

**An-Najah National University**  
**Faculty of Graduate Studies**

# **Singular Two Points Boundary Value Problem**

**By**

**Sawsan Mohammad Hamdan**

**Supervised by**

**Dr. Samir Matar**

**This thesis is submitted in partial fulfillment of the requirements for the  
Degree of Master in Computational Mathematics, Faculty of Graduate  
Studies at An-Najah National University, Nablus, Palestine**

**2010**



S. Matar

## Singular Two Points Boundary Value Problem

By:

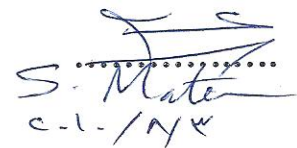
Sawsan Mohammad Hamdan

This Thesis was defended successfully on 14 \07\2010 and approved by:

### Committee Members

### Signature

1. Dr. Samir Matar (supervisor)



S. Matar  
c.l. / M<sup>W</sup>

2. Dr. Mohammad Najib Ass'ad (Internal Examiner)



3. Dr. Sae'd Mallak (External Examiner)



## **Dedication**

*To My Parents, My Husband, My children Raghad, Saif Al-deen, Mayar, My Sisters, Brothers, and to the Soul of the martyr Ihsan and to all who helped me to fulfill this thesis, I dedicate it.*

## **Acknowledgments**

First, my greatest thanks for allah for helping me finish this work as good as I hope.

Then my all thanks and wishes for my supervisor Dr. Samir Matar who directed me to finish this research successfully.

Dr. Mohammad N. Ass'ad who supported and helped me to be one of researcher students and granted me his trust.

Finally, my all thanks and wishes for my mother, father and husband especially, for their help and encouragement, and to my friends for all kinds of support and concern.

## الإقرار

أنا الموقع أدناه مقدم الرسالة التي تحمل العنوان:

# Singular Two Points Boundary Value Problem

## المعادلات التفاضلية الحدودية غير محددة القيمة

أقر بأن ما اشتملت عليه هذه الرسالة إنما هو نتاج جهدي الخاص، باستثناء ما تمت الإشارة إليه حيثما ورد، وان هذه الرسالة ككل، أو أي جزء منها لم يقدم من قبل لنيل أية درجة علمية أو بحث علمي أو بحثي لدى أية مؤسسة تعليمية أو بحثية أخرى.

## Declaration

The work provided in this thesis, unless otherwise referenced, is the researcher's own work, and has not been submitted elsewhere for any other degree or qualification.

**Student's name:**

اسم الطالب:

**Signature:**

التوقيع:

**Date:**

التاريخ:

## Table Of CONTENTS

<b>Declaration</b>	V
<b>List Of Tables</b>	VIII
<b>List of Figures</b>	IX
<b>Abstract</b>	X
<b>Chapter 1: Introduction</b>	
1.1 Introduction	2
1.2 Differential Equation	3
1.3 Ordinary Differential equations	4
1.4 Initial Value Problems	5
1.5 Boundary Value Problems (BVPs)	5
1.6 Singular BVPs	7
1.7 Previous Works	8
<b>Chapter 2: Some Numerical Methods for Solving Boundary Value Problems</b>	
2.1 Introduction	12
2.2 General Forms For The Differential Equations	13
2.3 General Forms For The Boundary Conditions	15
2.4 Type of Boundary Conditions	17
2.5 Linear Second-Order BVPs	19
2.6 Shooting Method	19
2.6.1 Shooting For Linear Problems	21
2.6.2 Shooting For Non-Linear Problems	31
2.7 Finite Difference Methods	44

2.7.1 Simple One-Step Schemes For Linear Systems	45
2.7.2 Finite Difference Method For Linear Problems	47
2.7.3 Neumann Boundary Conditions	49
2.7.4 Finite Difference Method For Nonlinear Problems	59
<b>Chapter 3: Singular Two-Points BVP</b>	
3.1 Introduction	70
3.2 Regular Singular Point, Singularities of The First Kind	70
3.3 Irregular Singular Point	74
3.4 Other Singular Problem	75
3.5 Finite Difference (Pade Based) Method	78
3.5.1 A Numerical Method Based on the (2,0) Pade Approximant	80
3.5.2 A Numerical Method Based on the (3,0) Pade Approximant	83
<b>Appendix</b>	99
<b>References</b>	136
الملخص	ب

**List of Tables**

<b>Table</b>	<b>page</b>
<b>Table (2.1)</b> : The approximate and exact solution for example 2.1.	<b>26</b>
<b>Table (2.2)</b> : The approximate and exact solution for example 2.2.	<b>29</b>
<b>Table (2.3)</b> : The approximate and exact solution for example 2.3.	<b>39</b>
<b>Table (2.4)</b> : The approximate and exact solution for example 2.4.	<b>42</b>
<b>Table (2.5)</b> : The approximate and exact solution for example 2.5.	<b>54</b>
<b>Table (2.6)</b> : The approximate and exact solution for example 2.6.	<b>57</b>
<b>Table (2.7)</b> : The approximate and exact solution for example 2.7.	<b>65</b>
<b>Table (2.8)</b> : The approximate and exact solution for example 2.8.	<b>67</b>
<b>Table (3.1)</b> : The approximate and exact solution for example 3.1.	<b>87</b>
<b>Table (3.2)</b> : The approximate and exact solution for example 3.1	<b>89</b>
<b>Table (3.3)</b> : The approximate and exact solution for example 3.1.	<b>91</b>
<b>Table (3.4)</b> : The approximate and exact solution for example 3.1.	<b>92</b>
<b>Table (3.5)</b> : The approximate and exact solution for example 3.2.	<b>97</b>



### List of Figures

<b>Figures</b>	<b>Page</b>
<b>Figure (1)</b> : shows the approximate and the exact solution for example (2.1) that was solved by shooting method.	<b>27</b>
<b>Figure(2)</b> : shows the approximate and the exact solution for example (2.2) that was solved by shooting method.	<b>30</b>
<b>Figure (3)</b> : shows the approximate and the exact solution for example (2.3) that was solved by shooting method.	<b>40</b>
<b>Figure (4)</b> : shows the approximate and the exact solution for example (2.4) that was solved by shooting method.	<b>43</b>
<b>Figure (5)</b> : shows the approximate and the exact solution for example (2.5) that was solved by finite difference method.	<b>55</b>
<b>Figure (6)</b> : shows the approximate and the exact solution for example (2.6) that was solved by finite difference method.	<b>58</b>
<b>Figure( 7)</b> : shows the approximate and the exact solution for example (2.7) that was solved by finite difference method.	<b>66</b>
<b>Figure( 8)</b> : shows the approximate and the exact solution for example (2.8) that was solved by finite difference method.	<b>68</b>
<b>Figure(9)</b> : shows the approximate and the exact solution for example (3.1) that was solved by finite difference method.	<b>88</b>
<b>Figure(10)</b> : shows the approximate and the exact solution for example (3.1) that was solved by shooting method.	<b>90</b>
<b>Figure( 11)</b> : Emden problem – BVP with singular term.	<b>96</b>

**Singular Two Points  
Boundary Value Problem  
By  
Sawsan Mohammad Hamdan  
Supervised by  
Dr. Samir Matar  
Abstract**

A singular Two points boundary value problem occur frequently in mathematical modeling of many practical problems. To solve singular two points boundary value problem for certain ordinary differential equations having singular coefficients. Many numerical method such as shooting method, finite difference method and pade approximation methods, have been studied and analysed.

# **Chapter one**

# Introduction

## 1.1 Introduction

Mathematics is the body of knowledge centered on such concepts as quantity, structure, space, and change, and also the academic discipline that studies them. Benjamin Pierce called it "the science that draws necessary conclusions".

Other practitioners of mathematics maintain that mathematics is the science of pattern, and that mathematicians seek out patterns whether found in numbers, space, science, computers, imaginary abstractions, or elsewhere.

Mathematicians explore such concepts, aiming to formulate new conjectures and establish their findings by rigorous deduction from appropriately chosen axioms and definitions.

Though the use of abstraction and logical reasoning, mathematics evolved from counting, calculation, measurement, and the systematic study of the shapes and motions of physical objects. Knowledge and use of basic mathematics have always been an inherent and integral part of individual and group life. Refinements of the basic ideas are visible in mathematical texts originating in the ancient Egyptian, Mesopotamian, Indian, Chinese, Greek and Islamic worlds. Rigorous arguments first appeared in Greek mathematics, most notably in Euclid's elements. The development continued in fitful bursts until the renaissance period of the 16<sup>th</sup> century, when mathematical innovations interacted with new scientific discoveries, leading to an acceleration in research that continues to the present day.

Today, mathematics is used throughout the world in many fields, including natural science, engineering, medicine, and the social sciences such as economics.

Applied mathematics, the application of mathematics to such fields, inspires and makes use of new mathematical discoveries and sometimes leads to the development of entirely new disciplines. See [13]

## **1.2 Differential Equation**

A differential equation is a mathematical equation for an unknown function of one or several variables that relates the values of the function itself and of its derivatives of various orders. Differential equations play a prominent role in engineering, physics, economics and other disciplines.

Differential equations arise in many areas of science and technology, whenever a deterministic relationship involving some continuously changing quantities (modeled by functions) and their rates of change (expressed as derivatives) is known or postulated. This is well illustrated by classical mechanics, where the motion of a body is described by its position and velocity as the time varies. Newton's laws allow one to relate the position, velocity, acceleration and various forces acting on the body and state this relation as a differential equation for the unknown position of the body as a function of time. In many cases, this differential equation may be solved explicitly, yielding the law of motion. See [3]&[13]

### 1.3 Ordinary Differential Equations

In mathematics, an ordinary differential equation (or ODE) is a relation that contains functions of only one independent variable, and one or more of the function's derivatives with respect to that independent variable.

A simple example is Newton's second law of motion, which leads to the differential equation

$$m \frac{d^2 x(t)}{dt^2} = F(x(t)) \quad (1.1)$$

For the motion of a particle of mass  $m$ . In general, the force  $F$  depends upon the position of the particle  $x(t)$  at time  $t$ , and thus the unknown function  $x(t)$  and its derivatives appears on both sides of the differential equation .

Ordinary differential equations are to be distinguished from partial differential equations where there are several independent variables involving partial derivatives.

Ordinary differential equations arise in many different contexts including geometry, mechanics, astronomy and population modeling. Many famous mathematicians have studied differential equations and contributed to the field, including Newton, the Bernoulli family, Reccati, Clairaut and Euler.

Many studies has been devoted to the solution of ordinary differential equations. In the case where the equation is linear, it can be solved by analytical methods, but the most of the interesting differential equations are non-linear and can't be solved exactly. Numerical methods that approximate solutions can be established by using computer. See[3]&[13]

## 1.4 Initial Value Problems

In mathematics, in the field of differential equations, an initial value problem (IVP) is an ordinary differential equation together with specified values, called the initial conditions, of the unknown function at a given point in the domain of the solution. In physics or other sciences, modeling a system frequently amounts to solving an initial value problem the differential equation is an evolution equation specifying how, given initial conditions.

A simple form of initial value problem (IVP) is a differential equation

$$y'(t) = f(t, y(t)) \quad (1.2)$$

with initial condition  $y(t_0) = y_0$ .

A solution to an initial value problem is a function  $y$  that is a solution to the differential equation and satisfies the initial condition  $y(t_0) = y_0$ .

## 1.5 Boundary Value Problems

A boundary value problems (BVP) is a differential equation together with a set of additional restrictions on the boundaries, called the boundary conditions. A solution to the boundary value problem is a solution to the differential equation which also satisfies the boundary conditions.

Boundary value problems arise in several branches of science. For example in physical differential equation for some problems involving the wave equation, such as the determination of normal modes, are often stated as boundary value problems.

To be useful in applications, a boundary value problem should be well-posed this means that given the input to the problem there exists a unique solution, which depends continuously on the input. Much theoretical work in the field of partial differential equation is devoted to proving that boundary value problems arising from scientific and engineering applications are in fact well-posed.

For a boundary value problem, information about a solution to the differential equation(s) may be generally specified at more than one point . Often there are two points, which correspond physically to the boundaries of some region, so that it is a two-points boundary value problem. A simple and common form for a two-points boundary value problem involve a second-order differential equation is:

$$y'' = f(x, y, y') \quad , \quad a \leq x \leq b \quad (1.3)$$

together with the boundary conditions

$$y(a) = \alpha \quad \text{and} \quad y(b) = \beta$$

where  $\alpha$  and  $\beta$  are known constants and the known endpoints  $a$  and  $b$  may be finite or infinite. See [1]&[3]

A more mathematical way to picture the difference between an initial value problem and a two-points boundary value problem is that (IVP) has all of the conditions specified at the same value of the independent variable in the equation ( and that value is at the lower boundary of the domain , thus the term "initial value" ).

On the other hand, a two-points boundary value problem has conditions specified at the extremes of the independent variable.



For example, if the independent variable is time  $t$  over the domain  $[0,1]$ , an initial value problem would specify a value of  $y(t)$  and / or  $y'(t)$  at time  $t = 0$ , while a two-points boundary value problem specify values for  $y(t)$  or  $y'(t)$  at both point's  $t = 0$  and  $t = 1$ . See [13]

## 1.6 Singular BVPs

Many problems in varied fields as thermodynamics, electrostatics, physics, and statistics give rise to ordinary differential equations of the form

$$-(p y')' + q y = w f$$

On some interval of the real line with some boundary conditions. Very often singularities are encountered at one or more points in that interval. Singular two-points boundary value problem occur frequently in mathematical modeling of many practical problems.

Singular point of a differential equation, a point at which the coefficients are not expandable in a Taylor series.

We mention here three examples to illustrate the point.

(1) The equation

$$-\frac{1}{\sin(\varphi)}[\Phi'(\varphi)\sin(\varphi)]' + \lambda\Phi(\varphi) = 0, \quad \varphi \in [0, \pi]$$

Appears when separation of variables is attempted on the heat equation in a solid sphere or the electrostatic potential in the sphere. The source of the singularity here is the vanishing of the function  $p$  at the endpoints.

(2) The equation

$$-((1-x^2)u')' = f(x), \quad x \in [-1,1]$$

Represents the steady state temperature distribution in a bar extending from -1 to 1 if the thermal conductivity is  $(1-x^2)$ . The same type of singularity occurs here also. See [5]

(3) An example of a class of singular BVP s is:

$$(x^\alpha y')' = f(x, y) \tag{1.4}$$

$$0 < x \leq 1, \quad y(0) = A, \quad y(1) = B$$

In which  $0 < \alpha \leq 1$  and  $A, B$  are finite constants. We assume also that for  $0 < x < 1$ , the real-valued function  $f(x, y)$  is continuous  $\frac{\partial f}{\partial y}$  exists and is continuous and that  $\frac{\partial f}{\partial y} > 0$ . See [10]

The obvious difficulty of the equation above is the behavior of the term  $y'/x$  near  $x = 0$ .

## 1.7 Previous Works

Many previous works have been done on studied numerical methods for solving singular BVPs, Gustafsson used some numerical methods that treated only scalar problems, not systems, and does not deal at all with existence or uniqueness of solutions. Natterer has treated systems, using a projection method and has get  $O(h^2 [\ln h]^r)$  accuracy. He also has dealt

with existence and uniqueness of solutions, but has used unnatural looking boundary conditions, and has not state when the problem will have a solution, only when the operator is Fredholm with index zero (not when the operator's inverse exists). Jamet also has treated only scalar equations and has used three-point finite difference schemes, which, for a model problem, with  $O(h^{1-\sigma})$  accurate solutions ( $\sigma \in (0,1)$  is a parameter of the problem). Shampine has dealt with a class of nonlinear second order scalar equations, all with the same linear differential operator. He has proved existence and uniqueness of solutions of this equation for certain boundary value problems and the convergence of collocation and finite difference methods. See [2]

[10] Twizell (1988) has developed numerical methods for this class of BVPs (1.4). Twizell's methods gave more accurate numerical results than those previously available (such as those of Chawla and Katti (1982) ). They are also more economical and easier to implement. See [10]

In this thesis we have explored some numerical methods for solving singular two-points boundary value problem and we have written some codes in matlab.

This thesis contains three chapters. Chapter 2 contains the general forms for the differential equation, and the type of boundary conditions, then we discuss a numerical methods to solve BVP, Shooting method, and Finite Difference method, for linear and nonlinear BVP.

Chapter 3 is devoted to singular two-points BVP. We discuss regular singular point, singularities of the first kind, irregular singular point, infinite interval problem, and other singular problem. Then some numerical methods were used to solve singular two-points BVP.

In this work, some numerical methods for solving these problems have been studied and analysed.

MATLAB is used as a computational tool during the development of this thesis.

## **Chapter Two**

# Some Numerical methods for Solving Boundary Value Problems

## 2.1 Introduction

A system of ordinary differential equations may have many solutions.

Commonly a solution of interest is determined by specifying the values of all its components at a single point  $x = a$ . This point and a direction of integration define an initial value problem (IVP).

In many applications the solution of interest is determined in a more complicated way. A boundary value problem (BVP) specifies values or equations for solution components at more than one point in the range of the independent variable  $x$ . Generally IVP has a unique solution, but this is not true for BVPs. Like a system of linear algebraic equations, a BVP may not have a solution at all, or may have a unique solution, or may have more than one solution. Because there might be more than one solution, BVP solvers require an estimate (guess) for the solution of interest. Often there are parameters that must be determined in order for the BVP to have a solution. Associated with a solution there might be just one set of parameters, a finite number of possible sets, or an infinite number of possible sets. See [9]

## 2.2 General Forms for the Differential Equations

For a second order non-linear BVPs we have the general form

$$y''(x) = f(x, y(x), y'(x)) \quad a \leq x \leq b$$

and the particular form that can be derived from the general one

$$y''(x) = f(x, y(x)) \quad a \leq x \leq b$$

These differential equations, valid in some interval  $[a, b]$ , together with (boundary) conditions imposed on the dependent variable and / or its first derivative at the two points  $x = a$  and  $x = b$  give rise to the second order general and special boundary value problems respectively.

For a linear boundary value problem which has the form

$$y''(x) = p(x)y' + q(x)y + r(x) \quad a \leq x \leq b, ,$$

with boundary conditions  $y(a) = \alpha, y(b) = \beta$

where  $p, q$  and  $r$  continuous functions on the interval  $[a, b]$ .

Usually one assumes that a general ordinary differential equation can be written as a first-order system

$$\mathbf{y}' = \mathbf{f}(x, \mathbf{y}) \quad a < x < b \quad (2.1)$$

where  $\mathbf{y}(x) = (y_1(x), y_2(x), \dots, y_n(x))^T$  is the unknown vector function  $\mathbf{y} \in \mathbb{R}^n$  and  $\mathbf{f}(x, \mathbf{y}) = (f_1(x, \mathbf{y}), f_2(x, \mathbf{y}), \dots, f_n(x, \mathbf{y}))^T$  is the (generally nonlinear) right-hand side. The interval ends  $a$  and  $b$  are finite or infinite constants. For a linear problem, the ODE simplifies to

$$\mathbf{y}' = \mathbf{A}(x)\mathbf{y} + \mathbf{q}(x) \quad a < x < b \quad (2.2)$$

where the matrix  $A$  and the vector  $\mathbf{q}$  are functions of  $x$ ,  $A(x) \in \mathbb{R}^{n \times n}$  and  $\mathbf{q}(x) \in \mathbb{R}^n$ . The linear system (2.2) is called homogeneous if  $\mathbf{q} = \mathbf{0}$ , and it is non-homogeneous otherwise.

High-order ODEs can normally be converted to the first-order form.

Given any scalar differential equation

$$u^{(n)} = f(x, u, u', \dots, u^{(n-1)}) \quad a < x < b \quad (2.3)$$

let  $\mathbf{y}(x) = (y_1(x), y_2(x), \dots, y_n(x))^T$  be defined by

$$\begin{aligned} y_1(x) &= u(x) \\ y_2(x) &= u'(x) \end{aligned} \quad (2.4)$$

.

.

$$y_n(x) = u^{(n-1)}(x)$$

Then the ODE can be converted to the equivalent first-order Form

$$\begin{aligned} y_1' &= y_2 \\ y_2' &= y_3 \\ &\vdots \\ &\vdots \\ y_{n-1}' &= y_n \\ y_n' &= f(x, y_1, y_2, \dots, y_n) \end{aligned}$$

This is in the form (2.1).



## 2.3 General Forms for the Boundary Conditions

A first-order system of ODEs like (2.1) has normally  $n$  boundary conditions (BCs)

$$\mathbf{g}(\mathbf{y}(a), \mathbf{y}(b)) = \mathbf{0} \quad (2.5)$$

where  $\mathbf{g} = (g_1, \dots, g_n)^T$  is a (generally nonlinear) vector function and  $\mathbf{0}$  is a vector of  $n$  zeros. The simplest instance of  $\mathbf{g}$  is the case for an IVP. Then the solution is given at the initial point; that is,

$$\mathbf{y}(a) = \boldsymbol{\alpha} \quad (2.6)$$

where  $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_n)^T \in \mathbb{R}^n$  is a known vector of initial conditions which uniquely determines  $\mathbf{y}(x)$  near  $a$ .

