Numerical solutions to second-order differential equations using cubic Hermite spline interpolation

This thesis is presented to the Department of Mathematics
in the College of Natural and Computational Sciences
at Addis Ababa University, in partial fulfillment of the requirements
for the Master of Science degree in Mathematics

By

Mulugeta Alemnew

Advisor: Dr Temesgen Alemu

August 20, 2024

Addis Ababa, Ethiopia

Approval

We, the undersigned, have reviewed this thesis and recommend its acceptance by the Department of Mathematics. The thesis, titled **Numerical solutions to second-order differential equations using cubic Hermite spline interpolation**, authored by **Mulugeta Alemnew**, fulfills the requirements for the Master of Science degree in Mathematics.

Advisor: Dr. Temesgen Alemu

Signature _____

Date _____

Examiners :

1. Dr. Markos Fiseha

   Signature _____

   Date _____

2. Dr.Kassahune Workalemahu

   Signature _____

   Date _____

Acknowledgements

First and foremost, I wish to express my profound gratitude to the Almighty for His guidance and support throughout this journey. This project would not have been possible without the contributions and assistance of many remarkable individuals.

I am deeply grateful to my advisor, Dr. Temesgen Alemu, for his generous time, insightful guidance, and unwavering support. His encouragement and expertise were crucial in the successful completion of this project.

A heartfelt thank you to my family, especially my mother, whose constant encouragement and inspiration gave me the strength to undertake and complete this project.

I also extend my sincere appreciation to Addis Ababa University's School of Graduate Studies for their support and resources.

Table of Contents

Abstract

In this thesis, we introduce cubic Hermite spline as a highly effective interpolation method for approximating curves using discrete data points. While traditional cubic splines provide a strong foundation for curve interpolation, they often fail to accurately capture curve behavior, particularly in situations where tangent information is essential. Cubic Hermite splines overcome this limitation by incorporating tangent data at each point, leading to smoother and more accurate curve approximations.

We offer an in-depth examination of cubic Hermite splines, detailing their mathematical formulation, computational methods, and practical implementation strategies. Using tangent information, cubic Hermite splines deliver greater flexibility in modeling complex curve shapes, making them especially valuable in fields such as computer graphics, animation, and engineering design. Additionally, we present comparative analyses highlighting the benefits of cubic Hermite splines over traditional interpolation techniques, demonstrating their effectiveness in maintaining curve characteristics and reducing interpolation errors.

Through both theoretical analysis and practical examples, we demonstrate the versatility and efficiency of cubic Hermite splines in various real-world applications. This thesis aims to be a useful resource for researchers, practitioners, and enthusiasts interested in advanced interpolation techniques for curve approximation and modeling.

CHAPTER ONE

Introduction

Historically, mathematicians have sought solutions to linear and nonlinear two-point boundary value problems (BVPs) of the form:

$$\frac{d^2y}{dx^2} + \alpha_1(x)\frac{dy}{dx} + \alpha_2(x)y = f(x), \quad x \in [a, b] \tag{1.1}$$

Boundary value problems (BVPs) are essential in various fields, including astronomy, biology, boundary layer theory, cable deflection, diffusion processes, electromagnetism, and heat transfer. Analytical solutions to these problems are often elusive, which necessitates the use of numerical methods. These methods include the collocation method, B-spline interpolation, cubic Hermite collocation, finite difference method, nonlinear shooting method, geometric Hermite interpolation, quintic B-spline collocation, polynomial and non-polynomial spline approaches, quartic spline solution, cubic spline collocation method, and finite volume element method.

This thesis focuses on the Cubic Hermite Collocation Method (CHCM), which utilizes cubic Hermite basis functions to tackle complex mathematical problems. CHCM is employed to address a range of linear and nonlinear differential equations with various boundary conditions, including Dirichlet, Neumann, and Robin conditions. The study also demonstrates that CHCM effectively solves elliptic problems under Neumann and Dirichlet conditions using a decoupling technique. It underscores CHCM's capability to handle both linear and nonlinear boundary value problems, as supported by recent research, and shows that CHCM provides superior accuracy compared to other methods such as the finite difference method (FDM), finite element method (FEM), finite volume method, B-spline method, and polynomial and non-polynomial spline techniques, achieving fourth-order convergence.

The thesis is organized into five chapters. Chapter One introduces the problem. Chapter Two outlines the preliminary concepts. Chapter Three discusses cubic splines. Chapter Four explores the numerical solutions of cubic Hermite splines and presents two numerical examples. Finally, Chapter Five offers the conclusions.

1.1 Literature Review

Cubic Hermite splines are a distinct category of cubic splines that ensure continuity not only in the function values but also in their first derivatives. These splines are particularly beneficial for solving boundary value problems (BVPs) in numerical analysis, especially when both the function values and derivatives are known or can be accurately approximated at specific points (knots).

These splines are constructed by specifying both the function values and their derivatives at each knot, facilitating smooth interpolation between points. This method is especially effective in representing complex function behaviors, which is crucial in engineering and scientific applications.

In their 2014 study, Ahmad, Arora, and Kukreja explored the application of the cubic Hermite collocation method for solving BVPs under various boundary conditions, including Dirichlet, Neumann, and Robin conditions [10]. Their approach involves selecting collocation points where the differential equation is satisfied, typically where both function values and derivatives are specified. Their research indicates that this method offers higher accuracy compared to traditional techniques, particularly when precise control over both function values and slopes is necessary.

The theoretical foundations of cubic Hermite splines are extensively covered in numerical analysis literature, such as the work by Burden and Faires (2005) [12]. Their texts elaborate on the mathematical formulation of cubic Hermite splines, including the derivation of spline coefficients and criteria for maintaining derivative continuity. They also address various boundary conditions, demonstrating the flexibility and broad applicability of cubic Hermite splines in applied mathematics and engineering.

Jain (2007) provides a comprehensive overview of numerical methods, with a focus on the implementation of cubic Hermite splines for solving differential equations [1]. His work highlights the importance of numerical stability and accuracy, which are crucial for practical engineering applications.

Kincaid and Cheney (2002) delve into the mathematical foundations of cubic Hermite splines within the context of scientific computing, offering detailed algorithms for their implementation [2]. Their book is a valuable resource for understanding

the computational efficiency and practical considerations of using these splines.

Sastry (2003, 2012) introduces fundamental numerical analysis methods, including cubic Hermite splines, with an emphasis on their real-world applications [3]. His texts serve as essential resources for students and practitioners seeking to understand the basics and applications of numerical methods.

Plato (2003) and Rolston (2001) discuss various numerical methods, including cubic Hermite splines, emphasizing their precision in solving boundary value problems [4].

Ware and Ashine (2021) investigate the integration of cubic splines with finite difference methods for solving boundary value problems in ordinary differential equations [7]. Their research highlights the effectiveness of these combined approaches in achieving accurate solutions.

Rashidinia et al. (2008) present the cubic spline method for two-point boundary value problems, addressing both theoretical and practical aspects of using cubic Hermite splines [9].

Ganaie, Arora, and Kukreja (2014) specifically focus on the cubic Hermite collocation method for solving BVPs with various boundary conditions, underscoring the method's benefits in terms of accuracy and computational efficiency [10].

Collectively, these references illustrate the significant role of cubic Hermite splines in numerical analysis, particularly in solving boundary value problems with high precision and reliability.

1.2 General Objective

- Solving differential equations using numerical analysis involves a variety of methods that approximate solutions to these equations, which may not be solvable analytically
- To explore, analyze, and implement cubic Hermite splines for efficient interpolation and data fitting in various applications, while evaluating their performance and accuracy in comparison to other spline methods.

1.3 Specific Objectives

- Formulate and apply the Cubic Hermite Collocation Method (CHCM) to solve boundary value problems involving differential equations.
- Evaluate the performance of CHCM by comparing its accuracy and efficiency with other numerical methods, such as the finite difference , b-spline, cubic spline and finite volume .
- Investigate how the choice of collocation points affects the convergence and stability of CHCM.
- Compile and present the research findings, highlighting the advantages and limitations of using CHCM for boundary value problems.

1.4 Advantages and Limitations

ADVANTAGES

- High Accuracy: Cubic Hermite splines offer high accuracy by being able to approximate function values and derivatives effectively.
- Smoothness: These splines maintain smoothness at the interpolation points, ensuring a visually and analytically pleasing representation of the function.

LIMITATIONS

- Complexity: Setting up and solving the cubic Hermite spline equations can be more complex than simpler interpolation methods.
- Computational Cost: The computational cost can be higher, particularly when compared to linear or quadratic spline methods.

CHAPTER TWO

Preliminaries

2.1 Polynomial Interpolation Theory

Given the ordered data points $x_0 < x_1 < x_2 < \ldots < x_n$ and the corresponding function values $f(x_i) = y_i$ for $i = 0, 1, 2, \ldots, n$, where each x_i is unique, our goal is to determine a polynomial $P(x)$ of degree n that interpolates these data points. The polynomial $P(x)$ is expressed as:

$$P(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n \tag{2.1}$$

To ensure that the polynomial passes through each of the given data points, we enforce the condition $P(x_i) = f(x_i)$ for all i. This results in the following system of equations:

$$\begin{aligned} a_0 + a_1 x_0 + a_2 x_0^2 + \cdots + a_n x_0^n &= y_0, \\ a_0 + a_1 x_1 + a_2 x_1^2 + \cdots + a_n x_1^n &= y_1, \\ &\vdots \\ a_0 + a_1 x_n + a_2 x_n^2 + \cdots + a_n x_n^n &= y_n \end{aligned} \tag{2.2}$$

This results in a system of $(n+1)$ linear equations with $(n+1)$ unknowns $a_0, a_1, a_2, \ldots, a_n$. The matrix representation of this system is given by:

$$\mathrm{X}\mathbf{a} = \mathbf{Y} \tag{2.3}$$

where

$$\mathrm{X} = \left[x_i^j\right]_{i,j=0,1,2,\ldots,n}, \quad \mathbf{a} = \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}, \quad \text{and} \quad \mathbf{Y} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}. \tag{2.4}$$

The matrix X is known as the Vandermonde matrix. Solving the system of equations is equivalent to solving the polynomial interpolation problem.

Theorem 2.1.1. *Given $n+1$ distinct points $x_0 < x_1 < x_2 < \ldots < x_n$ and $n+1$ real values $y_0, y_1, y_2, \ldots, y_n$, there exists a unique polynomial P of degree $\leq n$ that interpolates the data.*

x_0	x_1	x_2	$\cdots$	x_n
$y_0 = f(x_0)$	y_1	y_2	$\cdots$	y_n

Proof. It can be shown that the determinant of the matrix X is given by:

$$\mathrm{Det(X)} = \prod_{0 \le j < i \le n} (x_i - x_j)$$

Since this determinant is non-zero, the matrix X is invertible. Consequently, the system of equations has a unique solution. □

To prove the uniqueness of $P(x)$, assume there exists another polynomial $P^*(x)$ that also satisfies $P^*(x_i) = f(x_i)$ for $i = 0, 1, 2, \ldots, n$.

Consider the polynomial $Q(x) = P(x) - P^*(x)$. Since both $P(x)$ and $P^*(x)$ are polynomials of degree at most n, the polynomial $Q(x)$ will also have a degree of at most n. It satisfies $Q(x_i) = P(x_i) - P^*(x_i) = 0$ for $i = 0, 1, 2, \ldots, n$.

Therefore, $Q(x)$ is a polynomial of degree $\le n$ with $n+1$ distinct roots, namely $x_0, x_1, x_2, \ldots, x_n$. However, a polynomial of degree n can have at most n roots unless it is the zero polynomial. Thus, $Q(x) \equiv 0$, which implies that $P(x) = P^*(x)$.

Consequently, the polynomial $P(x)$ that interpolates the given data points is unique, ensuring that the coefficients $a_0, a_1, a_2, \ldots, a_n$ of the interpolating polynomial of degree n are also unique.

2.2 Lagrange Interpolation Polynomial

Definition 2.2.1. *Given a set of $n+1$ distinct nodes $\{x_0, x_1, \cdots, x_n\}$ where $x_j \neq x_m$ for $j \neq m$, the Lagrange basis polynomials $\{l_0(x), l_1(x), \cdots, l_n(x)\}$ are defined as follows. Each $l_j(x)$ is a polynomial of degree n that satisfies:*

$$l_j(x_m) = \begin{cases} 1 & \text{if } m = j, \\ 0 & \text{if } m \neq j. \end{cases}$$

The Lagrange basis polynomial $l_j(x)$ is given by:

$$l_j(x) = \frac{\prod_{\substack{0 \le m \le n \\ m \neq j}} (x - x_m)}{\prod_{\substack{0 \le m \le n \\ m \neq j}} (x_j - x_m)}$$

Here, the numerator $\prod_{m\neq j}(x-x_m)$ has n roots at the nodes $\{x_m\}_{m\neq j}$, while the denominator $\prod_{m\neq j}(x_j-x_m)$ normalizes the polynomial so that $l_j(x_j)=1$.

The Lagrange interpolating polynomial for the $n+1$ nodes with corresponding values $\{y_0, y_1, \cdots, y_n\}$ is given by:

$$L_n(x) = \sum_{j=0}^{n} y_j l_j(x)$$

Each basis polynomial $l_j(x)$ has degree n, and thus the polynomial $L_n(x)$ also has degree $\leq n$ and interpolates the data.

Proof. Let $f(x)$ be a function with values $f(x_0), f(x_1), \ldots, f(x_n)$ at the distinct points $x_0, x_1, \ldots, x_n$. The Lagrange interpolating polynomial $L_n(x)$ that represents $f(x)$ is given by:

$$L_n(x) = \sum_{j=0}^{n} f(x_j) \prod_{\substack{0\leq m\leq n \\ m\neq j}} \frac{x-x_m}{x_j-x_m}$$

To determine the coefficients of the polynomial, consider the expanded form of $L_n(x)$:

$$L_n(x) = a_0(x-x_1)(x-x_2)\cdots(x-x_n) + a_1(x-x_0)(x-x_2)\cdots(x-x_n) + \cdots$$

Evaluating $L_n(x_i)$ for each i yields:

$$L_n(x_i) = f(x_i) = a_0(x_i-x_1)(x_i-x_2)\cdots(x_i-x_n)+a_1(x_i-x_0)(x_i-x_2)\cdots(x_i-x_n)+\cdots+a_n(x_i-x_0)(x_i-x_1)$$

To solve for the coefficients a_j, we have:

$$a_j = \frac{f(x_j)}{\prod_{\substack{0\leq m\leq n \\ m\neq j}}(x_j-x_m)}$$

Therefore, the Lagrange interpolating polynomial $L_n(x)$ can be expressed as:

$$L_n(x) = \sum_{j=0}^{n} \frac{f(x_j)}{\prod_{\substack{0\leq m\leq n \\ m\neq j}}(x_j-x_m)} \prod_{\substack{0\leq m\leq n \\ m\neq j}} (x-x_m)$$

Substituting a_j into $L_n(x)$, we obtain:

$$L_n(x) = \sum_{j=0}^{n} f(x_j) l_j(x)$$

where $l_j(x)$ are the Lagrange basis polynomials, satisfying $l_j(x_i) = 0$ for $i \neq j$ and $l_j(x_j) = 1$. This confirms that $L_n(x)$ interpolates the data points exactly.

