ANALYSIS OF CUBIC SPLINE METHOD, B-SPLINE METHOD AND FINITE DIFFERENCE METHOD FOR SOLVING BOUNDARY VALUE PROBLEMS

BY
NURU AHMED

A THESIS SUBMITTED TO
THE PROGRAM OF APPLIED MATHEMATICS
SCHOOL OF APPLIED NATURAL SCIENCE

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTERS OF SCIENCE IN APPLIED MATHEMATICS

OFFICE OF GRADUATE STUDIES
ADAMA SCIENCE AND TECHNOLOGY UNIVERSITY

ADAMA, ETHIOPIA
JUNE, 2017

Analysis of Cubic Spline Method, B-Spline Method and Finite Difference Method for Solving Boundary Value Problems

By
Nuru Ahmed
Advisor
Tekle Gemechu (PhD)

A Thesis Submitted to
The Program of Applied Mathematics
School of Applied Natural Science

Presented in Partial Fulfillment of the Requirement for
the Degree of Masters to Science in Applied Mathematics

Office of Graduate Studies
Adama Science and Technology University

Adama, Ethiopia
June, 2017

Table of Contents

List of Figures

List of Tables

Abstract

The thesis deals with analysis of cubic spline and cubic B-spline interpolation and finite difference for boundary value problems. Investigation on the major problem specially on the behavior of the solutions and how to solve the ordinary differential equations using cubic spline method, cubic B- spline interpolation and finite difference methods. Cubic B-spline functions play important roles in both mathematics and engineering. To describe a numerical method for solving the boundary value problem with second-order using cubic B-spline, first, the cubic B-spline basis functions are introduced, then we use the linear combination of cubic B-spline basis to approximate the solution. Finally, we obtain the numerical solution by solving tri-diagonal equations. The finite difference method needs discretization with equal mesh sizes and replacing the BVP by difference equations,which also involves tri-diagonal systems for the solution. Convergence and stability of the methods will be discussed.The absolute errors in test examples are estimated , the comparison of approximate values and exact values and absolute error at the nodal point are shown tabularly and graphically . Further, shown that cubic B-spline method produces accurate result comparing with cubic spline and finite difference methods.

Acknowledgements

First of all I would like to thanks almighty ALLAH,who helped me to do every activity in every time. Next to this I would like to appreciate and thanks my advisor Dr Tekle Gemechu for his constructive criticism valuable suggestion and for giving supportive materials. I would like to also thanks my families for their valuable support during the process of the study. I also express my thanks to all my friends and other individuals who contribute directly or indirectly to the study.

Adama Science and Technology University Nuru Ahmed
June , 2017

Chapter 1

Introduction

1.1 Background of the study

Numerical data is usually difficult to analyze. To this end, the idea of the interpolation was developed. In the mathematical field of numerical analysis, interpolation is a method of constructing new data points within the range of a discrete set of known data points. Interpolation provides a means of estimating the value at the new data points within the range of parameters. Splines and particularly cubic splines are very popular models for interpolation by Kia Wang.(2013).

Historically, a spline was a common drafting tool, a flexible rod that was used to help draw smooth curves connecting widely spaced points. The break points of a spline are also referred to as its knots. The world of splines extends far beyond the basic one-dimensional, cubic, interpolatory spline we are describing here. There are multidimensional, high-order, variable knot, and approximating splines. The cubic spline curve accomplishes the same result for an interpolation problem. The spline technology has applications in computer aided design (CAD), computer aided manufacturing (CAM), and computer graphics systems. The cubic spline interpolation is a piece-wise continuous curve, passing through each of the values in the table. There is a separate cubic polynomial for each interval, each with its own coefficients Shang (2011).

The goal of cubic spline interpolation is to get an interpolation formula that is continuous in both the first and second derivatives, both within the intervals and at the

interpolating nodes. This will give us a smoother interpolating function. The first derivative and the second derivative of a cubic spline are continuous.The continuity of first derivative means that the graph y=s(x) will not have sharp corners. And also the continuity of second derivative means that the radius of curvature is defined at each point.
The fundamental idea behind cubic spline interpolation is based on the engineers tool used to draw smooth curves through a number of points. This spline consists of weights attached to a flat surface at the points to be connected. A flexible strip is then bent across each of these weights, resulting in a pleasingly smooth curve. The mathematical spline is similar in principle. The points, in this case, are numerical data. The weights are the coefficients on the cubic polynomials used to interpolate the data. These coefficients bend the line so that it passes through each of the data points without any erratic behaviour or breaks in continuity by Wolberg(2002).

It is well-known that many real life phenomena in physics and engineering can be modelled by systems of linear and non-linear differential equations. One class of these systems is of second order boundary value problems.

Ordinary Differential Equations (ODE) has a long history and widely applied in many fields. The numerical solution of ODE has made great development in the 20th century. There have been emerged many new ideas as well as many complex methods for solving ODE, so that the numerical methods for solving ODE has been deepened. Systems of ordinary differential equation have been applied to many problems in physics, engineering, biology and so on. The theory of spline functions is a very active field of approximation theory for boundary value problems (BVPs) when numerical aspects are considered Nazan and Hikmet(2006).There are many studies on the solutions of linear and non-linear systems of second order boundary value problems by Wolberg (2002). The main purpose of our present study is to apply a cubic spline and B-spline methods and finite difference method in solving Eq. (1.1.1).

There is a large body of work in the field of cubic spline interpolation. The earliest work in this area can be traced back to that of Chebyshev, his work was motivated by the Wolberg, Alfy (2002) need to design a stable governor for a steam engine.Currently,

work in this area is motivated by diverse applications in many industrial problems, including CAD,CAM, VLSI and signal processing. Recent work in this area dates back to Schweikerts work on splines in tension, where exponential splines were used as approximates. Tension parameters were used to control shape. All of these methods were global, interpolatory, and twice continuous.

In mathematics, finite-difference methods are numerical methods for solving differential equations by approximating them with difference equations, in which finite differences approximate the derivatives. Finite difference methods are thus discretization methods. In this thesis, we will discuss a direct method based on cubic spline method, B-spline and finite difference methods for two-point boundary value problems of second-order ordinary differential equation. There are many publications dealing with this problem with some methods. In the research, a cubic B-spline is used to solve two-point boundary value problems as the following systems which are assumed to have a unique solution in the interval [a, b].

$$\begin{cases} y''(x) + p(x)y'(x) + q(x)y(x) = f(x), & a < x < b \\ y(a) = \alpha, & y(b) = \beta \end{cases} \tag{1.1.1}$$

This type of boundary value problem is assumed to have a unique solution, $y(x)$ if $p(x)$, $q(x)$ and $f(x)$ are continuous on $[a, b]$.

An Ordinary differential equation(ODE): is an equation that relates a function y(x) to some of its derivatives $y^{(n)}(x) = \frac{d^n y}{dx^n}$.

In general we write as $F(x, y, y^{(1)}, y^{(2)}, \cdots, y^{(n)}) = 0$

Boundary value problem (BVP): A problem, typically an ODE or a PDE, which has values assigned on the physical boundary of the domain in which the problem is specified, is called a boundary value problem(BVP).

Example 1.1.1. $y''(x) + p(x)y' + q(x)y = f(x), 0 \leq x \leq 1, y(0) = 0$ *and* $y(1) = 0$ *where p,q and f are smooth functions*

1.2 Statement of the Problem

The proposed study intended to focus analysis of cubic spline method,cubic B-spline method and finite difference method for boundary value problems. A number of researchers have addressed solution of boundary value problems by using cubic spline, B-spline and finite difference methods. But in each of the reviews, there is no clear comparison on the stability, consistency and convergence of cubic spline method, B-spline method and finite difference method.

Hence the proposed study aims to answer the following research problems.

- Which method is better convergent?
- Which method is more stable?
- How can we apply the MATLAB programs for cubic B-spline method, finite difference method and cubic spline method?
- How to Compare cubic B-spline method,finite difference method and cubic spline method?

1.3 Objective of the Study

1.3.1 General objective

The main objective of this thesis is to analysis cubic spline method, B-spline method and finite difference methods for solving boundary value problems of ordinary differential equations(ODEs).

1.3.2 specific objective

The specific objectives of the thesis are:

1, To Determine the convergence of cubic spline method, B-spline method and finite difference method.

2, To Determine the stability of cubic spline method, B-spline method and finite difference method.

3, To Apply the MATLAB programs for cubic spline method, B-spline method and finite difference method.

4, To Comparing results of cubic spline method, B-spline method and finite difference methods and exact value.

1.4 Significance of the study

This study may help to develop skills and knowledge in cubic spline method, B-spline method and finite difference methods for boundary values problems, further may provide useful result and information for the advancement of numerical algorithm, the advancement of research and further for students will be doing on such types of topics. Moreover, it may serve as references material for those needs to conduct research on this area.

1.5 Limitation of the study

There are problems or shortages under study this thesis work. some of them are:

- Lack of important reference books.
- Shortage of time
- Network access is limited.

1.6 Delimitation of the study

Cubic spline method, B-spline method and finite difference methods are perhaps the most important methods in applied mathematics. It arises in varieties of mathematical and physical systems. B-spline,Finite difference and cubic spline methods have many applications in the real world. The thesis under study is mainly delimited to comparison of cubic spline method, B-spline method and finite difference methods for boundary value problems by explaining some behaviors and solving problems on it.So this study is delimited to a class of boundary value problems. The equations of the problem is given by:

$$\begin{cases} y''(x) + p(x)y'(x) + q(x)y(x) = f(x), & a < x < b \\ y(a) = \alpha, & y(b) = \beta \end{cases}$$

Chapter 2

Review of the related literature

2.1 Interpolation

We sometimes know the value of a function $f(x)$ at a set of points $x_1, x_2, \cdots, x_n$ (say, with $x_1 < x_2 < \cdots < x_n$), but we don't have an analytic expression for f(x) that lets us to calculate its value at an arbitrary point. For example, $f(x_i)$'s might result from some physical measurement or from long numerical calculation that cannot be cast into a simple functional form. Often the x_i's are equally spaced, but not necessarily. The task is now estimating $f(x)$ for arbitrary x by, in some sense, drawing a smooth curve through (and perhaps beyond) the x_i. If the desired x is in between the largest and smallest of the x_i 's , the problem is called interpolation .

2.1.1 Types of interpolation

There are several different interpolation methods based on the accuracy, how expensive is the algorithm of implementation, smoothness of interpolation function, etc.

Piece-wise constant interpolation

This is the simplest interpolation, which allows allocating the nearest value and assigning it to the estimating point. This method may be used in the higher dimensional multivariate interpolation, because of its calculation speed and simplicity.

Linear interpolation:

Linear interpolation takes two data points, say (x_a, y_a) and (x_b, y_b), and the interpolation function at the point (x, y) is given by the following formula:

$$y = y_a + (y_b - y_a)\frac{x - x_a}{x_b - x_a}$$

Linear interpolation is quick and easy, but not very precise.

Polynomial interpolation:

Polynomial interpolation is a generalization of linear interpolation. It replaces the interpolating function with a polynomial of higher degree. If we have n data points, there is exactly one polynomial of degree at most $n - 1$ going through all the data points:

$$P(x) = a_n x_n + a_{n-1} x_{n-1} + \cdots + a_2 x_2 + a_1 x + a_0$$

With higher degree polynomial ($n > 1$), the interpolation error can be very small. So, we see that polynomial interpolation overcomes most of the problems of linear interpolation. However, polynomial interpolation also has some disadvantages. For example, calculating the interpolating polynomial is computationally expensive compared to linear interpolation. More generally, the shape of the resulting curve, especially for very high or low values of the independent variable, may be contrary to common sense. These disadvantages can be reduced by using spline interpolation Wang(2013).

Spline interpolation:

Spline interpolation is an alternative approach to data interpolation. Compared to polynomial interpolation using on single formula to correlate all the data points, spline interpolation uses several formulas; each formula is a low degree polynomial to pass through all the data points. These resulting functions are called splines . Spline interpolation is preferred over polynomial interpolation because the interpolation error can be made small even when using low degree polynomials for the spline. The mathematical model for spline interpolation can be described as following: For $i = 1, \cdots, n$ data points, interpolate between all the pairs of knots (x_{i-1}, y_{i-1}) and (x_i, y_i) with polynomials $y = q_i(x)$, $i = 1, 2, \cdots, n$.

As the spline will take a function (shape) more smoothly (minimizing the bending), both y' and y'' should be continuous everywhere and at the knots. Therefore:
$q_i'(x_i) = q_{i+1}'(x_{i+1})$ and $q_i''(x_i) = q_{i+1}''(x_{i+1})$,where $1 \leq i \leq n-1$ This can only be achieved if polynomials of degree 3 or higher are used. The classical approach uses polynomials of degree 3, which is the case of cubic splines Rainer Kress(1998).

Cubic Spline Interpolation

Numerical methods have been introduced to generate approximation solution of the problem as equation (1.1.1) is difficult to be solved analytically. Shooting method and finite difference method used to solve linear and non linear boundary value problems and commonly used in explaining numerical methods in solving ordinary differential equations. In 1968, Fyfe introduced cubic spline interpolation method on approximating two-point boundary value problem, which then had successfully leads to many further investigation on application of spline in boundary value problems . Part of them are that focused on the application of cubic spline interpolation on boundary value problems. Fang et al(2003). had proposed a comparison on finite difference, finite element and finite volume methods applied to two-point boundary value problem. The details and the results can be found in and the references therein. Five years later, Caglar et al(2006). had proposed to compare the performance of the techniques studied by Fang et al(2003). with the new approach that Caglar et al(2006). developed as improved version of cubic spline interpolation in solving two-point boundary value problem. The method is known as cubic B-spline interpolation by Nazan and Hikmet (2009).

2.2 Finite difference method

The finite difference theory for general initial value problems and parabolic problems then had an intense period of development during 1950s and 1960s, when the concept of stability was explored. Independently of the engineering applications, a number of papers appeared in the mathematical literature in the mid-1960s which were concerned with the construction and analysis of finite difference method by the Rayleigh-Ritz procedure with piecewise linear approximating functions. Finite difference method is one of the most widely used numerical method to solve differential equations. The general concept of finite difference methods in details such as stability, convergence conditions, have been presented in literature for solving differential equations. The challenge in analysing finite

difference method for new classes of problems is often to find an approximate definition of stability that allow one to prove convergence and to estimate the error in approximation LeVeque (1998).