The general form of linear two-point BC for a first-order system (or for a higher-order ODE) is

$$\mathbf{B}_a \mathbf{y}(a) + \mathbf{B}_b \mathbf{y}(b) = \boldsymbol{\beta} \quad (2.7)$$

Here  $\mathbf{B}_a$  and  $\mathbf{B}_b \in \mathbb{R}^{n \times n}$  and  $\boldsymbol{\beta} \in \mathbb{R}^n$ .

we see that for the linear BVP (2.2) and (2.7) to have a unique solution, it is necessary but not sufficient that these BCs be linearly independent; that is, the matrix  $(\mathbf{B}_a, \mathbf{B}_b)$  have  $n$  linearly independent columns, or simply  $\text{rank}(\mathbf{B}_a, \mathbf{B}_b) = n$ .

BC of the general form (2.7) are called non-separated BC, since each involve information about  $\mathbf{y}(\mathbf{x})$  at both endpoints. However it frequently

happens that  $\text{rank}(\mathbf{B}_a) < n$  or  $\text{rank}(\mathbf{B}_b) < n$ , or both. If either holds we call the boundary condition partially separated.

In the case  $\text{rank}(\mathbf{B}_b) = q < n$ , the BVP can be transformed to one where the BC have the form

$$\mathbf{B}_{a1}\mathbf{y}(a) = \boldsymbol{\beta}_1$$

$$\mathbf{B}_{a2}\mathbf{y}(a) + \mathbf{B}_{b2}\mathbf{y}(b) = \boldsymbol{\beta}_2 \quad (2.8)$$

where  $\mathbf{B}_{a1} \in \mathbb{R}^{p \times n}$  ( $p := n - q$ ),  $\mathbf{B}_{a2}$  and  $\mathbf{B}_{b2} \in \mathbb{R}^{q \times n}$ ,  $\boldsymbol{\beta}_1 \in \mathbb{R}^p$  and  $\boldsymbol{\beta}_2 \in \mathbb{R}^q$ .

The BC are called separated if they simplify further to

$$\mathbf{B}_{a1}\mathbf{y}(a) = \boldsymbol{\beta}_1$$

$$\mathbf{B}_{b2}\mathbf{y}(b) = \boldsymbol{\beta}_2 \quad (2.9)$$

The nonlinear BC (2.5) can also occur in partially separated or separated form. Thus, the boundary conditions are separated if they are of the form

$$\mathbf{g}_1(\mathbf{y}(a)) = \mathbf{0}_1$$

$$\mathbf{g}_2(\mathbf{y}(b)) = \mathbf{0}_2 \quad (2.10)$$

where  $\mathbf{g}_1, \mathbf{0}_1 \in \mathbb{R}^p$  and  $\mathbf{g}_2, \mathbf{0}_2 \in \mathbb{R}^q$  with  $n = p + q$ .

In fact, a significant portion of the currently available software for BVPs assumes that the BC are separated. See [1]

## 2.4 Types of Boundary Conditions

For linear boundary value problems there are three types of conditions:

1. Functional boundary conditions i.e.  $y(a) = A$  and  $y(b) = B$  are given.
2. Derivative boundary conditions i.e.  $y'(a) = \alpha$  and  $y'(b) = \beta$  are given.
3. Mixed boundary condition i.e. conditions in the form

$$p_0 y(a) + q_0 y'(a) = r_0$$

$$p_1 y(b) + q_1 y'(b) = r_1$$

All three types of linear boundary conditions may be expressed in vector – matrix form as :

$$\begin{bmatrix} q_0 & p_0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} y'(a) \\ y(a) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ q_1 & p_1 \end{bmatrix} \begin{bmatrix} y'(b) \\ y(b) \end{bmatrix} = \begin{bmatrix} r_0 \\ r_1 \end{bmatrix}$$

so that  $q_0 = q_1 = 0$  gives type 1,  $p_0 = p_1 = 0$  gives type 2 and type 3 occurs when all four constants are non-zero. Type 3 boundary conditions can be written in the vector form (2.7).

### Theorem 2.1

Suppose the function  $f$  in the boundary-value problem which has the form  $y''$

$$= f(x, y, y'), \quad a \leq x \leq b$$

where  $y(a) = \alpha$ ,  $y(b) = \beta$  is continuous on the set

$$D = \{(x, y, y') \mid a \leq x \leq b, -\infty < y < \infty, -\infty < y' < \infty\}$$

and that the partial derivatives  $f_y$  and  $f_{y'}$  are also continuous on  $D$ . If

(i)  $f_y(x, y, y') > 0$  For all  $(x, y, y') \in D$  and

(ii) a constant  $M$  exists with

$$|f_{y'}(x, y, y')| \leq M \text{ for all } (x, y, y') \in D$$

Then the boundary-value problem has a unique solution. See[11]

\* Note that theorem (2.1) gives the conditions under which the general BVP with type 1 boundary condition has unique solution (existence and uniqueness).

When  $f(x, y, y')$  has the form

$$f(x, y, y') = p(x)y' + q(x)y + r(x)$$

the differential equation (1.3) is said to be linear.

Theorem (2.1) can be replaced by the following theorem:

### **Theorem 2.2**

If the linear boundary value problem:

$$y''(x) = p(x)y' + q(x)y + r(x) \tag{2.11}$$

$$a \leq x \leq b, \quad y(a) = \alpha, \quad y(b) = \beta$$

satisfies:

(i)  $p(x), q(x)$  and  $r(x)$  are continuous on  $[a, b]$

(ii)  $q(x) > 0$  on  $[a, b]$

then (2.11) has a unique solution. See [11]

## 2.5 Linear Second-Order BVP s

Consider the linear second order BVP (2.11). let  $y_0 = y$ ,  $y_1 = y'$  then (2.11) can be written as the system of first order differential equations:

$$\begin{bmatrix} y_0' \\ y_1' \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -q & -p \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \end{bmatrix} + \begin{bmatrix} 0 \\ r \end{bmatrix}$$

Which can be written in vector-matrix form as:

$$\mathbf{D} \mathbf{y} = \mathbf{Q} \mathbf{y} + \mathbf{P} \quad (2.12)$$

With boundary conditions  $y_0(0) = A$ ,  $y_1(1) = B$ . See [10]

## Numerical Methods To Solve BVP.

### 2.6 Shooting Method:

The simplest initial value method for BVPs is the single shooting method, it's one of the more successful numerical techniques for solving the general BVP with type 1 boundary conditions based on the idea of reformulating the problem as a sequence of IVPs of the form (1.3) with  $\mathbf{y}(a) = \mathbf{A}$

$$\mathbf{y}'(a) = \mathbf{z}_i, \quad i = 0, 1, \dots \quad (2.13)$$

To do this all conditions must be specified at one point. Suppose we choose to impose some initial condition, at  $t = a$ , where there are some boundary conditions are already known.

We guess the remaining boundary conditions at this point and, for the moment, ignore the known boundary conditions at  $t = b$ . We now have an IVP which can be solved using Runge Kutta or any other appropriate

method to obtain a numerical solution at  $t = b$ . These numerical values are then compared with the known boundary condition at  $t = b$ . If the guessed initial conditions are correct, there will be no discrepancies with the known boundary conditions at  $t = b$ , and the solution to the IVP will be the solution to BVP. If not, we need to modify the guessed initial conditions at  $t = a$ . This is called the shooting method, for obvious reasons. See [3]

It is probably clear to the reader that 'shooting methods' are so-called because of the analogy of firing missiles at a stationary target. Starting with the parameter  $z_0$ , which determines the initial elevation at which the missile is discharged from the point  $(x, y) = (a, A)$ . The trajectory of the missile is computed by solving the initial value problem given by (1.3) and (2.13) with  $i > 0$ . If the point of landing,  $(x, y) = (b, y(b, z_0))$  is not sufficiently close to  $(b, B)$ , the approximation is corrected by choosing another elevation  $z_1$ , and so on, until  $y(b, z_k)$  is acceptably close to the 'target'  $y(b) = B$ . See [11]

**Definition :** A function  $f(t, y)$  is said to satisfy a Lipschitz condition in the variable  $y$  on a set  $D \subset R^2$  if a constant  $L > 0$  exists with

$$|f(t, y_1) - f(t, y_2)| \leq L|y_1 - y_2|$$

Whenever  $(t, y_1), (t, y_2) \in D$ . The constant  $L$  is called a Lipschitz constant for  $f$ .

### 2.6.1 Shooting For Linear Problems

Consider the initial-value problems

$$y'' = p(x)y' + q(x)y + r(x) \quad a \leq x \leq b \quad (2.14)$$

$$y(a) = \alpha \quad (2.15)$$

$$y'(a) = 0 \quad (2.16)$$

And

$$y'' = p(x)y' + q(x)y \quad a \leq x \leq b \quad (2.17)$$

$$y(a) = 0 \quad (2.18)$$

$$y'(a) = 1 \quad (2.19)$$

If  $p, q, r$  continuous and  $q > 0$  on  $[a, b]$  then the Lipschitz condition exists and

(2.14) to (2.19) have unique solutions.

Take  $y_1(x)$  solution of (2.14) to (2.16), and  $y_2(x)$  solution of (2.17) to

(2.19), and take

$$y(x) = y_1(x) + \frac{\beta - y_1(b)}{y_2(b)} y_2(x) \quad y_2(b) \neq 0 \quad (2.20)$$

Where  $y_1(b)$  is the approximated solution for (2.14) to (2.16) at  $x = b$ ,

and  $y_2(b)$  is the approximated solution for (2.17) to (2.19) at  $x = b$ .

Can be checked to be unique solution of BVP (2.11), and

$$y'(x) = y_1'(x) + \frac{\beta - y_1(b)}{y_2(b)} y_2'(x) \quad (2.21)$$

$$y''(x) = y_1''(x) + \frac{\beta - y_1(b)}{y_2(b)} y_2''(x) \quad (2.22)$$

So

$$\begin{aligned} y'' &= p(x)y_1' + q(x)y_1 + r(x) + \frac{\beta - y_1(b)}{y_2(b)} (p(x)y_2' + q(x)y_2) \\ &= p(x)\left(y_1' + \frac{\beta - y_1(b)}{y_2(b)} y_2'\right) + q(x)\left(y_1 + \frac{\beta - y_1(b)}{y_2(b)} y_2\right) + r(x) \\ &= p(x)y'(x) + q(x)(y(x) + r(x)). \end{aligned}$$

The shooting method for linear equations is based on the replacement of the linear boundary-value problem by two initial-value problems.

See [3]&[12].



**Algorithm 2.1****Linear shooting**

To approximate the solution of the boundary-value problem

$$-y'' + p(x)y' + q(x)y + r(x) = 0. \quad a \leq x \leq b. \quad y(a) = \alpha, \quad y(b) = \beta :$$

INPUT : endpoints a, b; boundary conditions  $\alpha$ ,  $\beta$ ; number of subintervals N.

OUTPUT : approximations  $W_{1,i}$  to  $y(x_i)$ ;  $W_{2,i}$  to  $y'(x_i)$  for each  $i = 0, 1, \dots, N$ .

Step 1 Set  $h = (b-a) / N$ :

$$u_{1,0} = \alpha;$$

$$u_{2,0} = \mathbf{0};$$

$$v_{1,0} = \mathbf{0};$$

$$v_{2,0} = \mathbf{1}.$$

Step 2 For  $i = 0, \dots, N-1$  do steps 3 and 4.

(The Runge-Kutta method for systems is used in steps 3 and 4).

Step 3 Set  $x = a + ih$ .

Step 4 Set

$$k_{1,1} = hu_{2,i};$$

$$k_{1,2} = h[p(x)u_{2,i} + q(x)u_{1,i} + r(x)];$$

$$k_{2,1} = h[u_{2,i} + \frac{1}{2}k_{1,2}];$$

$$k_{2,2} = h[p(x + h/2)(u_{2,i} + \frac{1}{2}k_{1,2}) + q(x + h/2)(u_{1,i} + \frac{1}{2}k_{1,1}) + r(x + h/2)];$$

$$k_{3,1} = h[u_{2,i} + \frac{1}{2}k_{2,2}];$$

$$\begin{aligned}
k_{3,2} &= h[p(x+h/2)(u_{2,i} + \frac{1}{2}k_{2,2}) \\
&\quad + q(x+h/2)(u_{1,i} + \frac{1}{2}k_{2,1}) + r(x+h/2)]; \\
k_{4,1} &= h[u_{2,i} + k_{3,2}]; \\
k_{4,2} &= h[p(x+h)(u_{2,i} + k_{3,2}) + q(x+h)(u_{1,i} + k_{3,1}) + r(x+h)]; \\
u_{1,i+1} &= u_{1,i} + \frac{1}{6}[k_{1,1} + 2k_{2,1} + 2k_{3,1} + k_{4,1}]; \\
u_{2,i+1} &= u_{2,i} + \frac{1}{6}[k_{1,2} + 2k_{2,2} + 2k_{3,2} + k_{4,2}]; \\
k'_{1,1} &= hv_{2,i}; \\
k'_{1,2} &= h[p(x)v_{2,i} + q(x)v_{1,i}]; \\
k'_{2,1} &= h[v_{2,i} + \frac{1}{2}k'_{1,2}]; \\
k'_{2,2} &= h[p(x+h/2)(v_{2,i} + \frac{1}{2}k'_{1,2}) + q(x+h/2)(v_{1,i} + \frac{1}{2}k'_{1,1})]; \\
k'_{3,1} &= h[v_{2,i} + \frac{1}{2}k'_{2,2}]; \\
k'_{3,2} &= h[p(x+h/2)(v_{2,i} + \frac{1}{2}k'_{2,2}) + q(x+h/2)(v_{1,i} + \frac{1}{2}k'_{2,1})]; \\
k'_{4,1} &= h[v_{2,i} + k'_{3,2}]; \\
k'_{4,2} &= h[p(x+h)(v_{2,i} + k'_{3,2}) + q(x+h)(v_{1,i} + k'_{3,1})]; \\
v_{1,i+1} &= v_{1,i} + \frac{1}{6}[k'_{1,1} + 2k'_{2,1} + 2k'_{3,1} + k'_{4,1}]; \\
v_{2,i+1} &= v_{2,i} + \frac{1}{6}[k'_{1,2} + 2k'_{2,2} + 2k'_{3,2} + k'_{4,2}].
\end{aligned}$$

$$w_{1,0} = \alpha;$$

Step 5 Set  $w_{2,0} = \frac{\beta - u_{1,N}}{v_{1,N}};$

OUTPUT  $(\alpha, w_{1,0}, w_{2,0})$

Step 6 For  $i = 1, \dots, N$

$$\begin{aligned}
&W1 = u_{1,i} + w_{2,0}v_{1,i}; \\
\text{Set } &W2 = u_{2,i} + w_{2,0}v_{2,i};
\end{aligned}$$

$$x = a + ih;$$

OUTPUT  $(x, W1, W2)$ , (Output is  $\alpha, w_{1,i}, w_{2,i}$ .)

Step 7 STOP. (The process is complete.) See [3]

### Example 2.1

The linear boundary-value problem

$$y'' = -\frac{2}{x} y' + \frac{2}{x^2} y + \frac{\sin(\ln x)}{x^2}, \quad 1 \leq x \leq 2$$

$$y(1) = 1, \quad y(2) = 2$$

has the exact solution

$$y = c_1 x + \frac{c_2}{x^2} - \frac{3}{10} \sin(\ln x) - \frac{1}{10} \cos(\ln x)$$

Where

$$c_2 = \frac{1}{70} [8 - 12 \sin(\ln 2) - 4 \cos(\ln 2)] \approx -0.03920701320$$

And

$$c_1 = \frac{11}{10} - c_2 \approx 1.1392070132$$

Applying shooting method to this problem requires approximating the solutions to the initial-value problems

$$y_1'' = -\frac{2}{x} y_1' + \frac{2}{x^2} y_1 + \frac{\sin(\ln x)}{x^2}, \quad 1 \leq x \leq 2$$

$$y_1(1) = 1, \quad y_1'(1) = 0, \quad \text{and}$$

$$y_2'' = -\frac{2}{x} y_2' + \frac{2}{x^2} y_2, \quad 1 \leq x \leq 2 \quad y_2(1) = 0, \quad y_2'(1) = 1$$

Algorithm 2.1 uses the fourth-order Runge-Kutta technique to find the approximation to  $y_1(x)$  and  $y_2(x)$ .

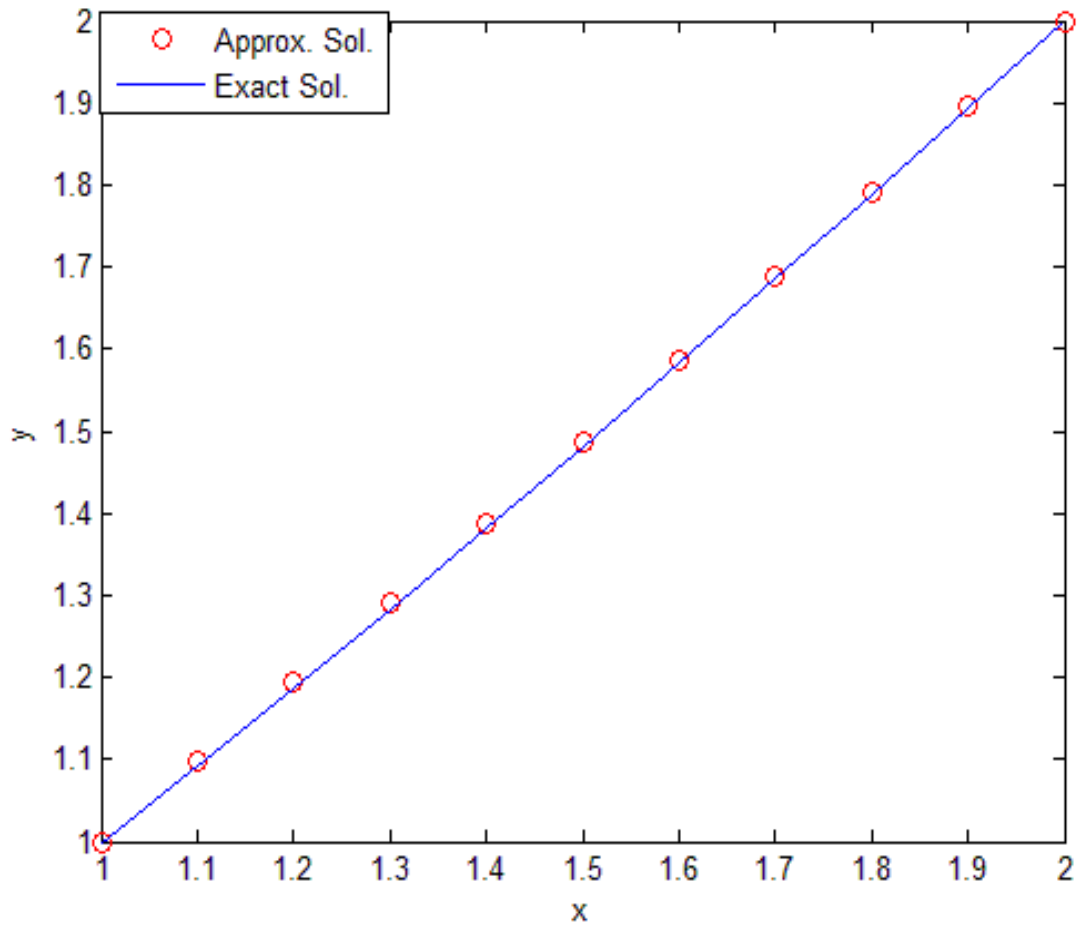
The results of the calculations with  $N = 10$  and  $h = 0.1$  are given in table (2.1). The value listed as  $w_i$  approximates and  $y(x_i)$  is the exact solution and  $ee_i$  is the error between the exact solution and the approximate solution.

$$y(x_i) = y_1(x_i) + \frac{2 - y_1(2)}{y_2(2)} y_2(x_i) . \text{ See [3] Program 1}$$

**Table (2.1) :** The approximate and exact solution for example 2.1.

$x_i$	$w_i$	$y(x_i)$	$ee_i =  y(x_i) - w_i $
1.000	1.000000	1.000000	0.000000
1.100	1.098134	1.092629	0.005505
1.200	1.194476	1.187084	0.007391
1.300	1.290881	1.283382	0.007498
1.400	1.388198	1.381445	0.006753
1.500	1.486792	1.481159	0.005633
1.600	1.586783	1.582392	0.004391
1.700	1.688171	1.685013	0.003157
1.800	1.790895	1.788898	0.001997
1.900	1.894870	1.893929	0.000941
2.000	2.000000	2.000000	0.000000

The maximum error is 0.007498.