□

2.3 Lagrange Interpolating Polynomial Error Bound Theorem

Theorem 2.3.1. *Let $x_0, x_1, \ldots, x_n$ be distinct points in the interval $[a, b]$, and suppose $f \in C^{n+1}[a, b]$. Then for every $x \in [a, b]$, there exists a point $\xi(x)$ such that $\xi(x)$ is located between $x_0, x_1, \ldots, x_n$ and consequently within (a, b). This point $\xi(x)$ satisfies:*

$$f(x) = L_n(x) + \frac{f^{(n+1)}(\xi(x))}{(n+1)!} \prod_{i=0}^{n} (x - x_i), \tag{2.5}$$

where $L_n(x)$ denotes the Lagrange interpolating polynomial given by:

$$L_n(x) = \sum_{j=0}^{n} l_j(x) f(x_j),$$

and $l_j(x)$ represents the Lagrange basis polynomials defined as:

$$l_j(x) = \frac{\prod_{\substack{0 \leq m \leq n \\ m \neq j}} (x - x_m)}{\prod_{\substack{0 \leq m \leq n \\ m \neq j}} (x_j - x_m)}.$$

Proof. If $x = x_k$ for some $k \in \{0, 1, 2, \ldots, n\}$, the second term in the error bound equation (2.5) vanishes, and thus $f(x_k) = L_n(x_k)$.

To prove the theorem, consider the function $g(t)$ defined for $t \in [a, b]$ by

$$g(t) = f(t) - L_n(t) - [f(x) - L_n(x)] \prod_{i=0}^{n} \frac{t - x_i}{x - x_i}.$$

Given that $f \in C^{n+1}[a, b]$ and $L_n(x)$ is a polynomial of degree n, the function $g(t)$ is also in $C^{n+1}[a, b]$. Moreover, $g(x_i) = 0$ for $i = 0, 1, 2, \ldots, n$.

From the definition of $g(x)$, we have:

$$f(x) - L_n(x) = [f(x) - L_n(x)] \prod_{i=0}^{n} \frac{t - x_i}{x - x_i}.$$

By applying the generalized Rolle's Theorem, there exists a point $\xi(x)$ in (a, b) such that

$$g^{(n+1)}(\xi(x)) = 0.$$

We can express the $(n+1)$-th derivative of $g(t)$ as:

$$\frac{d^{(n+1)}}{dt^{(n+1)}}\left[f(t) - L_n(t) - [f(x) - L_n(x)]\prod_{i=0}^{n}\frac{t-x_i}{x-x_i}\right]_{t=\xi(x)}.$$

Since $L_n(x)$ is a polynomial of degree n, its $(n+1)$-th derivative is zero:

$$\frac{d^{(n+1)}}{dt^{(n+1)}}L_n(t) = 0.$$

Thus, we have:

$$\frac{d^{(n+1)}}{dt^{(n+1)}}\left[f(t) - L_n(t) - [f(x) - L_n(x)]\prod_{i=0}^{n}\frac{t-x_i}{x-x_i}\right]_{t=\xi(x)}$$

Rearranging this, we obtain:

$$0 = f^{(n+1)}(\xi(x)) - \frac{(n+1)!}{\prod_{i=0}^{n}(x-x_i)}\left[f(x) - L_n(x)\right].$$

Solving for $f(x)$, we derive:

$$f(x) = L_n(x) + \frac{f^{(n+1)}(\xi(x))}{(n+1)!}\prod_{i=0}^{n}(x-x_i). \tag{2.6}$$

□

Example 2.3.1. *Consider the function $f(x) = \frac{1}{x}$ on the interval $[2,4]$. Using the nodes $x_0 = 2$, $x_1 = 2.75$, and $x_2 = 4$, derive the error form for the Lagrange polynomial and determine the maximum error when this polynomial approximates $f(x)$ over $x \in [2,4]$.*

Solution 2.3.1. *Given $f(x) = \frac{1}{x}$, we have $f'(x) = -x^{-2}$, $f''(x) = 2x^{-3}$, and $f'''(x) = -6x^{-4}$.*

The error form for the second Lagrange polynomial is given by:

$$f(x) - L_2(x) = \frac{f^{(3)}(\xi(x))}{3!}\prod_{i=0}^{2}(x-x_i),$$

where $\xi(x)$ is a number between x_0, x_1, and x_2, and hence in $(2,4)$.

Substituting $f^{(3)}(x) = -6x^{-4}$, we have:

$$f(x) - L_2(x) = \frac{-6\xi(x)^{-4}}{6}\prod_{i=0}^{2}(x-x_i) = -\xi(x)^{-4}(x-2)(x-2.75)(x-4).$$

To determine the maximum error, we find the maximum value of $|\xi(x)^{-4}|$ *on the interval* $[2, 4]$. *The maximum value occurs when* $\xi(x)$ *is closest to 2, so* $|\xi(x)^{-4}|$ *is maximized when* $\xi(x) = 2$, *giving* $\xi(x)^{-4} = 2^{-4} = \frac{1}{16}$.

Now we seek to find the maximum value of $|(x-2)(x-2.75)(x-4)|$ *over the interval* $[2, 4]$.

The polynomial $g(x)$ *is:*

$$g(x) = (x - 2)(x - 2.75)(x - 4).$$

To find the maximum value, we compute the derivative:

$$g'(x) = 3x^2 - \frac{35}{2}x + \frac{49}{4}.$$

Setting the derivative equal to zero, we find the critical points:

$$3x^2 - \frac{35}{2}x + \frac{49}{4} = 0 \quad \Rightarrow \quad x = \frac{35 \pm \sqrt{1225 - 588}}{12}.$$

Simplifying, we find the critical points are $x \approx 2.8$ *and* $x \approx 3.5$.

Evaluating $g(x)$ *at these points and the endpoints:*

$$g(2) = 0,$$

$$g(2.75) = -0.234375,$$

$$g(4) = 0,$$

$$g(2.8) \approx -0.144,$$

$$g(3.5) \approx -0.039.$$

The maximum value of $|g(x)|$ *on* $[2, 4]$ *is* $|g(2.75)| = 0.234375$.

Thus, the maximum error is:

$$\left| \frac{f^{(3)}(\xi(x))}{3!}(x - 2)(x - 2.75)(x - 4) \right| \leq \frac{1}{16 \cdot 6} \cdot 0.234375 = \frac{0.234375}{96} \approx 0.00244.$$

2.4 Hermite's Interpolation Formula

The interpolation methods discussed previously rely solely on function values. We now introduce an interpolation approach that incorporates both function values and their first derivatives

at each interpolation point, known as Hermite's interpolation formula. The goal of this interpolation problem is to find a polynomial of the smallest degree, denoted as $H_{2n+1}(x)$, which satisfies the following conditions: given a set of data points (x_i, y_i, y_i') for $i = 0, 1, \ldots, n$, the polynomial should fulfill

$$H_{2n+1}(x_i) = y_i \quad \text{and} \quad H'_{2n+1}(x_i) = y'_i$$

for each i. This polynomial $H_{2n+1}(x)$ will interpolate the given data points, matching both the values and the derivatives at those points.

$$H_{2n+1}(x_i) = y_i \quad \text{and} \quad H'_{2n+1}(x_i) = y'_i$$

for $i = 0, 1, \ldots, n$. Here, $H_{2n+1}(x)$ is the polynomial that interpolates the given data points while matching both the function values and their first derivatives.

$$H_{2n+1}\left(x_i\right) = y_i \quad \text{and} \quad H'_{2n+1}\left(x_i\right) = y'_i \quad \text{for} \quad i = 0, 1, \ldots, n, \tag{2.7}$$

In this context, the prime notation signifies differentiation with respect to x. The polynomial $H_{2n+1}(x)$ is referred to as Hermite's interpolation polynomial. Given that there are $2n + 2$ conditions, it follows that $2n + 2$ coefficients need to be determined, and thus the polynomial's degree is $2n + 1$. Analogous to the Lagrange interpolation formula, we aim to express the polynomial in a similar manner:

$$H_{2n+1}(x) = \sum_{i=0}^{n} u_i(x) y_i + \sum_{i=0}^{n} v_i(x) y'_i, \tag{2.8}$$

where $u_i(x)$ and $v_i(x)$ are polynomials in x with a degree of $2n+1$. Utilizing the conditions from (2.6), we derive...

$$\begin{cases} u_i(x_j) = \begin{cases} 1 & \text{for } i = j, \\ 0 & \text{for } i \neq j \end{cases} \\ \frac{d}{dx} v_i(x_j) = \begin{cases} 1 & \text{for } i = j, \\ 0 & \text{for } i \neq j \end{cases} \end{cases} \tag{2.9}$$

Given that $u_i(x)$ and $v_i(x)$ are polynomials in x with a degree of $2n + 1$, we can express them as follows:

$$u_i(x) = A_i(x)\left[l_i(x)\right]^2 \quad \text{and} \quad v_i(x) = B_i(x)\left[l_i(x)\right]^2, \tag{2.10}$$

where $l_i(x)$ denotes the Lagrange basis polynomials. It is clear that $A_i(x)$ and $B_i(x)$ are linear functions with respect to x.

Thus, we write

$$u_i(x) = (a_i x + b_i)\left[l_i(x)\right]^2 \quad \text{and} \quad v_i(x) = (c_i x + d_i)\left[l_i(x)\right]^2. \tag{2.11}$$

Using the constraints provided in equations (2.7) and (2.9), we derive the following results:

$$\begin{cases} a_i x_i + b_i = 1, \\ c_i x_i + d_i = 0, \end{cases} \tag{2.12}$$

and

$$\begin{cases} a_i + 2l_i'(x_i) = 0, \\ c_i = 1. \end{cases} \tag{2.13}$$

From this analysis, we can establish the following relationships:

$$\begin{cases} a_i = -2l_i'(x_i), \\ b_i = 1 + 2x_i l_i'(x_i), \\ c_i = 1, \\ d_i = -x_i. \end{cases} \tag{2.14}$$

Consequently, the formulas for $u_i(x)$ and $v_i(x)$ can be written as follows:

$$u_i(x) = \left[1 - 2(x - x_i)l_i'(x_i)\right]\left[l_i(x)\right]^2 \tag{2.15}$$

and

$$v_i(x) = (x - x_i)\left[l_i(x)\right]^2. \tag{2.16}$$

By substituting the above expressions for $u_i(x)$ and $v_i(x)$ into equation (2.6), we get

$$H_{2n+1}(x) = \sum_{i=0}^{n} \left\{\left[1 - 2(x - x_i)l_i'(x_i)\right]\left[l_i(x)\right]^2 y_i + (x - x_i)\left[l_i(x)\right]^2 y_i'\right\}, \tag{2.17}$$

which is the Hermite interpolation formula.

The following examples demonstrate the application of Hermite's formula.

Example 2.4.1. *Determine the third-order Hermite polynomial that interpolates the points* (x_i, y_i, m_i)*, for* $i = 0, 1$.

By setting $n = 1$ *in Hermite's formula, we derive*

$$H_3(x) = \left[1 + \frac{2(x - x_0)}{h_1}\right] \frac{(x_1 - x)^2}{h_1^2} y_0 + \left[1 + \frac{2(x - x_1)}{h_1}\right] \frac{(x - x_0)^2}{h_1^2} y_1 + (x - x_0)\frac{(x_1 - x)^2}{h_1^2} y_0' + (x - x_1)\frac{(x - x_0)^2}{h_1^2} y_1',$$

where $h_1 = x_1 - x_0$.

Given the Lagrange basis polynomials

$$l_0(x) = \frac{x - x_1}{x_0 - x_1} \quad \textit{and} \quad l_1(x) = \frac{x - x_0}{x_1 - x_0},$$

with their derivatives

$$l_0'(x) = -\frac{1}{h_1} \quad \textit{and} \quad l_1'(x) = \frac{1}{h_1},$$

we can express

$$H_3(x) = \left[1 + \frac{2(x - x_0)}{h_1}\right] \frac{(x_1 - x)^2}{h_1^2} y_0 + \left[1 + \frac{2(x - x_1)}{h_1}\right] \frac{(x - x_0)^2}{h_1^2} y_1 + (x - x_0)\frac{(x_1 - x)^2}{h_1^2} y_0' + (x - x_1)\frac{(x - x_0)^2}{h_1^2} y_1'.$$

This is the desired third-order Hermite interpolation polynomial.

Example 2.4.2. *Determine the Hermite polynomial of degree 5 that fits the following data and use it to estimate the value of* $\ln 2.7$.

x	$y = \ln x$	$y' = \frac{1}{x}$
2.0	*0.69315*	*0.5*
2.5	*0.91629*	*0.4000*
3.0	*1.09861*	*0.33333*

First, compute the Lagrange basis polynomials $l_i(x)$*:*

$$l_0(x) = \frac{x - x_1}{x_0 - x_1}, \quad l_1(x) = \frac{x - x_0}{x_1 - x_0}, \quad l_2(x) = \frac{x - x_2}{x_0 - x_2}.$$

Substituting the given data points, we have:

$$l_0(x) = \frac{x - 2.5}{2.0 - 2.5} = \frac{x - 2.5}{-0.5} = -2(x - 2.5),$$

$$l_1(x) = \frac{x - 2.0}{2.5 - 2.0} = \frac{x - 2.0}{0.5} = 2(x - 2.0).$$

The derivatives of these polynomials are:

$$l_0'(x) = -2, \quad l_1'(x) = 2, \quad l_2'(x) = 0.$$

Next, define the Hermite basis functions:

$$u_0(x) = (6x - 11)\left(2x^2 - 11x + 15\right)^2,$$

$$v_0(x) = (x - 2)\left(2x^2 - 11x + 15\right)^2,$$

$$u_1(x) = (4x^2 - 20x + 24)\left(4x^2 - 20x + 24\right)^2,$$

$$v_1(x) = (x - 2.5)\left(4x^2 - 20x + 24\right)^2,$$

$$u_2(x) = (19 - 6x)\left(2x^2 - 9x + 10\right)^2,$$

$$v_2(x) = (x - 3)\left(2x^2 - 9x + 10\right)^2.$$

Using the Hermite interpolation formula, we construct the polynomial $H_5(x)$:

$$\begin{aligned} H_5(x) =& (6x - 11)\left(2x^2 - 11x + 15\right)^2 (0.69315) \\ &+ \left(4x^2 - 20x + 24\right)^2 (0.91629) \\ &+ (19 - 6x)\left(2x^2 - 9x + 10\right)^2 (1.09861) \\ &+ (x - 2)\left(2x^2 - 11x + 15\right)^2 (0.5) \\ &+ (x - 2.5)\left(4x^2 - 20x + 24\right)^2 (0.4) \\ &+ (x - 3)\left(2x^2 - 9x + 10\right)^2 (0.33333). \end{aligned}$$

Evaluating this polynomial at $x = 2.7$, we find:

$$\ln(2.7) \approx H_5(2.7) = 0.993252.$$

This value is accurate to six decimal places and demonstrates greater precision compared to the Lagrange interpolation approach.