The basic steps for a finite difference method are as follows: first, choose a knot on the interval of interest, that is, for $[a, b]$

$a = x_0, x_1 \cdots x_{n-1}, x_n = b$ such that the approximate solution will be sought at these knot points; second, form the algebraic equations required to satisfy the differential equation and the boundary conditions by replacing derivatives with difference quotients involving only the knot points; and last, solve the algebraic system of equations.Methods involving finite differences for solving boundary-value problems replace each of the derivatives in the differential equation with an appropriate difference-quotient. The particular difference quotient is chosen to maintain a specified order of error. The finite-difference method for the second-order boundary-value problem,(1.1.1) requires that difference-quotient approximations be used for approximating both y' and y''. First, we select an integer $N > 0$ and divide the interval $[a, b]$ into $(N + 1)$ equal subintervals whose end points are the knot points $x_i = a + ih$, for $i = 0, 1, \cdots, N + 1$, where $h = \frac{(b-a)}{(N+1)}$. At the interior mesh points,x_i, for $i = 1, 2, \cdots, N$, the differential equation to be approximated is

$y''(x_i) = p(x_i)y'(x_i) + q(x_i)y(x_i) + f(x_i)$ by Burden and Faires (2002).

In general the idea of the finite difference method is to replace each derivative involved in boundary value problem in term of central difference approximations, which then leads to system of equations. The solution of the system of equations which later turned to a problem of solving a system of tridiagonal matrix, yields approximations to the solution of the original differential equations at discrete points.

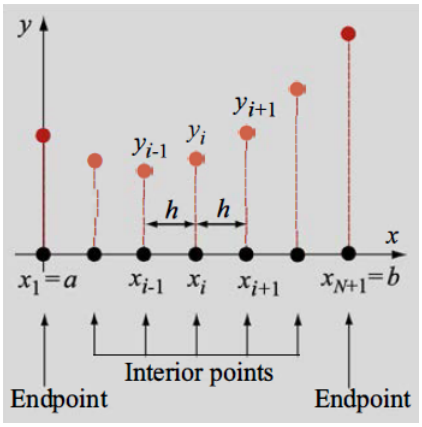

Figure 2.1: Discritization of the domain

Chapter 3

Research methodology

3.1 Study design

The study involves numerical implementation with MATLAB. For the study, the domain is discretized by cubic spline, B-spline and finite difference. Then the domain is discretized into a system of linear equations and obtain a function value at each node or the numerical approximation to the solution of our cubic B spline and finite difference method. Finally, the resulting linear system is solved using matrix for the unknown values on the node.

3.2 Study procedure

In order to achieve the stated objectives,the study will follow the following procedures:

- Define the methods.
- Discretize the given domain (or interval).
- Replacing the differential equations by cubic spline, cubic B-spline and finite difference approximations.
- Obtaining the tri-diagonal system which can be solved by any numerical methods.
- Writing a code for the problem by using MATLAB language
- Validating the methods using numerical examples.

Chapter 4

Results and Discussion

4.1 Cubic Spline Interpolation

Definition 4.1.1 (Cubic spline Interpolation)**.** *Given the n data points* $(x_1, y_1), \ldots, (x_n, y_n)$*, where* x_i *are distinct and in increasing order.* ***A cubic spline*** *S(x) through the data points* $(x_1, y_1), , (x_n, y_n)$ *is a set of cubic polynomials:*

$$S(x) = \begin{cases} s_1(x) = y_1 + b_1(x - x_1) + c_1(x - x_1)^2 + d_1(x - x_1)^3, & on\ [x_1, x_2] \\ s_2(x) = y_2 + b_2(x - x_2) + c_2(x - x_2)^2 + d_2(x - x_2)^3, & on\ [x_2, x_3] \\ \vdots & \\ s_{n-1}(x) = y_{n-1} + b_{n-1}(x - x_{n-1}) + c_{n-1}(x - x_{n-1})^2 + d_{n-1}(x - x_{n-1})^3, & on\ [x_{n-1}, x_n] \end{cases}$$

with the following properties:

a, $S_i(x_i) = y_i$ *and* $S_i(x_{i+1}) = y_{i+1}$

b, $S'_{i-1}(x_i) = S'_i(x_i)$*, for* $i = 2, \ldots, n-1$

c, $S''_{i-1}(x_i) = S''_i(x_i)$*, for* $i = 2, , n-1$

4.2 Construction of cubic spline

How to determine the unknown coefficients b_i, c_i and d_i of the cubic spline $S(x)$? So that we can construct it. Given S(x) is cubic spline that has all the properties as in the

definition,

$$S_i(x) = y_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3, on[x_i, x_{i+1}] \tag{4.2.1}$$

for $i = 1, 2, \ldots, n-1$
The first and second derivatives are:

$$S_i'(x) = 3d_i(x - x_i)^2 + 2c_i(x - x_i) + b_i \tag{4.2.2}$$

$$S_i''(x) = 6d_i(x - x_i) + 2c_i \tag{4.2.3}$$

for $i = 1, 2, \cdots, n-1$
From the first property of cubic spline, S(x) will interpolate all the data points, and we can have $S_i(x_i) = y_i$. Since the curve S(x) must be continuous across its entire interval, it can be concluded that each sub-function must join at the data points that is
$S_i(x_i) = S_{i-1}(x_i)$ Therefore,
$y_i = S_{i-1}(x_i)$

$$S_{i-1}(x_i) = y_{i-1} + b_{i-1}(x_i - x_{i-1}) + c_{i-1}(x_i - x_{i-1})^2 + d_{i-1}(x_i - x_{i-1})^3 \tag{4.2.4}$$

$$y_i = y_{i-1} + b_{i-1}(x_i - x_{i-1}) + c_{i-1}(x_i - x_{i-1}) + d_{i-1}(x_i - x_{i-1})^3, for i = 2, 3, \ldots, n-1. \tag{4.2.5}$$

Letting $h = x_i - x_{i-1}$ in (4.2.3), we have:

$$y_i = y_{i-1} + b_{i-1}h + c_{i-1}h^2 + d_{i-1}h^3, for i = 2, 3, \cdots, n-1 \tag{4.2.6}$$

Also, with properties (b) of cubic spline, the derivatives must be equal at the data points, that is:

$$S_{i-1}'(x_i) = S_i'(x_i) \tag{4.2.7}$$

By using (4.2.2)

$$S_i'(x_i) = b_i \text{ and}$$

$$S_{i-1}'(x_i) = 3d_{i-1}(x_i - x_{i-1})^2 + 2c_{i-1}(x_i - x_{i-1}) + b_{i-1} \tag{4.2.8}$$

Therefore,

$$b_i = 3d_{i-1}(x_i - x_{i-1})^2 + 2c_{i-1}(x_i - x_{i-1}) + b_{i-1} \tag{4.2.9}$$

Again, letting $h = x_i - x_{i-1}$, we find:

$$b_i = 3d_{i-1}h^2 + 2c_{i-1}h + b_{i-1}, for\ i = 2, 3, \ldots, n-1 \tag{4.2.10}$$

From (4.2.3)

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

$$S_i''(x_i) = 2c_i \tag{4.2.11}$$

Since $S_i''(x)$ should be continuous across the interval, therefore

$$S_{i-1}''(x_i) = S_i''(x_i)$$

$$S_{i-1}''(x_i) = 6d_{i-1}(x_i - x_{i-1}) + 2c_{i-1} \tag{4.2.12}$$

$$2c_i = 6d_{i-1}(x_i - x_{i-1}) + 2c_{i-1}$$

Letting $h = x_i - x_{i-1}$

$$2c_i = 6d_{i-1}(x_i - x_{i-1}) + 2c_{i-1}$$

$$2c_i = 6d_{i-1}h + 2c_{i-1} \tag{4.2.13}$$

Simplified these equation above by substituting M_i for $S_i''(x_i)$,

from (4.6.59)

$S_i''(x_i) = 2c_i, M_i = 2c_i$

$$c_i = \frac{M_i}{2} \tag{4.2.14}$$

From (4.6.61), we have

$$d_{i-1} = \tfrac{2c_i - 2c_{i-1}}{6h}$$

substitute, c_i we get

$$d_{i-1} = \frac{M_i - M_{i-1}}{6h} \tag{4.2.15}$$

From (4.2.6):

$$b_i = \tfrac{(y_{i+1} - y_i - c_i h^2 - d_i h^3}{h}) = \tfrac{(y_{i+1} - y_i)}{h} - c_i h - d_i h^2$$

Substitute c_i and d_i, we get

$$bi = \frac{(y_{i+1} - y_i)}{h} - \frac{h(2M_i + M_{i+1})}{6} \tag{4.2.16}$$

In (4.2.10) substitute b_i, c_i, d_i we get , Put these systems into matrix form as follow:

$$3h^2\frac{(M_i - M_{i-1})}{6h} + \frac{2hM_{i-1}}{2} + \frac{y_i - y_{i-1}}{h} - h\frac{(2M_{i-1} + M_i)}{6} = \frac{(y_{i+1} - y_i)}{h} - h\frac{(2M_i + M_{i+1})}{6}$$

$$\frac{h}{6}(M_{i-1} + 4M_i + M_{i+1}) = \frac{(y_{i-1} - 2y_i + y_{i+1})}{h}$$

$$M_{i-1} + 4M_i + M_{i+1} = 6\frac{(y_{i-1} - 2y_i + y_{i+1})}{h^2}, i = 2, 3, \ldots, n-1 \tag{4.2.17}$$

$$M_1 + 4M_2 + M_3 = 6\frac{(y_1 - 2y_2 + y_3)}{h^2}$$
$$M_2 + 4M_3 + M_4 = 6\frac{(y_2 - 2y_3 + y_4)}{h^2}$$
$$\vdots$$
$$M_{n-2} + 4M_{n-1} + M_n = 6\frac{(y_{n-2} - 2y_{n-1} + y_n)}{h^2}$$

Transform in to the matrix equation:

$$\begin{pmatrix} 1 & 4 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 4 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 4 & 1 & \cdots & 0 \\ & & & \vdots & & & \\ 0 & 0 & \cdots & 1 & 4 & 1 & 0 \\ 0 & 0 & \cdots & 0 & 1 & 4 & 1 \end{pmatrix} \begin{pmatrix} M_1 \\ M_2 \\ M_3 \\ \vdots \\ M_{n-1} \\ M_n \end{pmatrix} = \frac{6}{h^2} \begin{pmatrix} y_1 - 2y_2 + y_3 \\ y_2 - 2y_3 + y_4 \\ \vdots \\ y_{n-3} - 2y_{n-2} + y_{n-1} \\ y_{n-2} - 2y_{n-1} + y_n \end{pmatrix} \tag{4.2.18}$$

This system has n-2 rows and n columns, it is under-determined. In order to construct a unique cubic spline, two other conditions must be imposed upon the system.

4.3 Cubic spline method for solving Boundary value problems

By using cubic spline approximation equations for solving problem (1.1.1) a finite set of grid points $x_i, i = 0, 1, 2, \cdots, n-1, n$ are established by partitioning the interval $[a, b]$

into $n+1$ uniformly subinterval:
$x_i = a + ih, x_0 = a, x_n = b, h = \frac{b-a}{n+1}$.
We consider the boundary value problem defined by (1.1.1).Let $S(x)$ be the cubic spline approximating the function $y(x)$ and let $S''(x_i) = M_i$.To derive the governing equation of $S(x)$, we observe that second derivative of a third-order polynomial is a linear equation. This means that within each spline the second derivative is a linear function of x. For the $[x_{i-1}, x_i]$ interval, this linear function can be written in the Lagrange form:

$$S_i''(x) = \frac{x - x_i}{x_{i-1} - x_i} S_i''(x_{i-1}) + \frac{x - x_{i-1}}{x_i - x_{i-1}} S_i''(x_i) = \frac{1}{h}[(x_i - x)M_{i-1} + (x - x_{i-1})M_i] \quad (4.3.19)$$

where the values of the second derivative of the third-ordered polynomial at the endpoints (knots) of the ith interval are $S_i''(x_{i-1})$ and $S_i''(x_i)$. The third order polynomial in interval i can be determined by integrating Eq. (4.3.19) twice. The resulting expression contains two constants of integration. These two constants can be determined from the condition that the values of the polynomial at the knots are known: $S_i''(x_{i-1}) = y_{i-1}$ and$S_i''(x_i) = y_i$ Once the constants of integration are determined, the equation of the third-order polynomial in interval i is given by:

$$S_i(x) = \frac{1}{h}[\frac{(x_i - x)^3}{6} M_{i-1} + \frac{(x - x_{i-1})^3}{6} M_i + (y_{i-1} - \frac{h^2}{6} M_{i-1})(x_i - x) + (y_i - \frac{h^2}{6} M_i)(x - x_{i-1})] \quad (4.3.20)$$

In (4.3.20), the spline second derivativesM_i, are still not known. To determine them , we use the condition of first derivative continuity. From (4.3.20) , we obtain by differentiation:

$$S_i'(x) = \frac{1}{h}[\frac{-3(x_i - x)^2}{6} M_{i-1} + \frac{3(x - x_{i-1})^2}{6} M_i - (y_{i-1} - \frac{h^2}{6} M_{i-1}) + (y_i - \frac{h^2}{6} M_i)] \quad (4.3.21)$$

Setting $x = x_i$ in (4.3.21), we obtain the left hand derivative:

$$S_i'(x_i-) = \frac{h}{2} M_i - \frac{1}{h}(y_{i-1} - \frac{h^2}{6} M_{i-1}) + \frac{1}{h}(y_i - \frac{h^2}{6} M_i = \frac{1}{h}(y_i - y_{i-1}) + \frac{h}{6} M_{i-1} + \frac{h}{6} M_i \quad (4.3.22)$$

$i = 1, 2 \cdots, n$ and the right-hand derivative becomes:

$$S_{i+1}'(x_i+) = \frac{1}{h}(y_{i+1} - y_i) + \frac{h}{6} M_i + \frac{h}{6} M_{i+1} \quad (4.3.23)$$

$i = 0, 1, 2 \cdots, n$
Equality of(4.3.22)and(4.3.23)produces the recurrence relation:

$$M_{i-1} + 4M_i + M_{i+1} = \frac{6}{h^2}(y_{i+1} - 2y_i + y_{i-1}), i = 1, 2, \cdots, n-1 \quad (4.3.24)$$

Then, at $x = x_i$ the differential equation(1.1.1)gives

$$M_i + p_i S_i'(x_i) + q_i y_i = f_i \tag{4.3.25}$$

But from (4.3.22)and (4.3.23),we have

$$S'(x_i-) = \frac{h}{6}(2M_i + M_{i-1}) + \frac{1}{h}(y_i - y_{i-1}) \tag{4.3.26}$$

and

$$S'(x_i+) = -\frac{h}{6}(2M_i + M_{i+1}) + \frac{1}{h}(y_{i+1} - y_i) \tag{4.3.27}$$

