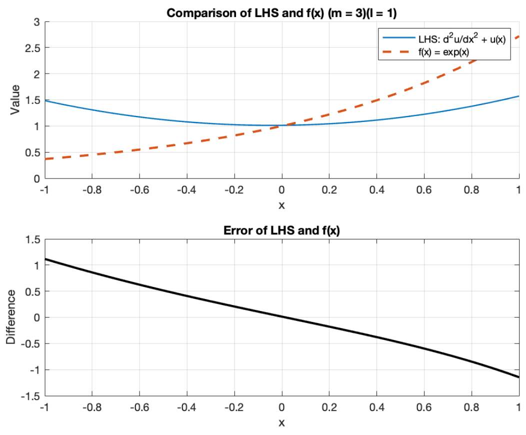


(f) $m = 20$
$l = 1$

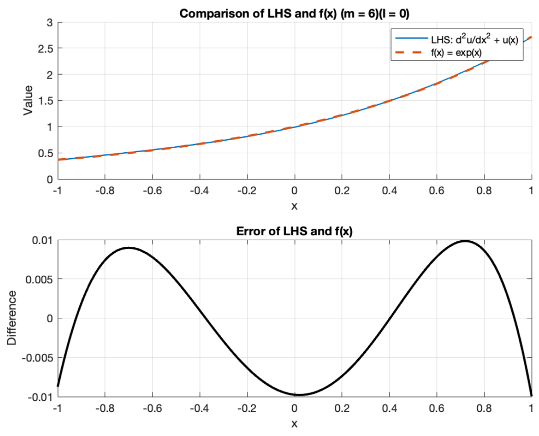
Figure 9: Comparison for LHS and $f(x) = \sin(x)$



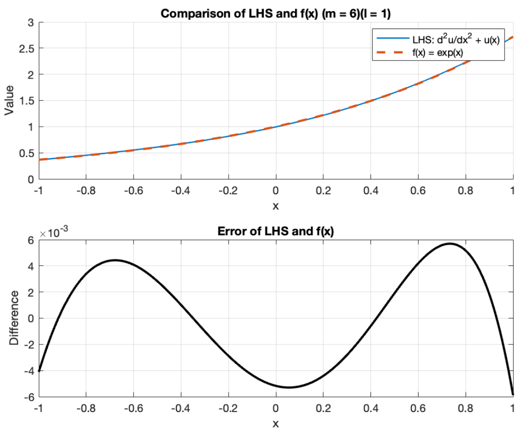
(a) $m = 3$
$l = 0$



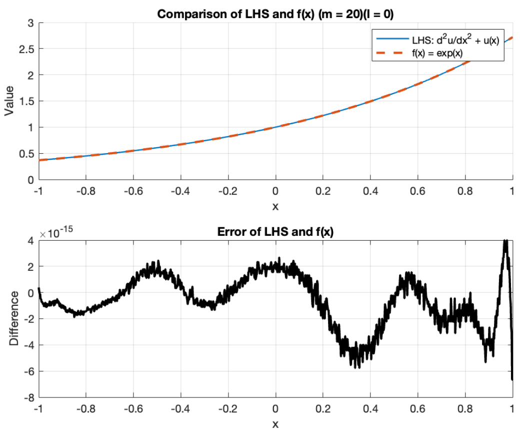
(b) $m = 3$
$l = 1$



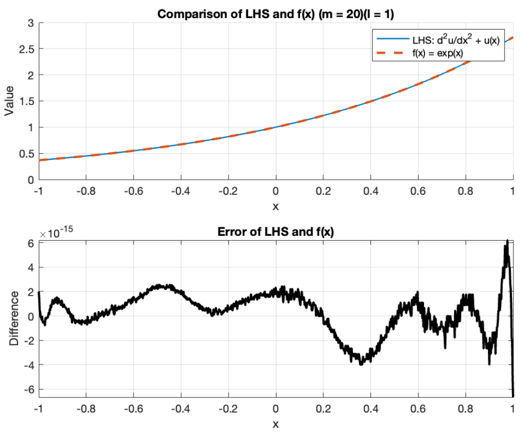
(c) $m = 6$
$l = 0$



(d) $m = 6$
$l = 1$



(e) $m = 20$
$l = 0$



(f) $m = 20$
$l = 1$

Figure 10: Comparison for LHS and $f(x) = e^x$

11 Reference

References

[1] J. P. Boyd, *Chebyshev and Fourier Spectral Methods*, Dover Publications, 2001.

[2] T. Siauw and A. M. Bayen, *An Introduction to MATLAB Programming and Numerical Methods for Engineers*, Elsevier, 2015.