2.5 Spline Interpolation

The term "spline" refers to a variety of functions used for interpolating and smoothing data.

Let's consider f as a real-valued function defined over the interval $[a, b]$. The interpolation data points are as shown in the table below.

For simplicity, assume $a = x_0 < x_1 < x_2 < \ldots < x_n = b$.

x	x_0	x_1	$\cdots$	x_n
$y = f(x)$	y_0	y_1	$\cdots$	y_n

Table 2.1: Data points for spline interpolation.

2.6 Definition

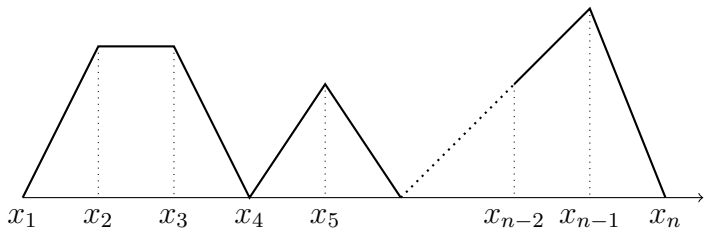
A function S is termed a spline of degree k if it fulfills the following conditions:

1. S is defined on the interval $[a, b]$.
2. The derivatives $S^{(r)}$ are continuous over (a, b) for $0 \leq r \leq k - 1$.
3. On each subinterval $[x_i, x_{i+1}]$, for $i = 0, 1, 2, \ldots, n - 1$, S is a polynomial of degree at most k.
4. The values of S at the knots match the values of a given function f such that $S(x_i) = f(x_i)$ for $x_i \in [a, b]$.

It is important to note that, unlike polynomial interpolation, the degree of splines remains constant regardless of the number of data points. Instead of a single polynomial of increasing degree, splines use multiple polynomials of a fixed degree.

2.7 Linear Spline Interpolating Function

An example of piecewise polynomial interpolation is piecewise linear interpolation. This method involves connecting a series of data points, as listed in Table 2.1, with straight line segments, as illustrated in Figure 1.



Suppose $f(x)$ is a real-valued function defined on an interval $[a, b]$. We aim to create a piecewise linear polynomial function $S(x)$ that interpolates $f(x)$ at the specified data points $a = x_0 < x_1 < x_2 < \ldots < x_n = b$.

By applying the equation of a straight line, the interpolating function $S(x)$ can be written as

$$\begin{aligned} S_i(x) &= f(x_i) + \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}(x - x_i), \quad i = 0, 1, 2, \ldots, n-1 \\ &= f(x_i) + f[x_{i+1}, x_i](x - x_i) \quad \text{on each subinterval } [x_i, x_{i+1}] \end{aligned} \tag{2.18}$$

Outside the interval $[a, b]$, the function $S(x)$ is typically defined as follows:

$$S(x) = \begin{cases} S_1(x), & \text{if } x < a \\ S_{n-1}(x), & \text{if } x > b \end{cases} \tag{2.19}$$

The points $x_2, x_3, \ldots, x_{n-1}$, where $S(x)$ transitions from one polynomial to another, are known as break points or knots. Since $S(x)$ is continuous on $[a, b]$, it is referred to as a spline of degree one.

Example 2.7.1. *Construct a first-degree interpolating polynomial for the following data points:*

x	*1*	*1.5*	*2*	*2.5*	*3*
$f(x)$	*1*	*3*	*7*	*10*	*15*

And use the resulting spline to approximate $f(2.2)$.

Solution: Using the linear interpolation formula, we have

$$S_1(x) = f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_0) = 1 + \frac{3-1}{1.5-1}(x-1) = 4x - 3,$$
$$S_2(x) = f(x_1) + \frac{f(x_2) - f(x_1)}{x_2 - x_1}(x - x_1) = 3 + \frac{7-3}{2-1.5}(x-1.5) = 8x - 9,$$
$$S_3(x) = f(x_2) + \frac{f(x_3) - f(x_2)}{x_3 - x_2}(x - x_2) = 7 + \frac{10-7}{2.5-2}(x-2) = 6x - 5,$$
$$S_4(x) = f(x_3) + \frac{f(x_4) - f(x_3)}{x_4 - x_3}(x - x_3) = 10 + \frac{15-10}{3-2.5}(x-2.5) = 10x - 15.$$

Therefore, the piecewise linear interpolating function $S(x)$ is:

$$S(x) = \begin{cases} 4x - 3 & \text{if } x \in [1, 1.5], \\ 8x - 9 & \text{if } x \in [1.5, 2], \\ 6x - 5 & \text{if } x \in [2, 2.5], \\ 10x - 15 & \text{if } x \in [2.5, 3]. \end{cases}$$

Since $x = 2.2$ is within the interval $[2, 2.5]$, we use $S_3(x)$ to approximate $f(2.2)$:

$$f(2.2) \approx S_3(2.2) = 6(2.2) - 5 = 13.2 - 5 = 8.2.$$

This demonstrates how a first-degree spline can be used to approximate a function at a specific point.

2.8 Theorem (First-degree Spline Accuracy)

Theorem 2.8.1. *Let f be a real-valued function that is twice differentiable and continuous on the interval $[a, b]$. Consider $P(x)$ as a first-degree spline that interpolates f at the points $a = x_1 < x_2 < \ldots < x_n = b$. The error bound can be expressed as:*

$$|f(x) - P(x)| \leq \frac{1}{8}Mh^2, \quad \textit{for } a \leq x \leq b,$$

where $h = \max_i(x_{i+1} - x_i)$ and M is the maximum value of $|f''(x)|$ over (a, b).

Proof. The error for Lagrange interpolation is given by:

$$f(x) - P(x) = \frac{1}{n!}f^{(n)}(\xi)\prod_{i=1}^{n}(x - x_i),$$

where n represents the number of interpolation points. For a first-degree spline over $[a, b]$, $n = 2$, thus:

$$f(x) - P(x) = \frac{1}{2} f''(\xi)(x - a)(x - b),$$

for some ξ within (a, b). Since $|f''(\xi)| \leq M$ on (a, b), and knowing that the maximum value of $|(x - a)(x - b)|$ on $[a, b]$ occurs at the midpoint, we get:

$$\max_{x \in [a,b]} |(x - a)(x - b)| = \frac{(b - a)^2}{4}.$$

Therefore, the error bound becomes:

$$|f(x) - P(x)| \leq \frac{1}{2} M \cdot \frac{(b - a)^2}{4} = \frac{1}{8} M h^2.$$

□

From the theorem, we understand that if the second derivative of a function is bounded, the interpolation error will diminish as h approaches zero. In polynomial interpolation with 10 data points, the error estimate considers the 10^{th} derivative.

Assuming the constant M is known, our goal is to determine the minimum number of equally spaced knots n such that the error bound for a first-degree spline is within a specified tolerance ε.

Solution: Given that $|f''(x)| \leq M$, we have $\frac{1}{8} M h^2 \leq \varepsilon$. Solving for h, we find $h \leq \sqrt{\frac{8\varepsilon}{M}}$. With $h = \frac{b-a}{n-1}$, it follows that:

$$(b - a)\sqrt{\frac{M}{8\varepsilon}} \leq n - 1.$$

Rearranging for n, we get:

$$n = 1 + \left\lceil (b - a)\sqrt{\frac{M}{8\varepsilon}} \right\rceil,$$

where $\lceil x \rceil$ is the ceiling function, representing the smallest integer greater than or equal to x.

Question 1. *Examine the function*

$$f(x) = \begin{cases} 2x - 1, & \textit{for } x \in [-1, 1], \\ -x + 2, & \textit{for } x \in [1, 2], \\ 5x, & \textit{for } x \in [2, 3]. \end{cases}$$

Is this function a first-degree spline?

Each segment of the function is a linear expression:

For $x \in [-1, 1]$, $f(x) = 2x - 1$ is linear.

For $x \in [1, 2]$, $f(x) = -x + 2$ is linear.

For $x \in [2, 3]$, $f(x) = 5x$ is linear.

Thus, the function is piecewise linear.

Check if the function is continuous at the knots The knots are at $x = 1$ and $x = 2$. We need to check if the function is continuous at these points.

At $x = 1$:

$$\lim_{x \to 1^-} f(x) = 2(1) - 1 = 1, \quad \lim_{x \to 1^+} f(x) = -(1) + 2 = 1$$

Since both limits are equal, the function is continuous at $x = 1$.

At $x = 2$:

$$\lim_{x \to 2^-} f(x) = -(2) + 2 = 0, \quad \lim_{x \to 2^+} f(x) = 5(2) = 10$$

Since the limits are not equal, the function is not continuous at $x = 2$.

Since the function is not continuous at $x = 2$, it does not satisfy the conditions for being a first-degree spline. Therefore, this function is not a first-degree spline.

2.9 Quadratic Spline Interpolation

While first-degree splines can be useful in many scenarios, they have a significant drawback: they exhibit a lack of smoothness due to abrupt changes in the slope at each knot. This issue arises from the discontinuities in the first derivative. On the other hand, quadratic splines provide a smoother alternative by maintaining continuity in the first derivatives at the knots, though they do not guarantee continuity in the second derivatives.

A quadratic spline function is generally expressed as:

$$f(x) = a_i x^2 + b_i x + c_i$$

The key features of a quadratic spline are:

1. The function $S(x_i) = f(x_i)$ is true for $i = 0, 1, 2, \ldots, n$.

2. In each interval $[x_{i-1}, x_i]$ for $1 \leq i \leq n$, $S(x)$ is a quadratic polynomial, except potentially in the first or last interval.

3. The function $S(x)$ and its derivative $S'(x)$ are continuous over the entire domain (a, b).

If $S''(x_i) = M_i$ exists, then within each interval $[x_{i-1}, x_i]$, The function $f(x)$ can be estimated using:

$$S(x) = S_i(x) = a_i x^2 + b_i x + c_i \text{ for } i = 1, 2, \ldots, n$$

To find the coefficients a_i, b_i, c_i for $i = 1, 2, \ldots, n$, there are $3n$ unknowns.

The continuity of $S(x)$ at the interior points $x_1, x_2, \ldots, x_{n-1}$

$$S_i(x_i) = f(x_i) = a_i x^2 + b_i x + c_i \tag{2.20}$$

$$S_{i+1}(x_i) = f(x_i) = a_{i+1} x_i^2 + b_{i+1} x_i + c_{i+1}, \quad \text{for } i = 1, 2, \ldots, n-1 \tag{2.21}$$

This results in $2n - 2$ equations. Continuity of $S'(x)$ at the internal nodes yields additional conditions:

$$S_i'(x_i) = S_{i+1}'(x_i), \quad \text{or} \quad 2a_i x_i + b_i = 2a_{i+1} x_i + b_{i+1}, \quad \text{for } i = 1, 2, \ldots, n-1 \tag{2.22}$$

These equations provide $n - 1$ additional conditions. Boundary conditions at the endpoints x_0 and x_n are given by:

$$f(x_0) = a_1 x_0^2 + b_1 x_0 + c_1 \text{ and } f(x_n) = a_n x_n^2 + b_n x_n + c_n \tag{2.23}$$

Combining all these constraints, we have $(2n-2) + (n-1) + 2 = 3n - 1$ equations for $3n$ unknowns. To solve for the unknowns uniquely, an additional equation is needed. This can be provided in various ways, such as by specifying $M_0 = f''(x_0) = S_1''(x_0)$.

For instance, if we set $f''(x_0) = 0$, it results in $a_1 = 0$. Consequently, the quadratic spline over the interval $[x_0, x_1]$ simplifies to a linear function, forming a straight line between the first two points. The system of linear equations, which is of size $(3n) \times (3n)$, can then be solved to find the coefficients a_i, b_i, and c_i for $i = 1, 2, \ldots, n$.

Rearranging the equations appropriately allows solving for each set of unknowns using 3×3 systems. The following example illustrates this process:

Example 2.9.1. *Consider three intervals:* $[x_0, x_1], [x_1, x_2], [x_2, x_3]$. *The equations derived from continuity and boundary conditions are:*

$$a_1x_1^2 + b_1x_1 + c_1 = f(x_1), \quad a_2x_1^2 + b_2x_1 + c_2 = f(x_1),$$
$$a_2x_2^2 + b_2x_2 + c_2 = f(x_2), \quad a_3x_2^2 + b_3x_2 + c_3 = f(x_2),$$
$$2a_1x_1 + b_1 = 2a_2x_1 + b_2, \quad 2a_2x_2 + b_2 = 2a_3x_2 + b_3,$$
$$a_1x_0^2 + b_1x_0 + c_1 = f(x_0), \quad a_3x_3^2 + b_3x_3 + c_3 = f(x_3)$$

Choosing $M_0 = f''(x_0) = 0$ simplifies to $a_1 = 0$. Substituting the given values into the equations, we solve for b_1 and c_1 first, then for the other coefficients.

Example 2.9.2. *Given the following set of points:*

x	*0*	*1*	*2*	*3*
$f(x)$	*1*	*2*	*33*	*244*

Fit quadratic splines with $M(0) = f''(0) = 0$ and estimate $f(2.5)$.