Substituting (4.3.26)and (4.3.27) successively in (4.3.25),we obtain the equations

$$M_i + p_i[\frac{h}{6}(2M_i + M_{i-1}) + \frac{1}{h}(y_i - y_{i-1})] + q_i y_i = f_i \tag{4.3.28}$$

and

$$M_i + p_i[-\frac{h}{6}(2M_i + M_{i+1}) + \frac{1}{h}(y_{i+1} - y_i)] + q_i y_i = f_i \tag{4.3.29}$$

Simplifying (4.3.28)and (4.3.29) we get

$$a_i M_{i-1} + b_i M_i + c_i y_{i-1} + d_i y_i = f_i,\ i = 1, 2, \cdots, n. \tag{4.3.30}$$

and

$$e_i M_i + g_i M_{i+1} + h_i y_i + k_i y_{i+1} = f_i,\ i = 0, 1, \cdots, n-1. \tag{4.3.31}$$

Where

$$a_i = \frac{h}{6}p_i, b_i = 1+\frac{h}{3}p_i, c_i = -\frac{p_i}{h}, d_i = q_i+\frac{p_i}{h}, e_i = 1-\frac{h}{3}p_i, g_i = -\frac{h}{6}p_i, h_i = q_i-\frac{p_i}{h} \text{ and } k_i = \frac{p_i}{h} \tag{4.3.32}$$

Since from the boundary condition y_0 and y_n are known, equations (4.3.30)and (4.3.31) constitute a system of $2n$ equations in $2n$ unknowns,these unknowns are $M_0, M_1, \cdots, M_n, y_1, y_2, \cdots, y_{n-}$ we obtain the system of equation

$$AM = F$$

Where

$M = [M_0, M_1, \cdots, M_n, y_1, \cdots, y_{n-1}]^t$

$F = [f_1 - c_1, f_2, \cdots, f_n - d_n, f_0 - h_0, f_1, \cdots, f_{n-1} - k_{n-1}]^t$

and A is an $(2n) \times (2n)$ dimensional tri-diagonal matrix given by

$$A = \begin{pmatrix} a_1 & b_1 & 0 & 0 & \cdots & d_1 & 0 & 0 & \cdots & 0 \\ 0 & a_2 & b_2 & 0 & \cdots & c_2 & d_2 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ 0 & 0 & \cdots & 0 & a_{n-1} & b_{n-1} & 0 & \cdots & c_{n-1} & d_{n-1} \\ 0 & 0 & \cdots & 0 & 0 & a_n & b_n & 0 & \cdots & c_n \\ e_0 & g_0 & 0 & 0 & \cdots & k_0 & 0 & 0 & \cdots & 0 \\ 0 & e_1 & g_1 & 0 & \cdots & h_1 & k_1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ 0 & 0 & \cdots & 0 & e_{n-2} & g_{n-2} & 0 & \cdots & h_{n-2} & k_{n-2} \\ 0 & 0 & \cdots & 0 & 0 & e_{n-1} & g_{n-1} & 0 & \cdots & h_{n-1} \end{pmatrix}$$

4.3.1 Error analysis

Theorem 4.3.1. *If $y \in c^2[a,b], a = x_0 < x_1 < x_2 < \cdots < x_n = b$, and if $s(x)$ is the cubic spline for which*

$$s(x_i) = y_i, i = 0, 1, 2, \cdots, n$$

then

$$max_{x_0 \le x \le x_n} \mid y(x) - s(x) \mid \le \frac{1}{2} M h^2$$

where $h = x_{i+1} - x_i, i = 0, 1, 2, \cdots, n$ and $M = max \mid y''(x) \mid, x_0 \le x \le x_n$.
The errors in the spline derivatives can be obtained by using the operator notation. To find the errors in the first derivatives , we start with the recurrence relation

$$m_{i-1} + 4m_i + m_{i+1} = \frac{3}{h}(y_{i+1} - y_{i-1}).$$

That is

$$s'(x_{i-1}) + 4s'(x_i) + s'(x_{i+1}) = \frac{3}{h}(y_{i+1} - y_{i-1}).$$

Using the operator notation ,the above equation can be written as

$$(E^{-1} + 4 + E)s'(x_i) = \frac{3}{h}(E - E^{-1})y_i. \tag{4.3.33}$$

Since $E = e^{hD}$, where $D = \frac{d}{dx}$ equation (4.3.33) becomes

$$(e^{-hD} + 4 + e^{hD})s'(x_i) = \frac{3}{h}(e^{hD} - e^{-hD})y_i. \tag{4.3.34}$$

Now,

$$e^{hD} = 1 + hD + \frac{h^2D^2}{2!} + \frac{h^3D^3}{3!} + \frac{h^4D^4}{4!} + \frac{h^5D^5}{5!} + \cdots$$

and

$$e^{-hD} = 1 - hD + \frac{h^2D^2}{2!} - \frac{h^3D^3}{3!} + \frac{h^4D^4}{4!} - \frac{h^5D^5}{5!} + \cdots$$

Hence

$$e^{hD} + e^{-hD} = 2(1 + \frac{h^2D^2}{2} + \frac{h^4D^4}{24} + \frac{h^6D^6}{720} + \cdots)$$

and

$$e^{hD} - e^{-hD} = 2(hD + \frac{h^3D^3}{6} + \frac{h^5D^5}{120} + \cdots)$$

Using the above expression in (4.3.34), we obtain

$$[2(1+\frac{h^2D^2}{2}+\frac{h^4D^4}{24}+\cdots)+4]s'(x_i) = \frac{3}{h}\times 2(hD+\frac{h^3D^3}{6}+\frac{h^5D^5}{120}+\cdots)y_i = 6(D+\frac{h^2D^3}{6}+\frac{h^4D^5}{120}+\cdots)$$

The above equation simplifies to

$$\begin{aligned}
s'(x_i) &= \frac{6(D + h^2D^3/6 + h^4D^5/120 + \cdots)}{6 + h^2D^2 + h^4D^4/12 + \cdots}y_i = \frac{D + h^2D^3/6 + h^4D^5/120 + \cdots}{1 + h^2D^2/6 + h^4D^4/72 + \cdots}y_i \\
&= (D + \frac{h^2D^3}{6} + \frac{h^4D^5}{120} + \cdots)[1 + (\frac{h^2D^2}{6} + \frac{h^4D^4}{72} + \cdots)]^{-1}y_i \\
&= (D + \frac{h^2D^3}{6} + \frac{h^4D^5}{120} + \cdots)[1 - (\frac{h^2D^2}{6} + \frac{h^4D^4}{72} + \cdots) \\
&\qquad +(\frac{h^2D^2}{6} + \frac{h^4D^4}{72} + \cdots)^2 - \cdots]y_i \\
&= (D + \frac{h^2D^3}{6} + \frac{h^4D^5}{120} + \cdots)(1 - \frac{h^2D^2}{6} - \frac{h^4D^4}{72} - \cdots + \frac{h^4D^4}{36} + \cdots)y_i \\
&= (D + \frac{h^2D^3}{6} + \frac{h^4D^5}{120} + \cdots)(1 - \frac{h^2D^2}{6} + \frac{h^4D^4}{72} - \cdots)y_i \\
&= (D - \frac{h^2D^3}{6} - \frac{h^4D^5}{72} - \cdots + \frac{h^2D^3}{6} - \frac{h^4D^5}{36} + \frac{h^4D^5}{120})y_i \\
&= (D - \frac{1}{180}h^4D^5 + \cdots)y_i
\end{aligned}$$

Hence

$$s'(x_i) = y_i' - \frac{1}{180}h^4y_i^{(5)} + O(h^6). \tag{4.3.35}$$

In a similar manner , we can derive the relations:

$$s''(x_i) = y''(x_i) - \frac{1}{12}h^2y^{iv}(x_i) + \frac{1}{360}h^4y^{vi}(x_i) + O(h^6) \tag{4.3.36}$$

4.4 Cubic B-spline

The B-splines are 'non global'. These are basis functions.This basis allow the degree of the resulting curve to be changed without any change in the data.The B-spline can be of any degree but, in computer graphics and other applications , B-splines of degree 2 or 3 are generally found to be sufficient.We therefore restrict our study to a discussion of cubic B-splines only. The cubic B-spline resembles the ordinary cubic spline. In cubic spline a separate cubic is derived for each interval. Specifically,a cubic B-spline(or a B-spline of order four),denoted by $B_{i,3}(x)$, is a cubic spline with knots $x_i, x_{i+1}, x_{i+2}, x_{i+3}$ and x_{i+4}, which is zero everywhere except in the range $x_i < x < x_{i+4}$. In such a case, $B_{3,i}(x)$ said to have a support $[x_i, x_{i+4}]$. It may be noted that a B-spline need not necessarily pass through any or all of the data points. Similarly ,a B-spline of order n+1 (degree n), denoted by $B_{i,n}(x)$,is non-zero only in the range $x_i < x < x_{i+1}$.The B-spline may be defined in several ways. A useful representation is that based on divided difference.
Let the set of data points be (x_i, y_i), $i = 1, 2, \cdots, m$, and $a \leq x \leq b$. Let $s(x)$ be the cubic spline with knots $x_1, x_2, \cdots x_p$, where $a < x_1 < x_2 < \cdots < x_p < b$.

Definition 4.4.1 (**B-spline function**). *Let X be a set of $n+1$ non decreasing numbers such that $X = \{x_0, x_1, \cdots, x_n\}$ be a set of partition of $[a, b]$, x_i's are called knots, the set X the knot vector, and the half-open interval $[x_i, x_{i+1})$ the i-th knot span. the zero degree B-spline is defined as follows:*

$$B_{i,0} = \begin{cases} 1, & x_i \leq x \leq x_{i+1} \\ 0, & otherwise \end{cases}$$

The degree of these basis functions is p. The i-th B-spline basis function of degree p, written as $B_{i,p}(x)$, is defined recursively as follows:

$$B_{i,p}(x) = \frac{x - x_i}{x_{i+p} - x_i} B_{i,p-1}(x) + \frac{x_{i+p+1} - x}{x_{i+p+1} - x_{i+1}} B_{i+1,p-1}(x). \tag{4.4.37}$$

*Equation(4.4.37) is referred to as the **Cox-de Boor recursion formula**.*

Therefore, by using definition (4.4.37) and rearranging the equation will produce a

cubic B-spline basis function which represented by (4.4.38).

$$B_{i,3}(x)=\frac{1}{6h^3}\begin{cases}(x-x_i)^3, & x\in[x_i,x_{i+1}]\\ (x-x_i)^2(x_{i+2}-x)+(x-x_i)(x_{i+3}-x)(x-x_{i+1}) & \\ +(x-x_{i+1})^2(x_{i+4}-x), & x\in[x_{i+1},x_{i+2}]\\ (x-x_i)(x_{i+3}-x)^2+(x-x_{i+1})(x_{i+3}-x)(x_{i+4}-x) & \\ +(x-x_{i+2})(x_{i+4}-x)^2, & x\in[x_{i+2},x_{i+3}]\\ (x_{i+4}-x)^3, & x\in[x_{i+3},x_{i+4}]\end{cases} \tag{4.4.38}$$

The properties of B-spline functions

(1) Translation Invariance:

$$B_{i-1,p}(x)=B_{0,p}(x-(i-1)h), i=-3,-2,\cdots$$

(2)Non-negativity :

$$B_{i,p}(x)>0, x\in[x_i,x_{i+p+1})$$

(3) Compact Supported:

$$B_{i,p}(x)=0, x\notin[x_i,x_{i+p+1})$$

(4) Derivation formula:

$$B_{i,p}^{(n)}(x)=\frac{p!}{(p-k)!}\sum_{j=0}^{k}\alpha_{k,j}B_{i+j,p-k}$$

Where

$$\begin{cases}\alpha_{0,0}=1\\ \alpha_{k,0}=\frac{\alpha_{k-1,0}}{x_{i+p-k+1}-x_i}\\ \alpha_{k,k}=\frac{-\alpha_{k-1,k-1}}{x_{i+p+1}-x_{i+k}}\\ \alpha_{k,j}=\frac{\alpha_{k-j,j}-\alpha_{k-1,j-1}}{x_{i+p+j-k+1}-x_{i+j}}\end{cases}$$

4.5 Cubic B-spline method for boundary value problems (BVPs)

Let

$$y(x) = \sum_{i=-1}^{n+1} c_i B_{i,3}(x) \tag{4.5.39}$$

be an approximate solution of Eq.(1.1.1),where c_i's are unknown real coefficients and $B_{j,3}(x)$ are cubic B-spline functions. Let $x_0, x_1, \cdots, x_n$ are $n+1$ grid points in the interval $[a, b]$,so that $x_i = a + ih$, $i = 0, 1, \cdots, n$, $x_0 = a, x_n = b, h = \frac{(b-a)}{n}$.
From (4.5.39), we have

$$\begin{gathered} y(x_i) = c_{i-1}B_{i-1}(x_i) + c_iB_i(x_i) + c_{i+1}B_{i+1}(x_i) + c_{i+2}B_{i+2}(x_i), \\ y'(x_i) = c_{i-1}B'_{i-1}(x_i) + c_iB'_i(x_i) + c_{i+1}B'_{i+1}(x_i) + c_{i+2}B'_{i+2}(x_i), \\ y''(x_i) = c_{i-1}B''_{i-1}(x_i) + c_iB''_i(x_i) + c_{i+1}B''_{i+1}(x_i) + c_{i+2}B''_{i+2}(x_i), \end{gathered} \tag{4.5.40}$$

and these yield

$$\begin{aligned} &c_{i-1}[B''_{i-1}(x_i) + p(x_i)B'_{i-1}(x_i) + q(x_i)B_{i-1}(x_i)] + c_i[B''_i(x_i) + p(x_i)B'_i(x_i) + q(x_i)B_i(x_i)] \\ &+c_{i+1}[B''_{i+1}(x_i)+p(x_i)B'_{i+1}(x_i)+q(x_i)B_{i+1}(x_i)]+c_{i+2}[B''_{i+2}(x_i)+p(x_i)B'_{i+2}(x_i)+q(x_i)B_{i+2}(x_i)] = f(x_i), \end{aligned} \tag{4.5.41}$$

also by the properties of cubic B-spline functions, we obtain the following

$$\begin{gathered} B''_{i-1}(x_i) = \frac{1}{h^2}, B''_i(x_i) = -\frac{2}{h^2}, B''_{i+1}(x_i) = \frac{1}{h^2}, B''_{i+2}(x_i) = 0, \\ B'_{i-1}(x_i) = -\frac{1}{2h}, B'_i(x_i) = 0, B'_{i+1}(x_i) = \frac{1}{2h}, B'_{i+2}(x_i) = 0, \\ B_{i-1}(x_i) = \frac{1}{6}, B_i(x_i) = \frac{2}{3}, B_{i+1}(x_i) = \frac{1}{6}, B_{i+2}(x_i) = 0. \end{gathered} \tag{4.5.42}$$