**Figure (1)** : shows the approximate and the exact solution for example (2.1) that was solved by shooting method.

**Example 2.2**

The boundary-value problem

$$y'' = 4(y - x), \quad 0 \leq x \leq 1, \quad y(0) = 0, \quad y(1) = 1$$

has the exact solution

$$y(x) = e^2(e^4 - 1)^{-1}(e^{2x} - e^{-2x}) + x$$

Applying the shooting method to this problem requires approximating the solutions to the initial-value problems

$$y_1'' = 4(y_1 - x), \quad 0 \leq x \leq 1 \quad y_1(0) = 0 \quad y_1'(0) = 0$$

$$y_2'' = 4(y_2), \quad 0 \leq x \leq 1 \quad y_2(0) = 0 \quad y_2'(0) = 0$$

Algorithm 2.1 uses the fourth-order Runge-Kutta technique to find the approximation to  $y_1(x)$  and  $y_2(x)$ , which can be found in page 22.

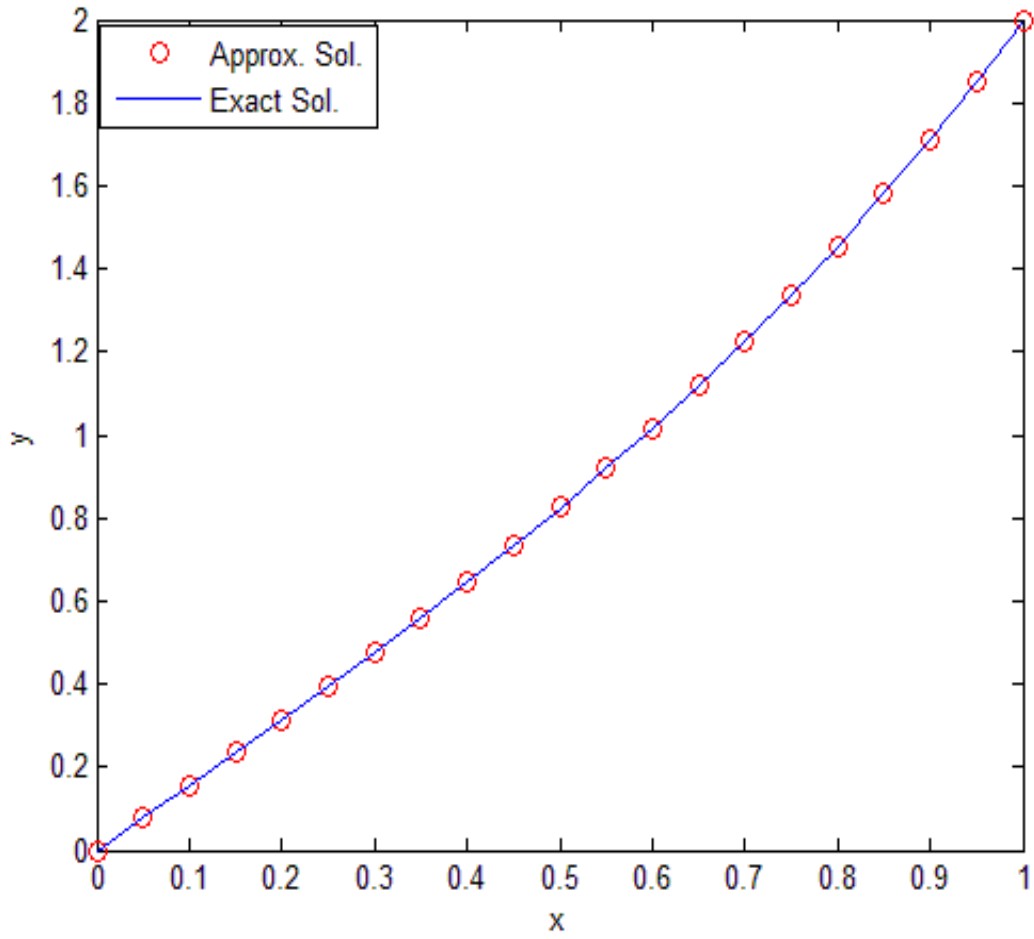
The results of the calculations with  $N = 20$  and  $h = 1/20$  are given in table (2.2). The value listed as  $w_i$  approximates and  $y(x_i)$  is the exact solution and  $ee_i$  is the error between the exact solution and the approximate solution.

$$y(x_i) = y_1(x_i) + \frac{1 - y_1(1)}{y_2(1)} y_2(x_i) . \text{ See [3] program 2}$$

**Table (2.2) :** The approximate and exact solution for example 2.2.

$x_i$	$w_i$	$y_i$	$ee_i =  y(x_i) - yy(x_i) $
0.000	0.000000	0.000000	0.000000
0.050	0.077831	0.077618	0.000213
0.100	0.155918	0.155512	0.000406
0.150	0.234541	0.233962	0.000579
0.200	0.313987	0.313252	0.000734
0.250	0.394548	0.393676	0.000872
0.300	0.476531	0.475538	0.000993
0.350	0.560254	0.559157	0.001097
0.400	0.646053	0.644869	0.001184
0.450	0.734285	0.733031	0.001253
0.500	0.825330	0.824027	0.001303
0.550	0.919596	0.918265	0.001331
0.600	1.017525	1.016189	0.001336
0.650	1.119593	1.118278	0.001314
0.700	1.226316	1.225055	0.001261
0.750	1.338260	1.337086	0.001174
0.800	1.456040	1.454992	0.001047
0.850	1.580329	1.579455	0.000874
0.900	1.711865	1.711217	0.000647
0.950	1.851459	1.851099	0.000359
1.000	2.000000	2.000000	0.000000

The maximum error is 0.001336



**Figure(2)** : shows the approximate and the exact solution for example (2.2) that was solved by shooting method.



## 2.6.2 Shooting For Non-Linear Problems

The shooting principle extends to nonlinear problems. Consider the following very simple model of a chemical reaction

$$u'' + e^u = 0 \quad 0 < x < 1 \quad (2.23)$$

$$u(0) = u(1) = 0 \quad (2.24)$$

As an initial value problem. With  $u(0) = 0$  and  $u'(0) = t$  the problem (2.23) has a unique solution. For each real  $t$ ; denoted by  $u(x, t)$ . Now if we find the correct "angle of shooting",  $t^*$  such that  $u(1; t^*) = 0$ , then the solution of the IVP also solves the BVP (2.23), (2.24).

Find  $t = t^*$  which satisfies the equation

$$u(1; t) = 0$$

This latter problem can be solved numerically by an iterative scheme. Note that each function evaluation in this iterative scheme involves the (numerical) solution of an IVP. See [1]

The shooting technique for the nonlinear second-order BVP (1.3) is similar to the linear technique, except that the solution to a nonlinear problem cannot be expressed as a linear combination to two initial value problems. Instead, the solution to the boundary value problem is approximate by using the solution to a sequence of initial value problem involving a parameter  $t$ . These problems have the form

$$y'' = f(x, y, y'), \quad a \leq x \leq b, \quad y(a) = \alpha, \quad y'(a) = t \quad (2.25)$$

We do this by choosing the parameters  $t = t_k$  in manner to ensure that

$$\lim_{k \rightarrow \infty} y(b, t_k) = y(b) = \beta$$

Where  $y(x, t_k)$  denotes the solution to the initial value problem (2.25) with  $t = t_k$  and  $y(x)$  denotes the solution to the boundary value problem (1.3).

Start with a parameter  $t_0$  that determines the initial elevation at which the object is fired from the point  $(a, \alpha)$  and along the curve described by the solution to the initial value problem:

$$y'' = f(x, y, y'), \quad a \leq x \leq b, \quad y(a) = \alpha, \quad y'(a) = t_0.$$

If  $y(b, t_0)$  is not sufficiently close to  $\beta$ , we correct our approximation by choosing elevations  $t_1, t_2$ , and so on, until  $y(b, t_k)$  is sufficiently close to  $\beta$ .

To determine the parameters  $t_k$ , suppose a boundary value problem of the form (1.3) satisfies theorem 2.1, 2.2. If  $y(x, t)$  denotes the solution to the initial value problem (2.25) we next determine  $t$  with

$$y(b, t) - \beta = 0 \tag{2.26}$$

This is a nonlinear equation that can be solved by Newton's method which use to generate the sequence  $\{ t_k \}$ , only one initial approximation,  $t_0$ , is needed.

The iteration has the form

$$t_k = t_{k-1} - \frac{y(b, t_{k-1}) - \beta}{\frac{dy}{dt}(b, t_{k-1})} \quad (2.27)$$

and it requires the knowledge of  $(dy/dt)(b, t_{k-1})$ . This presents a difficulty since an explicit representation for  $y(b, t)$  is not known; we know only the values  $y(b, t_0), y(b, t_1), \dots, y(b, t_{k-1})$ .

Suppose we rewrite the initial value problem (2.25), emphasizing that the solution depends on both  $x$  and the parameter  $t$ :

$$y''(x, t) = f(x, y(x, t), y'(x, t)), a \leq x \leq b, y(a, t) = \alpha, y'(a, t) = t \quad (2.28)$$

We have retained the prime notation to indicate differentiation with respect to  $x$ . Since we need to determine  $(dy/dt)(b, t)$  when  $t = t_{k-1}$ , we first take the partial derivative of (2.28) with respect to  $t$ . This implies that

$$\begin{aligned} \frac{dy''}{dt}(x, t) &= \frac{\partial f}{\partial t}(x, y(x, t), y'(x, t)) \\ &= \frac{\partial f}{\partial x}(x, y(x, t), y'(x, t)) \frac{\partial x}{\partial t} + \frac{\partial f}{\partial y}(x, y(x, t), y'(x, t)) \frac{\partial y}{\partial t}(x, t) \\ &\quad + \frac{\partial f}{\partial y'}(x, y(x, t), y'(x, t)) \frac{\partial y'}{\partial t}(x, t). \end{aligned}$$

Since  $x$  and  $t$  are independent,  $\partial x / \partial t = 0$  and

$$\frac{\partial y''}{\partial t}(x, t) = \frac{\partial f}{\partial y}(x, y(x, t), y'(x, t)) \frac{\partial y}{\partial t}(x, t) + \frac{\partial f}{\partial y'}(x, y(x, t), y'(x, t)) \frac{\partial y'}{\partial t}(x, t) \quad (2.29)$$

For  $a \leq x \leq b$ . This initial conditions give

$$\frac{\partial y}{\partial t}(a, t) = 0 \quad \text{and} \quad \frac{\partial y'}{\partial t}(a, t) = 1$$

If we simplify the notation by using  $z(x, t)$  to denote  $(\partial y / \partial t)(x, t)$  and assume that the order of differentiation of  $x$  and  $t$  can be reversed, (2.29) with the initial conditions becomes the initial value problem

$$z''(x, t) = \frac{\partial f}{\partial y}(x, y, y')z(x, t) + \frac{\partial f}{\partial y'}(x, y, y')z'(x, t) \quad (2.30)$$

$$a \leq x \leq b, \quad z(a, t) = 0, \quad z(b, t) = 1$$

Newton's method therefore requires that two initial value problems be solved for each iteration, (2.28) and (2.30). Then from (2.27),

$$t_k = t_{k-1} - \frac{y(b, t_{k-1}) - \beta}{z(b, t_{k-1})} \quad (2.31)$$

Of course, none of these initial value problems are solved exactly; the solution are approximated. See [3] & [6]

## Algorithm 2.2

### Nonlinear Shooting with Newton's Method

To approximate the solution of the nonlinear boundary-value problem

$$y'' = f(x, y, y'), \quad a \leq x \leq b, \quad y(a) = \alpha, \quad y(b) = \beta :$$

INPUT : endpoints  $a, b$ ; boundary conditions  $\alpha, \beta$ ; number of subintervals  $N \geq 2$ ; tolerance TOL; maximum number of iterations  $M$ .

OUTPUT : approximations  $w_{1,i}$  to  $y(x_i)$ ;  $w_{2,i}$  to  $y'(x_i)$  for each  $i = 0, 1, \dots, N$  or a message that the maximum number of iterations was exceeded.

Step 1 Set  $h = (b-a) / N$ ;

$$K = 1;$$

$$TK = (\beta - \alpha) / (b - a). \text{ (note: TK could also be input.)}$$

Step 2 While  $(k \leq M)$  do steps 3-10.

$$w_{1,0} = \alpha$$

$$w_{2,0} = TK;$$

Step 3 Set  $u_1 = 0$ ;

$$u_2 = 1.$$

Step 4 For  $i = 1, \dots, N$  do steps 5 and 6.

(The Runge-Kutta method for systems is used in steps 5 and 6.)

Step 5 Set  $x = a + (i-1)h$ .

Step 6 Set

$$k_{1,1} = hw_{2,i-1};$$

$$k_{1,2} = hf(x, w_{1,i-1}, w_{2,i-1});$$

$$k_{2,1} = h(w_{2,i-1} + \frac{1}{2}k_{1,2});$$

$$k_{2,2} = hf(x + h/2, w_{1,i-1} + \frac{1}{2}k_{1,1}, w_{2,i-1} + \frac{1}{2}k_{1,2});$$

$$k_{3,1} = h(w_{2,i-1} + \frac{1}{2}k_{2,2});$$

$$k_{3,2} = hf(x + h/2, w_{1,i-1} + \frac{1}{2}k_{2,1}, w_{2,i-1} + \frac{1}{2}k_{2,2});$$

$$k_{4,1} = h(w_{2,i-1} + k_{3,2});$$

$$k_{4,2} = hf(x + h, w_{1,i-1} + k_{3,1}, w_{2,i-1} + k_{3,2});$$

$$w_{1,i} = w_{1,i-1} + (k_{1,1} + 2k_{2,1} + 2k_{3,1} + k_{4,1})/6;$$

$$w_{2,i} = w_{2,i-1} + (k_{1,2} + 2k_{2,2} + 2k_{3,2} + k_{4,2})/6;$$

$$k'_{1,1} = hu_2;$$

$$k'_{1,2} = h[f_y(x, w_{1,i-1}, w_{2,i-1})u_1 + f_{y'}(x, w_{1,i-1}, w_{2,i-1})u_2];$$

$$k'_{2,1} = h[u_2 + \frac{1}{2}k'_{1,2}];$$

$$k'_{2,2} = h[f_y(x + h/2, w_{1,i-1}, w_{2,i-1})(u_1 + \frac{1}{2}k'_{1,1}) \\ + f_{y'}(x + h/2, w_{1,i-1}, w_{2,i-1})(u_2 + \frac{1}{2}k'_{1,2})];$$

$$k'_{3,1} = h(u_2 + \frac{1}{2}k'_{2,2});$$

$$k'_{3,2} = h[f_y(x + h/2, w_{1,i-1}, w_{2,i-1})(u_1 + \frac{1}{2}k'_{2,1}) \\ + f_{y'}(x + h/2, w_{1,i-1}, w_{2,i-1})(u_2 + \frac{1}{2}k'_{2,2})];$$

$$k'_{4,1} = h(u_2 + k'_{3,2});$$

$$k'_{4,2} = h[f_y(x + h, w_{1,i-1}, w_{2,i-1})(u_1 + k'_{3,1}) \\ + f_{y'}(x + h, w_{1,i-1}, w_{2,i-1})(u_2 + k'_{3,2})];$$

$$u_1 = u_1 + \frac{1}{6}[k'_{1,1} + 2k'_{2,1} + 2k'_{3,1} + k'_{4,1}];$$

$$u_2 = u_2 + \frac{1}{6}[k'_{1,2} + 2k'_{2,2} + 2k'_{3,2} + k'_{4,2}].$$

Step 7 If  $|w_{1,N} - \beta| \leq \text{TOL}$  then do steps 8 and 9.

Step 8 For  $i = 0, 1, \dots, N$

set  $x = a + ih$ ;

OUTPUT  $(x, w_{1,i}, w_{2,i})$ .

Step 9 (The procedure is complete.)

STOP.

Step 10 Set  $TK = TK - \frac{w_{1,N} - \beta}{u_1}$

(Newton's method is used to compute TK.)

$k = k+1$ .

Step 11 OUTPUT ('Maximum number of iterations exceeded');

(The procedure was unsuccessful.)

STOP. See[3]

**Example 2.3**

Consider the boundary-value problem

$$y'' = \frac{1}{8}(32 + 2x^3 - yy'), \quad 1 \leq x \leq 3, \quad y(1) = 17, \quad y(3) = \frac{43}{3}$$

has the exact solution

$$y(x) = x^2 + \frac{16}{x}$$

Applying shooting method to this problem requires approximating the solutions to the initial-value problems

$$y'' = \frac{1}{8}(32 + 2x^3 - yy'), \quad 1 \leq x \leq 3, \quad y(1) = 17, \quad y'(1) = t_k$$

$$z'' = -\frac{1}{8}(y'z + yz'), \quad 1 \leq x \leq 3, \quad z(1) = 0, \quad z'(1) = 1$$

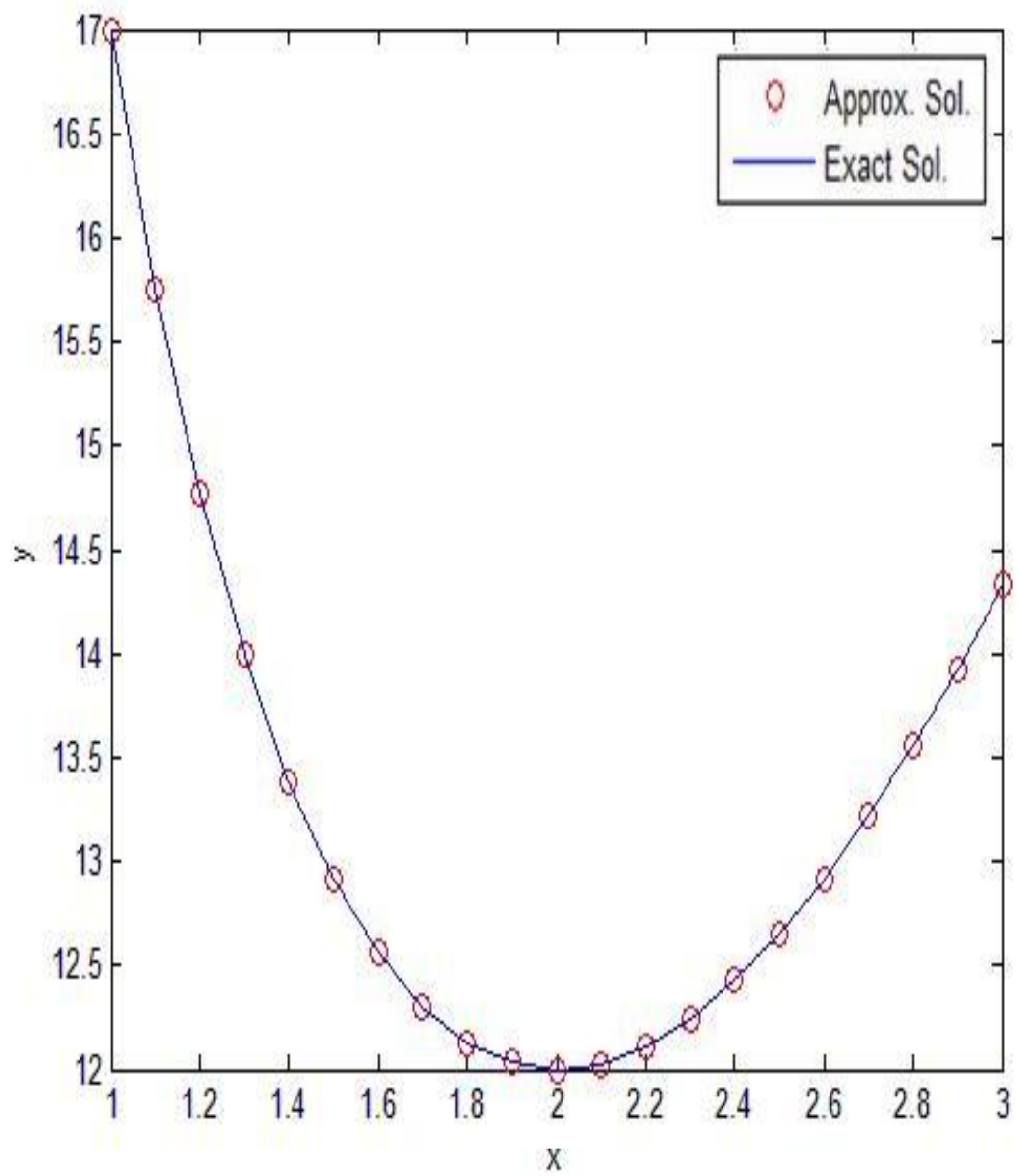
Algorithm 2.2 uses the Runge-Kutta method of order four to approximate both solutions required by Newton's method. The value listed as  $w_i$  approximates  $y(x_i)$  is the exact solution and  $ee_i$  is the error between the exact solution and the approximate solution. The results of the calculations with  $N = 20$  are given in table (2.3), See [3] program 3