Solution 2.9.1. *The quadratic spline approximation is given by:*

$$S(x) = \begin{cases} S_1(x) = a_1x^2 + b_1x + c_1, & \textit{for } 0 \le x \le 1 \\ S_2(x) = a_2x^2 + b_2x + c_2, & \textit{for } 1 \le x \le 2 \\ S_3(x) = a_3x^2 + b_3x + c_3, & \textit{for } 2 \le x \le 3 \end{cases}$$

Since $M(0) = f''(0) = 0$, we have $a_1 = 0$. Solving the equations:

$$\begin{cases} b_1(0) + c_1 = 1 \\ b_1 + c_1 = 2 \end{cases}$$

$$\begin{cases} a_2 + b_2 + c_2 = 2 \\ 4a_2 + 2b_2 + c_2 = 33 \\ 2a_2 + b_2 = 1 \end{cases}$$

Solving these yields $a_2 = 10$, $b_2 = -19$, $c_2 = 11$. Similarly, solving for a_3, b_3, c_3 results in $a_3 = 20$, $b_3 = -59$, $c_3 = 31$.

The quadratic splines are:

$$S_1(x) = x + 1 \quad \textit{for} \quad 0 \le x \le 1$$
$$S_2(x) = 10x^2 - 19x + 11 \quad \textit{for} \quad 1 \le x \le 2$$
$$S_3(x) = 20x^2 - 59x + 31 \quad \textit{for} \quad 2 \le x \le 3$$

Thus, the estimated value at $x = 2.5$ is $f(2.5) = 86.25$.

2.10 Higher-Degree Spline Interpolation

Higher-degree spline interpolations, which are of degree two or higher, are employed when it is necessary to achieve smoothness and an accurate approximation of the function's shape.

To guarantee that the approximating spline has a continuous m^{th}-derivative, we select a spline of degree at least $(m+1)$. Suppose we have knots $x_1 < x_2 < \ldots < x_n$. We consider a piecewise polynomial of degree m, where each segment is joined at the knots to create a spline S that maintains m continuous derivatives. For each internal knot, the spline must satisfy the following conditions:

On the left side of t, the spline $S(x)$ is defined by the polynomial $p(x)$, while on the right side of t, it is defined by $q(x)$, with both p and q being polynomials of degree m.

The requirement for the m-th derivative $S^{(m)}$ to be continuous ensures that all lower-order derivatives $S^{(m-1)}, S^{(m-2)}, \ldots, S^{(1)}, S$ are also continuous.

Thus, for any k:

$$\lim_{x \to t^-} S^{(k)}(x) = \lim_{x \to t^+} S^{(k)}(x) \quad \text{for} \quad 0 \leq k \leq m$$

From this, it follows that:

$$\lim_{x \to t^-} p^{(k)}(x) = \lim_{x \to t^+} q^{(k)}(x) \quad \text{for} \quad 0 \leq k \leq m$$

In this context, $\lim_{x \to t^+}$ denotes the limit of x as it approaches t from the right (where $x - t$ is positive), while $\lim_{x \to t^-}$ represents the limit as x approaches t from the left (where $x - t$ is negative). Given that p and q are both polynomials, their derivatives of all orders are continuous. Consequently, we obtain:

$$p^{(k)}(t) = q^{(k)}(t) \quad \text{for } 0 \leq k \leq m$$

This requirement indicates that p and q must be identical polynomials. By utilizing Taylor's theorem:

$$p(x) = \sum_{k=0}^{m} \frac{1}{k!} p^{(k)}(t)(x-t)^k = \sum_{k=0}^{m} \frac{1}{k!} q^{(k)}(t)(x-t)^k = q(x)$$

This reasoning can be extended to each knot $x_2, x_3, \ldots, x_{n-1}$, showing that S essentially behaves as a single polynomial throughout the interval from x_1 to x_n. Consequently, to construct a spline function, one must use a piecewise polynomial of degree $m+1$ with m continuous

derivatives. This approach ensures that the spline behaves as a unified polynomial over the whole range. In general, ordinary polynomials are less effective for fitting curves.

To achieve the desired level of smoothness and specific shape characteristics in the approximating function, higher-degree splines are employed.

CHAPTER THREE

Cubic Spline Interpolation

Cubic spline interpolation is a prevalent method for constructing a piecewise polynomial approximation, employing cubic polynomials between consecutive nodes. Each cubic polynomial is defined by four constants, which facilitate the interpolant's continuous differentiability and ensure a smooth second derivative. The primary objective of cubic spline interpolation is to create an interpolation formula that ensures continuity in both the first and second derivatives, leading to smooth transitions between intervals and at the interpolation nodes. This smoothness ensures that the graph of $y = S(x)$ is devoid of sharp corners and has a well-defined radius of curvature at every point.

In the development of a cubic spline, it is not required for the interpolant's derivatives to coincide with those of the approximated function at the nodes. Given a function f defined over the interval $[x_0, x_n]$ with nodes $\{x_0, x_1, \ldots, x_n\}$ such that $x_0 = a$ and $x_n = b$, the cubic spline interpolation S for f adheres to the following conditions:

1. On each subinterval $[x_i, x_{i+1}]$ for $i = 0, 1, \ldots, N-1$, $S(x)$ is represented by a cubic polynomial, denoted as $S_i(x)$.

2. The cubic spline satisfies the condition $S(x_i) = f(x_i)$ for all $i = 0, 1, 2, \ldots, N$.

3. Continuity of the spline S is maintained such that $S_{i+1}(x_{i+1}) = S_i(x_{i+1})$ for all $i = 0, 1, 2, \ldots, N-2$.

4. The first derivative continuity condition $S'_{i+1}(x_{i+1}) = S'_i(x_{i+1})$ is satisfied for all $i = 0, 1, 2, \ldots, N-2$.

5. The second derivative continuity condition $S''_{i+1}(x_{i+1}) = S''_i(x_{i+1})$ is ensured for all $i = 0, 1, 2, \ldots, N-2$.

6. The spline must satisfy one of the following boundary conditions:

   - For natural boundary conditions, $S''(x_0) = 0$ and $S''(x_N) = 0$.
   - For clamped boundary conditions, $S'(x_0) = f'(x_0)$ and $S'(x_N) = f'(x_N)$.

Remark 3.0.1. *To construct the cubic spline interpolant S for the function f, consider the sequence of points $a = x_0 < x_1 < \ldots < x_N = b$. On each subinterval $[x_i, x_{i+1}]$, the cubic spline S is defined by:*

$$S(x) = S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3, \quad \textit{for } x_i \leq x \leq x_{i+1}.$$

Definition 3.0.1. *A cubic spline $S(x)$ is a piecewise-defined function that meets the following criteria:*

1. *On each subinterval $[x_i, x_{i+1}]$ for $i = 0, 1, \ldots, N-1$, $S(x) = S_i(x)$ is expressed as a cubic polynomial.*
2. *For $i = 0, 1, \ldots, N$, $S(x_i) = u_i$ (where S matches the specified data points).*
3. *The functions $S(x)$, $S'(x)$, and $S''(x)$ are continuous over the interval $[a, b]$, ensuring that S is smooth.*

Thus, the N cubic polynomial segments are represented as:

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3, \quad \text{for } i = 0, 1, \ldots, N-1$$

where a_i, b_i, c_i, and d_i denote the four unknown coefficients for each cubic polynomial.
The first and second derivatives of the cubic spline are given by:

$$\begin{aligned} S'(x) &= b_i + 2c_i(x - x_i) + 3d_i(x - x_i)^2 \\ S''(x) &= 2c_i + 6d_i(x - x_i) \end{aligned} \tag{3.1}$$

Given that $S(x_i) = u_i$ for $i = 1, 2, \ldots, N-1$ and x_i lies within the interval $[x_i, x_{i+1}]$, it follows that:

$$u_i = a_i \quad \text{for each } i = 1, 2, \ldots, N-1 \tag{3.2}$$

Considering the continuity constraints of cubic splines across the intervals, we obtain:

$$\begin{aligned} S(x_i) &= S_i(x_i) \\ &= S_{i-1}(x_i) \quad \text{for } i = 2, 3, \ldots, N \end{aligned} \tag{3.3}$$

From the above, $S_i(x_i) = a_i$, and

$$\begin{aligned} S_{i-1}(x_i) &= a_{i-1} + b_{i-1}(x_i - x_{i-1}) + c_{i-1}(x_i - x_{i-1})^2 + d_{i-1}(x_i - x_{i-1})^3 \\ a_i &= a_{i-1} + b_{i-1}h + c_{i-1}h^2 + d_{i-1}h^3 \quad \text{for } i = 2, 3, \ldots, N-1 \text{ and } h = x_i - x_{i-1} \end{aligned} \tag{3.4}$$

To guarantee continuity and smoothness at the data points, the derivatives of the cubic splines must match across adjacent intervals:

$$\begin{aligned} S_i'(x_i) &= S_{i-1}'(x_i) \\ b_i &= b_{i-1} + 2c_{i-1}h + 3d_{i-1}h^2 \quad \text{for } i = 1, 2, \ldots, N-1 \end{aligned} \tag{3.5}$$

Using the derivative expression from the earlier equation, we have:

$$S_i''(x) = 6d_i(x - x_i) + 2c_i$$

$$S_i''(x_i) = 2c_i \quad \text{for } i = 2, 3, \ldots, N-2 \tag{3.6}$$

Finally, to ensure that the second derivative of S_i is continuous across the interval, we have:

$$\begin{aligned} S_i''(x_{i+1}) &= 6d_i(x_{i+1} - x_i) + 2c_i \\ S_{i+1}''(x_{i+1}) &= 6d_i(x_{i+1} - x_i) + 2c_i \end{aligned} \tag{3.7}$$

To simplify, let $h = x_{i+1} - x_i$. Using the above equations, we obtain:

$$\begin{aligned} &\text{From } S_{i+1}''(x_{i+1}) = 6d_ih + 2c_i \text{ and} \\ &2c_{i+1} = 6d_ih + 2c_i \end{aligned} \tag{3.8}$$

The equation can be simplified by using M_i to denote $S_i''(x_i)$ and expressing the coefficients in terms of M_i and u_i. Specifically:

$$S_i''(x_i) = 2c_i \Rightarrow M_i = 2c_i \Rightarrow c_i = \frac{M_i}{2} \tag{3.9}$$

Similarly, using equation (3.8), d_i can be expressed as:

$$\begin{aligned} &2c_{i+1} = 6d_ih + 2c_i \\ &\Rightarrow 6d_ih = 2c_{i+1} - 2c_i \\ &\Rightarrow d_i = \frac{2c_{i+1} - 2c_i}{6h} = \frac{2\left(\frac{M_{i+1}}{2}\right) - 2\left(\frac{M_i}{2}\right)}{6h} \\ &\Rightarrow d_i = \frac{M_{i+1} - M_i}{6h} \end{aligned} \tag{3.10}$$

From equation (3.4), b_i can be written as:

$$\begin{aligned} & a_{i+1} = a_i + b_i h + c_i h^2 + d_i h^3 \\ & \Rightarrow b_i h = -a_i - c_i h^2 - d_i h^3 + a_{i+1} \\ & \Rightarrow b_i = \frac{-a_i - c_i h^2 - d_i h^3 + a_{i+1}}{h} \\ & \Rightarrow b_i = \frac{u_{i+1} - u_i}{h} - \frac{h}{6}\left(M_{i+1} + 2M_i\right) \end{aligned} \tag{3.11}$$

We can determine the coefficients for $N-1$ intervals using the following equations:

$$\begin{aligned} & a_i = u_i, \\ & b_i = \frac{u_{i+1} - u_i}{h} - \frac{h}{6}\left(M_{i+1} + 2M_i\right), \\ & c_i = \frac{M_i}{2}, \\ & d_i = \frac{M_{i+1} - M_i}{6h} \end{aligned} \tag{3.12}$$

These equations can be conveniently represented in matrix form as follows:

$$\begin{aligned} & \text{According to the relationship, } b_{i+1} = b_i + 2c_i h + 3d_i h^2 \text{ for } i = 1, 2, \ldots, N-1 \\ & \text{Thus, } 3d_i h^2 + 2c_i h = b_{i+1} - b_i \end{aligned} \tag{3.13}$$

When we substitute the values from equation (3.12) into (3.13), we have

$$\begin{aligned} & 3\left(\frac{M_{i+1} - M_i}{6h}\right) h^2 + 2\left(\frac{M_i}{2}\right) h + \frac{u_{i+1} - u_i}{h} - \left(\frac{M_{i+1} + 2M_i}{6}\right) h \\ & = \frac{u_{i+2} - u_{i+1}}{h} - \left(\frac{M_{i+2} + 2M_{i+1}}{6}\right) h \\ & 3\left(\frac{M_{i+1} - M_i}{6h}\right) h^2 + 2\left(\frac{M_i}{2}\right) h + \frac{u_{i+1} - u_i}{h} - \left(\frac{M_{i+1} + 2M_i}{6}\right) h + \left(\frac{M_{i+2} + 2M_{i+1}}{6}\right) h \\ & = \frac{u_{i+2} - u_{i+1}}{h} - \frac{u_{i+1} - u_i}{h} \\ & h\left(\frac{M_{i+1} - M_i}{2}\right) + \frac{M_i}{3} - \frac{M_{i+1} + 2M_i}{6} + \frac{M_{i+2} + 2M_{i+1}}{6} \\ & = \frac{u_i - 2u_{i+1} + u_{i+2}}{h} \\ & \frac{h}{6}\left(M_i + 4M_{i+1} + M_{i+2}\right) = \frac{u_i - 2u_{i+1} + u_{i+2}}{h} \\ & M_i + 4M_{i+1} + M_{i+2} = \frac{6}{h^2}\left(u_i - 2u_{i+1} + u_{i+2}\right) \quad \text{for } i = 1, 2, \ldots, N-1 \end{aligned} \tag{3.14}$$

This leads to the matrix equation:

$$\begin{bmatrix} 1 & 4 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & 4 & 1 & \cdots & 0 & 0 & 0 \\ 0 & 0 & 1 & 4 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 4 & 1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 1 & 4 & 1 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 & 4 \end{bmatrix} \begin{bmatrix} M_1 \\ M_2 \\ M_3 \\ \vdots \\ M_{N-3} \\ M_{N-2} \\ M_{N-1} \\ M_N \end{bmatrix} \tag{3.15}$$

$$= \frac{6}{h^2} \begin{bmatrix} u_1 - 2u_2 + u_3 \\ u_2 - 2u_3 + u_4 \\ u_3 - 2u_4 + u_5 \\ \vdots \\ u_{N-3} - 2u_{N-2} + u_{N-1} \\ u_{N-2} - 2u_{N-1} + u_N \end{bmatrix} \quad \text{for } i = 1, 2, \ldots, N-1$$

This matrix equation provides a systematic approach to solve for the second derivatives M_i, crucial for cubic spline interpolation.