If we combine (4.5.41) and (4.5.42), we obtain

$$c_{i-1}[\frac{6}{h^2} - \frac{3}{h}p(x_i) + q(x_i)] + c_i[\frac{-12}{h^2} + 4q(x_i)] + c_{i+1}[\frac{6}{h^2} + \frac{3}{h}p(x_i) + q(x_i)] \tag{4.5.43}$$

Now we apply the boundary conditions:

$$\begin{gathered} y(x_0) = c_{-1}B_{-1}(x_0) + c_0B_0(x_0) + c_1B_1(x_0) + c_2B_2(x_0) = \alpha, \\ y(x_n) = c_{n-1}B_{n-1}(x_n) + c_nB_n(x_n) + c_{n+1}B_{n+1}(x_n) + c_{n+2}B_{n+2}(x_n) = \beta. \end{gathered} \tag{4.5.44}$$

where the value of $B_i(x)$ at $x = x_0$ and $x = x_n$ are given below

$$B_{-1}(x_0) = \frac{1}{6} = B_{n-1}(x_n), B_0(x_0) = \frac{4}{6} = B_n(x_n),$$
$$B_1(x_0) = \frac{1}{6} = B_{n+1}(x_n), B_2(x_0) = 0 = B_{n+2}(x_n) \tag{4.5.45}$$

Therefore,

$$c_{-1} + 4c_0 + c_1 = 6\alpha, \tag{4.5.46}$$

$$c_{n-1} + 4c_n + c_{n+1} = 6\beta. \tag{4.5.47}$$

Now that we have found all the constant coefficients in (4.5.43), (4.5.46), and (4.5.47), we can write a system of $n+1$ linear equations in $n+1$ unknowns. This system is represented in (4.5.43) where the coefficient matrix is an $(n+1) \times (n+1)$ matrix.

$$\begin{pmatrix} 1 & 4 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ a_0(x_0) & b_0(x_0) & c_0(x_0) & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & a_1(x_1) & b_1(x_1) & c_1(x_1) & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & a_n(x_n) & b_n(x_n) & c_n(x_n) \\ 0 & 0 & 0 & 0 & 0 & \cdots & 1 & 4 & 1 \end{pmatrix} \begin{pmatrix} c_{-3} \\ c_{-2} \\ \vdots \\ \vdots \\ c_{n-1} \\ c_n \end{pmatrix} = 6 \begin{pmatrix} \alpha \\ f(x_0) \\ \vdots \\ \vdots \\ f(x_n) \\ \beta \end{pmatrix} \tag{4.5.48}$$

Equation(4.5.48) written as:

$$AB = F$$

Where $B = [c_{-3}, c_{-2}, c_{-1}, \cdots, c_{n-1}]^t$
$F = [\alpha, f(x_0), f(x_1), \cdots, f(x_n), \beta]^t$ and A is an $(n+3) \times (n+3)$ dimensional tri-diagonal matrix given by

$$A = \begin{pmatrix} 1 & 4 & 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ a_0(x_0) & b_0(x_0) & c_0(x_0) & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & a_1(x_1) & b_1(x_1) & c_1(x_1) & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & a_n(x_n) & b_n(x_n) & c_n(x_n) \\ 0 & 0 & 0 & 0 & 0 & \cdots & 1 & 4 & 1 \end{pmatrix} \tag{4.5.49}$$

also the coefficients in the matrix A have the following form

$$\begin{cases} a_i(x_i) = \frac{6}{h^2} - \frac{3}{h}p(x_i) + q(x_i), & i = 0, 1, \cdots, n \\ b_i(x_i) = \frac{-12}{h^2} + 4q(x_i), & i = 0, 1, \cdots, n \\ c_i(x_i) = \frac{6}{h^2} + \frac{3}{h}p(x_i) + q(x_i), & i = 0, 1, \cdots, n \end{cases}$$

4.5.1 Error analysis

Truncation error

Suppose $S(x)$ is the cubic B-spline interpolating $y(x_i)$, then

$$S(x) = \sum_{i=-1}^{n+1} c_i B_{i,3}(x). \tag{4.5.50}$$

Thus, the approximation at the point, x_i can be simplified to:

$$S(x_i) = c_{i-1}B_{i-1}(x_i) + c_i B_i(x_i) + c_{i+1}B_{i+1}(x_i) \approx y(x_i) \tag{4.5.51}$$

and we can easily get

$$S'(x_i) = c_{i-1}B'_{i-1}(x_i) + c_i B'_i(x_i) + c_{i+1}B'_{i+1}(x_i) \approx y'(x_i) \tag{4.5.52}$$

$$S''(x_i) = c_{i-1}B''_{i-1}(x_i) + c_i B''_i(x_i) + c_{i+1}B''_{i+1}(x_i) \approx y''(x_i), i = 0, 1, 2 \cdots, n \tag{4.5.53}$$

By substituting the blending function (4.4.38) into Eqs. (4.5.51)-(4.5.53), we have
$S(x_i) = (\frac{1}{6})c_{i-1} + (\frac{2}{3})c_i + (\frac{1}{6})c_{i+1}$
$S'(x_i) = (-\frac{1}{2h})c_{i-1} + (\frac{1}{2h})c_{i+1}$
$S''(x_i) = (\frac{1}{h^2})c_{i-1} + (-\frac{2}{h^2})c_i + (\frac{1}{h^2})c_{i+1}$
Then, the following relationships can be obtained:

$$h[(\frac{1}{6})S'_{i-1}(x_{i-1}) + (\frac{2}{3})S'_i(x_i) + (\frac{1}{6})S'_{i+1}(x_{i+1})] = \frac{1}{2}[y(x_{i+1}) - y(x_{i-1})] \tag{4.5.54}$$

$$h^2 S''(x_i) = 6[S(x_{i+1}) - S(x_i)] - 2h[2S'(x_i) + S'(x_{i+1})]. \tag{4.5.55}$$

By using the operator notation $E(S(x_i)) = S(x_{i+1})$, Eq. (4.5.54) can be written as

$$h[(\frac{1}{6})E^{-1} + (\frac{2}{3}) + (\frac{1}{6})E]S'_i(x_i) = \frac{1}{2}(E - E^{-1})y(x_i) \tag{4.5.56}$$

Since $E = e^{hD}$ where $D = d/dx$, notation E can be written in the expansion form of powers hD,

$$e^{hD} = 1 + hD + \frac{h^2D"}{2!} + \frac{h^3D^3}{3!} + \frac{h^4D^4}{4!} + \frac{h^5D^5}{5!} + \cdots$$

and

$$e^{-hD} = 1 - hD + \frac{h^2D^2}{2!} - \frac{h^3D^3}{3!} + \frac{h^4D^4}{4!} - \frac{h^5D^5}{5!} + \cdots$$

and we can get

$$e^{hD} + e^{-hD} = 2(1 + \frac{h^2D^2}{2} + \frac{h^4D^4}{24} + \frac{h^6D^6}{720} + \cdots)$$

and

$$e^{hD} - e^{-hD} = 2(hD + \frac{h^3D^3}{6} + \frac{h^5D^5}{120} + \frac{h^7D^7}{5040} + \cdots)$$

Therefore, the above Eq. (4.5.56) can be expressed as

$$h[\frac{2}{3} + \frac{1}{3}(1 + \frac{h^2D^2}{2} + \frac{h^4D^4}{24} + \frac{h^6D^6}{720} + \cdots)]s'(x_i) = (hD + \frac{h^3D^3}{6} + \frac{h^5D^5}{120} + \frac{h^7D^7}{5040} + \cdots)y(x_i)$$

Or

$$[1 + \frac{1}{3}(1 + \frac{h^2D^2}{2} + \frac{h^4D^4}{24} + \frac{h^6D^6}{720} + \cdots)]s'(x_i) = (D + \frac{h^2D^3}{6} + \frac{h^4D^5}{120} + \frac{h^6D^7}{5040} + \cdots)y(x_i)$$

And, it can be simplified into

$$\begin{aligned}
s'(x_i) &= (D + \frac{h^2D^3}{6} + \frac{h^4D^5}{120} + \cdots)[1 + (\frac{h^2D^2}{6} + \frac{h^4D^4}{72} + \frac{h^6D^6}{2160} + \cdots)]^{-1}y_i \\
&= (D + \frac{h^2D^3}{6} + \frac{h^4D^5}{120} + \cdots)[1 - (\frac{h^2D^2}{6} + \frac{h^4D^4}{72} + \frac{h^6D^6}{2160} + \cdots) \\
&\qquad + (\frac{h^2D^2}{6} + \frac{h^4D^4}{72} + \cdots)^2 - \cdots]y_i \\
&= (D + \frac{h^2D^3}{6} + \frac{h^4D^5}{120} + \cdots)(1 - \frac{h^2D^2}{6} - \frac{h^4D^4}{72} - \cdots + \frac{h^4D^4}{36} + \cdots)y_i \\
&= (D + \frac{h^2D^3}{6} + \frac{h^4D^5}{120} + \cdots)(1 - \frac{h^2D^2}{6} + \frac{h^4D^4}{72} - \frac{h^6D^6}{2160} - \cdots)y_i \\
&= (D - \frac{h^2D^3}{6} - \frac{h^4D^5}{72} - \cdots + \frac{h^2D^3}{6} - \frac{h^4D^5}{36} + \frac{h^4D^5}{120})y_i \\
&= (D - \frac{1}{180}h^4D^5 + \frac{1}{1512}h^6D^7 - \cdots)y_i
\end{aligned}$$

Hence,

$$s'(x_i) = y'_i - \frac{1}{180}h^4 y_i^{(5)} + O(h^6). \tag{4.5.57}$$

By using the same approach for Eq. (4.5.55), we can derive:

$$s''(x_i) = y''(x_i) - \frac{1}{12}h^2 y^{iv}(x_i) + \frac{1}{360}h^4 y^{vi}(x_i) + O(h^6) \tag{4.5.58}$$

4.6 Finite Difference method for boundary value problems (BVPs)

The idea of the finite difference method is to replace each derivative involved in boundary value problem in term of central difference approximations, which then leads to system of equations. The solution of the system of equations which later turned to a problem of solving a system of tridiagonal matrix, yields approximations to the solution of the original differential equations at discrete points. By considering Taylor polynomial expansion about x_i which evaluated at x_{i+1} and x_{i-1} , we have

$$y(x_{i+1}) = y(x_i + h) = y(x_i) + \frac{h}{1!}y'(x_i) + \frac{h^2}{2!}y''(x_i) + \frac{h^3}{3!}y'''(x_i) + \cdots \tag{4.6.59}$$

$$y(x_{i-1}) = y(x_i - h) = y(x_i) - \frac{h}{1!}y'(x_i) + \frac{h^2}{2!}y''(x_i) + \frac{h^3}{3!}y'''(x_i) + \cdots \tag{4.6.60}$$

By adding and subtracting (4.6.59) and (4.6.60) respectively and rearrange them in terms of y' and y'', resulting to central difference approximations for derivatives y' and y''.
That is:

$$y'(x_i) = \frac{y(x_{i+1}) - y(x_{i-1})}{2h} + O(h^2) \tag{4.6.61}$$

$$y''(x_i) = \frac{y(x_{i+1}) - 2y(x_i) + y(x_{i-1})}{h^2} + O(h^2) \tag{4.6.62}$$

Then, we get a differential equation which truncation error is (h^2) .
putting (4.6.61) and (4.6.62) into (1.1.1), then, we have get

$$\frac{1}{h^2}[y_{i-1} - 2y_i + y_{i+1}] = \frac{p(x_i)[y_{i-1} - y_{i+1}]}{2h} - q(x_i)y_i + f(x_i) \tag{4.6.63}$$

Also, combined with the boundary conditions

$$y(a) = \alpha, y(b) = \beta$$

Then we get the system of equations

$$\begin{cases} [-2+h^2q(x_1)]y_1+[1+\frac{h}{2}p(x_1)]y_2=h^2f(x_1)+[-1+\frac{h}{2}p(x_1)]\alpha, & i=1 \\ [1-\frac{h}{2}p(x_i)]y_{i-1}+[-2+h^2q(x_i)]y_i+[1+\frac{h}{2}p(x_i)]y_{i+1}=h^2f(x_i), & i=2,3\cdots,n-2 \\ [1-\frac{h}{2}p(x_n)]y_{n-2}+[-2+h^2q(x_n)]y_{n-1}=h^2f(x_n)-[1+\frac{h}{2}p(x_n)]\beta, & i=n-1 \end{cases} \tag{4.6.64}$$

Transform into matrix form

$$AY=F$$

$$\begin{pmatrix} -2+h^2q(x_1) & 1+\frac{h}{2}p(x_1) & 0 & \cdots & 0 \\ 1-\frac{h}{2}p(x_2) & \ddots & 1+\frac{h}{2}p(x_2) & \cdots & \vdots \\ 0 & [1-\frac{h}{2}p(x_3) & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 1+\frac{h}{2}p(x_{n-2}) \\ 0 & \cdots & 0 & 1-\frac{h}{2}p(x_{n-1}) & -2+h^2q(x_{n-1}) \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{n-2} \\ y_{n-1} \end{pmatrix} =$$

$$\begin{pmatrix} h^2f(x_1)+[-1+\frac{h}{2}p(x_1)]\alpha \\ h^2f(x_2) \\ \vdots \\ h^2f(x_{n-2}) \\ h^2f(x_n)-[1+\frac{h}{2}p(x_{n-1})]\beta \end{pmatrix} \tag{4.6.65}$$

4.6.1 Consistency, Stability and Convergence analysis

To study the accuracy and the computability of the difference approximation $\{y_i\}_{i=0}^{n+1}$, we introduce the concepts of consistency, stability and convergence of finite difference methods. We can write (1.1.1)as

$$Ly(x)=-y''+p(x)y'+q(x)y=f(x), x\in I=[a,b], y(a)=g_0, y(b)=g_1$$

We assume that the functions p, q and f are smooth on I, q is positive, and $p*$ and $q*$ are positive constants such that

$$\mid p(x) \mid \leq p*, 0<q* \leq q(x), x\in I. \tag{4.6.66}$$

Let $\Pi=\{x_i\}_{i=0}^{n+1}$ denote a uniform partition of the interval I.

Definition 4.6.1. *(Consistency). Let*

$$\tau_{i,\Pi}[w] \equiv L_h w(x_i) - Lw(x_i), i = 1, 2, \cdots, n$$

where w is a smooth function on I. Then the difference problem (4.6.63) is consistent with the differential problem (1.1.1) if

$$|\tau_{i,\Pi}| \to 0 \ as \ h \to 0.$$

The quantities $\tau_{i,\Pi}[w], i = 1, 2, \cdots, n$, are called the local truncation (or local discretization) errors.