**Table (2.3) :** The approximate and exact solution for example 2.3.

$x_i$	$w_{1,i}$	$y(x_i)$	$ee_i =  w_{1,i} - y(x_i) $
1.000	17.000000	17.000000	0.000000
1.100	15.755485	15.755454	0.000038
1.200	14.773372	14.773333	0.000035
1.300	13.997727	13.997692	0.000027
1.400	13.388599	13.388571	0.000017
1.500	12.916684	12.916666	0.000007
1.600	12.560007	12.560000	0.000002
1.700	12.301761	12.301764	0.000012
1.800	12.128876	12.128888	0.000021
1.900	12.031031	12.031052	0.000028
2.000	11.999971	12.000000	0.000036
2.100	12.029011	12.029047	0.000042
2.200	12.112684	12.112727	0.000047
2.300	12.246473	12.246521	0.000052
2.400	12.426614	12.426666	0.000056
2.500	12.649943	12.650000	0.000059
2.600	12.913786	12.913846	0.000062
2.700	13.215863	13.215925	0.000064
2.800	13.554221	13.554285	0.000066
2.900	13.927175	13.927241	0.000067
3.000	14.333266	14.333333	0.000000

The maximum error is 0.000067



**Figure (3)** : shows the approximate and the exact solution for example (2.3) that was solved by shooting method.

**Example 2.4**

Consider the non-linear boundary-value problem

$$y'' = y^3 - yy', \quad 1 \leq x \leq 2, \quad y(1) = \frac{1}{2}, \quad y(2) = \frac{1}{3}$$

has the exact solution

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

Applying shooting method to this problem requires approximating the solutions to the initial-value problems

$$y'' = y^3 - yy', \quad 1 \leq x \leq 2, \quad y(1) = \frac{1}{2}, \quad y'(1) = t_k$$

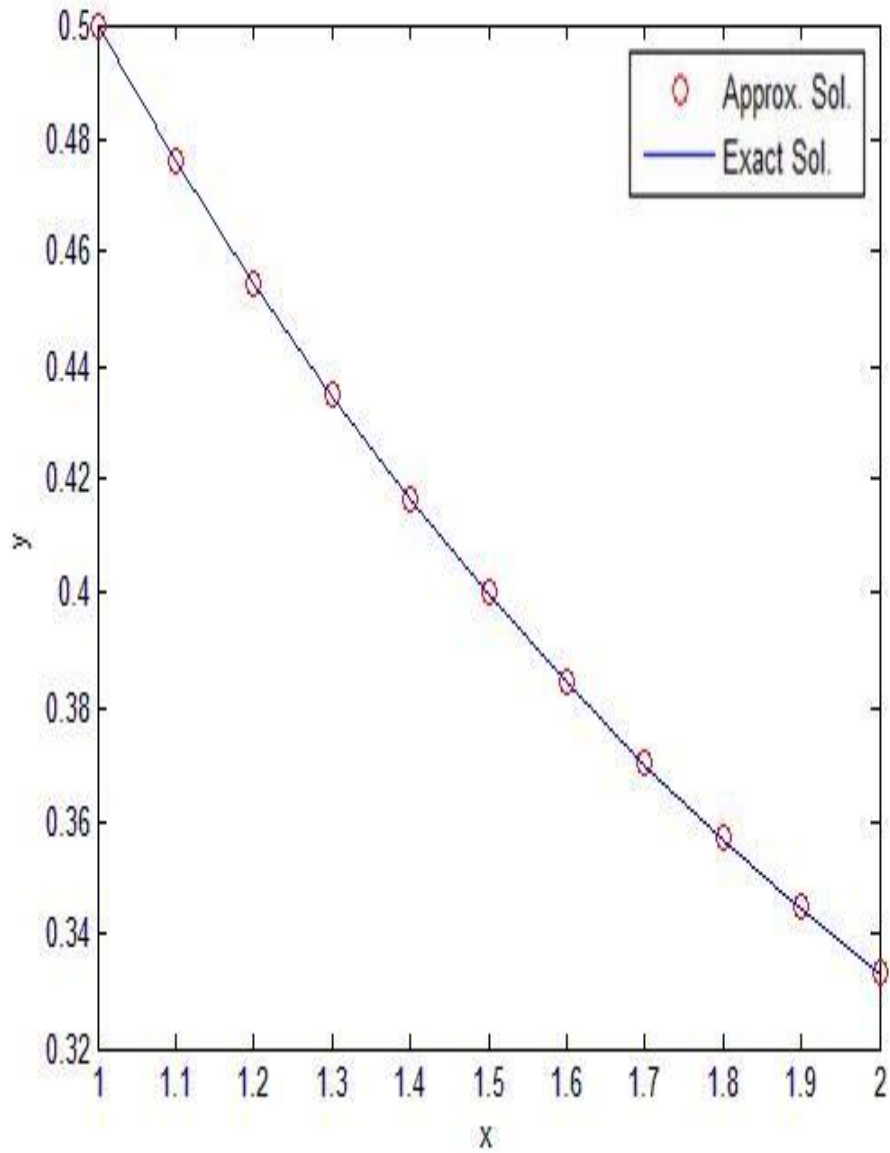
$$z'' = -(y'z + yz'), \quad 1 \leq x \leq 2, \quad z(1) = 0, \quad z'(1) = 1$$

Algorithm 2.2 uses the Runge-Kutta method of order four to approximate both solutions required by Newton's method (page 35). The value listed as  $w_i$  approximates  $y(x_i)$  is the exact solution and  $ee_i$  is the error between the exact solution and the approximate solution. The results of the calculations with  $N = 10$  are given in table (2.4). See [3] program 3

**Table (2.4)** :The approximate and exact solution for example 2.4.

$x_i$	$w_{1,i}$	$y(x_i)$	$ee_i =  w_{1,i} - y(x_i) $
1.000	0.500000	0.500000	0.000000
1.100	0.476191	0.476190	0.000002
1.200	0.454547	0.454545	0.000003
1.300	0.434786	0.434782	0.000004
1.400	0.416671	0.416666	0.000006
1.500	0.400006	0.400000	0.000007
1.600	0.384622	0.384615	0.000008
1.700	0.370378	0.370370	0.000009
1.800	0.357152	0.357142	0.000010
1.900	0.344838	0.344827	0.000011
2.000	0.333345	0.333333	0.000000

The maximum error is 0.000011



**Figure (4):** shows the approximate and the exact solution for example (2.4) that was solved by shooting method.

## 2.7 Finite Difference Methods

In these methods, no initial value problems are explicitly integrated. Rather, an approximate solution representation is sought over the entire interval of interest. Thus, these methods are sometimes referred to as global methods.

The basic steps of a finite difference method are outlined as follows,

we choose a mesh  $\Omega$  to the interval  $[a,b]$ , where  $\Omega = (a = x_1 < x_2 < \dots < x_{N+1} = b)$  then approximate solution values are then sought at these mesh points  $x_i$  for  $i=2, 3, \dots, n$

Form a set of algebraic equations for the approximate solution values by replacing derivatives with difference quotients in the differential equations and boundary conditions that the exact solution satisfies.

Finally, solve the resulting system of equations for the approximate solution, this gives a set of discrete solution values  $y_i = y_\Omega(x_i)$ .

Finite difference methods proceed by replacing the derivatives in the differential equations by finite difference approximations. This gives a large algebraic system of equations to be solved in place of differential equation.

To approximate  $y'$  we can use one-sided approximation

$$D_+ y(x) = y' \cong \frac{y(x+h) - y(x)}{h} \quad \text{or} \quad D_- y(x) = y' \cong \frac{y(x) - y(x-h)}{h}$$

Or we can use centered approximation :

$$D_0 y(x) = y' \cong \frac{y(x+h) - y(x-h)}{2h}$$

Which is the average of the two one-sided approximations. It is clear that  $D_0 y(x)$  gives a better approximation than either of the one-sided approximations also it gives us a second order accurate approximation .

We can use finite difference to solve a differential equation consider the second order differential equation

$$y''(x) = f(x), \quad 0 < x < 1 \quad (2.32)$$

$$y(0) = \alpha \quad y(1) = \beta$$

The function  $f(x)$  is specified and we wish to determine  $y(x)$  in the interval  $0 < x < 1$ . This problem is called two points boundary value problem. Since boundary conditions are given at the two distinct points 0 and 1.

### 2.7.1 Simple One-Step Schemes for Linear first-order Systems

Consider now the linear first-order system

$$\mathbf{y}' = \mathbf{A}(x) \mathbf{y} + \mathbf{f}(x), \quad x \in [a, b], \quad \mathbf{y} \in \mathbb{R}^n \quad (2.33)$$

$$\mathbf{B}_a \mathbf{y}(a) + \mathbf{B}_b \mathbf{y}(b) = \boldsymbol{\beta} \quad (2.34)$$

and we seek numerical methods which work equally well for non-uniform meshes. This naturally leads to one-step schemes, schemes which define the difference operator based only on values related to one subinterval

$[\mathbf{x}_i, \mathbf{x}_{i+1}]$  of the mesh  $\Omega$  at a time. The two simplest such finite difference schemes are the midpoint and the trapezoidal schemes.

For a (generally non-uniform) mesh  $\Omega$ , a discrete numerical solution

$\mathbf{y}_\Omega = (y_1, y_2, \dots, y_{N+1})^T$  is sought, Where  $y_i$  is to approximate component-wise the exact solution  $y(x)$  at  $\mathbf{x} = \mathbf{x}_i$ ,  $i = 2, 3, \dots, N$ . The numerical solution (in all methods based on one-step schemes) is required to satisfy the boundary condition (2.34).

For the interior mesh points, two difference schemes are presented. For each subinterval  $[\mathbf{x}_i, \mathbf{x}_{i+1}]$ ,  $i=1,2,3,\dots,N-1,N$ , of  $\Omega$  the derivative in (2.33) is

replaced by  $\frac{y_{i+1} - y_i}{h_i}$ . This approximation is centered at

$$x_{i+1/2} := x_i + \frac{1}{2} h_i, \text{ with } h_i := (x_{i+1} - x_i) \text{ at the middle of the subinterval.}$$

Then  $A(x)y(x) + f(x)$  is approximated by a centered approximation,

yielding second-order accuracy. The trapezoidal scheme is defined by:

$$\frac{y_{i+1} - y_i}{h_i} = \frac{1}{2} [A(x_{i+1})y_{i+1} + A(x_i)y_i] + \frac{1}{2} [f(x_{i+1}) + f(x_i)] \quad 1 \leq i \leq N \quad (2.35)$$

and the midpoint scheme is defined by:

$$\frac{y_{i+1} - y_i}{h_i} = \frac{1}{2} A(x_{i+1/2})(y_{i+1} + y_i) + f(x_{i+1/2}) \quad 1 \leq i \leq N \quad (2.36)$$



The latter scheme (2.36) is also known as the box scheme. In vector-matrix form, both of these methods can be written as

$$Ay_{\Omega} = F \quad (2.37)$$

and, in detail,

$$\begin{bmatrix} S_1 R_1 & & & & & \\ & S_2 R_2 & & & & \\ & & \cdot & & & \\ & & & \cdot & & \\ & & & & S_N R_N & \\ B_a & & & & & B_b \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ y_{N+1} \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \cdot \\ \cdot \\ f_N \\ \beta \end{bmatrix} \quad (2.38)$$

Where  $S_i, R_i$  are  $n \times n$  matrices. For the trapezoidal scheme

$$\begin{aligned} \mathbf{S}_i &= -h_i^{-1}I - \frac{1}{2}A(x_i), \quad \mathbf{R}_i = h_i^{-1}I - \frac{1}{2}A(x_{i+1}) \\ f_i &= \frac{1}{2}[f(x_{i+1}) + f(x_i)] \quad 1 \leq i \leq N \end{aligned} \quad (2.39)$$

while for the midpoint scheme

$$\begin{aligned} \mathbf{S}_i &= -h_i^{-1}I - \frac{1}{2}A(x_{i+1/2}), \quad \mathbf{R}_i = h_i^{-1}I - \frac{1}{2}A(x_{i+1/2}) \\ f_i &= f(x_{i+1/2}) \quad 1 \leq i \leq N \end{aligned} \quad ;(2.40) \text{ See [1]}$$

## 2.7.2 Finite-Difference Methods for Linear Problems

Consider more general linear equation

$$y'' = p(x)y' + q(x)y + r(x)$$

Together with two boundary conditions, "Dirichlet conditions "



The discretization used above, while second order accurate, may not be the best discretization to use for certain problems of this type. Often the physical problem has certain properties that we would like to preserve with our discretization, and it is important to understand the underlying problem and be aware of its mathematical properties before blindly applying numerical method. See [8]

### 2.7.3 Neumann Boundary Conditions

Consider we have one or more Neumann boundary conditions, instead of Dirichlet boundary conditions, meaning that a boundary condition on the derivative  $y'$  is given rather than a condition on the value of  $y$  itself. We might have heat flux at a specified rate giving  $y' = \alpha$  at this boundary.

Consider the equation (2.32) with boundary conditions:

$$y'(0) = \alpha \quad y(1) = \beta \quad (2.41)$$

to solve this problem numerically, we need to introduce one more unknown than we previously had:  $Y_0$  at the point  $x_0 = 0$  since this is now an unknown value. We also need to augment the system (2.37) with one more equation that models the boundary conditions (2.41). As a first try, we might use a one-sided expression for  $y'(0)$  such as:

$$\frac{y_1 - y_0}{h} = \alpha \quad (2.42)$$



We have now reduced the system to one with only  $N+1$  equation for the unknowns  $Y_0, Y_1, \dots, Y_N$ . The matrix is exactly the same as the matrix in (2.43) which came from the one-sided approximation, the only difference in the linear system is that the first element in the right hand side of (2.43) is now changed from  $\alpha$  to  $\alpha + \frac{h}{2} f(x_0)$  we can view the left hand side of (2.45) as a centered approximation to  $y'(x_0 + \frac{h}{2})$  and the right hand side as the first two terms in the Taylor series expansion of this value

$$y'(x_0 + \frac{h}{2}) = y'(x_0) + \frac{h}{2} y''(x_0) + \dots = \alpha + \frac{h}{2} f(x_0) + \dots$$

### Algorithm 2.3

#### Linear Finite-Difference

To approximate the solution of the boundary-value problem

$$y'' = p(x)y' + q(x)y + r(x), \quad a \leq x \leq b, \quad y(a) = \alpha, \quad y(b) = \beta:$$

INPUT : endpoints  $a, b$ ; boundary conditions  $\alpha, \beta$ ; integer  $N \geq 2$ , and elements of matrix  $A$  and the right hand side.

OUTPUT : approximations  $w_i$  to  $y(x_i)$  for each  $i = 0, 1, \dots, N+1$ .

Step 1 Set

$$\begin{aligned}h &= (b - a) / (N + 1); \\x &= a + h; \\a_1 &= 2 + h^2 q(x); \\b_1 &= -1 + (h / 2) p(x); \\d_1 &= -h^2 r(x) + (1 + (h / 2) p(x)) \alpha.\end{aligned}$$

Step 2 For  $i = 2, \dots, N-1$

Set

$$\begin{aligned}x &= a + ih; \\a_i &= 2 + h^2 q(x); \\b_i &= -1 + (h / 2) p(x); \\c_i &= -1 - (h / 2) p(x); \\d_i &= -h^2 r(x).\end{aligned}$$

Step 3 Set

$$\begin{aligned}x &= b - h; \\a_N &= 2 + h^2 q(x); \\c_N &= -1 - (h / 2) p(x); \\d_N &= -h^2 r(x) + (1 - (h / 2) p(x)) \beta.\end{aligned}$$

Step 4 Set  $l_1 = a_1$ ; (steps 4-8 solve a tridiagonal linear system algorithm.)

$$\begin{aligned}u_1 &= b_1 / a_1; \\z_1 &= d_1 / l_1.\end{aligned}$$

Step 5 For  $i = 2, \dots, N-1$  set

$$l_i = a_i - c_i u_{i-1}$$

$$u_i = b_i / l_i$$

$$z_i = (d_i - c_i z_{i-1}) / l_i$$

Step 6 Set

$$l_N = a_N - c_N u_{N-1};$$

$$z_N = (d_N - c_N z_{N-1}) / l_N.$$

Step 7 Set

$$w_0 = \alpha;$$

$$w_{N+1} = \beta;$$

$$w_N = z_N.$$

Step 8 For  $i = N-1, \dots, 1$  set  $w_i = z_i - u_i w_{i+1}$ .

Step 9 For  $i = 0, \dots, N+1$  set  $x = a + ih$ ;

OUTPUT  $(x, w_i)$ .

Step 10 STOP. (The procedure is complete.)

**Example 2.5**

Algorithm (2.3) will be used to approximate the solution to the boundary-value problem

$$y'' = -\frac{2}{x} y' + \frac{2}{x^2} y + \frac{\sin(\ln x)}{x^2}, \quad 1 \leq x \leq 2$$

$$y(1) = 1, \quad y(2) = 2$$

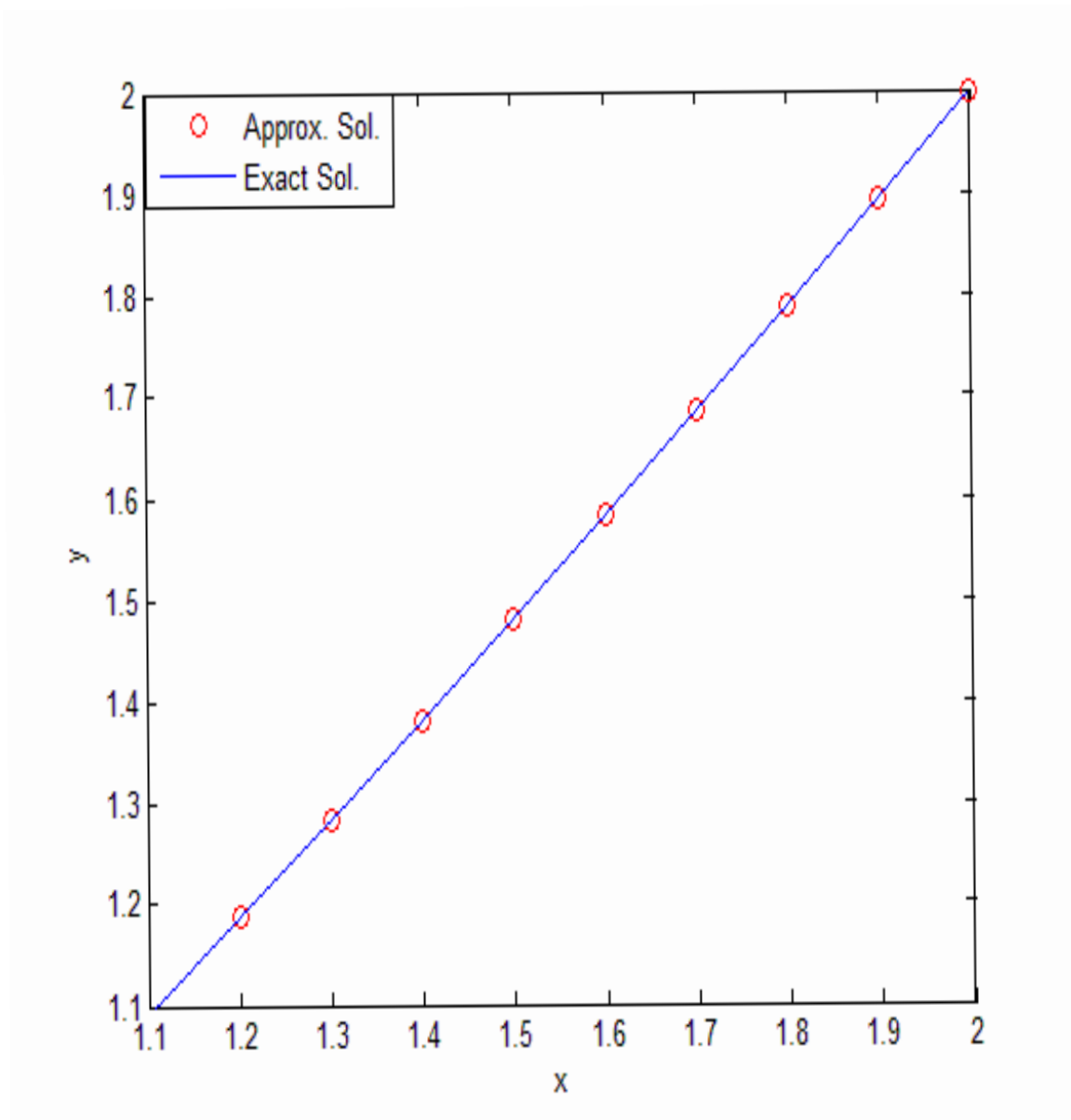
with  $h = 0.1$  which was also approximated by the shooting method in example (2.1), gives the results listed in table (2.5). The value listed as  $w_i$  approximates and  $y(x_i)$  is the exact solution and  $ee_i$  is the error between the exact solution and the approximate solution. program 4

**Table(2.5) :** The approximate and exact solution for example 2.5.

$x_i$	$w_i$	$y_i$	$ee_i =  w(x_i) - y(x_i) $
1.000	1.000000	1.000000	0.000000
1.100	1.092600	1.092629	0.000028
1.200	1.187043	1.187084	0.000041
1.300	1.283336	1.283382	0.000045
1.400	1.381402	1.381445	0.000043
1.500	1.481120	1.481159	0.000039
1.600	1.582359	1.582392	0.000032
1.700	1.684989	1.685013	0.000024
1.800	1.788881	1.788898	0.000016
1.900	1.893921	1.893929	0.000008
2.000	2.000000	2.000000	0.000000

The maximum error is 0.000045





**Figure (5):** shows the approximate and the exact solution for example (2.5) that was solved by finite difference method.