3.1 Four Types of Splines

3.1.1 Clamped Spline

Theorem 3.1.1. *Given a function f defined on $a = x_0 < x_1 < \ldots < x_n = b$ and differentiable at both endpoints a and b, there exists a unique cubic spline interpolation s that satisfies the clamped boundary conditions $s'(a) = f'(a)$ and $s'(b) = f'(b)$ at the nodes $x_0, x_1, \ldots, x_n$.*

3.1.2 Natural Spline

In the natural spline approach, the boundary conditions are set such that the second derivatives at the endpoints are zero:

$$M_0 = M_n = 0 \tag{3.16}$$

This condition ensures that the spline behaves like a straight line beyond the boundary points. Consequently, the matrix used for computing the second derivatives M_0 through M_n should be modified to incorporate these boundary conditions.

$$\begin{pmatrix} 1 & 0 & 0 & 0 & \cdots & \cdots & \cdots & 0 & 0 & 0 & 0 \\ 0 & 1 & 4 & 1 & \cdots & \cdots & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 4 & \cdots & \cdots & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \cdots & \cdots & 4 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & \cdots & \cdots & 1 & 4 & 1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & \cdots & \cdots & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} M_1 \\ M_2 \\ M_3 \\ \vdots \\ \vdots \\ \vdots \\ M_{n-2} \\ M_{n-1} \\ M_n \end{pmatrix} = \frac{6}{h^2} \begin{pmatrix} y_1 - 2y_2 + y_3 \\ y_2 - 2y_3 + y_4 \\ y_3 - 2y_4 + y_5 \\ \vdots \\ \vdots \\ \vdots \\ y_{n-4} - 2y_{n-3} + y_{n-2} \\ y_{n-3} - 2y_{n-2} + y_{n-1} \\ y_{n-2} - 2y_{n-1} + y_n \end{pmatrix} \tag{3.17}$$

For simplicity, the columns corresponding to M_1 and M_n can be removed from the matrix, as these values are both zero.

$$\begin{pmatrix} 4 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 4 & 1 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 4 & 1 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 4 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 & 4 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 1 & 4 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 & 4 & 1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 1 & 4 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 1 & 4 \end{pmatrix} \begin{pmatrix} M_2 \\ M_3 \\ M_4 \\ \vdots \\ M_{n-3} \\ M_{n-2} \\ M_{n-1} \end{pmatrix} = \frac{6}{h^2} \begin{pmatrix} y_1 - 2y_2 + y_3 \\ y_2 - 2y_3 + y_4 \\ y_3 - 2y_4 + y_5 \\ \vdots \\ y_{n-4} - 2y_{n-3} + y_{n-2} \\ y_{n-3} - 2y_{n-2} + y_{n-1} \\ y_{n-2} - 2y_{n-1} + y_n \end{pmatrix} \tag{3.18}$$

This process leads to an $(n-2) \times (n-2)$ matrix, which is used to find the values of M_2 through M_{n-1}. Once these values are determined, the spline interpolation is uniquely defined.

3.2 Parabolic Runout Spline

The parabolic runout spline requires that the second derivatives at the boundary points M_1 and M_n match the values of M_2 and M_{n-1} respectively:

$$M_1 = M_2 \quad \text{and} \quad M_{n-1} = M_n \tag{3.19}$$

This condition ensures that the spline transitions into a parabolic shape at the boundary points. Parabolic runout splines are especially effective for modeling periodic and exponential datasets.

The matrix equation governing this type of spline is:

$$\begin{pmatrix} 5 & 1 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 1 & 4 & 1 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 1 & 4 & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \\ 0 & 0 & 0 & \cdots & 4 & 1 & 0 & \\ 0 & 0 & 0 & \cdots & 1 & 4 & 1 & \\ 0 & 0 & 0 & \cdots & 0 & 1 & 5 & \end{pmatrix} \begin{pmatrix} M_2 \\ M_3 \\ M_4 \\ \vdots \\ M_{n-3} \\ M_{n-2} \\ M_{n-1} \end{pmatrix} = \frac{6}{h^2} \begin{pmatrix} y_1 - 2y_2 + y_3 \\ y_2 - 2y_3 + y_4 \\ y_3 - 2y_4 + y_5 \\ \vdots \\ y_{n-4} - 2y_{n-3} + y_{n-2} \\ y_{n-3} - 2y_{n-2} + y_{n-1} \\ y_{n-2} - 2y_{n-1} + y_n \end{pmatrix} \tag{3.20}$$

We can now compute the values for M_2 through M_{n-1}, given that the values for M_1 and M_n have already been established.

3.3 Cubic Runout Spline

The cubic runout spline features a unique endpoint condition where $M_1 = 2M_2 - M_3$ and $M_n = 2M_{n-1} - M_{n-2}$. This specification leads to the spline being represented by a single cubic polynomial over the final two intervals, as opposed to handling them as separate cubic functions.

The matrix equation for this type is

$$\begin{pmatrix} 6 & 0 & 0 & 0 & . & . & . & 0 & 0 & 0 & 0 \\ 1 & 4 & 1 & 0 & . & . & . & 0 & 0 & 0 & 0 \\ 0 & 1 & 4 & 0 & . & . & . & 0 & 0 & 0 & 0 \\ . & . & . & . & & & & . & . & . & . \\ . & . & . & . & & & & . & . & . & . \\ 0 & 0 & 0 & 0 & . & . & . & . & 4 & 1 & 0 \\ 0 & 0 & 0 & 0 & . & . & . & . & 4 & 4 & 1 \\ 0 & 0 & 0 & 0 & . & . & . & . & 0 & 0 & 6 \\ & & & & & & & & & & \end{pmatrix} \begin{pmatrix} M_2 \\ M_3 \\ M_4 \\ . \\ . \\ . \\ M_{n-3} \\ M_{n-2} \\ M_{n-1} \end{pmatrix} = \frac{6}{h^2} \begin{pmatrix} y_1 - 2y_2 + y_3 \\ y_2 - 2y_3 + y_4 \\ y_3 - 2y_4 + y_5 \\ . \\ . \\ . \\ y_{n-4} - 2y_{n-3} + y_{n-2} \\ y_{n-3} - 2y_{n-2} + y_{n-1} \\ y_{n-2} - 2y_{n-1} + y_n \end{pmatrix} \tag{3.21}$$

3.4 Error in the Cubic Spline and Its Derivatives

Evaluating the error associated with cubic splines and their derivatives is essential for accurate practical applications.

The natural cubic spline is known for offering a reliable approximation of a smooth function and its derivatives, as supported by the subsequent theorem:

Theorem: Let $y \in C^2[a, b]$ with $a = x_0 < x_1 < \ldots < x_n = b$, and let $s(x)$ be the natural cubic spline satisfying $s(x_i) = y_i$ for $i = 0, 1, \ldots, n$. Then:

$$\max_{x_0 \le x \le x_n} |y(x) - s(x)| \le \frac{Mh^2}{2}$$

where $h = x_{i+1} - x_i$ for $i = 0, 1, \ldots, n$ and $M = \max |y''(x)|$ over the interval $[x_0, x_n]$.

As the interval width h decreases, the cubic spline approximation improves, unlike Lagrange interpolation which can suffer from instability with increased degrees. The error in spline derivatives can be analyzed using operator notation.

For the first derivative error estimation, we use:

$$m_{i-1} + 4m_i + m_{i+1} = \frac{3}{h}(y_{i+1} - y_{i-1}) \quad \text{(based on the Cubic Hermite Spline method)}$$

which can be expressed as:

$$\left(E^{-1} + 4 + E\right) s'(x_i) = \frac{3}{h}\left(E - E^{-1}\right) y_i$$

Here, $E = e^{hD}$ where $D = \frac{d}{dx}$. Thus, equation (3.22) becomes:

$$\left(e^{-hD} + 4 + e^{hD}\right) s'(x_i) = \frac{3}{h}\left(e^{hD} - e^{-hD}\right) y_i$$

Now, $e^{hD} = 1 + hD + \frac{h^2D^2}{2!} + \frac{h^3D^3}{3!} + \ldots$ and $e^{-hD} = 1 - hD + \frac{h^2D^2}{2!} - \frac{h^3D^3}{3!} + \ldots$. Therefore, $e^{-hD} + e^{hD} = 2\left(1 + \frac{h^2D^2}{2!} + \frac{h^4D^4}{24} + \frac{h^6D^6}{720} + \ldots\right)$ and $e^{hD} - e^{-hD} = 2\left(hD + \frac{h^3D^3}{6} + \frac{h^5D^5}{120} + \ldots\right)$.

Substituting these expansions into (3.23), we obtain:

$$\left[2\left(1 + \frac{h^2D^2}{2!} + \frac{h^4D^4}{24} + \frac{h^6D^6}{720} + \ldots\right) + 4\right] s'(x_i) = 6\left(D + \frac{h^2D^3}{6} + \frac{h^4D^5}{120} + \ldots\right) y_i$$

$$s'(x) = \left(D - \frac{1}{180}h^4D^5 + \ldots\right) y_i \quad \text{Hence} \quad s'(x_i) = y_i' - \frac{1}{180}h^4 y_i^{(5)} + O(h^6)$$

Similarly, we can derive the relationships

$$\begin{aligned} y''(x_i) &= s''(x_i) + \tfrac{1}{12}h^2 y''(x_i) + O(h^4) \\ y'''(x_i) &= \tfrac{1}{2}\left[s'''(x_i+) + s'''(x_i-)\right] + O(h^2) \\ y''(x_i) &= \tfrac{1}{2}\left[s'''(x_i+) - s'''(x_i-)\right] + O(h^4) \end{aligned}$$

3.5 Cubic Spline Method for Two-Point Boundary Value Problems

3.5.1 Description of the Method

Consider a uniform mesh Δ with nodal points x_i on the interval $[a, b]$, defined as follows:

$$\Delta : a = x_0 < x_1 < x_2 < \ldots < x_{N-1} < x_N = b$$

where each x_i is given by $x_i = a + ih$ for $i = 0, 1, \ldots, N$, and h represents the mesh spacing, calculated as $h = \frac{b-a}{N}$.

Let $S_\Delta(x)$ be a non-polynomial function of class $C^2[a, b]$ that interpolates the function $u(x)$ at the mesh points x_i for $i = 0, 1, \ldots, N$. This function $S_\Delta(x)$ is parameterized by τ and is designed to converge to an ordinary cubic spline on the interval $[a, b]$ as τ approaches zero.

The spline function under consideration is expressed as

$$T_3 = \text{Span}\{1, x, \cos(\tau x), \sin(\tau x)\},$$

where τ represents the frequency parameter associated with the trigonometric components of the spline. This parameter τ can be either a real number or a purely imaginary number, which

enhances the method's accuracy. Consequently, within each subinterval $x_i \leq x \leq x_{i+1}$, the function takes the form:

$\text{Span}\{1, x, \cos\tau x, \sin\tau x\}$ or

$\text{Span}\{1, x, x^2, x^3\}$.(when $\tau \to 0$)

This becomes apparent when examining the relationship between polynomial and non-polynomial spline basis functions in the following way:

The spline function T_3 is given by:

$$T_3 = \text{Span}\{1, x, \cos(\tau x), \sin(\tau x)\}$$

which can also be expressed as:

$$\text{Span}\left\{1, x, \frac{2}{\tau^2}(1 - \cos(\tau x)), \frac{6}{\tau^3}(\tau x - \sin(\tau x))\right\}.$$

It follows that as τ approaches zero, T_3 converges to:

$$\lim_{\tau \to 0} T_3 = \{1, x, x^2, x^3\}.$$

For each interval $[x_i, x_{i+1}]$, where $i = 0, 1, \ldots, N-1$, the non-polynomial spline $s_\Delta(x)$ takes the following form:

$$s_\Delta(x) = a_i + b_i(x - x_i) + c_i \sin(\tau(x - x_i)) + d_i \cos(\tau(x - x_i)), \tag{3.22}$$

with a_i, b_i, c_i, and d_i representing the coefficients specific to each segment.

Let u_i denote an approximation to $u(x_i)$, obtained using the spline function $s_\Delta(x)$ that interpolates through the points (x_i, u_i) and (x_{i+1}, u_{i+1}). In order to determine the coefficients specified in equation (3.22), it is necessary not only for $s_\Delta(x)$ to satisfy the interpolation conditions at x_i and x_{i+1}, but also for the first derivative of $s_\Delta(x)$ to be continuous at the points (x_i, u_i).

To derive an expression for the coefficients of (3.22) in terms of u_i, u_{i+1}, M_i and M_{i+1}, we first denote:

$$\begin{aligned} S_\Delta(x_i) &= u_i, \quad S_\Delta(x_{i+1}) = u_{i+1}, \\ S''_\Delta(x_i) &= M_i, \quad S''_\Delta(x_{i+1}) = M_{i+1}. \end{aligned} \tag{3.23}$$

By performing algebraic manipulation, we obtain the following expression:

The set of equations is given by:

$$\begin{aligned} a_i &= u_i + \tfrac{M_i}{\tau^2}, & b_i &= \tfrac{u_{i+1}-u_i}{h} + \tfrac{M_{i+1}-M_i}{\tau\theta} \\ c_i &= \tfrac{M_i\cos\theta - M_{i+1}}{\tau^2\sin\theta}, & d_i &= -\tfrac{M_i}{\tau^2} \end{aligned} \tag{3.24}$$

where $\theta = \tau h$ and $i = 0, 1, \ldots, N-1$.