Definition 4.6.2. *The difference problem (4.6.63) is locally p^{th} order accurate if, for sufficiently smooth data, there exists a positive constant C,independent of h, such that*

$$max_{1 \le i \le n} |\tau_{i,\Pi}[w]| \le Ch^p.$$

The following lemma demonstrates that the difference problem (4.6.63) is consistent with (1.1.1) and is locally secondorder accurate.

Lemma 4.6.1. *If $w \in C^4(I)$, then*

$$\tau_{i,\Pi}[w] = -\frac{h^2}{12}[w^{(4)}(\nu_i) - 2p(x_i)w^{(3)}(\theta_i)]'$$

where ν_i and θ_i lie in (x_{i-1}, x_{i+1}).

Proof. : By definition

$$\tau_{i,\Pi}[w] = -[\frac{w(x_{i+1}) - 2w(x_i) + w(x_{i-1})}{h^2} - w''(x_i)] + p_i[\frac{w(x_{i+1}) - w(x_{i-1})}{2h} - w'(x_i)], i = 1, 2, \cdots, n \tag{4.6.67}$$

using Taylor's theorem we obtain

$$\frac{w(x_{i+1}) - w(x_{i-1})}{2h} - w'(x_i) = \frac{h^2}{6} w^{(3)}(\theta_i), \theta_i \in (x_{i-1}, x_{i+1}). \tag{4.6.68}$$

Also,

$$\frac{w(x_{i+1}) - 2w(x_i) + w(x_{i-1})}{h^2} - w''(x_i) = \frac{h^2}{12} w^{(4)}(\nu_i), \nu_i \in (x_{i-1}, x_{i+1}). \tag{4.6.69}$$

Now substituting (4.6.68) and (4.6.69)in (4.6.67) we obtain the desired result. □

Definition 4.6.3. *(Stability). The linear difference operator L_h is stable if, for sufficiently small h, there exists a constant K, independent of h, such that*

$$|v_i| \leq K\{max(|v_0|, |v_{n+1}|) + max_{1\leq j\leq n}|L_h v_j|\}, i = 0, 1 \cdots, n+1,$$

for any mesh function $\{v_i\}_{i=0}^{n+1}$.

Theorem 4.6.2. *If the functions p and q satisfy (4.6.66), then the difference operator L_h of (4.6.63) is stable for $h < 2/p*$, with $K = max\{1, 1/q*\}$.*

Proof. If

$$|v_i^*| = max_{0\leq i\leq n+1}|v_i|, 1 \leq\leq i^* \leq n,$$

then, from (4.6.64), we obtain

$$d_i^* v_i^* = -e_i^* v_i^* + 1 - c_i^* v_i^* - 1 + h^2 L_h v_i * .$$

Thus

$$d_i^*|v_i^*| \leq (|e_i^*| + |c_i^*|)|v_i^*| + h^2 max_{1\leq i\leq n}|L_h v_i|.$$

If $h < 2/p^*$, then

$$d_i^* = |e_i^*| + |c_i^*| + h^2 q_{i*},$$

and it follows that

$$h^2 q_{i*}|v_i^*| \leq h^2 max_{1\leq i\leq n}|L_h v_i|,$$

or

$$|v_i^*| \leq \frac{1}{q_*} max_{1\leq i\leq n}|L_h v_i|.$$

Thus, if $max_{0\leq i\leq n+1}|v_i|$ occurs for $1 \leq i \leq n$ then

$$max_{0\leq i\leq n+1}|v_i| \leq \frac{1}{q*} max_{1\leq i\leq n}|L_h v_i|$$

and clearly

$$max_{0\leq i\leq n+1}|v_i| \leq K\{max(|v_0|, |v_{n+1}|) + max_{1\leq i\leq n}|L_h v_i|\}, \tag{4.6.70}$$

with$K = max\{1, 1/q*\}$. If $max_{0\leq i\leq n+1}|v_i| = \{max(|v_0|, |v_{n+1}|)\}$, then (4.6.70) follows immediately. □

Definition 4.6.4. *(Convergence:)Let u be the solution of the boundary value problem (1.1.1) and $\{uj\}_{i=0}^{n+1}$ the difference approximation defined by (4.6.63). The difference approximation converges to u if*

$$max_{1\leq i\leq n}|u_i - u(x_i)| \to 0, \ as \ h \to 0.$$

The difference $u_i - u(x_i)$ is the global truncation (or discretization) error at the point $x_i, i = 1, 2, \cdots, n$.

Theorem 4.6.3. *Suppose $u \in C^4(I)$ and $h < 2/p^*$. Then the difference solution $\{u_i\}_{i=0}^{n+1}$ of (4.6.63) is convergent to the solution u of (1.1.1). Moreover,*

$$max_{1\leq i\leq n}|u_i - u(x_i)| \leq Ch^2$$

Proof. Under the given conditions, the difference problem (4.6.63) is consistent with the boundary value problem (1.1.1) and the operator L_h is stable.
Since

$$L_h[u_i - u(x_i)] = f(x_i) - L_h u(x_i) = Lu(x_i) - L_h u(x_i) = -\tau_{i,\Pi}[u],$$

and $u_0 - u(x_0) = u_{n+1} - u(x_{n+1}) = 0$, the stability of L_h implies that

$$|u_i - u(x_i)| \leq \frac{1}{q_*} max_{1\leq i\leq n}|\tau_{i,\Pi}[u]|.$$

The desired result follows from Lemma (4.6.1). □

4.7 Numerical Examples and Results

In this study work two second order boundary value problems with $h = 0.1$ and $h = 0.2$ respectively, have been solved,whose exact solutions are known.the exact solution, the approximate solution and absolute errors are tabulated in tables (4.7.1)-(4.8) and comparison are shown in figures (4.7.1)-(4.6). The results of this study are compared with exact solution of the two problems and finite difference method, cubic spline method and B-spline method of the two problems.The methods are tested on second order two-point boundary value problems, for subintervals of $n = 10$. For each problem, Max-norm is calculated. Let $Y(x)$ and $S(x)$, as the analytical and approximated solutions of equation (1.1.1), respectively. The Max-norm of the approximations then, calculated based on equation (4.7.71). Matlab software is applied as the programming language in analyzing

the methods on the tested linear two-point boundary value problems.

$$Max - norm = \{max\}_{i=0}^{n} \mid S(x_i) - Y(x_i) \mid \tag{4.7.71}$$

Example 4.7.1. *Solve the following boundary value problem*

$$\begin{cases} y''(x) - y'(x) = -e^{x-1} - 1, & 0 \le x \le 1 \\ y(0) = 0, & y(1) = 0 \end{cases}$$

The analytical solution is

$$y(x) = x - xe^{x-1}$$

Solution

Method 1(cubic spline method): *by comparing the given equation with (1.1.1) we have*

$$p_i = p(x_i) = -1,\ q_i = q(x_i) = 0 \text{ and } f_i = f(x_i) = -e^{(x-1)} - 1 \tag{4.7.72}$$

and take $h = 0.1$ *then* $x_i = 0, 0.1, 0.2, 0.3, \cdots, 0.9, 1$ *where* $i = 0, 1, 2, \cdots, 10$
$f_i = f(x_i), i = 0, 1, \cdots, 10.$ *then*
$f_0 = f(0) = -1.367879441171442,\ f_1 = -1.406569659740599\ f_2 = f(0.2) = -1.449328964117222,$
$f_3 = f(0.3) = -1.496585303791409,\ f_4 = f(0.4) = -1.548811636094027,$
$f_5 = f(0.5) = -1.606530659712633, f_6 = f(0.6) = -1.670320046035639,$
$f_7 = f(0.7) = -1.740818220681718,\ f_8 = f(0.8) = -1.818730753077982,$
$f_9 = f(0.9) = -1.90483741803596$ *and* $f_{10} = f(1) = -2$
By using (4.3.32) we get
$a_i = -0.016666666666667, b_i = 0.966666666666667, c_i = 10, d_i = -10,$
$e_i = 1.033333333333333, g_i = 0.016666666666667, h_i = 10, k_i = -0.1$
substitute the above values into (4.3.30) and (4.3.31) then we get
$-0.016666666666667M_{i-1} + 0.966666666666667M_i + 10y_{i-1} - 10y_i = f_i,\ i = 1, 2, \cdots, 10.$
and
$1.033333333333333M_i + 0.016666666666667M_{i+1} + 10y_i - 0.1y_{i+1} = f_i,\ i = 0, 1, \cdots, 9.$
These two equations gives 20×20 *system of equations*
Therefore,by solving the tri-diagonal system we get

$S_1 = 0.059418338557575,\ S_2 = 0.110278103903226,\ S_3 = 0.180725204495857,$
$S_4 = 0.197014964250349,\ S_5 = 0.198097936749207,\ S_6 = 0.181700820275073,$
$S_7 = 0.181700820275073,\ S_8 = 0.145240281572363,\ S_9 = 0.085782881513176$

The analytical solutions are given by

$y(x_1) = 0.059343034025940, y(x_2) = 0.110134207176556, y(x_3) = 0.151024408862577,$
$y(x_4) = 0.180475345562389, y(x_5) = 0.196734670143683, y(x_6) = 0.197807972378616,$
$y(x_7) = 0.181427245522798, y(x_8) = 0.145015397537615, y(x_9) = 0.085646323767636.$

The error for cubic spline method are calculated as

$|y(x_1) - S_1| = 0.000075304531634908, |y(x_2) - S_2| = 0.000143896726670298,$
$|y(x_3) - S_3| = 0.000203136001628829, |y(x_4) - y_4| = 0.000249858933467573,$
$|y(x_5) - y_5| = 0.000280294106665718, |y(x_6) - y_6| = 0.000289964370590590,$
$|y(x_7) - y_7| = 0.000273574752275463, |y(x_8) - y_8| = 0.000224884034748490,$
$|y(x_9) - y_9| = 0.00013655774553962,$

Table 4.1: Cubic spline results and absolute errors for Example (4.7.1)

x_i	Cubic spline	Exact	*Absolute Error*
0.1	*0.059418338557575*	*0.059343034025940*	*0.000075304531634908*
0.2	*0.110278103903226*	*0.110134207176556*	*0.000143896726670298*
0.3	*0.180725204495857*	*0.151024408862577*	*0.000203136001628829*
0.4	*0.197014964250349*	*0.180475345562389*	*0.000249858933467573*
0.5	*0.198097936749207*	*0.196734670143683*	*0.000280294106665718*
0.6	*0.181700820275073*	*0.197807972378616*	*0.000289964370590590*
0.7	*0.181700820275073*	*0.181427245522798*	*0.000273574752275463*
0.8	*0.145240281572363*	*0.145015397537615*	*0.000224884034748490*
0.9	*0.085782881513176*	*0.085646323767636*	*0.00013655774553962*

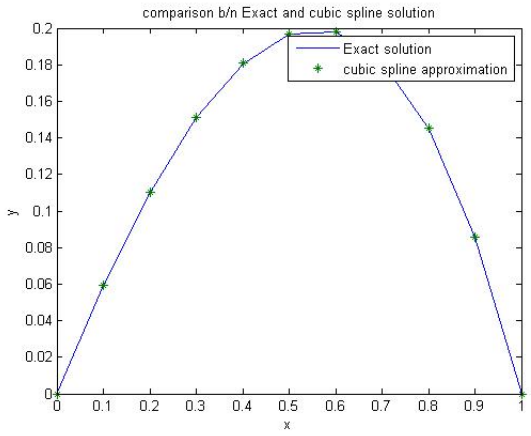

Figure 4.1: Cubic spline approximation of example (4.7.1)

Method 2(cubic B-spline method): *By using (4.7.72)and (4.5.49) we get*

$a_i(x_i) = \frac{6}{0.1^2} - \frac{3}{0.1}(-1) + 0 = 630$

$b_i(x_i) = -\frac{12}{0.1^2} + 4(0) = -1200$

$c_i(x_i) = \frac{6}{0.1^2} + \frac{3}{0.1}(-1) + 0 = 570$

we can get the coefficient matrix A for $n = 10$.

$$A = \begin{pmatrix} -1200 & 570 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 630 & -1200 & 570 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 630 & -1200 & 570 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 630 & -1200 & 570 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 630 & -1200 & 570 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 630 & -1200 & 570 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 630 & -1200 & 570 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 630 & -1200 & 570 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 630 & -1200 \end{pmatrix}$$

And

$F = [0, -8.20728, -8.43942, -8.69598, -8.97954, -9.29286, -9.63918, -10.02192, -10.44492,$
$-10.912379999999999, -11.42904, -12, 0]^T$

$B = [-0.065740019188057, 0.001224965186413, 0.060840158442406, 0.111924635199029,$
$0.153130320035298, 0.182919866433278, 0.199541891399468, 0.201002761098940,$
$0.185035090766777, 0.149062191978598, 0.090158145949558, 0.005002726654303, -0.110169052566769]$

And we can get the function

$y(x) = \sum_{j=-3}^{n-1} c_j B_{j,3}(x)$

From this we obtain

$$y(x) = \begin{cases} y_0(x) = -0.2x^3 - 0.365x^2 + 0.6325x - 0.0000166667, & x \in [0, 0.1) \\ y_1(x) = -0.233x^3 - 0.355x^2 + 0.6315x + 0.000016, & x \in [0.1, 0.2) \\ y_2(x) = -0.25x^3 - 0.3449x^2 + 0.6294x + 0.00015, & x \in [0.2, 0.3) \\ y_3(x) = -0.3x^3 - 0.3x^2 + 0.616x + 0.0015, & x \in [0.3, 0.4) \\ y_4(x) = -0.32x^3 - 0.28x^2 + 0.608x + 0.002483, & x \in [0.4, 0.5) \\ y_5(x) = -0.4x^3 - 0.155x^2 + 0.5455x + 0.0129833, & x \in [0.5, 0.6) \\ y_6(x) = -0.417x^3 - 0.1255x^2 + 0.5275x + 0.0165833, & x \in [0.6, 0.7) \\ y_7(x) = -0.467x^3 - 0.02x^2 + 0.454x + 0.033733, & x \in [0.7, 0.8) \\ y_8(x) = -0.6x^3 + 0.3x^2 + 0.198x + 0.102, & x \in [0.8, 0.9) \\ y_9(x) = -0.6x^3 + 0.3x^2 + 0.1979x + 0.102, & x \in [0.9, 1) \end{cases}$$

then from this we get

$y_1 = 0.0593827, y_2 = 0.110234, y_3 = 0.1512,$

$y_4 = 0.1806167, y_5 = 0.1969833, y_6 = 0.198083334,$

$y_7 = 0.18165523, y_8 = 0.1452, y_9 = 0.08571$

the analytical solutions are given by

$y(x_1) = 0.0593, y(x_2) = 0.1101, y(x_3) = 0.1510,$

$y(x_4) = 0.1805, y(x_5) = 0.1967, y(x_6) = 0.1978,$

$y(x_7) = 0.1814, y(x_8) = 0.1450, y(x_9) = 0.0856.$

The error for cubic b-spline method are calculated as

$|y(x_1) - y_1| = 0.0000827, |y(x_2) - y_2| = 0.000134, |y(x_3) - y_3| = 0.0002,$