**Example 2.6**

Algorithm (2.3) will be used to approximate the solution to the boundary-value problem

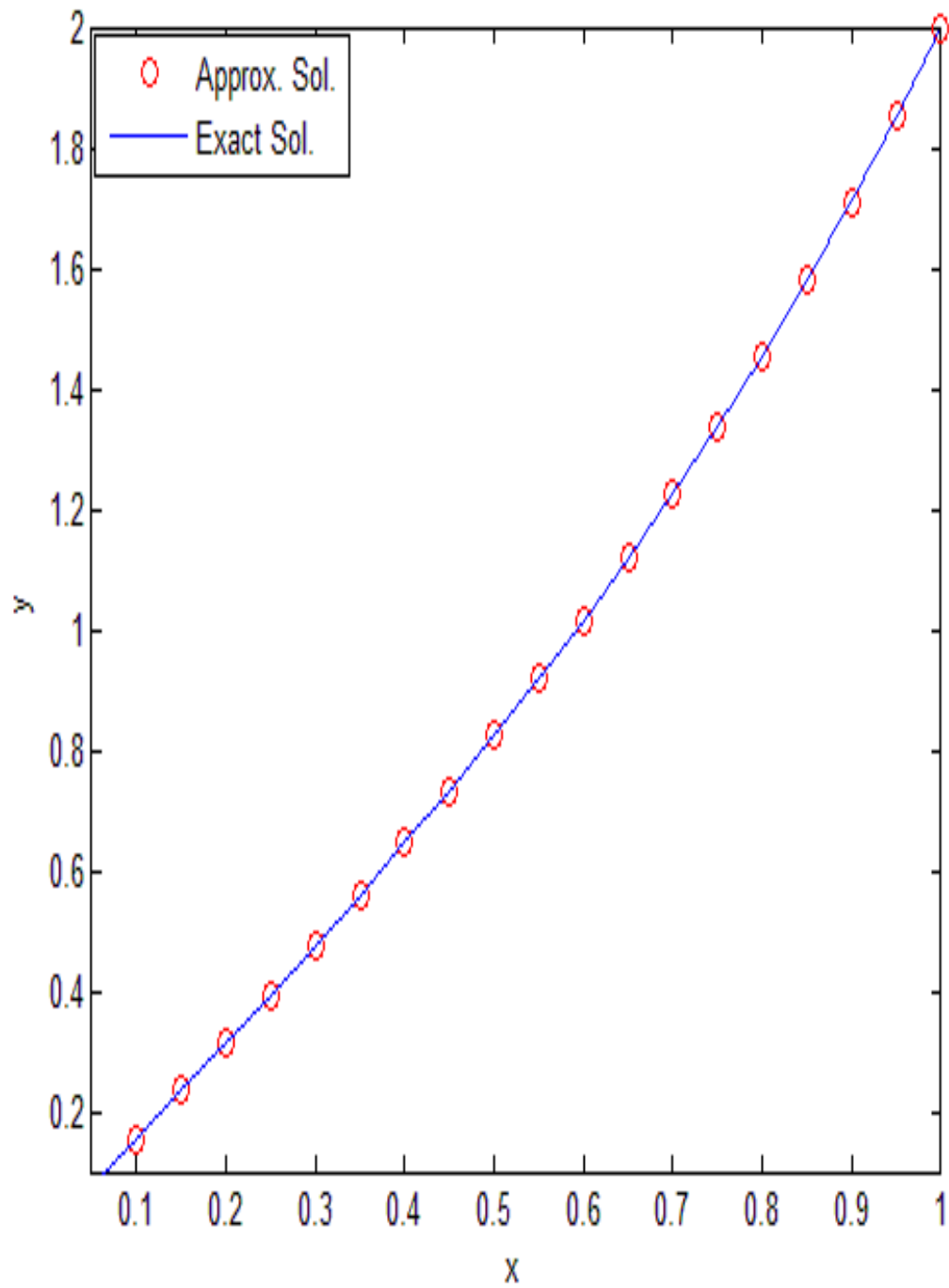
$$y'' = 4(y - x), \quad 0 \leq x \leq 1, \quad y(0) = 0, \quad y(1) = 1$$

with  $h = 1/20$ , which was also approximated by the shooting method in example (2.2), gives the results listed in table (2.6). The value listed as  $w_i$  approximates  $y(x_i)$  is the exact solution and  $ee_i$  is the error between the exact solution and the approximate solution. program 5

**Table (2.6) :** The approximate and exact solution for example 2.6.

$x_i$	$y_i$	$yy_i$	$ee_i =  y(x_i) - yy(x_i) $
0.000	0.000000	0.000000	0.000000
0.050	0.077616	0.077618	0.000001
0.100	0.155509	0.155512	0.000003
0.150	0.233957	0.233962	0.000005
0.200	0.313244	0.313252	0.000008
0.250	0.393664	0.393676	0.000012
0.300	0.475520	0.475538	0.000017
0.350	0.559132	0.559157	0.000024
0.400	0.644835	0.644869	0.000033
0.450	0.732986	0.733031	0.000045
0.500	0.823967	0.824027	0.000059
0.550	0.918188	0.918265	0.000076
0.600	1.016091	1.016189	0.000098
0.650	1.118155	1.118278	0.000123
0.700	1.224900	1.225055	0.000154
0.750	1.336894	1.337086	0.000191
0.800	1.454757	1.454992	0.000235
0.850	1.579168	1.579455	0.000286
0.900	1.710870	1.711217	0.000347
0.950	1.850682	1.851099	0.000417
1.000	2.000000	2.000000	0.000000

The maximum error is 0.000417



**Figure (6)** : shows the approximate and the exact solution for example (2.6) that was solved by finite difference method.

### 2.7.4 Finite-Difference Method for Nonlinear Problems

For the general nonlinear boundary-value problem  $y'' = f(x, y, y')$ ,  $a \leq x \leq b$ , if the function  $f$  satisfies the following conditions:

1.  $f$  and the partial derivatives  $f_y$  and  $f_{y'}$  are all continuous on

$$D = \{(x, y, y') \mid a \leq x \leq b, -\infty < y < \infty, -\infty < y' < \infty\};$$

2.  $f_y(x, y, y') \geq \delta$  on  $D$ , for some  $\delta > 0$ ;
3. Constants  $k$  and  $L$  exist, with

$$k = \max_{(x, y, y') \in D} |f_y(x, y, y')| \quad \text{and} \quad L = \max_{(x, y, y') \in D} |f_{y'}(x, y, y')|.$$

This ensures, by Theorem 2.1 page 17, that a unique solution exists.

Discretizing the interval  $a \leq x \leq b$  into  $N+1$  subintervals each of width  $h$  so that  $(N+1)h = b - a$ , a numerical method determines a vector  $\mathbf{Y} = [y_1, y_2, \dots, y_N]^T$ , where  $y_i$  is an approximation to  $y(x_i)$ . The derivatives  $y''(x)$  and  $y'(x)$  will be replaced by their second-order central difference approximants to the equation

$$y''(x_i) = f(x_i, y(x_i), y'(x_i))$$

for each  $i = 1, 2, \dots, N$  this gives

$$\frac{y(x_{i+1}) - 2y(x_i) + y(x_{i-1}))}{h^2} = f(x_i, y(x_i), \frac{y(x_{i+1}) - y(x_{i-1}))}{2h})$$

Where  $y_0 = \alpha$  and  $y_{N+1} = \beta$ . The solution is thus found by solving the  $N \times N$  nonlinear system

$$\begin{aligned}
2y_1 - y_2 + h^2 f(x_1, y_1, (y_2 - \alpha) / 2h) &= \alpha, \\
-y_1 + 2y_2 - y_3 + h^2 f(x_2, y_2, (y_3 - y_1) / 2h) &= 0, \\
&\dots \qquad \dots \\
-y_{N-2} + 2y_{N-1} - y_N + h^2 f(x_{N-1}, y_{N-1}, (y_N - y_{N-2}) / 2h) &= 0, \\
-y_{N-1} + 2y_N + h^2 f(x_N, y_N, (\beta - y_{N-1}) / 2h) &= \beta,
\end{aligned} \tag{2.48}$$

We use Newton's method for nonlinear systems, to approximate the solution to this system. A sequence of iterates  $\{(y_1^{(k)}, y_2^{(k)}, \dots, y_N^{(k)})^t\}$  is generated that may converges to the solution of the system (2.48), provided that the initial approximation  $(y_1^{(0)}, y_2^{(0)}, \dots, y_N^{(0)})^t$  is sufficiently close to the solution  $(y_1, y_2, \dots, y_N)^t$ , and that the Jacobian matrix for the system is nonsingular. For system (2.48), the Jacobian matrix  $J(y_1, \dots, y_N)$  is a tridiagonal with  $ij$ -th entry

$$J(y_1, \dots, y_N)_{ij} = \left\{ \begin{array}{l} -1 + \frac{h}{2} f_{y'}(x_i, y_i, \frac{y_{i+1} - y_{i-1}}{2h}), \quad \text{for } i=j-1 \quad \text{and} \quad j=2, \dots, N, \\ 2 + h^2 f_{y'}(x_i, y_i, \frac{y_{i+1} - y_{i-1}}{2h}), \quad \text{for } i=j \quad \text{and} \quad j=1, \dots, N, \\ -1 - \frac{h}{2} f_{y'}(x_i, y_i, \frac{y_{i+1} - y_{i-1}}{2h}), \quad \text{for } i=j+1 \quad \text{and} \quad j=1, \dots, N-1, \end{array} \right\}$$

Newton's method for nonlinear system requires that at each iteration the  $N \times N$  linear system

$$\begin{aligned}
& -(2y_1 - y_2 - \alpha + h^2 f(x_1, y_1, \frac{y_2 - \alpha}{2h})), \\
& -y_1 + 2y_2 - y_3 + h^2 f(x_2, y_2, \frac{y_3 - y_1}{2h}), \dots, \\
\mathbf{J}(y_1, \dots, y_N) (\mathbf{v}_1, \dots, \mathbf{v}_N)^t = & -y_{N-2} + 2y_{N-1} - y_N + h^2 f(x_{N-1}, y_{N-1}, \frac{y_N - y_{N-2}}{2h}), \\
& -y_{N-1} + 2y_N + h^2 f(x_N, y_N, \frac{\beta - y_{N-1}}{2h}) - \beta)^t
\end{aligned}$$

Be solved for  $v_1, v_2, \dots, v_N$ , since  $y_i^{(k)} = y_i^{(k-1)} + v_i$ , for each  $i = 1, 2, \dots, N$ .

See [3] & [10]

### Algorithm 2.4

#### Nonlinear Finite-Difference

To approximate the solution to the nonlinear boundary-value problem

$$y'' = f(x, y, y'), \quad a \leq x \leq b, \quad y(a) = \alpha, \quad y(b) = \beta:$$

INPUT : endpoints  $a, b$ ; boundary conditions  $\alpha, \beta$ ; integer  $N \geq 2$ , tolerance

TOL; maximum number of iterations  $M$ .

OUTPUT : approximations  $w_i$  to  $y(x_i)$  for each  $i = 0, 1, \dots, N+1$  or a

message that the maximum number of iterations was exceeded.

Step 1 Set

$$h = (b - a) / (N + 1);$$

$$w_0 = \alpha;$$

$$w_{N+1} = \beta.$$

Step 2 For  $i = 1, \dots, N$  set  $w_i = \alpha + i(\frac{\beta - \alpha}{b - a})h$ .

Step 3 Set  $k = 1$ .

Step 4 While  $K \leq M$  do steps 5-16.

Step 5 Set

$$\begin{aligned}x &= a + h; \\t &= (w_2 - \alpha)/(2h); \\a_1 &= 2 + h^2 f_y(x, w_1, t); \\b_1 &= -1 + (h/2) f_{y'}(x, w_1, t); \\d_1 &= -(2w_1 - w_2 - \alpha + h^2 f(x, w_1, t)).\end{aligned}$$

Step 6 For  $i = 2, \dots, N-1$

set

$$\begin{aligned}x &= a + ih; \\t &= (w_{i+1} - w_{i-1})/(2h); \\a_i &= 2 + h^2 f_y(x, w_i, t); \\b_i &= -1 + (h/2) f_{y'}(x, w_i, t); \\c_i &= -1 - (h/2) f_{y'}(x, w_i, t); \\d_i &= -(2w_i - w_{i+1} - w_{i-1} + h^2 f(x, w_i, t)).\end{aligned}$$

Step 7 Set

$$\begin{aligned}x &= b - h; \\t &= (\beta - w_{N-1})/(2h); \\a_N &= 2 + h^2 f_y(x, w_N, t); \\c_N &= -1 - (h/2) f_{y'}(x, w_N, t); \\d_N &= -(2w_N - w_{N+1} - \beta + h^2 f(x, w_N, t)).\end{aligned}$$

Step 8 Set  $l_1 = a_1$ ; (steps 8-12 solve a tridiagonal linear system.)

$$\begin{aligned}u_1 &= b_1 / a_1; \\z_1 &= d_1 / l_1.\end{aligned}$$



Step 9 For  $i = 2, \dots, N-1$  set

$$\begin{aligned}l_i &= a_i - c_i u_{i-1}; \\u_i &= b_i / l_i; \\z_i &= (d_i - c_i z_{i-1}) / l_i.\end{aligned}$$

Step 10 Set

$$\begin{aligned}l_N &= a_N - c_N u_{N-1}; \\z_N &= (d_N - c_N z_{N-1}) / l_N.\end{aligned}$$

Step 11 Set

$$\begin{aligned}v_N &= z_N; \\w_N &= w_N + v_N.\end{aligned}$$

Step 12 For  $i = N-1, \dots, 1$  set

$$\begin{aligned}v_i &= z_i - u_i v_{i+1}; \\w_i &= w_i + v_i.\end{aligned}$$

Step 13 If  $\|v\| \leq TOL$  Then do steps 14 and 15.

Step 14 For  $i = 0, \dots, N+1$  set  $x = a + ih$ ;

OUTPUT  $(x, w_i)$ .

Step 15 STOP. (The procedure was successful.)

Step 16 Set  $k = k + 1$ .

Step 17 OUTPUT ('Maximum number of iterations exceeded');

(The procedure was unsuccessful.)

STOP.

**Example 2.7**

Consider the boundary-value problem

$$y'' = \frac{1}{8}(32 + 2x^3 - yy'), \quad 1 \leq x \leq 3, \quad y(1) = 17, \quad y(3) = \frac{43}{3}$$

has the exact solution

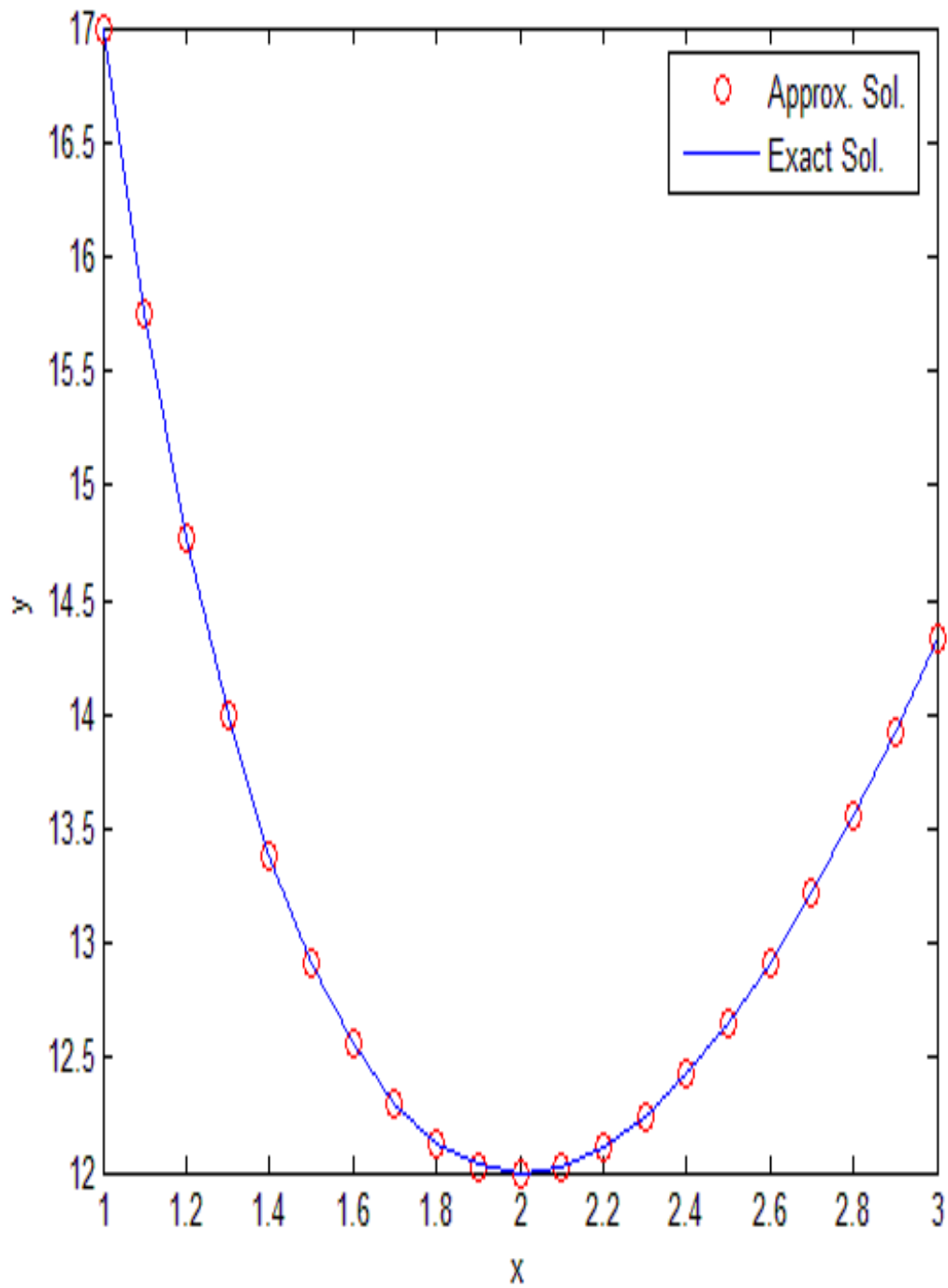
$$y(x) = x^2 + \frac{16}{x}$$

Applying finite difference method to this problem the solutions results in table (2.7). The value listed as  $w_i$  approximates and  $y(x_i)$  is the exact solution and  $ee_i$  is the error between the exact solution and the approximate solution. program 6

**Table (2.7) :** The approximate and exact solution for example 2.7.

$x_i$	$w_i$	$y_i$	$ee_i =  w(x_i) - y(x_i) $
1.000	17.000000	17.000000	0.000000
1.100	15.754502	15.755454	0.000951
1.200	14.771740	14.773333	0.001593
1.300	13.995677	13.997692	0.002014
1.400	13.386296	13.388571	0.002274
1.500	12.914252	12.916666	0.002413
1.600	12.557538	12.560000	0.002461
1.700	12.299326	12.301764	0.002438
1.800	12.126529	12.128888	0.002359
1.900	12.028813	12.031052	0.002238
2.000	11.997915	12.000000	0.002084
2.100	12.027142	12.029047	0.001905
2.200	12.111019	12.112727	0.001707
2.300	12.245024	12.246521	0.001496
2.400	12.425388	12.426666	0.001278
2.500	12.648944	12.650000	0.001055
2.600	12.913012	12.913846	0.000833
2.700	13.215311	13.215925	0.000614
2.800	13.553885	13.554285	0.000400
2.900	13.927046	13.927241	0.000195
3.000	14.333333	14.333333	0.000000

The maximum error is 0.002461



**Figure( 7):** shows the approximate and the exact solution for example (2.7) that was solved by finite difference method.