Based on the continuity of the first derivative at (x_i, u_i), as described in [3.26], we have:

$$S'_{\Delta_{i-1}}(x_i) = S'_{\Delta_i}(x_i) \tag{3.25}$$

The following relation is obtained for the coefficients α and β, and the second derivatives M_{i+1}, M_i, and M_{i-1}:

$$\alpha M_{i+1} + 2\beta M_i + \alpha M_{i-1} = \frac{1}{h^2}\left(u_{i+1} - 2u_i + u_{i-1}\right) \tag{3.26}$$

where α and β are defined as:

$$\alpha = \frac{1}{h^2}(\theta\csc\theta - 1), \quad \beta = \frac{1}{h^2}(1 - \theta\cot\theta), \tag{3.27}$$

and $\theta = \tau h$. As $\tau \to 0$, α approaches $\frac{1}{6}$ and β approaches $\frac{1}{3}$. Consequently, equation (3.28) simplifies to the consistency relation for cubic splines:

$$M_{i+1} + 4M_i + M_{i-1} = \frac{6}{h^2}\left(u_{i+1} - 2u_i + u_{i-1}\right). \tag{3.28}$$

Initially, we express equation (1.1) in the following manner:

$$\begin{aligned} &\frac{d^2u}{dx^2} + q(x)\frac{du}{dx} + r(x)u = f(x), \quad \text{for } a < x < b, \\ &u(a) = u(b) = 0. \end{aligned} \tag{3.29}$$

At the grid points x_i, the differential equation given by (3.29) can be discretized as follows:

$$\begin{aligned} u_i'' + q_i u_i' + r_i u_i &= f_i, \\ &\text{for } i = 1, 2, \ldots, n \end{aligned} \tag{3.30}$$

where $q_i = q(x_i)$, $r_i = r(x_i)$, and $f_i = f(x_i)$.

By applying the spline method to equation (3.30), we derive:

$$M_i + q_i u_i' + r_i u_i = f_i \tag{3.31}$$

Following reference [3.24], we approximate the first derivatives of u as:

$$u_i' = \frac{u_{i+1} - u_{i-1}}{2h} \tag{3.32}$$

$$u_{i+1}' = \frac{3u_{i+1} - 4u_i + u_{i-1}}{2h} \tag{3.33}$$

$$u_{i-1}' = \frac{-u_{i+1} + 4u_i - 3u_{i-1}}{2h} \tag{3.34}$$

By substituting equations (3.31), (3.32), (3.33), and (3.34) into equation (3.28) and simplifying, we obtain a tridiagonal system. This system provides approximations $u_1, u_2, \ldots, u_{N-1}$ for the solution $u(x)$ at the points $x_1, x_2, \ldots, x_{N-1}$:

$$\begin{aligned}
&\left(\frac{3}{2}\alpha h q_{i-1} + \beta h q_i - \frac{1}{2}\alpha h q_{i+1} - h^2 \alpha r_{i-1} - 1\right) u_{i-1} \\
&\quad + \left(-2\alpha h q_{i-1} + 2\alpha h q_{i+1} - h^2 2\beta r_i + 2\right) u_i \\
&\quad + \left(\frac{1}{2}\alpha h q_{i-1} - \beta h q_i - \frac{3}{2}\alpha h q_{i+1} - h^2 \alpha r_{i+1} - 1\right) u_{i+1} \\
&= \mathrm{h}^2 \left(\alpha f_{i-1} + 2\beta f_i + \alpha f_{i+1}\right)
\end{aligned} \tag{3.35}$$

With the boundary conditions $u(a) = 0$ and $u(b) = 0$ applied for $i = 1, 2, \ldots, N-1$,

Solve the second-order differential equation:

$$\frac{d^2y}{dx^2} + xy = \left(3 - x - x^2 + x^3\right)\sin(x) + 4\cos(x), \quad x \in [a, b]$$

3.6 Convergence Analysis

The tridiagonal linear system given by equation (3.35) can be expressed in matrix form as follows:

$$AU + h^2 DF = G \tag{3.36}$$

where A is a tridiagonal matrix of order $N-1$ that is diagonally dominant, given by

$$A = P + hBQ - h^2 BR.$$

In this context, $P = (p_{ij})$ denotes a tridiagonal matrix with the elements

The elements of $P = (p_{ij})$ are defined as follows:

$$p_{ij} = \begin{cases} 2, & \text{for } i = j \text{ and } i = 1, 2, \ldots, N-1, \\ -1, & \text{for } |i-j| = 1, \\ 0, & \text{otherwise.} \end{cases}$$

and $BQ = (z_{ij})$, $BR = (R_{ij})$ are tridiagonal matrices defined by

$$z_{ij} = \begin{cases} 2\alpha(-q_0 + q_2), & \text{if } i = j = 1, \\ \frac{3}{2}\alpha q_{i-1} + \beta q_i - \frac{1}{2}\alpha q_{i+1}, & \text{if } i > j, \\ 2\alpha(-q_{i-1} + q_{i+1}), & \text{if } i = j, \\ \frac{1}{2}\alpha q_{i-1} - \beta q_i - \frac{3}{2}\alpha q_{i+1}, & \text{if } i < j, \\ 2\alpha(-q_{N-2} + q_N), & \text{if } i = j = N-1, \end{cases}$$

$$R_{ij} = \begin{cases} 2\beta r_i, & \text{if } i = j = 1, 2, 3, \ldots, N-1, \\ \alpha r_{i-1}, & \text{if } i > j, \\ \alpha r_{i+1}, & \text{if } i < j, \end{cases}$$

and

The vectors F and U are defined as:

$$F = \begin{pmatrix} f_1 & f_2 & \cdots & f_{N-1} \end{pmatrix},$$
$$U = \begin{pmatrix} u_1 & u_2 & \cdots & u_{N-1} \end{pmatrix}.$$

The tridiagonal matrix D is specified by

$$D = \begin{bmatrix} 2\beta & \alpha & 0 & 0 & \cdots & 0 \\ \alpha & 2\beta & \alpha & 0 & \cdots & 0 \\ 0 & \alpha & 2\beta & \alpha & \cdots & 0 \\ 0 & 0 & \alpha & 2\beta & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \ddots & \alpha \\ 0 & 0 & 0 & \cdots & \alpha & 2\beta & \alpha \\ 0 & 0 & 0 & \cdots & 0 & \alpha & 2\beta \end{bmatrix}$$

and

$G = (g_1, 0, \ldots, 0, g_{N-1})^T$,

with

$g_1 = -h^2\alpha f_0$

$g_i = 0, \quad i = 2, 3, \cdots N-2$

$g_{N-1} = -h^2\alpha f_N.$

We assume that

$\bar{U} = \left(u\left(x_1\right), u\left(x_2\right), \ldots, u\left(x_{N-1}\right)\right)^T$

The exact solution of the given boundary value problem (1.1) at nodal points $X_i, i = 0, 1, \ldots, N-1$, is:

$$A\bar{U} + h^2 DF = T(h) + G \tag{3.37}$$

where $T = (T(x_1), T(x_2), \ldots, T(x_{N-1}))^T$ denotes the truncation error vector. By expanding equation (3.35) in a Taylor series around x_i and using equation (3.30), we derive the following local truncation error:

$$\begin{aligned} T_i(h) =& [-1 + 2(\alpha+\beta)]h^2 u''(\xi_i) \\ &+ \frac{1}{3}(\alpha q_{i+1} - \beta q_i + \alpha q_{i-1})h^4 u'''(\xi_i) \\ &+ \frac{1}{12}\left[(-1+12\alpha) + \alpha h(q_{i+1} - q_{i-1})\right] h^4 u^{(4)}(\xi_i) + O(h^5), \end{aligned} \tag{3.38}$$

where $x_{i-1} < \xi_i < x_{i+1}$.

From equations (3.36) and (3.37), we obtain:

$$A(\bar{U} - U) = AE = T(h),$$

where the error vector E is expressed as

$$E = \bar{U} - U = \begin{pmatrix} e_1 & e_2 & \cdots & e_{N-1} \end{pmatrix}^T.$$

The primary objective is to establish a bound for $\|E\|$. To achieve this, we require the following lemma.

Lemma 3.1: Let W be an $N \times N$ matrix such that $\|W\| < 1$. Then, the matrix $(I+W)^{-1}$ exists, and the norm of its inverse satisfies

$$\left\|(I+W)^{-1}\right\| < \frac{1}{1 - \|W\|}.$$

From the above equation , it follows that:

$$
\begin{aligned}
E &= A^{-1}T \\
&= [P + hBQ]^{-1} T \\
&= \left[I + hP^{-1}BQ\right]^{-1} P^{-1}T.
\end{aligned}
$$

which implies

$$
\|E\| \leq \left\| \left[I + hP^{-1}BQ\right]^{-1} \right\| \left\|P^{-1}\right\| \|T\|. \tag{3.39}
$$

$$
\|E\| \leq \frac{\|P^{-1}\| \|T\|}{1 - h \|P^{-1}\| \|BQ\|}.
$$

Provided that $h \|P^{-1}\| \|BQ\| \leq 1$,

Following [3.23] we have

$$
\left\|P^{-1}\right\| \leq \frac{(b-a)^2}{8h^2}. \tag{3.40}
$$

and then we have

$$
\|BQ\| \leq q(8\alpha + 2\beta), \tag{3.41}
$$

where q is defined as the maximum of $|q(x_i)|$ for $a < x_i < b$.

At present $\alpha+\beta = \frac{1}{2}$ and $\alpha \neq \frac{1}{12}$, we have $\|T\| \leq \eta_1 h^4 M_4$, where $M_4 = \max_{a\leq\xi\leq b} \left|u^{(4)}(\xi)\right|$, then we have

$$
\|E\| \leq \frac{\eta_1 (b-a)^2 h^2 M_4}{1 - \omega q} \equiv O(h^2). \tag{3.42}
$$

However, with the parameters chosen as $\alpha = \frac{1}{12}$ and $\beta = \frac{5}{12}$, we obtain:

$$
\begin{aligned}
T_i(h) =& \frac{1}{360}(\alpha q_{i+1} - \beta q_i + \alpha q_{i-1}) h^4 u'''(\xi_i) \\
&+ \frac{1}{144}(q_{i+1} - q_{i-1}) h^4 u^{(4)}(\xi_i) + O(h^5),
\end{aligned} \tag{3.43}
$$

where $x_{i-1} < \xi_i < x_{i+1}$.

Thus, under these conditions, the method achieves optimal second-order accuracy.

Remark 3.1. When $\alpha + \beta = \frac{1}{2}$, the method attains second-order accuracy.

Remark 3.2. Specifically, with $\alpha = \frac{1}{12}$ and $\beta = \frac{5}{12}$, the method reaches optimal performance for second-order accuracy.

3.7 Illustrations

To demonstrate the effectiveness and convergence of our method, we solve two second-order boundary value problems with known exact solutions. We present the maximum absolute errors at the nodal points, $\max |u(x_i) - u_i|$, in Tables 3.1 and 3.2 for various parameter values. Furthermore, we compare the performance of our method with other techniques, including finite difference, finite element, B-spline, and finite volume methods, by evaluating the maximum absolute error norm, as shown in Table 3.2.

Problem 1: Examine the boundary value problem given by:

$$u''(x) - u'(x) = -(e^{x-1} + 1), \quad 0 < x < 1,$$

with boundary conditions $u(0) = u(1) = 0$. The exact solution to this problem is $u(x) = x(1 - e^{x-1})$.

Table 3.1 Maximum absolute errors in the solution of Problem 1.

Methods h	$\frac{1}{8}$	$\frac{1}{16}$	$\frac{1}{32}$
cubic spline ($\alpha = \frac{1}{4}, \beta = \frac{1}{4}$)	5.50×10^{-5}	1.37×10^{-5}	3.44×10^{-6}
cubic spline ($\alpha = \frac{1}{6}, \beta = \frac{1}{3}$)	4.51×10^{-5}	1.12×10^{-5}	2.82×10^{-6}
cubic spline ($\alpha = \frac{1}{14}, \beta = \frac{3}{7}$)	3.38×10^{-5}	8.44×10^{-6}	2.11×10^{-6}
cubic spline ($\alpha = \frac{1}{18}, \beta = \frac{4}{9}$)	3.19×10^{-5}	7.96×10^{-6}	1.99×10^{-6}
cubic spline ($\alpha = \frac{1}{24}, \beta = \frac{11}{24}$)	3.02×10^{-5}	7.55×10^{-6}	1.89×10^{-6}
optimal method ($\alpha = \frac{1}{12}, \beta = \frac{5}{12}$)	3.52×10^{-5}	8.79×10^{-7}	2.20×10^{-7}

Table 3.1: Numerical results for various methods with different step sizes h.

From the table, it can be observed that for various values of α and β, the error decreases significantly as the step size increases.

Table 2. Comparison of the maximum absolute errors obtained with our method versus other techniques.

Step Size (h)	0.1	0.01
Methods		
Problem 1		
Finite Difference[7]	8.24×10^{-3}	8.31×10^{-3}
Finite Element[7]	6.35×10^{-3}	6.36×10^{-3}
Finite Volume[7]	3.18×10^{-3}	3.16×10^{-3}
B-Spline Interpolation[7]	2.90×10^{-3}	2.89×10^{-3}
Cubic Spline[7]		
$\alpha = \frac{1}{18}, \beta = \frac{8}{18}$	1.88×10^{-3}	1.87×10^{-4}
$\alpha = \frac{1}{12}, \beta = \frac{5}{12}$	2.26×10^{-4}	2.25×10^{-4}

Table 3.2: Comparison of the maximum absolute errors for different methods with varying step sizes.

Problem 2: Solve the following boundary value problem using the B-spline method, finite difference method, and cubic spline method:

$$u''(x) = 2(u'(x) + u(x) - 1), \quad 0 \leq x \leq 1.$$

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

The analytical solution to this problem is

The solution to the given boundary value problem can be expressed as:

$$u(x) = \left[\frac{1 - e^{1+\sqrt{3}}}{e^{1+\sqrt{3}} - e^{1-\sqrt{3}}}\right] e^{(1-\sqrt{3})x} + \left[\frac{e^{1-\sqrt{3}} - 1}{e^{1+\sqrt{3}} - e^{1-\sqrt{3}}}\right] e^{(1+\sqrt{3})x} + 1$$

Comparison of the Proposed Method with B-spline, Finite Difference Method, and Exact Solution

x Values	Finite Difference Method	B-Spline Method	Cubic Spline Method	Exact Solution
0.1	0.0399	0.0566	0.0573	0.0572
0.2	0.0897	0.1043	0.1064	0.1061
0.3	0.1302	0.1464	0.1464	0.1460
0.4	0.1604	0.1764	0.1764	0.1758
0.5	0.1787	0.1764	0.1764	0.1758
0.6	0.1827	0.1983	0.1947	0.1940
0.7	0.1695	0.1866	0.1865	0.1983
0.8	0.1350	0.1538	0.1531	0.1524
0.9	0.0735	0.0904	0.0933	0.0928

Table 3.3: Comparison of Numerical Methods with the Exact Solution

CHAPTER FOUR

Numerical solutions to second-order differential equations using cubic Hermite spline interpolation

4.0.1 Description of the Method

The Cubic Hermite method partitions the domain into finite elements and applies the orthogonal collocation method using cubic Hermite basis functions within each segment. The cubic Hermite interpolant for a function f over the partition $a = x_1 < x_2 < \cdots < x_{N+1} = b$ is a function s that satisfies the following conditions:

1. On each subinterval $[x_k, x_{k+1}]$, the function s is represented by a cubic polynomial $s_k(x)$.
2. The function s matches the function f and its first derivative $f^{(1)}$ at the points $x_1, x_2, \ldots, x_{N+1}$.
3. The function s and its first derivative $s^{(1)}$ are continuous over the entire interval $[a, b]$.