$|y(x_4) - y_4| = 0.000117, |y(x_5) - y_5| = 0.0002833, |y(x_6) - y_6| = 0.00023334,$

$|y(x_7) - y_7| = 0.00025523, |y(x_8) - y_8| = 0.0002, |y(x_9) - y_9| = 0.00011,$

Table 4.2: Cubic B-spline results and absolute errors for Example (4.7.1)

x_i	Cubic B-spline	Exact	*Absolute Error*
0	*0*	*0*	*0*
0.1	*0.0593827*	*0.0593*	*0.0000827*
0.2	*0.110234*	*0.1101*	*0.000134*
0.3	*0.1512*	*0.1510*	*0.0002*
0.4	*0.1806167*	*0.1805*	*0.000117*
0.5	*0.1969833*	*0.1967*	*0.0002833*
0.6	*0.198083334*	*0.1978*	*0.00023334*
0.7	*0.198083334*	*0.1814*	*0.00025523*
0.8	*0.1452*	*0.1450*	*0.0002*
0.9	*0.08571*	*0.0856*	*0.00011*
1	*0*	*0*	*0*

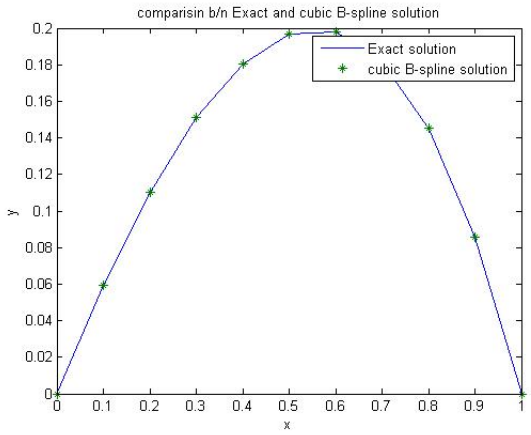

Figure 4.2: Cubic B-spline approximation of example (4.7.1)

Method 3(Finite difference method of example (4.7.1)):*To solve this BVP by FDM first discretize the domain , divide* $[0, 1]$ *into 10 equal parts then* $h = 0.1$.*Also by using (4.7.72)and (4.6.64) we obtain*

$1 - \frac{h}{2}p(x_i) = 1 - \frac{0.1}{2}(-1) = 1.05$

$-2 + h^2 q(x_i) = -2 + 0.01(0) = -2$

$1 + \frac{h}{2}p(x_i) = 1 + \frac{0.1}{2}(-1) = 0.95$

Hence, we get the coefficient matrix

$$A = \begin{pmatrix} -2 & 0.95 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1.05 & -2 & 0.95 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.05 & -2 & 0.95 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.05 & -2 & 0.95 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.05 & -2 & 0.95 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.05 & -2 & 0.95 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.05 & -2 & 0.95 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1.05 & -2 & 0.95 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.05 & -2 \end{pmatrix}$$

and

$f_i = f(x_i), i = 0, 1, \cdots, 10.$ *then*

$f_0 = f(0) = -136788,\ f_1 = -1.40657\ f_2 = f(0.2) = -144933,$

$f_3 = f(0.3) = -1.49659,\ f_4 = f(0.4) = -1.54881,$

$f_5 = f(0.5) = -1.60653, f_6 = f(0.6) = -1.67032,$

$f_7 = f(0.7) = -174082,\ f_8 = f(0.8) = -1.81873,$

$f_9 = f(0.9) = -1.90484$ *and* $f_{10} = f(1) = -2$

Then we get $F = [-0.0140657, -0.0144933, -0.0149659, -0.0154881, -0.0160653 - 0.0167032,$

$- 0.0174082, -0.0181873, -0.0190484]^t$

The system of equation become

$$\begin{pmatrix} -2 & 0.95 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1.05 & -2 & 0.95 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.05 & -2 & 0.95 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.05 & -2 & 0.95 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.05 & -2 & 0.95 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.05 & -2 & 0.95 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1.05 & -2 & 0.95 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1.05 & -2 & 0.95 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.05 & -2 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \\ y_9 \end{pmatrix} = \begin{pmatrix} -0.0140657 \\ -0.0144933 \\ -0.0149659 \\ -0.0154881 \\ -0.0160653 \\ -0.0167032 \\ -0.0174082, \\ -0.0181873 \\ -0.0190484 \end{pmatrix}$$

We solve this system of equation by using any numerical method

Therefore,by solving the above tri-diagonal system by inverse method and we get

$y_1 = 0.059384070199395, y_2 = 0.110213095156621, y_3 = 0.151136438530397,$
$y_4 = 0.180613923311939, y_5 = 0.196891038070486, y_6 = 0.197970691224669,$
$y_7 = 0.181581676289818, y_8 = 0.145143080835510, y_9 = 0.085724317438643,$

Table 4.3: Finite Difference results and absolute errors for Example (4.7.1)

x_i	Finite Difference	Exact	*Absolute Error*
0.1	*0.059384070199395*	*0.0593*	*0.00041036173454904*
0.2	*0.110213095156621*	*0.1101*	*0.000078887980065306*
0.3	*0.151136438530397*	*0.1510*	*0.000112029667819813*
0.4	*0.180613923311939*	*0.1805*	*0.00013857774954959*
0.5	*0.196891038070486*	*0.1967*	*0.000156367926802709*
0.6	*0.197970691224669*	*0.1978*	*0.000162718846052595*
0.7	*0.181581676289818*	*0.1814*	*0.000154430767020458*
0.8	*0.145143080835510*	*0.1450*	*0.000127683297895503*
0.9	*0.085724317438643*	*0.0856*	*0.000077993671006638*

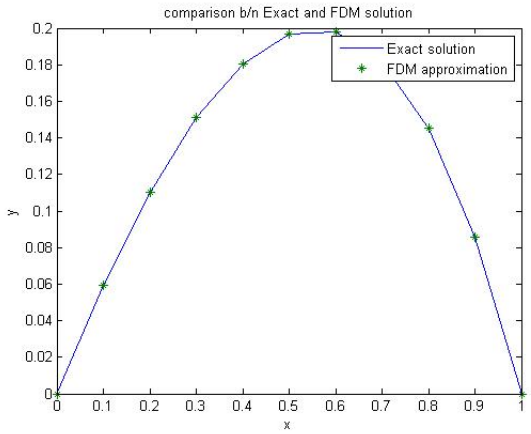

Figure 4.3: Finite difference approximation of example(4.7.1)

Table (4.4) shows the max-absolute errors of example (4.7.1) for the three methods with respect to the exact solution.

Table 4.4: max-absolute errors for Example (4.7.1)

Methods	*h*	*Max-absolute errors*
Cubic spline method	*0.1*	*0.000289964370590590*
Cubic B-spline method	*0.1*	*0.00025523*
Finite difference method	*0.1*	*0.00041036173454904*

Example 4.7.2. *Solve the following boundary value problem*

$$\begin{cases} y''(x) - \frac{4x}{1+x^2}y'(x) - \frac{2}{1+x^2}y(x) = 0, & 0 \le x \le 2 \\ y(0) = 1, & y(2) = 0.2 \end{cases} \tag{4.7.73}$$

The exact solution is given by:

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

Solution

Method 1(cubic spline method):*From (4.7.73) we have*

$$p(x) = -\frac{4x}{1+x^2}, q(x) = -\frac{2}{1+x^2} \text{ and } f(x) = 0 \tag{4.7.74}$$

Taking $h = 0.2$ *then we have* $x_i = 0, 0.2, 0.4, \cdots, 1.6, 1.8, 2.$ *where* $i = 0, 1, 2, \cdots, 10$ *using (4.7.74)we get*

$p(x_i) = -\frac{4x_i}{1+x_i^2}$

$p_0 = 0, p_1 = -0.769230769230769,\ p_2 = -1.379310344827586,$

$p_3 = -1.764705882352941,\ p_4 = -1.951219512195122,\ p_5 = -2,$

$p_6 = -1.967213114754098,\ p_7 = -1.891891891891892,$

$p_8 = -1.797752808988764,\ p_9 = -1.698113207547170,\ p_{10} = -1.6$

$q(x_i) = -\frac{2}{1+x_i^2}$

$q_0 = -2, q_1 = -1.923076923076923, q_2 = -1.724137931034483,$

$q_3 = -1.470588235294118, q_4 = -1.219512195121951, q_5 = -1,$

$q_6 = -0.819672131147541, q_7 = -0.675675675675676,$

$q_8 = -0.561797752808989, q_9 = -0.471698113207547, q_{10} = -0.4$

substitute p_i *and* q_i *in equation (4.3.32)and using (4.3.31) then we obtain* 20×20 *system of equation and we solve it by using MATLab then which gives.*

$S_1 = 0.848610272653757, S_2 = 0.743621572116475, S_3 = 0.666845068855886,$
$S_4 = 0.606242929998805, S_5 = 0.553879516416257, S_6 = 0.503580225141704,$
$S_7 = 0.449577605205405, S_8 = 0.385656753726227, S_9 = 0.304549519348779.$

We get the result below in the table with absolute error.

Table 4.5: Cubic spline results and absolute errors for Example(4.7.2)

x_i	Cubic spline	Exact	*Absolute Error*
0.2	*0.848610272653757*	*0.9615384615384613*	*0.112928188884704*
0.4	*0.743621572116475*	*0.862068965517241*	*0.118447393400766*
0.6	*0.666845068855886*	*0.735294117647059*	*0.068449048791173*
0.8	*0.606242929998805*	*0.609756097560976*	*0.003513167562171*
1	*0.553879516416257*	*0.5*	*0.053879516416257*
1.2	*0.503580225141704*	*0.409836065573771*	*0.093744159567933*
1.4	*0.449577605205405*	*0.337837837837838*	*0.111739767367567*
1.6	*0.385656753726227*	*0.280898876404494*	*0.104757877321733*
1.8	*0.304549519348779*	*0.235849056603774*	*0.068700462745005*

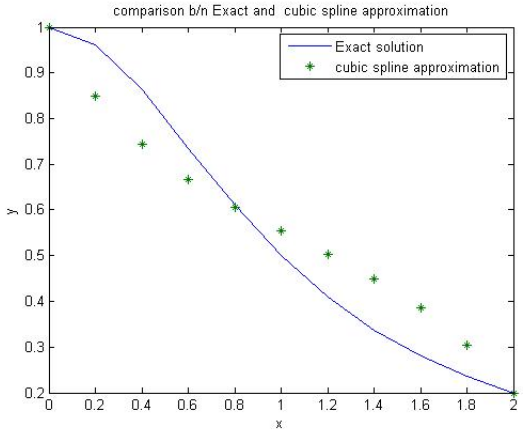

Figure 4.4: Cubic spline approximation of example (4.7.2)

Method 2(cubic B-spline method): *By using (4.7.72)and (4.7.74) we get*
$a_i(x_i) = \frac{6}{h^2} - \frac{3}{h}p(x_i) + q(x_i), i = 0, 1, \cdots, 10$
$a_0 = 148, a_1 = 159.6153846153846, a_2 = 168.9655172413793, a_3 = 175,$

$a_4 = 178.0487804878049, a_5 = 179, a_6 = 178.6885245901639,$
$a_7 = 177.7027027027027, a_8 = 176.4044943820225, a_9 = 175, a_{10} = 173.6$
$b_i(x_i) = \frac{-12}{h^2} + 4q(x_i), i = 0, 1, \cdots, 10$
$b_0 = -308, b_1 = -307.6923076923076, b_2 = -306.8965517241379, b_3 = -305.8823529411764,$
$b_4 = -304.8780487804878, b_5 = -304, b_6 = -303.2786885245901, b_7 = -302.7027027027026,$
$b_8 = -302.2471910112359, b_9 = -3.018867924528302, b_{10} = -301.6.$
$c_i(x_i) = \frac{6}{h^2} + \frac{3}{h}p(x_i) + q(x_i), i = 0, 1, \cdots, 10$
$c_0 = 148, c_1 = 136.5384615384615, c_2 = 127.5862068965517, c_3 = 122.0588235294117,$
$c_4 = 119.5121951219512, c_5 = 119, c_6 = 119.6721311475410,$
$c_7 = 120.9459459459459, c_8 = 122.4719101123595, c_9 = 124.0566037735849, c_{10} = 125.6$

we can get the coefficient matrix A for $n = 10$.