### Example 2.8

Consider the boundary-value problem

$$y'' = 2y^3 - 6y - 2x^3 \quad 1 \leq x \leq 2 \quad y(1) = 2 \quad y(2) = \frac{5}{2}$$

has the exact solution

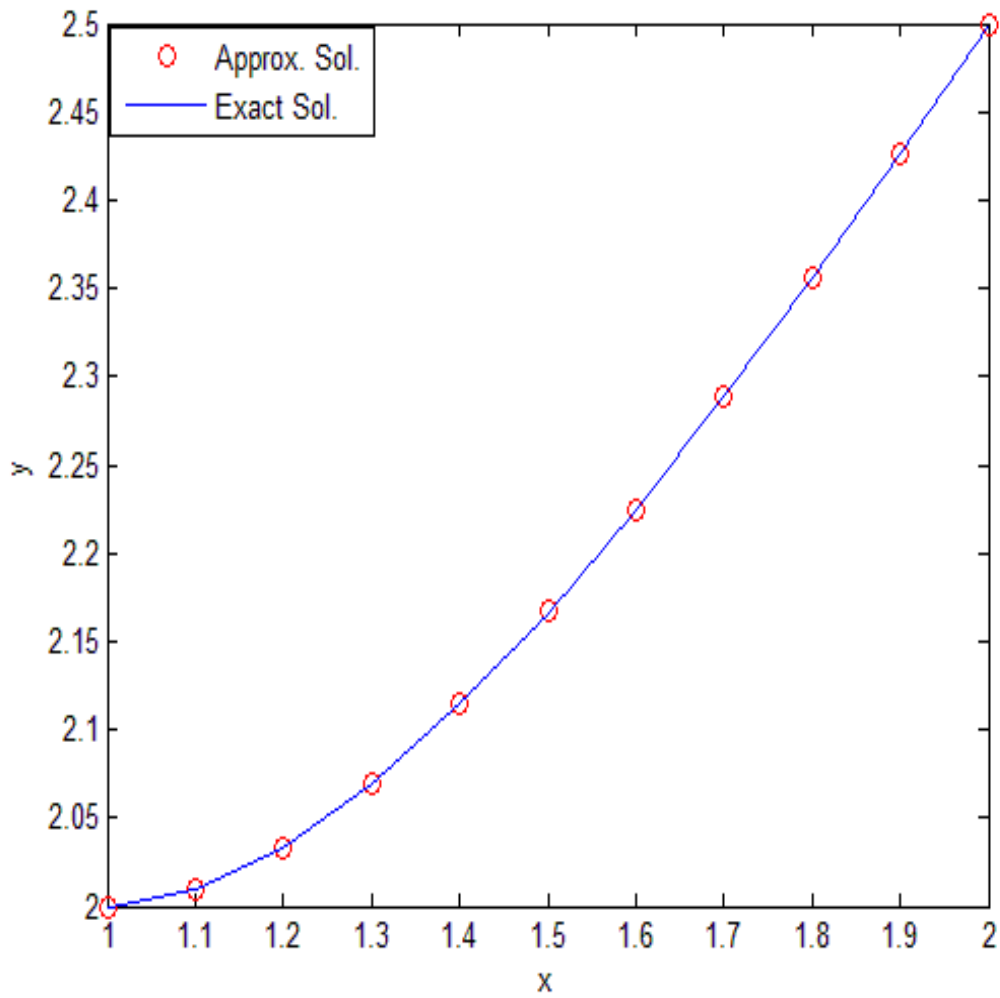
$$y(x) = x + x^{-1}$$

Applying finite difference method to this problem the solutions results in table (2.8). The value listed as  $w_i$  approximates and  $y(x_i)$  is the exact solution and  $ee_i$  is the error between the exact solution and the approximate solution. program 7

**Table (2.8) :** The approximate and exact solution for example 2.8.

$x_i$	$w_i$	$y_i$	$ee_i =  w(x_i) - y(x_i) $
1.000	2.000000	2.000000	0.000000
1.100	2.009256	2.009090	0.000165
1.200	2.033570	2.033333	0.000237
1.300	2.069431	2.069230	0.000200
1.400	2.114447	2.114285	0.000161
1.500	2.166795	2.166666	0.000129
1.600	2.225105	2.225000	0.000105
1.700	2.288321	2.288235	0.000085
1.800	2.355607	2.355555	0.000051
1.900	2.426252	2.426315	0.000063
2.000	2.500000	2.500000	0.000000

The maximum error is 0.000237



**Figure( 8 ) :** shows the approximate and the exact solution for example (2.8) that was solved by finite difference method.

## **Chapter Three**

## Singular Two-Points BVP

### 3.1 Introduction

Singular two-points boundary value problem occur frequently in mathematical modeling of many practical problems.

We consider first a system of linear ordinary differential equations on a finite interval with a singularity of the first kind at one endpoint. We treat the same problem with singularities at both endpoints and with a singularity on the interior of the interval.

Consider a class of singular BVPs:

$$(x^\alpha y')' = f(x, y), \quad 0 < x \leq 1, \quad y(0) = A, \quad y(1) = B \quad (3.1)$$

In which  $0 < \alpha \leq 1$  and  $A, B$  are finite constants, we assume also that for

$0 < x < 1$ , the real-valued function  $f(x, y)$  is continuous,  $\frac{df}{dy}$  exists and is

continuous and that  $\frac{df}{dy} > 0$ . See [2]

### 3.2 Regular Singular Point ,Singularities of The First Kind

Consider the ODE

$$(x^\alpha y')' - f(x, y) = 0 \quad 0 < x < 1 \quad (3.2)$$

If we assume here that  $\alpha = 1$  in (3.2). The assumptions on the regularity of a solution  $y(x)$  of (3.2) imply that  $\lim_{x \rightarrow 0} y(x)$  exists, as  $x$  decreases to 0.



This is needed in order to make the BVP for (3.2) meaningful, and is reasonable in most applications. These assumptions further yield that

$$\mathbf{f}(0, \mathbf{y}(0)) = \mathbf{0} \quad (3.3)$$

which must be compatible with the prescribed BC. In fact, the requirement (3.3) is often used to determine part of the BC.

To be more specific, let us consider now the linear BVP which has a singular point of the first kind

$$\mathbf{y}' = \mathbf{A}(x) \mathbf{y} + \mathbf{q}(x) \quad 0 < x < 1 \quad (3.4)$$

$$\text{where} \quad \mathbf{A}(x) = \frac{1}{x} \mathbf{R} + \tilde{\mathbf{A}}(x) \quad (3.5)$$

here  $\mathbf{y}(x)$ ,  $\mathbf{q}(x)$ , are  $n$  component vectors, and  $\mathbf{A}(x)$ ,  $\mathbf{R}$ ,  $\tilde{\mathbf{A}}(x)$  are  $n \times n$  matrices.  $\mathbf{R}$  is a constant matrix, and  $\mathbf{q}(x) \in C(0,1]$ .

For any solution  $\mathbf{y}(x)$  of (3.4), we require  $\mathbf{y}(x) \in C^1(0,1]$ , we also impose a linear system of two-points boundary conditions written as

$$\lim_{x \rightarrow 0^+} \{ \mathbf{B}_0 \mathbf{y}(x) \} + \mathbf{B}_1 \mathbf{y}(1) = \boldsymbol{\beta} \quad (3.6)$$

note that we cannot merely write

$$\mathbf{B}_0 \mathbf{y}(0) + \mathbf{B}_1 \mathbf{y}(1) = \boldsymbol{\beta} \quad (3.7)$$

because  $\mathbf{y}(x)$  is not even necessarily defined at  $x = 0$ . Notice also (3.7)

implies that  $\lim_{x \rightarrow 0^+} \mathbf{B}_0 \mathbf{y}(x)$  is bounded.

Let the fundamental solution matrix,  $Y(x)$ , for the homogeneous equation for (3.4). That is,  $Y(x)$  satisfies

$$Y' = A(x)Y, \quad x \in (0,1], \quad Y(x_0) = I, \quad x_0 \in (0,1] \quad (3.8)$$

Then every solution to (3.4) can be written

$$y(x) = cY(x) + y_p(x) \quad x \in (0,1] \quad (3.9)$$

where  $y(x)$  is any particular solution of (3.4) and where  $c$  is a constant vector. where the particular solution  $y_p(x)$  satisfies

$$y_p = \beta(x) \quad 0 < x \leq 1, \quad y_p(\delta) = 0 \quad (3.10)$$

Where  $0 < \delta < 1$ , See [2] & [7]

The smoothness of  $f(x, y)$  in (3.2) [or of  $A(x)$  and  $q(x)$  in (3.4)] does not imply corresponding smoothness of  $y(x)$  near  $x = 0$ .

For example, the IVP

$$xy' = \frac{1}{2}y, \quad y(0) = 0$$

has the solution  $y(x) = \sqrt{x}$ , which has an unbounded first derivative at  $x = 0$ . However, where often the solution  $y(x)$  is nonetheless smooth near the singularity. The performance of numerical methods for problems with singularities of the first kind where the solution is smooth at the singularities.

The situation is much less straightforward for some of the initial value. This is because not all fundamental solution components of (3.4) may be expected to be as smooth near the singularity as the solution  $y(x)$  is.

For example, the IVP

$$y' = \frac{-2}{x} y + \frac{2}{x}$$

$$y(0) = 1$$

has the solution  $y(x) = 1$  and a fundamental solution  $y(x) = \frac{1}{x^2}$ . Therefore a special treatment near  $x = 0$  is often required before a code based on an initial value approach can be used.

Such a special treatment may consist of power expansion of a fundamental solution in the vicinity of  $x = 0$ , followed by use of an initial value code when we are sufficiently far away from the singularity. Once a fundamental solution  $y(x)$ ,  $0 < x < \delta$ , has been found in this way, an appropriate particular solution can be found as well, and the boundary condition

$$\mathbf{B}_\delta \mathbf{y}(\delta) + \mathbf{B}_1 \mathbf{y}(1) = \tilde{\boldsymbol{\beta}} \quad (3.11)$$

can be constructed to replace (3.6). The location of the joint  $\delta > 0$  has to be small enough so that the power series expansion for  $Y(x)$  on  $[0, \delta]$  can be easily and efficiently constructed, and at the same time large enough so that

the BVP (3.4), (3.11) on  $[\delta,1]$  can be solved by a standard initial value method without difficulty. See [1]

### 3.3 Irregular Singular Point

There is at present no theoretical work justifying numerical methods for solving problems with irregular singular points. The main practical occurrence of such problems seems to be those formulated on infinite intervals and we examine some simple examples here. See [7]

Suppose that we have the ODE

$$\mathbf{y}' = \mathbf{f}(\mathbf{x}, \mathbf{y}) = \mathbf{A}(\mathbf{x}) \mathbf{y} \quad a > 0 \quad (3.12)$$

Then a transformation

$$t = \frac{a}{x} \quad (3.13)$$

reformulates (3.12) as an ODE defined on the interval  $(0,1]$ , namely

$$t^2 \frac{d\mathbf{y}}{dt} = -a\mathbf{f}\left(\frac{a}{t}, \mathbf{y}\right) \quad (3.14)$$

in which we recognize an ODE with a singularity of the second kind. In (3.13) we have assumed that  $a > 0$ . If  $a \leq 0$  then the transformation

$t = \frac{1}{x+1-a}$  and reformulates (3.12) as an ODE defined on the interval

$(0,1]$ , namely.

$$t^2 \frac{d\mathbf{y}}{dt} = -\mathbf{f}\left(\frac{1}{t} + a - 1, \mathbf{y}\right)$$

Since the formulation (3.12) is more natural, and since it turns out to be usually preferable for numerical discretization as well.

Of course, when it comes to numerical discretization the infinite interval  $[a, \infty]$  has to be replaced by a finite one, say  $[a, b]$  where  $b$  is "large". See [1]

### 3.4 Other Singular Problem

We now consider three cases of singularities . The first of three is the case of an equation with a singularity at both ends of the interval  $[0,1]$ :

$$\mathbf{y}' = \left( \frac{1}{x} \mathbf{R}_0 + \frac{1}{1-x} \mathbf{R}_1 + \tilde{\mathbf{A}}(x) \right) \mathbf{y} + \mathbf{b}(x), \quad x \in (0,1) \quad (3.15)$$

Where  $R_0$  and  $R_1$  are constant  $n \times n$  matrices, and  $\mathbf{b}(x) \in C(0,1)$ . We use the boundary conditions

$$\lim_{x \rightarrow 0^+} B_0 y(x) + \lim_{x \rightarrow 1^-} B_1 y(1) = \beta \quad (3.16)$$

Substituting the form of  $y(x)$  (3.9) & (3.10) into this boundary condition we have

$$\lim_{x \rightarrow 0^+} [B_0 c Y(x) + B_0 y_p(x)] + \lim_{x \rightarrow 1^-} [B_1 c Y(x) + B_1 y_p(x)] = \beta \quad (3.17)$$

For  $\tilde{\mathbf{A}}(x) \neq 0$ , then  $\tilde{\mathbf{A}}(x)$  may have singularities which are weaker than

$$\frac{1}{x} R_0 + \frac{1}{1-x} R_1.$$

The second case is the case of a singularity in the interior of the interval. The equation is the same as our original equation (3.4) but on the interval  $x \in [-1,0) \cup (0,1]$ .

$$\mathbf{y}' = \left(\frac{1}{x}\mathbf{R} + \tilde{\mathbf{A}}(x)\right)\mathbf{y} + \mathbf{b}(x), \quad x \in [-1,0) \cup (0,1] \quad (3.18)$$

We use a system of boundary conditions at -1 and 1:

$$\mathbf{B}_{-1}\mathbf{y}(-1) + \mathbf{B}_1\mathbf{y}(1) = \boldsymbol{\beta} \quad (3.19)$$

By a solution to (3.18), (3.19) we mean any of the functions

$$y(x) = cY(x) + y_p(x), \quad x \in [-1,0) \cup (0,1] \quad (3.20)$$

Which satisfies (3.19). In satisfying (3.19), we must have

$$[B_{-1}Y(-1) + B_1Y(1)]c = \boldsymbol{\beta} - B_{-1}y_p(-1) - B_1y_p(1) \quad (3.21)$$

Since  $Y(-1), Y(1), y_p(-1)$ , and  $y_p(1)$  exist with no singularities. Then here the singularity index is zero, so that if a solution is required for every  $\boldsymbol{\beta}$ ,  $B_{-1}Y(-1) + B_1Y(1)$  must be nonsingular. If  $B_{-1}Y(-1) + B_1Y(1)$  is singular, then  $\boldsymbol{\beta} - B_{-1}y_p(-1) - B_1y_p(1)$  must lie in its range .

The third and final case is simply treating the case of a regular differential equation on an infinite interval. We will illustrate this case for a semi-infinite interval, treating the problem

$$(a) \mathbf{y}' = \mathbf{A}(x)\mathbf{y} + \mathbf{b}(x), \quad x \in [0, \infty)$$

$$(b) \lim_{x \rightarrow \infty} B_{\infty}y(x) + B_0y(0) = \beta. \quad (3.22)$$

If we make the change of variable

$$t = \frac{1}{x+1}, \quad \text{or} \quad x = \frac{1}{t} - 1 \quad (3.23)$$

We map  $x \in [0, \infty)$  into  $t \in [1, 0)$ . Letting

$$\hat{\mathbf{y}}(t) = \mathbf{y}\left(\frac{1}{t} - 1\right), \hat{\mathbf{A}}(t) = \mathbf{A}\left(\frac{1}{t} - 1\right), \hat{\mathbf{b}}(t) = \mathbf{b}\left(\frac{1}{t} - 1\right),$$

the problem is then transformed into

$$(a) \hat{\mathbf{y}}'(t) = -\frac{1}{t^2} \hat{\mathbf{A}}(t)\hat{\mathbf{y}}(t) + \frac{1}{t^2} \hat{\mathbf{b}}(t), t \in [1, 0)$$

$$(b) \lim_{t \rightarrow 0^+} B_{\infty} \hat{\mathbf{y}}(t) + B_0 \hat{\mathbf{y}}(1) = \beta \quad (3.24)$$

Then a necessary and sufficient condition for (3.24) to have at most a singularity of the first kind at  $t = 0$  is and  $\mathbf{A}(\infty) = 0$ .

This statement implies that if

$$\mathbf{A}(x) = \frac{1}{x} \mathbf{R} + o\left(\frac{1}{x^2}\right) \text{ as } x \rightarrow \infty, \quad (3.25)$$

(3.24) will have exactly a singularity of the first kind if  $\mathbf{R}$  is not the zero matrix. See [2]

### 3.5 Finite Difference (Pade Based) Methods

We now describe schemes of finite difference methods based on pade rational approximation. These methods are based on rational approximants to the exponential function.

Pade approximants are defined as follows:

Let  $f(z), z \in \mathbb{C}$ , be function in a region of the complex plane containing the origin  $z = 0$ .

A pade approximant.  $R_{\mu, \kappa}(z)$  to the function  $f(z)$  is defined as by:

$$f(z) = \frac{P_{\kappa}(z)}{Q_{\mu}(z)}, \text{ where } P_{\kappa}(z) \text{ and } Q_{\mu}(z), \text{ are polynomials of degrees}$$

$\kappa$  and  $\mu$  respectively.

For the function  $f(z) = e^z$ , the polynomials  $P_{\kappa}(z)$  and  $Q_{\mu}(z)$ , are given explicitly as:

$$P_{\kappa}(z) = \sum_{j=0}^{\kappa} \frac{(\mu + \kappa - j)! \kappa!}{(\mu + \kappa)! j! (\kappa - j)!} (z^j)$$

And

$$Q_{\mu}(z) = \sum_{j=0}^{\mu} \frac{(\mu + \kappa - j)! \mu!}{(\mu + \kappa)! j! (\mu - j)!} (-z^j)$$



If  $e^z = \frac{P_\kappa(z)}{Q_\mu(z)} + T_{\mu,\kappa}(z)$ , then the remainder  $T_{\mu,\kappa}(z)$  is given

by:

$$T_{\mu,\kappa}(z) = \frac{(-1)^{\kappa+1} z^{(\mu+\kappa+1)}}{(\mu+\kappa)! Q_\mu(z)} \int_0^1 e^{(z(1-u))u^\kappa(1-u)^\mu} du$$

The Pade approximants for  $f(z) = e^z$  (for  $\mu=1,2,3,4$  and  $\kappa=1,2,3,4$ )

Can be generated from the above equations .

Example : When  $\kappa = 0$  &  $\mu = 2$  we will have

$$e^z = \frac{1}{1 - z + \frac{1}{2} z^2}$$

See Appendix for more function approximations.

### 3.5.1 A Numerical Method Based on The (2,0) Pade

#### Approximant

Consider the linear second order BVP (2.11). let  $y_0 = y$ ,  $y_1 = y'$  then (2.11) can be written as the system of first order differential equations:

$$\begin{bmatrix} y_0' \\ y_1' \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{1} \\ -\mathbf{q} & -\mathbf{p} \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{r} \end{bmatrix}$$

Which can be written in vector-matrix form as:

$$\mathbf{D} \mathbf{y} = \mathbf{Q} \mathbf{y} + \mathbf{P}$$

With boundary conditions  $y_0(0) = A$ ,  $y_1(1) = B$ . See [10]

To solve this class of singular BVP

$$(x^\alpha y')' = f(x, y), \quad 0 < x < 1, \quad y(0) = A, \quad y(1) = B \quad (3.26)$$

In which  $0 < \alpha \leq 1$  and A, B are finite constants.

$$y'' + \alpha \frac{x^{\alpha-1}}{x^\alpha} y' = f(x, y)$$

This problem can be written in vector-matrix form as

$$\mathbf{D} \underline{\mathbf{y}} = \mathbf{Q} \underline{\mathbf{y}} + \underline{\mathbf{p}} \quad (3.27)$$

With special case that is  $\underline{\mathbf{p}} = \underline{\mathbf{0}}$ .

The boundary conditions become  $y_0(0) = A$  ,  $y_1(1) = B$ . Using the relation,  $y(x + h) = e^{hD}y(x)$  and replacing the exponential term by its (2,0) Pade approximant, we get

$$[\mathbf{I} - h\mathbf{D} + \frac{h^2}{2}\mathbf{D}^2]y(x + h) = y(x) + o(h^3)$$

$$\mathbf{I} - h\mathbf{D}y + \frac{h^2}{2}\mathbf{D}(\mathbf{D}y) = y(x)$$

Using (3.27) and its second derivative and applying the resulting equation to the discrete point, of  $\Omega$  (where  $\Omega$  is the grid  $a = x_0 < x_1 < x_2, \dots, < x_N < x_{N+1} = b$  obtained by discretizing the interval  $[a, b]$  into  $N+1$  subintervals each of width  $h = \frac{b-a}{N+1}$   $N \in \mathbb{Z}^*$ ) leads to the finite-difference formula:

$$\mathbf{A}_{k+1}\mathbf{y}_{k+1} + \mathbf{B}_k\mathbf{y}_k = \mathbf{0} \quad , (k = 0, 1, 2, \dots, N) \quad (3.28)$$

Where  $\mathbf{B}_k = -\mathbf{I}$  ,  $\mathbf{I}$  is the identity matrix,  $\mathbf{y}_k = [y_{1k}, y_{0k}]^T$  and the elements of the matrix  $\mathbf{A}_{k+1}$  are

$$\mathbf{A}_{k+1} = \begin{bmatrix} a_{k+1,1,1} & a_{k+1,1,2} \\ a_{k+1,2,1} & a_{k+1,2,2} \end{bmatrix}$$

Such that



$\mathbf{A}_K$  and  $\mathbf{B}_K$  ( $k = 1, 2, \dots, N$ ) are as defined in (3.28)

$\mathbf{Y} = [y_1, y_2, \dots, y_{N+1}]^T$ , where  $\mathbf{y}_k = [y_{1k}, y_{0k}]^T$  ( $k = 1, 2, \dots, N$ ).