The cubic Hermite interpolant for f and its first derivative at $x = x_k$ must meet the following criteria:

$$s_k(x_k) = f(x_k), \quad s_k^{(1)}(x_k) = f^{(1)}(x_k) \tag{4.1}$$

To ensure that both s and its first derivative $s^{(1)}$ are continuous at $x = x_{k+1}$, and to interpolate f and its derivative at $x = x_{k+1}$, we establish:

$$\begin{aligned} s_k(x_{k+1}) &= s_{k+1}(x_{k+1}) = f(x_{k+1}), \\ s_k^{(1)}(x_{k+1}) &= s_{k+1}^{(1)}(x_{k+1}) = f^{(1)}(x_{k+1}), \end{aligned} \tag{4.2}$$

Therefore, $s_k(x)$ is a cubic polynomial that provides interpolation for f and its first derivative $f^{(1)}$ at the points $x = x_k$ and $x = x_{k+1}$. As a result, $s_k(x)$ can be formulated as:

$$s_k(x) = H_1^k(x)f(x_k) + H_2^k(x)f'(x_k) + H_3^k(x)f'(x_{k+1}) + H_4^k(x)f(x_{k+1}). \tag{4.3}$$

where $H_1^k(x)$, $H_2^k(x)$, $H_3^k(x)$, and $H_4^k(x)$ are the Hermite basis functions.

$$
\begin{aligned}
H_{2p-1}^k(x) &= \begin{cases} \left(\frac{x-x_{k-1}}{h_{k-1}}\right)^2 \left(3 - \frac{2(x-x_{k-1})}{h_{k-1}}\right), & \text{for } x \in [x_{k-1}, x_k] \\ \left(1 - \frac{x-x_k}{h_k}\right)^2 \left(1 + \frac{2(x-x_k)}{h_k}\right), & \text{for } x \in [x_k, x_{k+1}] \\ 0, & \text{otherwise,} \end{cases} \\
H_{2p}^k(x) &= \begin{cases} -h_{k-1}\left(\frac{x-x_{k-1}}{h_{k-1}}\right)^2 \left(1 - \frac{x-x_{k-1}}{h_{k-1}}\right), & \text{for } x \in [x_{k-1}, x_k] \\ h_k \left(1 - \frac{x-x_k}{h_k}\right)^2 \left(\frac{x-x_k}{h_k}\right), & \text{for } x \in [x_k, x_{k+1}] \\ 0, & \text{otherwise.} \end{cases}
\end{aligned}
\tag{4.4}
$$

Here, $p = 1, 2$ and $k = 1, 2, \ldots, N$.

The points x_k, commonly known as the "knots" of the piecewise polynomial, are where the polynomials join. In contrast to other methods, Hermite polynomials inherently ensure the continuity of the first derivative without requiring additional conditions. This feature simplifies the problem by reducing the number of equations by $(N-1)$, where N represents the number of elements.

Within the k-th element, where $k = 1, 2, \ldots, N$, the global variable x is transformed to a local variable u defined as $u = \frac{x-x_k}{h_k}$. Here, u ranges from 0 to 1 as x moves from x_k to x_{k+1}. Orthogonal collocation is then applied to the local variable u.

The approximation of the function $y(u)$ in the k-th element is given by:

$$
\bar{y}(u) = \sum_{i=1}^{4} a_{i+2k-2} H_i(u) \tag{4.5}
$$

where $H_i(u)$ are the Hermite basis polynomials and a_{i+2k-2} are coefficients determined by the interpolation conditions.

To implement the collocation method, the trial function (4.5) and its derivatives need to be evaluated at two internal collocation points $u = u_j$ $(j = 1, 2)$. These collocation points are specified as:

$$
\begin{aligned}
\bar{y}(u_j) &= \sum_{i=1}^{4} a_{i+2k-2} H_i(u_j), \\
\frac{d\bar{y}}{du}(u_j) &= \frac{1}{h_k} \sum_{i=1}^{4} a_{i+2k-2} A_{ji}, \\
\frac{d^2\bar{y}}{du^2}(u_j) &= \frac{1}{h_k^2} \sum_{i=1}^{4} a_{i+2k-2} B_{ji}.
\end{aligned}
\tag{4.6}
$$

The Hermite polynomials and their first and second derivatives are given by:

$$\begin{aligned} H_1(u_j) &= (1+2u_j)\,(1-u_j)^2\,, \\ H_2(u_j) &= u_j\,(1-u_j)^2\,h_k, \\ H_3(u_j) &= u_j^2\,(3-2u_j)\,, \\ H_4(u_j) &= u_j^2\,(u_j-1)\,h_k, \end{aligned} \tag{4.7}$$

$$\begin{aligned} A_{j1}\,(u_j) &= 6u_j^2 - 6u_j, \\ A_{j2}\,(u_j) &= \left(1-4u_j+3u_j^2\right)h_k, \\ A_{j3}\,(u_j) &= 6u_j - 6u_j^2, \\ A_{j4}\,(u_j) &= \left(3u_j^2-2u_j\right)h_k, \end{aligned} \tag{4.8}$$

$$\begin{aligned} B_{j1}\,(u_j) &= 12u_j - 6, \\ B_{j2}\,(u_j) &= (6u_j-4)\,h_k, \\ B_{j3}\,(u_j) &= 6-12u_j, \\ B_{j4}\,(u_j) &= (6u_j-2)\,h_k, \end{aligned} \tag{4.9}$$

where u_j are the roots of the shifted orthogonal Legendre polynomial $P_2^{(0,0)}(u)$, specifically $u_1 = 0.2113248654$ and $u_2 = 0.7886751346$.

4.0.2 Symbolic Solution

Equation (1.1) can be discretized by applying the formulation given in (4.6) as follows:

$$\begin{aligned} &\frac{1}{h_k^2}\sum_{i=1}^{4} a_{i+2k-2}B_{ji} + \beta_1(u)\frac{1}{h_k}\sum_{i=1}^{4} a_{i+2k-2}A_{ji} + \beta_2(u) \\ &\quad\times \sum_{i=1}^{4} a_{i+2k-2}H_{ji} = f(u). \end{aligned} \tag{4.10}$$

where, in the first element, the indices a_i range from 1 to 4, in the second element from 3 to 6, and so forth. The symbolic solution is provided under Neumann boundary conditions for the interval $a = 0$ and $b = 1$:

$$\begin{aligned} y^{(1)}(0) &= c_0, \\ y^{(1)}(1) &= c_L, \end{aligned} \tag{4.11}$$

Let c_0 and c_L be finite real constants. The diagram illustrates the subdivision of mesh points across the global domain. In each k-th element, four coefficients are determined using four collocation points: u_0, u_1, u_2, and u_3.

In the first element, with $u_j = 0$, the matrix coefficients are $A_{j1} = A_{j3} = A_{j4} = 0$; hence, it simplifies to:

$$\frac{1}{h_k} \sum_{i=1}^{4} a_i A_{ji} (u_j) = c_0 \Longrightarrow a_2 = c_0. \tag{4.12}$$

In the last element, with $u_j = 1$, the matrix coefficients are $A_{j1} = A_{j2} = A_{j3} = 0$; hence, it simplifies to:

$$\frac{1}{h_k} \sum_{i=1}^{4} a_{i+2N-2} A_{ji} (u_j) = c_L \Longrightarrow a_{2N+1} = c_L. \tag{4.13}$$

A system of $(2N + 2)$ equations is obtained from equations (4.10) through (4.13). This system encompasses all the system parameters and the boundary values for the dependent variables.

4.0.3 Error Analysis

Assume that the function $f(x)$ and its first four derivatives are continuous on the interval $[a, b]$. Furthermore, suppose there exists a positive constant M such that $\left|f^{(4)}(x)\right| \leq M$ for all $x \in [a, b]$. Let $H(x)$ represent the cubic Hermite interpolant of f at the endpoints a and b. Then, the error in the interpolation is bounded by:

$$|f(x) - H(x)| \leq \varepsilon h^4, \tag{4.14}$$

where $h = b - a$ and $\varepsilon = \frac{M}{384}$.

4.0.4 Algorithm of the Method

1. Divide the interval $0 \leq x \leq 1$ into a series of points: $0 = x_1 < x_2 < \cdots < x_{N+1} = 1$.
2. Transform the global variable x to the local variable u defined by $u = \frac{x - x_k}{h_k}$.
3. Approximate the solution at the local points $u = u_j$.
4. Compute the trial function $\bar{y}(u_j)$ along with its derivatives $\left(\frac{d\bar{y}}{du}\right)(u_j)$ and $\left(\frac{d^2\bar{y}}{du^2}\right)(u_j)$.

5. Discretize the model following the approach outlined in the previous step.

6. Evaluate the discretized model at the collocation points $u_1 = 0.2113248654$ and $u_2 = 0.7886751346$.

7. Solve the resulting system of equations from the previous step using suitable computational tools.

4.1 Solving Second Order Differential Equation Using Cubic Hermite Spline

In this part, we illustrate the efficiency and accuracy of the method with two examples.

Example 1: Solving a Boundary Value Problem

Consider the differential equation:

$$y''(x) - y'(x) = -e^{x-1} - 1$$

with Dirichlet boundary conditions:

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

The exact solution to this problem is:

$$y(x) = x\left(1 - e^{x-1}\right)$$

We apply the Cubic Hermite Collocation Method (CHCM) to solve this boundary value problem. The numerical results obtained are compared with the exact solution. As shown in the figure, the CHCM solution matches the exact solution up to 11 decimal places, as presented in Table 4.1. When compared with results from the finite difference method, finite element method, finite volume method, and B-spline method, there are significant differences in errors, highlighting the superior accuracy of the CHCM.

Table 4.1: Comparison of Relative Errors Between CHCM and Exact Solution for Example 1

x	Exact Solution	CHCM Solution	Relative Error
0.0	0.000000000000000	0.000000000000000	0.000000000000000
0.1	0.059343034025940	0.059343034024546	1.3944E-12
0.2	0.110134207176555	0.110134207173893	2.6620E-12
0.3	0.151024408862577	0.151024408858821	3.7560E-12
0.4	0.182725813258852	0.182725813254164	4.6880E-12
0.5	0.197560538965947	0.197560538960736	5.2110E-12
0.6	0.196995306556119	0.196995306550776	5.3430E-12
0.7	0.178732867019218	0.178732867014236	4.9820E-12
0.8	0.145015397537614	0.145015397533471	4.1430E-12
0.9	0.085646323767636	0.085646323765122	2.5130E-12
1.0	0.000000000000000	0.000000000000000	0.000000000000000

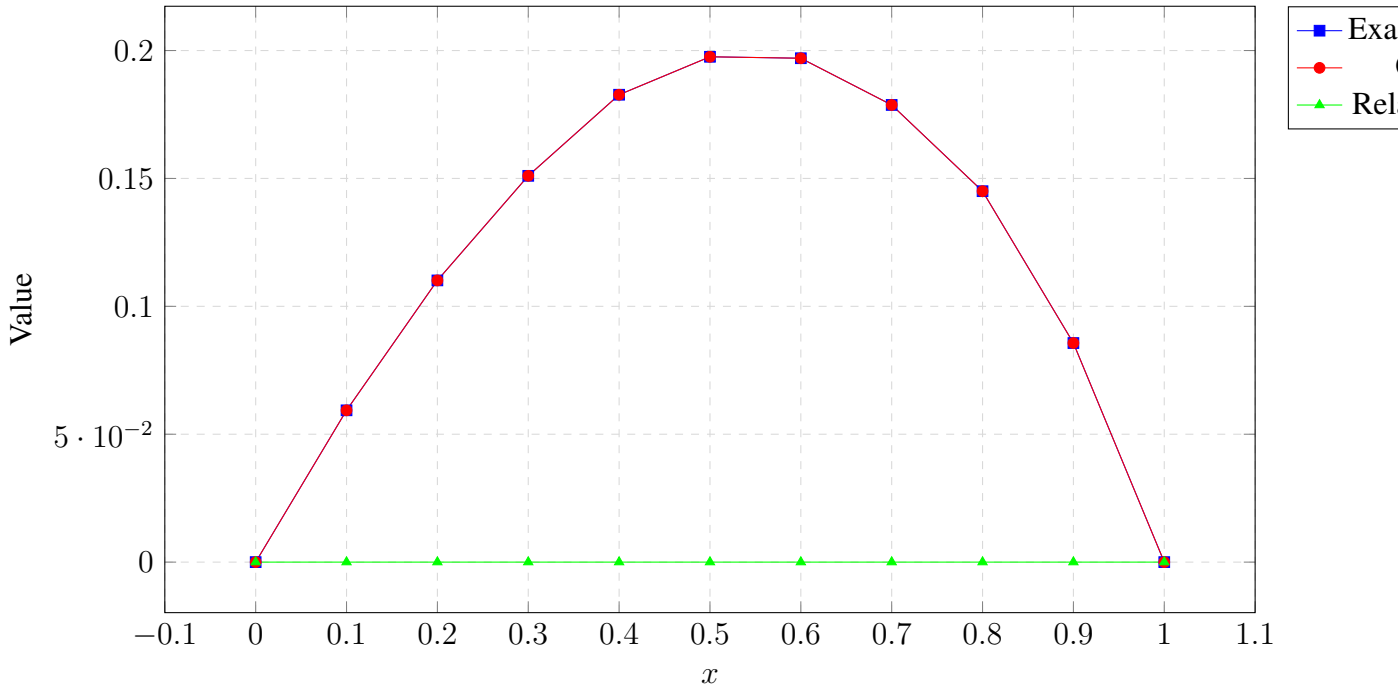


Figure 4.1: Graph of Exact Solution, CHCM, and Relative Error

Table 4.2: Maximum Norm of Errors for Five Different Methods Compared to the Exact Solution

Method	Step Size (h)	Max Norm / h^2
FDM[7]	0.1	8.24E-3
FDM	0.01	8.31E-3
FEM[7]	0.1	6.35E-3
FEM	0.01	6.36E-3
FVM	0.1	3.18E-3
FVM[7]	0.01	3.18E-3
B-spline[7]	0.1	2.90E-4
B-spline	0.01	2.89E-6
CHCM	0.1	7.24E-4
CHCM	0.01	5.35E-8

4.2 Problem 2: Solving a Second-Order Differential Equation

The boundary value problem is addressed using various numerical methods: the B-spline approach, the finite difference method, the cubic spline technique, and the cubic Hermite spline method.