And

$F = [6, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1.2]^t$

from these we have get:

$B = [1.232374715969299, 0.986666666666667, 0.820958617364036, 0.696620358379046,$
$0.588438909321551, 0.475873138455134, 0.337308202579968, 0.145885729418837,$
$0.147918715323922, 0.155863897362921, 0.171597707199726, 0.197707325802969, 0.237572989588399]^t$

The approximation of analytical solution of the cubic B-spline at subinterval, $n = 10$, *y(x) which represented as piecewise polynomial functions.*

$$y(x) = \begin{cases} y_0(x) = 1 + 0.0012115x + x^2 + 0.35187x^3, & x \in [0.0, 0.2) \\ y_1(x) = 0.99689 + 0.047877x - 1.2333x^2 + 0.74075x^3, & x \in [0.2, 0.4) \\ y_2(x) = 1.0048 - 0.011192x - 1.0857x^2 + 0.61769x^3, & x \in [0.4, 0.6) \\ y_3(x) = 1.072 - 0.4719x - 0.52565x^2 + 0.30658x^3, & x \in [0.6, 0.8) \\ y_4(x) = 1.1909 - 0.79313x + 0.031775x^2 + 0.074315x^3, & x \in [0.8, 1) \\ y_5(x) = 1.3009 - 1.1231x + 0.36179x^2 + 0.03569x^3, & x \in [1, 1.2) \\ y_6(x) = 1.3587 - 1.2676x + 0.48214x^2 - 0.06912x^3, & x \in [1.2, 1.4) \\ y_7(x) = 1.3578 - 1.2657x + 0.48078x^2 - 0.068797x^3, & x \in [1.4, 1.6) \\ y_8(x) = 1.3119 - 1.1797x + 0.42705x^2 - 0.057602x^3, & x \in [1.6, 1.8) \\ y_9(x) = 1.2386 - 1.0575x + 0.35915x^2 - 0.045028x^3, & x \in [1.8, 2.0] \end{cases}$$

Then from this piece-wise function we get

$y_1(x_1) = 0.9630594, y_2(x_2) = 0.86614336, y_3(x_3) = 0.66584728,$

$y_4(x_4) = 0.61478128, y_5(x_5) = 0.57528, y_6(x_6) = 0.41242224,$

$y_7(x_7) = 0.339369832, y_8(x_8) = 0.2816902, y_9(x_9) = 0.236142704$

Table 4.6: Cubic B-spline results and absolute errors for Example (4.7.2)

x_i	cubic B-spline	Exact	*Absolute Error*
0.2	*0.9630594*	*0.9615384615384613*	*0.001520938461539*
0.4	*0.86614336*	*0.862068965517241*	*0.004074394482759*
0.6	*0.66584728*	*0.735294117647059*	*0.069446837647059*
0.8	*0.61478128*	*0.609756097560976*	*0.005025182439024*
1	*0.57528*	*0.5*	*0.07528*
1.2	*0.41242224*	*0.409836065573771*	*0.002586174426229*
1.4	*0.339369832*	*0.337837837837838*	*0.001531994162162*
1.6	*0.2816902*	*0.280898876404494*	*0.000791323595506*
1.8	*0.236142704*	*0.235849056603774*	*0.000293647396226*

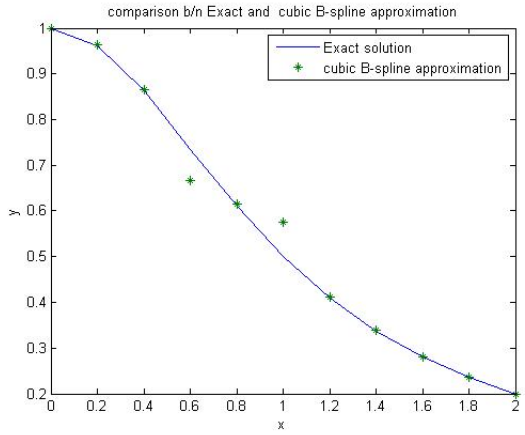

Figure 4.5: Cubic B-spline approximation for example (4.7.2)

Method 3(Finite difference method):*To solve this BVP by FDM first discretize the domain , divide* $[0, 2]$ *into 10 equal parts then* $h = 0.2$*.Also by using (4.7.74)and (4.6.64) we get*

$a_i = a(x_i) = 1 - \frac{h}{2}p(x_i), i = 1, 2, \cdots, 9$

$a_1 = 1.076923076923077, a_2 = 1.137931034482759, a_3 = 1.176470588235294,$

$a_4 = 1.195121951219512, a_5 = 1.2, a_6 = 1.19672131147541,$

$a_7 = 1.189189189189189, a_8 = 1.179775280898876, a_9 = 1.169811320754717$

$b_i = b(x_i) = -2 + h^2 q(x_i), i = 1, 2, \cdots, 9$

$b_1 = -2.076923076923077, b_2 = -2.068965517241379, b_3 = -2.058823529411765,$

$b_4 = -2.048780487804878, b_5 = -2.04, b_6 = -2.032786885245902,$

$b_7 = -2.027027027027027, b_8 = -2.022471910112360, b_9 = -2.018867924528302.$

$c_i = c(x_i) = 1 + \frac{h}{2} p(x_i), i = 1, 2, \cdots, 9$

$c_1 = 0.923076923076923, c_2 = 0.862068965517241, c_3 = 0.823529411764706,$

$c_4 = 0.804878048780488, c_5 = 0.8, c_6 = 0.80327868852459,$

$c_7 = 0.810810810810811, c_8 = 0.820224719101124, c_9 = 0.830188679245283.$ *Then we obtain a coefficient matrix A*

$F = [-1.076923076923077, 0, 0, 0, 0, 0, 0, 0, -0.1660377358490566]^t$

Therefore, from these we get the finite difference approximations:

$y_1 = 0.851122941012019, y_2 = 0.748359950610376, y_3 = 0.672581599329037,$

$y_4 = 0.612368354593484, y_5 = 0.560074042991815, y_6 = 0.5096362777389,$

$y_7 = 0.455295781657575, y_8 = 0.390772913460218, y_9 = 0.30867215546293.$

Table 4.7: Finite difference results and absolute errors for Example (4.7.2)

x_i	FDM	Exact	*Absolute Error*
0.2	*0.851122941012019*	*0.9615384615384613*	*0.110415520526442*
0.4	*0.748359950610376*	*0.862068965517241*	*0.113709014906865*
0.6	*0.672581599329037*	*0.735294117647059*	*0.062712518318022*
0.8	*0.612368354593484*	*0.609756097560976*	*0.002612257032508*
1	*0.560074042991815*	*0.5*	*0.060074042991815*
1.2	*0.5096362777389*	*0.409836065573771*	*0.099800212165129*
1.4	*0.455295781657575*	*0.337837837837838*	*0.119457943819737*
1.6	*0.390772913460218*	*0.280898876404494*	*0.109874037055724*
1.8	*0.30867215546293*	*0.235849056603774*	*0.072823098859156*

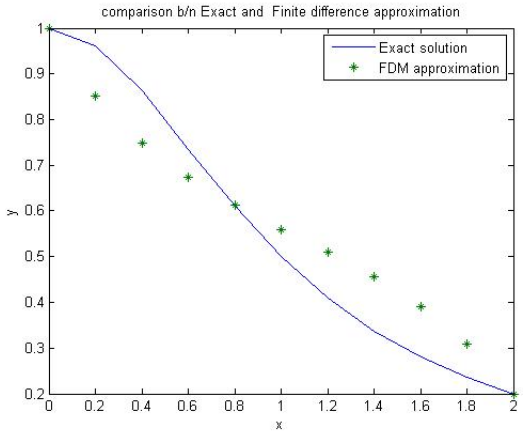

Figure 4.6: Finite difference approximation of example (4.7.2)

Table (4.8) shows the max-absolute errors of example (4.7.2) for the three methods with respect to the exact solution.

Table 4.8: max-absolute errors for Example (4.7.2)

Methods	*h*	*Max-absolute errors*
Cubic spline method	*0.2*	*0.118447393400766*
Cubic B-spline method	*0.2*	*0.07528*
Finite difference method	*0.2*	*0.119457943819737*

4.8 Discussion

We solved problems in examples(4.7.1 and 4.7.2) using the B-spline method , cubic spline method and finite difference method. The numerical results are displayed in the tables(4.1-4.3, 4.5- 4.7) . The observed maximum absolute errors associated with exact solutions, are given in tables (4.4 and 4.8).As it was shown above in tables and graphs, B-spline method produces numerical solutions that are better than those computed by the other methods. From the above discussions and examples we can conclude that B-spline method produce more accurate results than the cubic spline method and finite difference methods.

The numerical results for our examples are shown in tables, which show that there is a big difference for the errors between B-spline method and the other methods unless there is no remarkable difference among the accuracy of the other method in the case where f is sufficiently smooth.The detail of errors produced by each method at respective value of subintervals are presented in tables 4.1-4.3 and 4.5-4.7 while figure 4.1-4.6 represented comparison of each methods with exact solutions graphs. According to table 4.3 and 4.8, the approximations in examples 4.7.1 and 4.7.2 by these three methods show that B-spline method approximates better in term of max-norms and cubic spline method is also better than finite difference method as we seen in the tables. As we see from two examples (4.7.1 and 4.7.2) with $h = 0.1$ and $h = 0.2$ respectively, for example (4.7.1)the error approach to zero, whereas in example (4.7.2) the error does not approach to zero. This implies as the step size h goes to zero the approximation solution approach to the exact solution.

Chapter 5

Conclusion and Recommendation

5.1 Conclusion

Based on the numerical results, the Max-norms produced by B-spline method are always better than finite difference method and cubic spline method . One of the reasons is due to the errors produced by B-spline method are much close to zero and the resulting solution by B-spline method is constrained to satisfy the boundary conditions (1.1.1). Contrary, cubic spline method directly incorporate the boundary conditions given in the finite set of equations. The finite difference method derived based on finite difference approximations with truncation error of $O(h^2)$, was the reason on the max-norms generated by this method less accurate than those produced by cubic B-spline method. The cubic spline method can be made better in approximates by using higher approximation for y'', y' . Unfortunately, it will involve more computation to achieve higher accuracy approximation and possibility of having non-tridiagonal system of equations. Although cubic spline method is accurate, it requires a way of solving systems of equations, which is a time consuming operation especially when the subinterval is larger.

The B-spline method produced an approximation of analytical solution of the problem with respect to the selected number of subinterval, while cubic spline method and finite difference method approximate the discrete solution of the problems with respect to the selected number of subinterval. Furthermore, B-spline method required less calculations and costs. Even though the approximations by B-spline method are better than cubic spline method and finite difference method, it is noticeable based on the numerical results that as the subinterval are increasing (h smaller), the approximations by cubic spline

method are comparable to B-spline method . Therefore,cubic spline method has potential of giving good approximation solution for two-point boundary value problem.
As conclusion, these three methods are acceptable as part of approximation solutions to two-point boundary value problem. Each method are structured and approximated differently which caused the different level of accuracy on each problem tested. The main criteria of B-spline method , which is the approximation to the analytical solution of tested boundary value problems, caused this method to be the suggested method.

5.2 Recommendation

B-spline method is more stable method to approximate better than cubic spline and finite difference methods. If the two-point boundary value problems desired for approximation of analytical solutions,B-spline method has the prospective. Besides, higher order of boundary value problems should be considered in future work and the approximations of these methods should also be compared with other numerical methods in approximating solution to boundary value problems, such as finite element method, shooting method. Moreover, B-spline method is considered as among the new approach in approximating solution of two-point boundary value problems and it is improvable method. Recently, the extended B-spline interpolation method, which is simplification of B-spline method , has been studied as the improved version of B-spline method to boundary value problems .

APPENDIX

APPENDIX (1): Matlab code for example (1) cubic spline method

$h = 0.1; x = 0 : h : 1; x = 0 : h : 1;$

$alpha = 0; beta = 0; n = 10;$

$p = -1; q = 0; f = -exp(x - 1) - 1$

$a = h/6 * pb = 1 + h/3 * pc = -p/h; d = q + p/h;$

$e = 1 - h/3 * p; g = -h/6 * p; r = q - p/h; k = p/h;$

$F = zeros(1, 2 * n)$

$F(1, 1) = f(2) - c * alpha$

$F(n, 1) = f(n) - d * beta$

$F(n + 1, 1) = f(1) - r * alpha$

$F(2 * n, 1) = f(n - 1) - k * beta$

$for\ i = 2 : n$

$F(i, 1) = f(i)$

end

$for\ i = 1 : n - 1$

$F(n + i, 1) = f(i)$

end

$A = zeros(2 * n, 2 * n)$

$A(n + 2, 1) = d$

$A(n, n) = c$

$A(n + 1, n + 1) = k$

$A(2 * n, 2 * n) = r$

$for\ i = 1 : n$

$A(i, i) = a$

$A(i, i + 1) = b$

end

$for\ i = 1 : n - 1$

$A(n + i, i) = c$

$A(i, n + i + 1) = d$

end

$for\ i = 1 : n$

$A(i, n + i) = e$

$A(i + 1, n + i) = g$

end

$for\ i = 1 : n - 1$

$A(2 * n - i, 2 * n - i) = r\ A(2 * n - i, 2 * n - (i + 1)) = k$

end

$M = A\backslash F$

$y = x - x. * exp(x - 1)$

$z = [0, 0.059418338557575, 0.110278103903226, 0.151227544864206, 0.180725204495857, 0.19701496425$

$0.198097936749207, 0.181700820275073, 0.145240281572363, 0.085782881513176, 0]$

$plot(x, y,' -', x, z,' *');$

legend ('Exact solution', 'cubic spline approximation');

title ('comparison b/n Exact and FDM solution'), xlabel('x'),ylabel('y'); e=abs(z-y);

APPENDIX (1): Matlab code for example (1) cubic B-spline method

$h = 0.1$

$p = -1; q = 0;$

$a = 6/h^2 - 3./h * p + q, b = -12./h^2 + 4. * q, c = 6/h^2 + 3/h * p + q$

$A = [1410000000000; 630 - 12005700000000000; 0630 - 1200570000000000;$

$00630 - 120057000000000; 000630 - 12005700000000; 0000630 - 1200570000000;$

$00000630 - 120057000000; 000000630 - 12005700000; 0000000630 - 1200570000;$

$00000000630 - 120057000; 000000000630 - 12005700;$

$0000000000630 - 1200570; 0000000000141];$

$F = [0; -8.20728; -8.43942; -8.69598; -8.97954; -9.29286; -9.63918; -10.02192; -10.44492;$

$- 10.912379999999999; -11.42904; -12; 0];$

$C = A\backslash F$

$x = 0.1 : h : 0.9;$

$y = 1./(1 + x.^2);$

$x = 0 : 0.1 : 1;$

$y = x - x. * exp(x - 1);$

$z = [0, 0.059383, 0.110234, 0.1512, 0.180403, 0.1969833, 0.1978313, 0.181552, 0.1452, 0.08571, 0];$

$plot(x, y, '-', x, z, '*');$

legend ('Exact solution','cubic B-spline solution');

title ('comparisin b/n Exact and cubic B-spline solution'), xlabel('x'),ylabel('y');

APPENDIX (1): Matlab code for example (1) finite difference method

$\%\ y''(x) - y'(x) = -exp(x - 1) - 1, 0 \leq x \leq 1$

$\%y(0) = \alpha, y(l) = \beta$

$h = 0.2; x = 0 : h : 1; l = 1; \alpha = 0; \beta = 0 y(0) = \alpha; y(l) = \beta;$

$h = l/(n + 1);$

$x = 0 : h : l;$

$p(i) = p(x(i)) = -1); q(i) = q(x(i)) = 0; f(i) = f(x(i)) = -exp(x - 1) - 1$

$for\ i = 1 : n$

$a(i) = 1 - h./2. * p(i); d(i) = -2 + h^2. * q(i); c(i) = 1 + h./2. * p(i);$

end

% Making of the vector F

F = zeros (n , 1) ;

$F(1) = h^2 * f(1) - a(1) * \alpha; b(n) = h^2 * f(n) - c(1) * \beta;$

$for\ i = 2 : n - 1$

$F(i) = h^2 * f(i);$

end

% Making of the matrix A

A = zeros(n,n);