The last vector  $\mathbf{y}_{N+1} = [y_{1,N+1}, y_{0,N+1}]^T$  and in this case the two matrices

$\mathbf{B}_0$  and  $\mathbf{A}_{N+1}$  will be defined as  $\mathbf{B}_0 = \begin{bmatrix} 0 & -1 \\ 0 & 0 \end{bmatrix}$  and  $\mathbf{A}_{N+1} = \begin{bmatrix} a_{N+1,1,1} & 0 \\ a_{N+1,2,1} & 0 \end{bmatrix}$

respectively.

The vector  $\mathbf{G}$  is defined by

$$\mathbf{g}_k = \mathbf{0} \quad , \quad (k = 2, 3, 4, \dots, N), \quad \mathbf{g}_1 = [0, y_{0_0}]^T$$

$$\text{and } \mathbf{g}_{N+1} = [-a_{N+1,1,2} \quad y_{0_{N+1}}, -a_{N+1,2,2} \quad y_{0_{N+1}}]^T.$$

### 3.5.2 A Numerical Method Based on The(3,0)Pade

#### Approximant

Using the relation  $y(x+h) = e^{hD}y(x)$  and replacing the exponential term by its (3,0) Pade approximant gives:

$$[\mathbf{I} - h\mathbf{D} + \left(\frac{h^2}{2}\right)\mathbf{D}^2 - \left(\frac{h^3}{6}\right)\mathbf{D}^3]y(x+h) = y(x) + o(h^4)$$

Using (3.27) and its second and third derivatives leads to the finite difference method

$$\mathbf{A}_{k+1}\mathbf{y}_{k+1} + \mathbf{B}_k\mathbf{y}_k = \mathbf{0} \quad , \quad (k = 0, 1, 2, \dots, N) \quad (3.29)$$

Where  $B_k = -I$ ,  $I$  is the identity matrix,  $y_k = [y_{1_k}, y_{0_k}]^T$  and the elements of the matrix  $A_{k+1}$  are

$$a_{(k+1),1,1} = 1 + hp + \frac{h^2}{2}(-p' + p^2 - q) + \frac{h^3}{6}(p'' - 3pp' + 2q' + p^3 - 2pq)$$

$$a_{(k+1),1,2} = hq + \frac{h^2}{2}(-q' + pq) + \frac{h^3}{6}(q'' - 2p'q - pq' + p^2q - q^2)$$

$$a_{(k+1),2,1} = -h - \frac{h^2}{2}p + \frac{h^3}{6}(p' - p^2 + q)$$

$$a_{(k+1),2,2} = 1 - \frac{h^2}{2}q + \frac{h^3}{6}(q' - pq)$$

Here the functions  $p$ ,  $q$  and their first and second derivatives are functions of the independent variable  $x$  at the discrete point  $x_{k+1}$ .

After applying the vector-matrix equation (3.29) to the discrete points

$x_0, x_1, x_2, \dots, x_N$ . We obtain the system of linear equations with

$2(N+1)$  equations in  $2(N+1)$  unknowns:

$$A y = G$$

Where  $A_k$  and  $B_k$  are as defined in (3.29). The  $2(N+1)$  vector

$y = [y_0, y_1, y_2, \dots, y_N]^T$  contains the  $2 \times 1$  sub-vectors

$y_k$  ( $k = 0, 1, 2, \dots, N$ ), where  $y_k = [y_{1_k}, y_{0_k}]^T$ .

The last sub-vectors in  $y$  is  $y_{N+1} = [y_{1_{N+1}}, y_{1_0}]^T$ . Its structure depends on the type of boundary conditions. Also in this case the two matrices  $\mathbf{B}_0$  and  $\mathbf{A}_{N+1}$  are defined respectively as :

$$\mathbf{B}_0 = \begin{bmatrix} 0 & -1 \\ 0 & 0 \end{bmatrix} \text{ and } \mathbf{A}_{N+1} = \begin{bmatrix} a_{(N+1),1,1} & 0 \\ a_{(N+1),2,1} & 0 \end{bmatrix}$$

The vector  $\mathbf{G} = [g_1, g_2, \dots, g_{N+1}]^T$  contains the sub-vectors

$\mathbf{g}_k = \mathbf{0}$ , ( $k = 0, 1, 2, \dots, N$ ). The first and last sub-vectors  $\mathbf{g}_1$

and  $\mathbf{g}_{N+1}$  are updated, because of the boundary conditions, to

$\mathbf{g}_1 = [0, y_{0_0}]^T$  and  $\mathbf{g}_{N+1} = [-a_{(N+1),1,2} y_{0_{N+1}}, -a_{(N+1),2,2} y_{0_{N+1}}]^T$  respectively.

See [4]&[10]

**Example 3.1**

$$(x^\alpha y')' = \beta x^{(\alpha+\beta-2)}(\alpha + \beta - 1 + \beta x^\beta)y, \quad 0 < x \leq 1$$

With boundary conditions  $y(0) = 1$ ,  $y(1) = e$ .

With  $\alpha = 0.5$  and  $\beta = 4$ ,

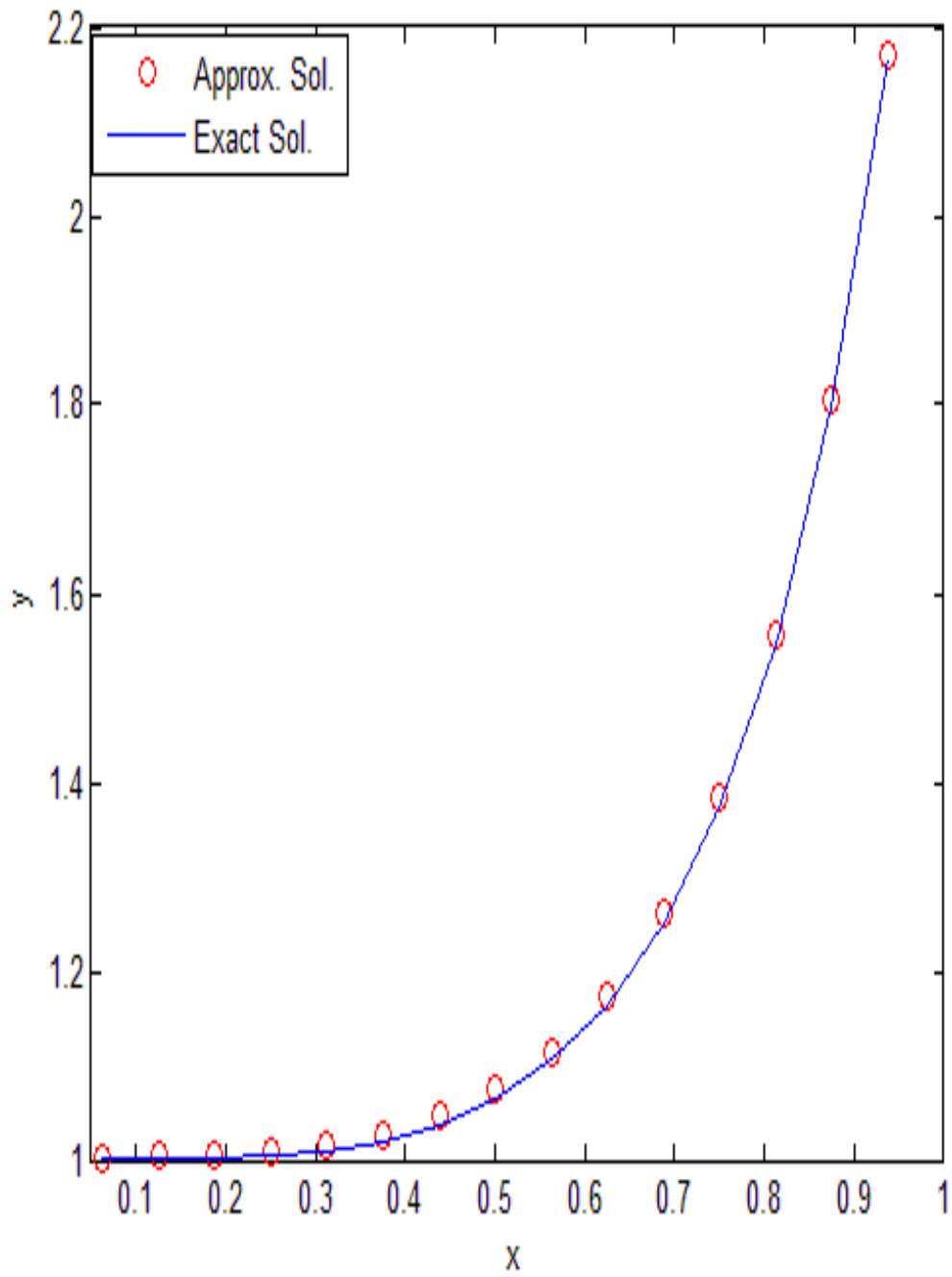
Which has the exact solution  $y(x) = e^{x^\beta}$ , this problem is linear and it has singularity at  $x = 0$ , The value listed as  $w_i$  approximates and  $y(x_i)$  is the exact solution and  $e e_i$  is the error between the exact solution and the approximate solution. Applying algorithm 2.3 to this problem the solutions results in table (3.1), program 8



**Table (3.1) :** The approximate and exact solution for example 3.1.

$x_i$	$w_i$	$y_i$	$ee_i =  w(x_i) - y(x_i) $
0.00000	1.000000	1.000000	0.000000
0.06250	1.002787	1.000015	0.002772
0.12500	1.004631	1.000244	0.004387
0.18750	1.006828	1.001236	0.005592
0.25000	1.010477	1.003913	0.006563
0.31250	1.016962	1.009582	0.007379
0.37500	1.028057	1.019972	0.008085
0.43750	1.046027	1.037315	0.008711
0.50000	1.073772	1.064494	0.009278
0.56250	1.115088	1.105295	0.009792
0.62500	1.175092	1.164844	0.010247
0.68750	1.260930	1.250325	0.010604
0.75000	1.382958	1.372187	0.010771
0.81250	1.556749	1.546209	0.010540
0.87500	1.806589	1.797113	0.009475
0.93750	2.171762	2.165120	0.006641
1.00000	2.718281	2.718281	0.000000

The maximum error is 0.010771



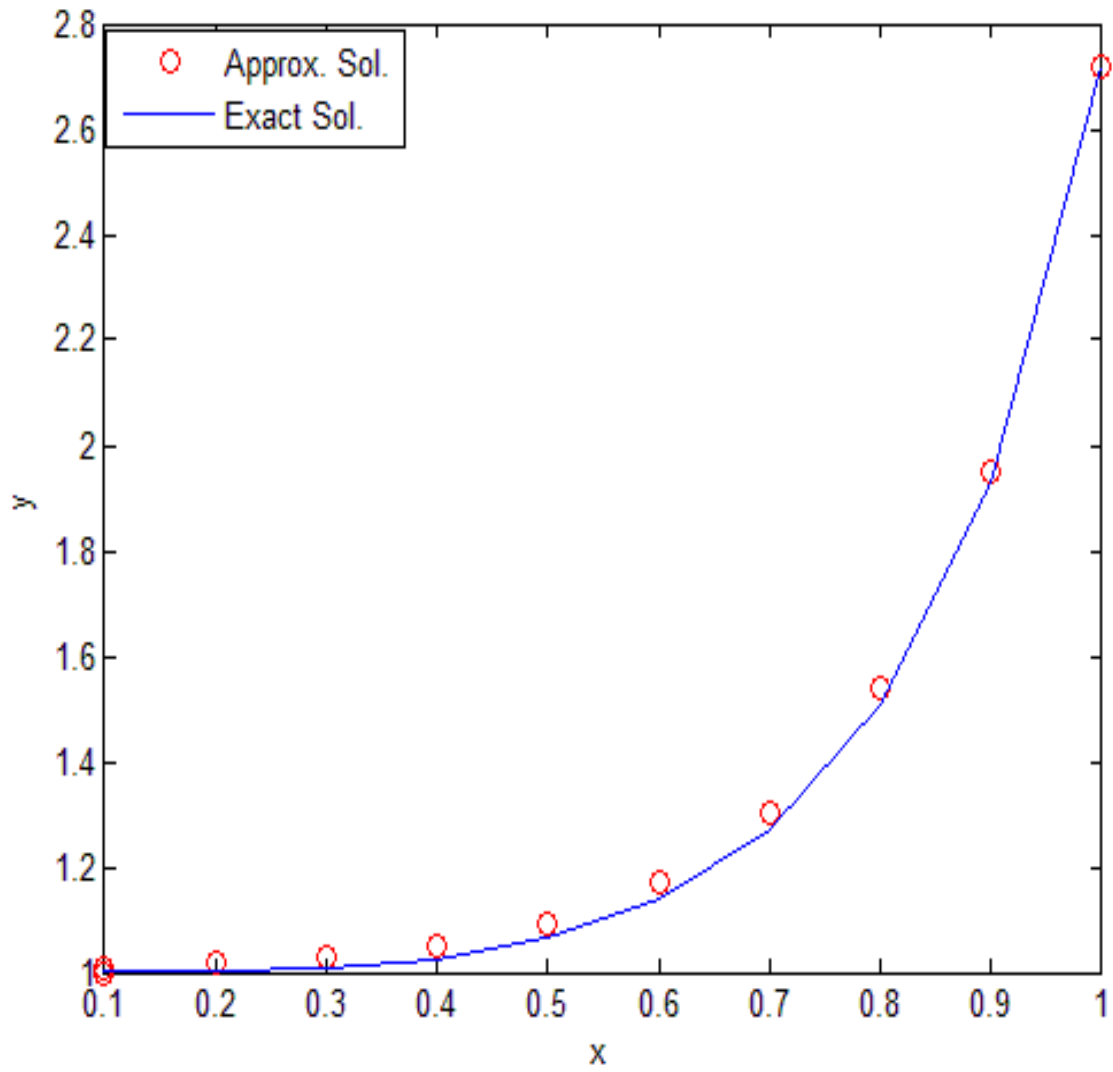
**Figure (9):** shows the approximate and the exact solution for example (3.1) that was solved by finite difference method.

Applying algorithm 2.1 to example 3.1 ,The results of the calculations with  $N = 10$  and  $h = 0.1$  are given in table (3.2).

**Table (3.2) :** The approximate and exact solution for example 3.1.

$x_i$	$w_i$	$y_i$	$ee_i =  w(x_i) - y(x_i) $
0.000	1.000000	1.000100	0.000100
0.100	1.009164	1.000100	0.009064
0.200	1.018424	1.001601	0.016823
0.300	1.030473	1.008132	0.022340
0.400	1.052422	1.025930	0.026491
0.500	1.094095	1.064494	0.029601
0.600	1.170135	1.138372	0.031762
0.700	1.304094	1.271376	0.032718
0.800	1.537564	1.506215	0.031348
0.900	1.951364	1.927261	0.024103
1.000	2.718281	2.718281	0.000000

The maximum error is 0.032718



**Figure (10):** shows the approximate and the exact solution for example (3.1) that was solved by shooting method.

To solve example (3.1) using pade (2,0) and pade (3,0) with  $\alpha = 0.5$  and  $\beta = 4$  the results in table (3.3) and (3.4) respectively.

**Table (3.3):** The approximate and exact solution for example 3.1.

1.0e+012 \*

$x_i$	$w_i$	$y_i$	$ee_i = \left  \frac{(w(x_i) - y(x_i))}{y(x_i)} \right $
0.000000000000006	-0.00000000000418	0.00000000000100	0.00000000000518
0.000000000000013	0.00000000002646	0.00000000000100	0.00000000002545
0.000000000000019	-0.00000000036398	0.00000000000100	0.00000000036453
0.000000000000025	0.00000000913803	0.00000000000100	0.00000000910140
0.000000000000031	-0.00000015338691	0.00000000000101	0.00000015193205
0.000000000000038	0.00000081648072	0.00000000000102	0.00000080049210
0.000000000000044	0.00000963647942	0.00000000000104	0.00000928982185
0.000000000000050	-0.00012760212962	0.00000000000106	0.00011987110841
0.000000000000056	-0.00001325637190	0.00000000000111	0.00001199350800
0.000000000000062	0.00600375866353	0.00000000000116	0.00515412737749
0.000000000000069	-0.01569498348483	0.00000000000125	0.01255271789526
0.000000000000075	-0.13140344030893	0.00000000000137	0.09576200872853
0.000000000000081	0.57807099917539	0.00000000000155	0.37386337482886
0.000000000000088	1.22280382687261	0.00000000000180	0.68042668483199
0.000000000000094	-9.02533317841833	0.00000000000217	4.16851236290096
0.000000000000100	0.00000000001333	0.00000000000272	0.00000000000391

**Table (3.4) :** The approximate and exact solution for example 3.1.

1.0e+014 \*

$x_i$	$w_i$	$y_i$	$ee_i = \left  \frac{(w(x_i) - y(x_i))}{y(x_i)} \right $
0.0000000000000000	-0.000000000000005	0.000000000000001	0.000000000000006
0.0000000000000000	0.000000000000078	0.000000000000001	0.000000000000077
0.0000000000000000	-0.00000000007366	0.000000000000001	0.00000000007358
0.0000000000000000	0.00000000366492	0.000000000000001	0.00000000365062
0.0000000000000000	-0.00000007717323	0.000000000000001	0.00000007644076
0.0000000000000000	0.00000053492313	0.000000000000001	0.00000052444872
0.0000000000000000	0.00000381175595	0.000000000000001	0.00000367463426
0.0000000000000001	-0.00006697411107	0.000000000000001	0.00006291635482
0.0000000000000001	0.00005736275989	0.000000000000001	0.00005189811113
0.0000000000000001	0.00294360222848	0.000000000000001	0.00252703376125
0.0000000000000001	-0.00953942058839	0.000000000000001	0.00762954963518
0.0000000000000001	-0.06223875592692	0.000000000000001	0.04535732302178
0.0000000000000001	0.31221754240677	0.000000000000002	0.20192451143853
0.0000000000000001	0.57403913840400	0.000000000000002	0.31942290277875
0.0000000000000001	-4.77720221691693	0.000000000000002	2.20643671625442
0.0000000000000001	0.00000000000012	0.000000000000003	0.000000000000003

The function `bvp4c` solves a class of singular BVPs of the form

$$y' = \frac{1}{x}Sy + f(x, y)$$

$$0 = g(y(0), y(b))$$

It can also accommodate unknown parameters for problems of the form

$$y' = \frac{1}{x}Sy + f(x, y, p)$$

$$0 = g(y(0), y(b), p)$$

Singular problems must be posed on an interval  $[0, b]$  with  $b > 0$ . Use `bvpset` to pass the constant matrix  $S$  to `bvp4c` as the value of the 'SingularTerm' integration property. Boundary conditions at  $x = 0$  must be consistent with the necessary condition for a smooth solution,  $Sy(0) = 0$ . An initial guess should also satisfy this necessary condition.

When you solve a singular BVP in Matlab using

```
sol = bvp4c(@odefun, @bcfun, solinit, options)
```

`bvp4c` requires that your function `odefun(x, y)` return only the value of the  $f(x, y)$  term.