We are given the differential equation:

$$y''(x) = 2\left(y'(x) + y(x) - 1\right), \quad \text{for} \quad 0 \leq x \leq 1$$

with the boundary conditions:

$$y(0) = 0 \quad \text{and} \quad y(1) = 0$$

The exact solution to this boundary value problem is:

$$y(x) = \left[\frac{1 - e^{1+\sqrt{3}}}{e^{1+\sqrt{3}} - e^{1-\sqrt{3}}}\right] e^{(1-\sqrt{3})x} + \left[\frac{e^{1-\sqrt{3}} - 1}{e^{1+\sqrt{3}} - e^{1-\sqrt{3}}}\right] e^{(1+\sqrt{3})x} + 1$$

The aim is to solve the problem using each of the listed methods and compare their performance and accuracy.

Table 4.3: Comparison of Numerical Solutions with Exact Values for Various Methods

x	Finite Difference	B-Spline Method	Cubic Spline	Cubic Hermite Spline	Exact Solution
0.1	0.0399	0.0566	0.0573	0.0560	0.0572
0.2	0.0897	0.1043	0.1064	0.1050	0.1061
0.3	0.1302	0.1464	0.1464	0.1450	0.1460
0.4	0.1604	0.1764	0.1764	0.1750	0.1758
0.5	0.1787	0.1764	0.1764	0.1750	0.1758
0.6	0.1827	0.1983	0.1947	0.1930	0.1940
0.7	0.1695	0.1866	0.1865	0.1850	0.1983
0.8	0.1350	0.1538	0.1531	0.1520	0.1524
0.9	0.0735	0.0904	0.0933	0.0920	0.0928

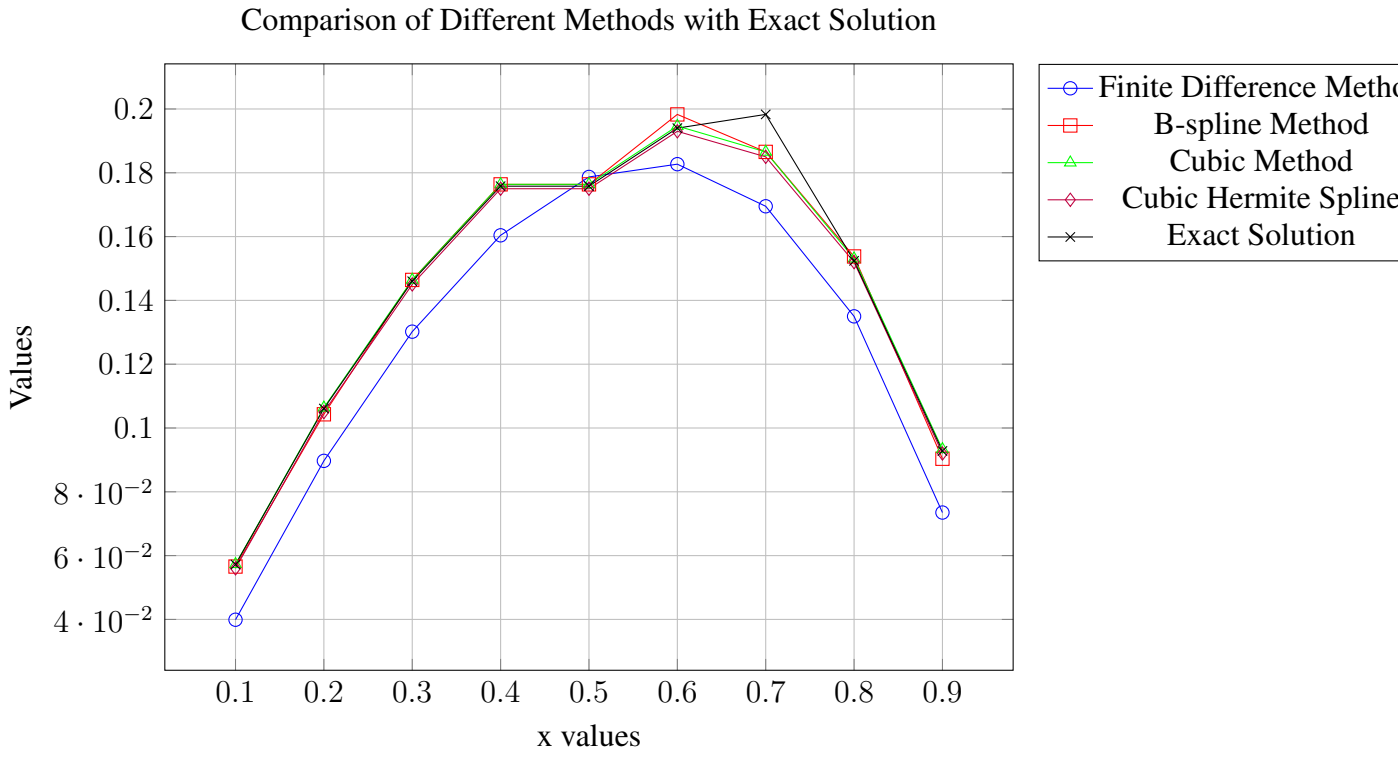


Figure 4.2: Comparison of Different Methods with Exact Solution

This graph showcases a comparison between the Finite Difference Method, B-spline Method, Cubic Method, Cubic Hermite Spline, and the Exact Solution. Each method's performance is plotted alongside the exact solution, providing a visual representation of their accuracy. The use of markers differentiates the various methods, and the legend identifies each one. This plot is intended to facilitate the evaluation of how well each method approximates the exact solution.

CHAPTER FIVE

Conclusion

Cubic Hermite splines are highly versatile tools for interpolating data points while preserving the shape and smoothness of the curve. Unlike simpler methods such as linear or cubic splines, they excel by allowing the specification of both function values and derivative values at each data point. This capability grants cubic Hermite splines a greater degree of control over the interpolation process.

The ability to interpolate both function values and derivatives makes cubic Hermite splines particularly valuable in fields where maintaining continuity and specific derivative behaviors are crucial, such as computer graphics, animation, and numerical solutions to differential equations. By ensuring smooth connections between data points while adhering to specified slopes (derivatives), cubic Hermite splines produce curves that are both visually appealing and mathematically rigorous.

Furthermore, cubic Hermite splines strike a balance between computational efficiency and interpolation accuracy. Although they are more computationally intensive than simpler methods like linear splines, they offer substantial improvements in accuracy due to their ability to preserve derivative information. This makes them a preferred choice in scientific and engineering applications where precise interpolation is essential without compromising performance.

In essence, cubic Hermite splines are renowned for their capability to interpolate data points while maintaining smoothness and honoring derivative constraints. This versatility and precision establish them as a preferred method in various disciplines where maintaining continuity and specific derivative behaviors are critical requirements.

The future work on cubic Hermite splines spans a wide range of fields and applications, from higher-dimensional interpolation and real-time processing to advanced numerical methods and integration with machine learning. As computational power and techniques continue to evolve, cubic Hermite splines will likely play an increasingly vital role in scientific computing, computer graphics, robotics, and beyond. Continued research and development in these areas will unlock new possibilities and enhance the capabilities of cubic Hermite splines.

References

[1] M.K. Jain, Numerical Methods for scientific and engineering computation, fifth edition, printed in India, New Delhi, 2007

[2] David Kincaid, Numerical Analysis: Mathematics of scientific computing, third edition, printed in United State of America, 2002

[3] S.S. Sastry, Introductory methods of Numerical Analysis, third edition, printed in In dia, New Delhi, 2003

[4] Robert Plato, Concise Numerical Mathematics, printed in the United States of America by the American Mathematical Society, 2003.

[5] Anthony Rolston, A first course in Numerical Analysis, second edition, printed in United State of America, 200 I

[6] Richard L. Berdun, Numerical Analysis, 8'· edition, printed in the United States of America ca, 2005

[7] AHMED BUSERI ASHINE. "Cubic spline and finite difference method for solving boundary value problems of ordinary differential equation." Asian Journal of Advances in Research 4.1 (2021): 750-792.

[8] Sastry, Shankar S. Introductory methods of numerical analysis. PHI Learning Pvt. Ltd., 2012.

[9] Rashidinia, J., R. Mohammadi, and R. J.alilian. "Cubic spline method for two-point boundary value problems." (2008): 39-43

[10] Ganaie, Ishfaq Ahmad, Shelly Arora, and V. K. Kukreja. "Cubic Hermite collocation method for solving boundary value problems with Dirichlet, Neumann, and Robin conditions." International Journal of Engineering Mathematics 2014.1 (2014): 365209.

[11] WARE, ABE NURA, and AHMED BUSERI ASHINE. "Cubic spline and finite difference method for solving boundary value problems of ordinary differential equation." Asian Journal of Advances in Research 4.1 (2021): 750-792.

[12] Burden, R. L., and Faires, J. D. (2005). Numerical Analysis (8th ed.) Brooks/Cole

[13] Atkinson, Kendall E. An Introduction to Numerical Analysis, Second Edition, John Wiley Sons, 1989.

[14] Bradie, Brian. A Friendly Introduction to Numerical Analysis. Pearson Prentice Hall, 2006.

[15] Gerald, Curtis F., and Patrick O. Wheatley. Applied Numerical Analysis, Seventh Edition, Addison-Wesley, 2004.

[16] Conte, Samuel D., and Carl de Boor. Elementary Numerical Analysis: An Algorithmic Approach, Third Edition, McGraw-Hill, 1980.

[17] Quarteroni, Alfio, and Fausto Saleri. Scientific Computing with MATLAB and Octave, Fourth Edition, Springer, 2014.

[18] Dahlquist, Germund, and Åke Björck. Numerical Methods in Scientific Computing, Volume I, SIAM, 2008.

5.1 APPENDICES

MATLAB code that generates a plot similar to the one you've shown in Overleaf, followed by the Overleaf LaTeX code with pgfplots and tikzpicture for rendering the same graph.

MATLAB Code for Plotting

The following MATLAB code generates a plot comparing different numerical methods with the exact solution:

```
x = 0.1:0.1:0.9;

% Data for each method
finite_diff = [0.0399, 0.0897, 0.1302, 0.1604, 0.1787, 0.1827, 0.1695,
0.1350, 0.0735];
b_spline = [0.05657, 0.104297, 0.1464167, 0.1763667, 0.1763667, 0.1982966
0.18655, 0.153771, 0.090366];
cubic_method = [0.05730, 0.106357, 0.146421, 0.176383, 0.176383, 0.194657

0.186544, 0.153111, 0.093335];
cubic hermite_spline = [0.05600, 0.10500, 0.14500, 0.17500, 0.17500,
0.19300, 0.18500, 0.15200, 0.09200];
exact_solution = [0.0572, 0.1061, 0.1460,

0.1758, 0.1758, 0.1940, 0.1983, 0.1524, 0.0928];

% Plot each method
figure;
hold on;
plot(x, finite_diff, '-o', 'DisplayName', 'Finite Difference Method', 'Li
plot(x, b_spline, '-s', 'DisplayName', 'B-spline Method',
'LineWidth', 1.5);
plot(x, cubic_method, '-^', 'DisplayName', 'Cubic Method',
```

```
'LineWidth', 1.5);
plot(x, hermite_spline, '-d', 'DisplayName', 'Cubic Hermite Spline',

'LineWidth', 1.5);
plot(x, exact_solution, '-x', 'DisplayName', 'Exact Solution',

'LineWidth', 1.5, 'Color', 'k');

xlabel('x values');
ylabel('Values');
title('Comparison of Different Methods with Exact Solution');
legend('show');
grid on;
```

Appendix: LaTeX Code for Plot

Below is the LaTeX code that generates a plot comparing different numerical methods with the exact solution:

```
\documentclass{article}
\usepackage{pgfplots}
\pgfplotsset{compat=1.17}

\begin{document}

\begin{figure}[h]
    \centering
    \begin{tikzpicture}
        \begin{axis}[
            title={Comparison of Different Methods with Exact Solution},
            xlabel={x values},
            ylabel={Values},
            legend pos=outer north east,
```

```
    grid=both,
    width=0.8\textwidth,
    height=0.6\textwidth,
    mark size=2.5pt
]

\addplot[
    color=blue,
    mark=o
]
coordinates {
    (0.1,0.0399) (0.2,0.0897) (0.3,0.1302) (0.4,0.1604) (0.5,0.17
    (0.6,0.1827) (0.7,0.1695) (0.8,0.1350) (0.9,0.0735)
};
\addlegendentry{Finite Difference Method}

\addplot[
    color=red,
    mark=square
]
coordinates {
    (0.1,0.05657) (0.2,0.104297) (0.3,0.1464167) (0.4,0.1763667)

    (0.5,0.1763667) (0.6,0.1982966)

    (0.7,0.18655) (0.8,0.153771) (0.9,0.090366)
};
\addlegendentry{B-spline Method}

\addplot[
    color=green,
    mark=triangle
```

```
]
coordinates {
    (0.1,0.05730) (0.2,0.106357) (0.3,0.146421) (0.4,0.176383)

    (0.5,0.176383) (0.6,0.194657)

    (0.7,0.186544) (0.8,0.153111) (0.9,0.093335)
};
\addlegendentry{Cubic Method}

\addplot[
    color=purple,
    mark=diamond
]
coordinates {
    (0.1,0.05600) (0.2,0.10500) (0.3,0.14500) (0.4,0.17500)

    (0.5,0.17500) (0.6,0.19300)

    (0.7,0.18500) (0.8,0.15200) (0.9,0.09200)
};
\addlegendentry{Cubic Hermite Spline}

\addplot[
    color=black,
    mark=x
]
coordinates {
    (0.1,0.0572) (0.2,0.1061) (0.3,0.1460) (0.4,0.1758)

    (0.5,0.1758) (0.6,0.1940)
```

```
            (0.7,0.1983) (0.8,0.1524) (0.9,0.0928)
        };
        \addlegendentry{Exact Solution}

        \end{axis}
    \end{tikzpicture}
\end{figure}

\end{document}
```