$A(1,1) = d(1); A(1,2) = c(1);$

$A(n, n-1) = a(n); A(n,n) = d(n);$

$for\ i = 2 : n-1$

$A(i, i-1) = a(i);$

$A(i,i) = d(i);$

$A(i, i+1) = c(i);$

end

$A = A;$

% The solution of the system

$z = A \backslash F$

% The exact solution

% $y = y(x)$

$\text{plot}(x, y, '-', x, z, '*');$

legend ('Exact solution', 'FDM approximation');

title ('comparison b/n Exact and Finite difference approximation'), xlabel('x'),ylabel('y');

APPENDIX (2): Matlab code for example (2) cubic spline method

$h = 0.2; x = 0 : 0.2 : 2;$

h=0.2; n=10; x=0:0.2:2

$p = [0, -0.769230769230769, -1.379310344827586, -1.764705882352941, -1.951219512195122, -2,$
$-1.967213114754098, -1.891891891891892, -1.797752808988764, -1.698113207547170, -1.6];$

$q = [-2, -1.923076923076923, -1.724137931034483, -1.470588235294118, -1.219512195121951,$
$-1, -0.819672131147541, -0.675675675675676, -0.561797752808989, -0.471698113207547, -0.4];$

$l = 2; n = 10 y(0) = 1; y(l) = 0.2;$

$h = 0.2; x = 0 : h : l;$

$`p(i) = p(x(i)); q(i) = q(x(i)); f(i) = f(x(i))$

% Making of the vector F

$F = zeros(1, 2 * n);$

$F(1,1) = f(1) - c(1) * \alpha; F(n,1) = f(n) - d(n) * \beta;$

$F(n+1,1) = f(n+1) - r(1) * \alpha; F(2*n,1) = f(2n) - k(n-1) * \beta;$

for $i = 2 : n-1$

$F(i,1) = f(i);$

end

% Making of the matrix A

A = zeros(2*n, 2*n);

for $i = 2 : 2n - 1$

$A(1,1) = d(1); A(1,2) = c(1);$

$A(n, n-1) = a(n); A(n,n) = d(n);$

for $i = 1 : n$

$A(i,i) = a(i);$

$A(i, i+1) = b(i);$

end for $i = 1 : n - 1 A(n+i, i) = c(i); A(n+i+1, i) = d(i);$

end for $i = 1 : n - 1 A(i, n+i) = e(i-1); A(i = 1, n+i) = g(i-1);$

end for $i = 1 : n - 1 A(2*n-i, 2*n-i) = h(i-1); A(2*n-(i+1), 2*n-i) = k(i-1);$

end

$A = A;$

% The solution of the system

$y = A \backslash F$

% The exact solution

% $y = y(x)$

$y = 1./(1 + x.^2);$

$z = [1, 0.856778685619307, 0.764461881193873, 0.706108913885861, 0.674687031267626, 0.669801210388$

$0.696396656881756, 0.764544183148095, 0.889814622958279, 1.093985233482091, 0.2];$

$plot(x, y, '-', x, z, '*');$

legend ('Exact solution', 'cubic spline approximation');

title ('comparison b/n Exact and cubic spline approximation'), xlabel('x'),ylabel('y');

$y = [0.961538461538461, 0.862068965517241, 0.735294117647059, 0.609756097560976, 0.5,$

$0.409836065573771, 0.337837837837838, 0.280898876404494, 0.235849056603774]$

$S = [0.856778685619307, 0.764461881193873, 0.706108913885861, 0.674687031267626, 0.6698012103882$

$0.696396656881756, 0.764544183148095, 0.889814622958279, 1.093985233482091]$

$E = abs(y - S);$

APPENDIX (2): Matlab code for example (2) cubic B-spline method

%MATLAB simulaion for cubic B-spline method of example 2

$h = 0.2$

$p = [0, -0.769230769230769, -1.379310344827586,$
$-1.764705882352941, -1.951219512195122, -2,$
$-1.967213114754098, -1.891891891891892,$
$-1.797752808988764, -1.698113207547170, -1.6];$
$q = [-2, -1.923076923076923, -1.724137931034483,$
$-1.470588235294118, -1.219512195121951, -1, -0.819672131147541,$
$-0.675675675675676, -0.561797752808989, -0.471698113207547, -0.4];$
$for i = 1 : 11$
$a(i) = 6./h^2 - 3./h. * p(i) + q(i);$
$b(i) = -12./h^2 + 4. * q(i);$
$c(i) = 6./h^2 + 3./h. * p(i) + q(i);$
end
$A = [1410000000000; 148 - 3081480000000000;$
$0159.6153846153846 - 307.6923076923076136.538461538461500000000;$
$00168.9655172413793 - 306.8965517241379127.586206896551700000000;$
$000175 - 305.8823529411764122.058823529411700000000; 0000178.0487804878049 - 304.878048780487811$
$00000179 - 30411900000; 00000078.6885245901639 - 303.2786885245901119.67213114754100000;$
$0000000177.7027027027027 - 302.7027027027026120.9459459459459000; 00000000176.4044943820225 -$
$302.2471910112359122.471910112359500;$
$000000000175 - 301.8867924528302124.05660377358490;$
$0000000000173.6 - 301.6125.6; 0000000000141];$
$F = [6; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 0; 1.2];$
$c = A \backslash F$
$x = 0 : 0.2 : 2;$
$y = 1./(1 + x.^2);$
$z = [1, 0.9630594, 0.86614336, 0.66584728, 0.61478128, 0.57528, 0.41242224, 0.339369832, 0.2816902, 0.23$
$\text{plot}(x, y, '-', x, z, '*');$
$\text{legend}('\text{Exact solution}', '\text{cubic B-spline approximation}');$
$\text{title}('\text{comparison b/n Exact and cubic B-spline approximation}'), \text{xlabel}('x'), \text{ylabel}('y');$
$x = 0.2 : h : 1.8;$
$y = 1./(1 + x.^2);$
$s = [0.9630594, 0.86614336, 0.66584728, 0.61478128, 0.57528, 0.41242224, 0.339369832, 0.2816902, 0.2361$
$E = abs(y - s)$

APPENDIX (2): Matlab code for example (2) finite difference method

$\%\ y''(x) + p(x)y' + q(x)y = f(x), 0 \le x \le 2$

$\%y(0) = \alpha, y(l) = \beta$

$h = 0.2; x = 0 : h : 2; l = 2; \alpha = 1; \beta = 0.2 y(0) = \alpha; y(l) = \beta;$

$h = l/(n+1);$

$x = 0 : h : l;$

$p(i) = p(x(i)) = -4x/(1+x^2); q(i) = q(x(i)) = -2/(1+x^2); f(i) = f(x(i)) = 0$

$for\ i = 1 : n$

$a(i) = 1 - h./2. * p(i); d(i) = -2 + h^2. * q(i); c(i) = 1 + h./2. * p(i);$

end

% Making of the vector F

F = zeros (n , 1) ;

$F(1) = h^2 * f(1) - a(1) * \alpha; b(n) = h^2 * f(n) - c(1) * \beta;$

$for\ i = 2 : n-1$

$F(i) = h^2 * f(i);$

end

% Making of the matrix A

A = zeros(n,n);

$A(1,1) = d(1); A(1,2) = c(1);$

$A(n, n-1) = a(n); A(n,n) = d(n);$

$for\ i = 2 : n-1$

$A(i, i-1) = a(i);$

$A(i,i) = d(i);$

$A(i, i+1) = c(i);$

end

$A = A;$

% The solution of the system

$z = A \backslash F$

% The exact solution

$\%\ y = y(x)$

$\text{plot}(x, y, '-', x, z, '*');$

legend ('Exact solution', 'FDM approximation');
title ('comparison b/n Exact and Finite difference approximation'), xlabel('x'),ylabel('y');

APPENDIX (3): Matlab program for cubic spline method

% cubic spline Method for BVPs
% $y''(x) + p(x)y' + q(x)y = f(x), 0 \leq x \leq l$
%$y(a) = \alpha, y(b) = \beta$
$y(0) = \alpha; y(l) = \beta;$
$h = l/(n+1);$
$x = 0 : h : l;$
$p(i) = p(x(i)); q(i) = q(x(i)); f(i) = f(x(i))$
$for i = 1 : n$
$a(i) = h./6. * p(i); b(i) = 1 + h./3. * p(i); c(i) = -p(i)./h;$
$d(i) = q(i)+p(i)./h; e(i) = 1-h./3.*p(i); g(i) = -h./6.*p(i); r(i) = q(i)-p(i)./h; k(i) = p(i)./h;$
end
% Making of the vector f
F = zeros (2n , 1) ;
$F(1) = f(1) - c(1) * \alpha; F(n) = f(n) - d(n) * \beta;$
$F(n+1) = f(n+1) - h(1) * \alpha; F(2n) = f(2n) - k(n-1) * \beta;$
$for\ i = 2 : n-1$
$F(i) = f(i);$
end
% Making of the matrix A
A = zeros(2n, 2n);
$for\ i = 2 : 2n-1$
$A(1,1) = d(1); A(1,2) = c(1);$
$A(n, n-1) = a(n); A(n,n) = d(n);$
$fori = 2 : n-1$
$A(i,i) = a(i);$
$A(i, i+1) = b(i);$
$A(i, n+i) = c(i); A(i, n+i+) = d(i);$
$A(n+i, i) = e(i-1); A(n+i, i+i) = g(i-1);$

$A(n+i, n+i) = h(i-1); A(n+i, n+i+1) = k(i-1);$

end

$A = A;$

% The solution of the system

$y = A \backslash F$

$ym = linspace(0, \beta, n+2);$

$ym(2 : n+1) = y$

% The exact solution

% $y = y(x)$

$y = y(x);$

APPENDIX (4): Matlab program for finite difference method

% Finite Difference Method for BVPs

% $y''(x) + p(x)y' + q(x)y = f(x), 0 \leq x \leq l$

%$y(0) = \alpha, y(b) = \beta$

$y(0) = \alpha; y(l) = \beta;$

$h = l/(n+1);$

$x = 0 : h : l;$

$p(i) = p(x(i)); q(i) = q(x(i)); f(i) = f(x(i))$

$for\ i = 1 : n$

$a(i) = 1 - h./2. * p(i); d(i) = -2 + h^2. * q(i); c(i) = 1 + h./2. * p(i);$

end

% Making of the vector F

F = zeros (n , 1) ;

$F(1) = h^2 * f(1) - a(1) * \alpha; b(n) = h^2 * f(n) - c(1) * \beta;$

$for\ i = 2 : n-1$

$F(i) = h^2 * f(i);$

end

% Making of the matrix A

A = zeros(n,n);

$A(1,1) = d(1); A(1,2) = c(1);$

$A(n, n-1) = a(n); A(n, n) = d(n);$

$for\ i = 2 : n-1$

$A(i, i-1) = a(i);$

$A(i,i) = d(i)$;

$A(i,i+1) = c(i)$;

end

$A = A$;

% The solution of the system

$z = A\backslash F$

% The exact solution

% $y = y(x)$

plot$(x, y, '-', x, z, '*')$;

legend ('Exact solution', 'FDM approximation');

title ('comparison b/n Exact and Finite difference approximation'), xlabel('x'),ylabel('y');

References

Ahmed Salem , Nur Nadiah Abd Hamid , Ahmad Izani Md. Ismail.(2016) Extended cubic Bspline method for solving a linear system of secondorder boundary value problems,School of Mathematical Sciences, University Sains Malaysia.

Al-Said, E.A.(1998). Cubic spline method for solving two-point boundary value problems, Korean J. Computational and Applied Mathematics.

Amos Gilot,Vish Subramaniam.(2014). Numerical Methods for Engineers and Scientists an Introduction with Applications using MATLAB,Third Edition,Department of Mechanical Engineering,The Ohio State University.

Curtis, F. Gerald,Patrick, O. Wheatley.(2006). Applied Numerical analysis seventh edition,California Polytechnic State University.

Doro Levy.(2010). Introduction to Numerical Analysis, Department of Mathematics and Center for Scientific Computation and Mathematical Modelling(CSCAMM) University of Maryland.

Friedman, J.Hastie, and Tibshirani.(2013). Splines and Applications: In The Elements of Statistical Learning,Bgyi Ibolya Applied Machine Learning.

Fyfe, D.J. (1968). The use of cubic splines in the solution of two-point boundary value problems, The Computer Journal.

George Wolberg.(2002). An energy-minimization framework for monotonic cubic spline interpolation, Journal of Computational and Applied Mathematics.

Grandine, T. A.(2005). The Extensive Use of Splines at Boeing, SIAM News.

Jeffrey R. Chasnov.(2012). Introduction to Numerical Methods Lecture notes,The Hong Kong University of Science and Technology.

John Bird.(2010). Higher Engineering Mathematics, Sixth edition,Amsterdam Heidelberg London.

Kai Wang.(2013). A Study of cubic spline interpolation,Riveir Academic Journal.

Khan, A.(2004). Parametric cubic spline solution of two point boundary value problems. Applied Mathematics and Computation.

M. M. chawla. C. P. katti.(1984). A finite difference method for a class of singular two point boundary value problems, IMA. J. Number. Anal.

Madhumangal Pal.(2006.) Numerical Analysis for Scientist and Engineers, Department of Applied Mathematics with Oceanology and Computer Programming Vidyasagar University.

Nazan Caglar, Hikmet Caglar.(2006). B-spline solution of singular boundary value problems, Applied Mathematics and Computation.

Nazan Caglar, Hikmet Caglar.(2009). B-spline method for solving linear system of second-order boundary value problems, Computers and Mathematics with Applications.

R.J. LeVeque . (1998). Finite Difference Methods for Differential Equations, University of Washington.

R. L. Burden, J.D. Faires.(2002). Numerical methods third edition,Brooks cole.

Rainer Kress.(1998). Graduate Text innumerical analysis,Department of mathematics San Francisco State University.

S.S.Sastry.(2006). Introductory methods of numerical analysis.2005).fourth edition,Asoke K. Ghosh,Prentice-Hall of India

Sabarina Shafie, Ahmad Abd. Majid.(2012) Approximation of Cubic B-spline Interpolation Method, Shooting and Finite Difference, Methods for Linear Problems on Solving Linear Two-Point Boundary Value Problems,World Applied Sciences Journal.

Shang Gao.(2011). Differentiation and Numerical Integral of the Cubic Spline Interpolation, Journal of computer.

Steven C. Chapra.(2010). Numerical Methods for Engineers, Sixth edition,Avenue of the Americas, New York.

Steven T. Karris.(2007). Numerical Analysis Using MATLAB and Excel, Third Edition,Orchard Publications.

Wang Ren-hong, Li Chong-jun and Zhu Chun-gang.(2008). Computational Geometry.