**Example 3.2**

**Emden's equation.** Emden's equation arises in modeling a spherical body of gas. The PDE of the model is reduced by symmetry to the ODE

$$y'' + \frac{2}{x}y' + y^5 = 0$$

on an interval  $[0, 1]$ . The coefficient  $2/x$  is singular at  $x = 0$ , but symmetry implies the boundary condition  $y'(0) = 0$ . With this boundary condition, the term

$$\frac{2}{x}y'(0)$$

is well-defined as  $x$  approaches 0. For the boundary condition  $y(1) = \sqrt{3}/2$ , this BVP has the solution

$$y(x) = \left(1 + \frac{x^2}{3}\right)^{-1/2}$$

**Rewrite the problem as a first-order system and identify the singular term.** Using a substitution  $y_1 = y$  and  $y_2 = y'$ , write the differential equation as a system of two first-order equations

$$\begin{aligned} y_1' &= y_2 \\ y_2' &= -\frac{2}{x}y_2 - y_1^5 \end{aligned}$$

The boundary conditions become

$$\begin{aligned} y_2(0) &= 0 \\ y_1(1) &= \sqrt{3}/2 \end{aligned}$$

Writing the ODE system in a vector-matrix form



$$\begin{bmatrix} y_1' \\ y_2' \end{bmatrix} = \frac{1}{x} \begin{bmatrix} 0 & 0 \\ 0 & -2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} + \begin{bmatrix} y_2 \\ -y_1^5 \end{bmatrix}$$

the terms are identified as

$$S = \begin{bmatrix} 0 & 0 \\ 0 & -2 \end{bmatrix}$$

and

$$f(x, y) = \begin{bmatrix} y_2 \\ -y_1^5 \end{bmatrix}$$

**Code the ODE and boundary condition functions.** Code the differential equation and the boundary conditions as functions that `bvp4c` can use.

```
function dydx = emdenode(x,y)
```

```
dydx = [ y(2)
```

```
    -y(1)^5 ];
```

```
function res = emdenbc(ya,yb)
```

```
res = [ ya(2)
```

```
    yb(1) - sqrt(3)/2 ];
```

**Setup integration properties.** Use the matrix as the value of the 'Singular Term' integration property.

```
S = [0,0;0,-2];
```

```
options = bvpset('SingularTerm',S);
```

Create an initial guess. This example starts with a mesh of five points and a constant guess for the solution.

$$y_1(x) \equiv \sqrt{3}/2$$

$$y_2(x) \equiv 0$$

Use `bvpinit` to form the guess structure

```
guess = [sqrt(3)/2;0];
```

```
solinit = bvpinit(linspace(0,1,5),guess);
```

**Solve the problem in Matlab.** Use the standard `bvp4c` syntax to solve the problem.

```
sol = bvp4c(@emdenode,@emdenbc,solinit,options);
```

**View the results.** This problem has solution

$$y(x) = \left(1 + \frac{x^2}{3}\right)^{-1/2}$$

The example evaluates the solution at 100 equally spaced points and plots it along with the numerical solution computed using `bvp4c`.

```
x = linspace(0,1);
```

```
truy = 1 ./ sqrt(1 + (x.^2)/3);
```

```
plot(x,truy,sol.x,sol.y(1,:),'ro');
```

```
title('Emden problem -- BVP with singular term.')
```

```
legend('Computed');
```

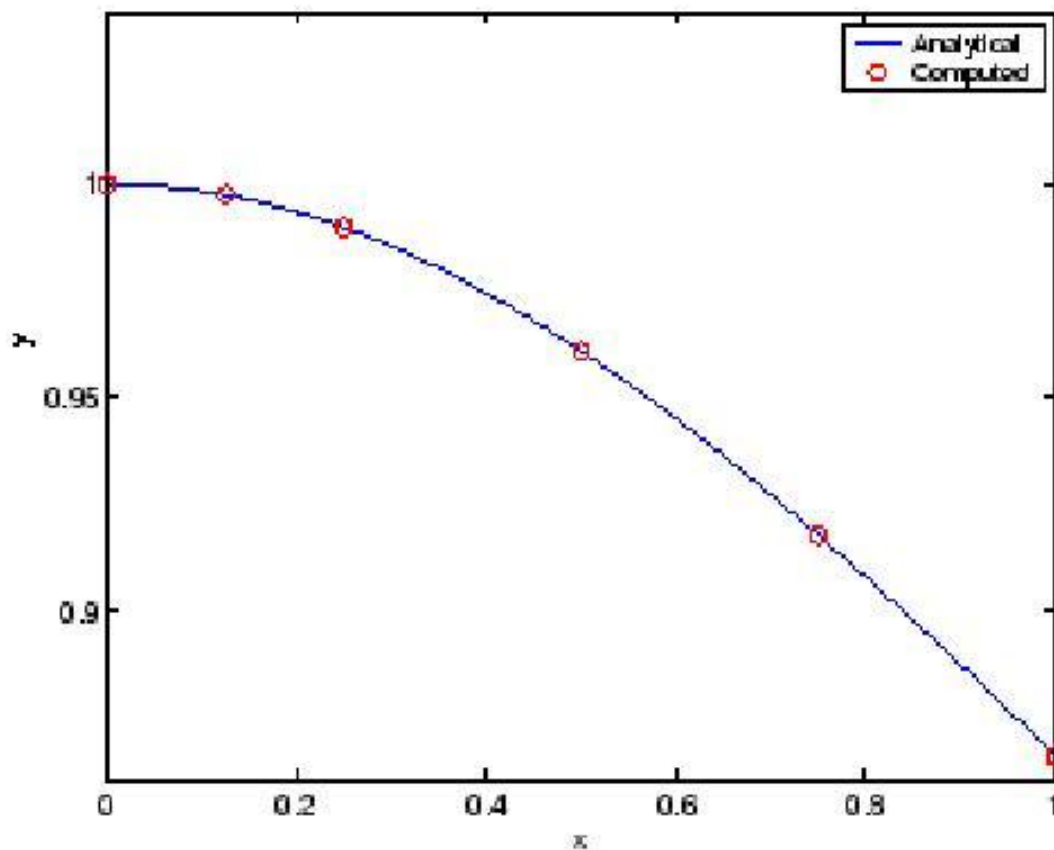
```
xlabel('x');
```

ylabel('solution y'); See [14]

**Table (3.5) :** The approximate and exact solution for example 3.2.

$x_i$	$w_i$	$y_i$	$ee_i =  w(x_i) - y(x_i) $
0.000	1.0000005109	1.0000000000	0.0000005109
0.125	0.9974064705	0.9974059619	0.0000005086
0.250	0.9897438277	0.9897433186	0.0000005091
0.500	0.9607707751	0.9607689228	0.0000018522
0.750	0.9176644937	0.9176629354	0.0000015582
1.000	0.8660254037	0.8660254037	0.0000000000

The maximum error is 0.0000018522



**Figure( 11):** Emden problem – BVP with singular term.

## **Conclusion and Results**

Various numerical methods namely: Finite Difference, Shooting, Pade Approximant, have been studied to compare the efficiency of these methods. It is desirable to develop the results to Pade Approximant to obtain more accurate for the approximate solution.

From the above work, we see that shooting method is more accurate for non-linear problems, on the other hand, finite difference method is more accurate for linear problems.

## **Appendix**

## Matlab Programs

### Program 1:

```
format long
```

```
`
```

```
clc
```

```
a=1;
```

```
b=2;
```

```
h=1/10;
```

```
n=10;
```

```
x(1)=1;
```

```
m(1)=1;
```

```
w(1)=0;
```

```
m1(1)=0;
```

```
w1(1)=1;
```

```
k=zeros(4,2);
```

```
for i=1:n
```

```
    k(1,1)=h*w(i);
```

```
    k(1,2)=h*[(-2/x(i))*w(i)+(2/(x(i))^2)*m(i)+(sin(log(x(i))))/(x(i))^2];
```

```
    k(2,1)=h*[w(i)+(1/2)*k(1,2)];
```

```
    k(2,2)=h*[(-2/(x(i)+(h/2)))*(w(i)+(1/2)*k(1,2))+2/(x(i)+(h/2))^2*  
(m(i)+(1/2)*k(1,1))+sin(log(x(i)+(h/2)))/(x(i))^2];
```

```
    k(3,1)=h*[w(i)+(1/2)*k(2,2)];
```

```
    k(3,2)=h*[(-2/(x(i)+(h/2)))*(w(i)+(1/2)*k(2,2))+2/(x(i)+(h/2))^2*  
(m(i)+(1/2)*k(2,1))+sin(log(x(i)+(h/2)))/(x(i))^2];;
```

```

k(4,1)=h*[w(i)+(1/2)*k(3,2)];
k(4,2)=h*[(-2/(x(i)+(h))))*(w(i)+k(3,2))+(2/(x(i)+(h))^2)*
(m(i)+k(3,1))+sin(log(x(i)+(h)))/(x(i))^2];
m(i+1)=m(i)+(1/6)*(k(1,1)+2*k(2,1)+2*k(3,1)+k(4,1));
w(i+1)=w(i)+(1/6)*(k(1,2)+2*k(2,2)+2*k(3,2)+k(4,2));
x(i+1)=a+h*i;
end
kk=zeros(4,2);
for i=1:n
    kk(1,1)=h*w1(i);
    kk(1,2)=h*[(-2/x(i))*w1(i)+(2/(x(i))^2)*m1(i)];
    kk(2,1)=h*[w1(i)+(1/2)*k(1,2)];
    kk(2,2)=h*[(-2/(x(i)+(h/2))))*(w1(i)+(1/2)*k(1,2))+(2/(x(i)+(h/2))^2)
*(m1(i)+(1/2)*k(1,1))];
    kk(3,1)=h*[w1(i)+(1/2)*k(2,2)];
    kk(3,2)=h*[(-2/(x(i)+(h/2))))*(w1(i)+(1/2)*k(2,2))+(2/(x(i)+(h/2))^2)
*(m1(i)+(1/2)*k(2,1))];
    kk(4,1)=h*[w1(i)+(1/2)*k(3,2)];
    kk(4,2)=h*[(-2/(x(i)+(h))))*(w1(i)+k(3,2))+(2/(x(i)+(h))^2)
*(m1(i)+k(3,1))];

    m1(i+1)=m1(i)+(1/6)*(kk(1,1)+2*kk(2,1)+2*kk(3,1)+kk(4,1));
    w1(i+1)=w1(i)+(1/6)*(kk(1,2)+2*kk(2,2)+2*kk(3,2)+kk(4,2));
end

```

```

y=m+((2-m(n+1))/m1(n+1))*m1;
for j=1:n+1
    yy(j)=1.1392070132*x(j)+(-0.03920701320/(x(j))^2)-
(3/10)*sin(log(x(j)))-(1/10)*cos(log(x(j)));
    ee(j)=abs(y(j)-yy(j));
end
[x' y' yy' ee']
plot (x,y,x,yy)
legend('Approx. Sol.','Exact Sol.');
```

## Program 2

```

format long
clear
clc
a=0;
b=1;
h=1/20;
n=20;
x(1)=0;
m(1)=0;
w(1)=0;
m1(1)=0;
w1(1)=1;
k=zeros(4,2);
```



for i=1:n

$$k(1,1)=h*w(i);$$

$$k(1,2)=h*[(0)*w(i)+(4)*m(i)+(-4*(x(i)))];$$

$$k(2,1)=h*[w(i)+(1/2)*k(1,2)];$$

$$k(2,2)=h*[(0)+(4)*(m(i)+(1/2)*k(1,1))-(4*(x(i)+(h/2)))];$$

$$k(3,1)=h*[w(i)+(1/2)*k(2,2)];$$

$$k(3,2)=h*[(0)+(4)*(m(i)+(1/2)*k(2,1))-(4*(x(i)+(h/2)))];$$

$$k(4,1)=h*[w(i)+(1/2)*k(3,2)];$$

$$k(4,2)=h*[(0)+(4)*(m(i)+k(3,1))-(4*(x(i)+h))];$$

$$m(i+1)=m(i)+(1/6)*(k(1,1)+2*k(2,1)+2*k(3,1)+k(4,1));$$

$$w(i+1)=w(i)+(1/6)*(k(1,2)+2*k(2,2)+2*k(3,2)+k(4,2));$$

$$x(i+1)=a+h*i;$$

end

kk=zeros(4,2);

for i=1:n

$$kk(1,1)=h*w1(i);$$

$$kk(1,2)=h*[(0)+4*m1(i)];$$

$$kk(2,1)=h*[w1(i)+(1/2)*kk(1,2)];$$

$$kk(2,2)=h*[0+(4)*(m1(i)+(1/2)*kk(1,1))];$$

$$kk(3,1)=h*[w1(i)+(1/2)*kk(2,2)];$$

$$kk(3,2)=h*[(0)+(4)*(m1(i)+(1/2)*kk(2,1))];$$

$$kk(4,1)=h*[w1(i)+(1/2)*kk(3,2)];$$

$$kk(4,2)=h*[(0)+(4)*(m1(i)+kk(3,1))];$$

$$m1(i+1)=m1(i)+(1/6)*(kk(1,1)+2*kk(2,1)+2*kk(3,1)+kk(4,1));$$

```

w1(i+1)=w1(i)+(1/6)*(kk(1,2)+2*kk(2,2)+2*kk(3,2)+kk(4,2));
end
y=m+((2-m(n+1))/m1(n+1))*m1;
for j=1:n+1
    yy(j)=(exp(1))^(2)*(((exp(1))^(4)-1)^(-1))*((exp(1))^(2*x(j))-((exp(1))^(-
2*x(j))))+x(j);
ee(j)=y(j)-yy(j);
end
[x' y' yy' ee']
plot (x,y,x,yy)
legend('Approx. Sol.','Exact Sol.');
```

### program 3

```

% To approximate the solution of the nonlinear boundary-value problem
%       $Y'' = F(X,Y,Y')$ ,  $A \leq X \leq B$ ,  $Y(A) = \text{ALPHA}$ ,  $Y(B) = \text{BETA}$ :
% INPUT:  Endpoints A,B; boundary conditions ALPHA, BETA; number
% of subintervals N; tolerance TOL; maximum number of iterations M.
% OUTPUT: Approximations W(1,I) TO Y(X(I)); W(2,I) TO Y'(X(I))
% for each I=0,1,...,N or a message that the maximum number of iterations
was exceeded.

syms('OK', 'A', 'B', 'ALPHA', 'BETA', 'TK', 'AA', 'N');
syms('TOL', 'NN', 'FLAG', 'NAME', 'OUP', 'H', 'K', 'W1', 'YY1', 'EE');
syms('W2', 'U1', 'U2', 'T', 'X', 'T', 'K11', 'K12', 'K21');
syms('K22', 'K31', 'K32', 'K41', 'K42', 'J', 's', 'x', 'y', 'z');
```

```

TRUE = 1;
FALSE = 0;
fprintf(1,'This is the Nonlinear Shooting Method.\n');
fprintf(1,'Input the function F(X,Y,Z) in terms of x, y, z.\n');
fprintf(1,'followed by the partial of F with respect to y on the \n');
fprintf(1,'next line followed by the partial of F with respect to \n');
fprintf(1,'z or y-prime on the next line. \n');
fprintf(1,'actual solution .\n');
fprintf(1,'For example: (32+2*x^3-y*z)/8 \n');
fprintf(1,'      -z/8 \n');
fprintf(1,'      -y/8 \n');
fprintf(1,'      x^2+16/x \n');
s = input(' ', 's');
F = inline(s,'x','y','z');
s = input(' ', 's');
FY = inline(s,'x','y','z');
s = input(' ', 's');
FYP = inline(s,'x','y','z');
s = input(' ', 's');
YY11 = inline(s,'x','y','z');
OK = FALSE;
while OK == FALSE
fprintf(1,'Input left and right endpoints on separate lines.\n');
A = input(' ');

```

```
B = input(' ');
if A >= B
fprintf(1,'Left endpoint must be less than right endpoint.\n');
else OK = TRUE;
end;
end;
fprintf(1,'Input Y(%.10e).\n', A);
ALPHA = input(' ');
fprintf(1,'Input Y(%.10e).\n', B);
BETA = input(' ');
TK = (BETA-ALPHA)/(B-A);
fprintf(1,'TK = %.8e\n', TK);
fprintf(1,'Input new TK? Enter Y or N.\n');
AA = input(' ','s');
if AA == 'Y' | AA == 'y'
fprintf(1,'input new TK\n');
TK = input(' ');
end;
OK = FALSE;
while OK == FALSE
fprintf(1,'Input an integer > 1 for the number of subintervals.\n');
N = input(' ');
if N <= 1
fprintf(1,'Number must exceed 1.\n');
```

```
else
OK = TRUE;
end;
end;
OK = FALSE;
while OK == FALSE
fprintf(1,'Input Tolerance.\n');
TOL = input(' ');
if TOL <= 0
fprintf(1,'Tolerance must be positive.\n');
else
OK = TRUE;
end;
end;
OK = FALSE;
while OK == FALSE
fprintf(1,'Input maximum number of iterations.\n');
NN = input(' ');
if NN <= 0
fprintf(1,'Must be positive integer.\n');
else
OK = TRUE;
end;
end;
```

```

if OK == TRUE

fprintf(1,'Choice of output method:\n');

fprintf(1,'1. Output to screen\n');

fprintf(1,'2. Output to text File\n');

fprintf(1,'Please enter 1 or 2.\n');

FLAG = input(' ');

if FLAG == 2

fprintf(1,'Input the file name in the form - drive:\\name.ext\n');

fprintf(1,'for example A:\\OUTPUT.DTA\n');

NAME = input(' ','s');

OUP = fopen(NAME,'wt');

else

OUP = 1;

end;

fprintf(OUP, 'NONLINEAR SHOOTING METHOD\n\n');

fprintf(OUP, ' X(I)      W1(I)      YY1(I)      EE(I)\n');

% STEP 1

W1 = zeros(1,N+1);

W2 = zeros(1,N+1);

H = (B-A)/N;

K = 1;

% TK already computed

OK = FALSE;

% STEP 2

```

```

while K <= NN & OK == FALSE
% STEP 3
W1(1) = ALPHA;
W2(1) = TK;
U1 = 0 ;
U2 = 1;
% STEP 4
% Runge-Kutta method for systems is used in STEPS 5 and 6
for I = 1 : N
% STEP 5
X = A+(I-1)*H;
T = X+0.5*H;
% STEP 6
K11 = H*W2(I);
K12 = H*F(X,W1(I),W2(I));
K21 = H*(W2(I)+0.5*K12);
K22 = H*F(T,W1(I)+0.5*K11,W2(I)+0.5*K12);
K31 = H*(W2(I)+0.5*K22);
K32 = H*F(T,W1(I)+0.5*K21,W2(I)+0.5*K22);
K41 = H*(W2(I)+K32);
K42 = H*F(X+H,W1(I)+K31,W2(I)+K32);
W1(I+1) = W1(I)+(K11+2*(K21+K31)+K41)/6;
W2(I+1) = W2(I)+(K12+2*(K22+K32)+K42)/6;
K11 = H*U2;

```

$$K12 = H*(FY(X,W1(I),W2(I))*U1+FYP(X,W1(I),W2(I))*U2);$$

$$K21 = H*(U2+0.5*K12);$$

$$K22=H*(FY(T,W1(I),W2(I))*(U1+0.5*K11)+FYP(T,W1(I),W2(I))*(U2+0.5*K21));$$

$$K31 = H*(U2+0.5*K22);$$

$$K32=H*(FY(T,W1(I),W2(I))*(U1+0.5*K21)+FYP(T,W1(I),W2(I))*(U2+0.5*K22));$$

$$K41 = H*(U2+K32);$$

$$K42=H*(FY(X+H,W1(I),W2(I))*(U1+K31)+FYP(X+H,W1(I),W2(I))*(U2+K32));$$

$$U1 = U1+(K11+2*(K21+K31)+K41)/6;$$

$$U2 = U2+(K12+2*(K22+K32)+K42)/6;$$

end;

% STEP 7

% test for accuracy

if abs(W1(N+1)-BETA) < TOL

% STEP 8

I = 0;

Fprintf(OUP, '%3d %13.8f %13.8f %13.8f\n', I, A, ALPHA, TK);

for I = 1 : N

J = I+1;

X = A+I\*H;



```

YY1(J)=YY11(X,W1(J),W2(J));
EE(J)=abs(YY1(J)-W1(J));
Fprintf(OUP, ' %13.8f %13.8f %13.8f %13.8f \n', X, W1(J),
YY1(J),EE(J));
end;
fprintf(OUP, 'Convergence in %d iterations\n', K);
fprintf(OUP, ' t = %14.7e\n', TK);
% STEP 9
OK = TRUE;
else
% STEP 10
% Newton's method applied to improve TK
TK = TK-(W1(N+1)-BETA)/U1;
K = K+1;
end;
end;
% STEP 11
% method failed
if OK == FALSE
fprintf(OUP, 'Method failed after %d iterations\n', NN);
end;
end;
if OUP ~= 1
fclose(OUP);

```

```
fprintf(1,'Output file %s created successfully \n',NAME);
```

```
end;
```

#### **program 4**

```
format long
```

```
clear
```

```
clc
```

```
a=1;
```

```
b=2;
```

```
h=1/10;
```

```
n=11;
```

```
y=zeros(10,1);
```

```
for i=1:10
```

```
    x(i)=a+(i)*h;
```

```
    yy(i)=1.1392070132*x(i)+(-0.03920701320/(x(i)^2))-
```

```
(3/10)*sin(log(x(i)))-(1/10)*cos(log(x(i)));
```

```
end
```

```
[x' yy'];
```

```
u=zeros(9,9);
```

```
bb=zeros(9,1);
```

```
u(1,1)=-((2*(h)^2/(x(1))^2)+2);
```

```
u(1,2)=(1+(h/x(1)));
```

```
u(2,1)=(1-(h/x(2)));
```

```
bb(1)=((h)^2)*sin(log(x(1)))/(x(1))^2-(1-(h/x(1)));
```

```
for j=2:8
```

```

for k=2:8
    if j==k
        u(j,k)=-1*((2*(h)^(2)/(x(j))^(2))+2);
    elseif j==k+1
        u(j,k)=(1-(h/x(j)));
    elseif j==k-1
        u(j,k)=(1)+(h/x(j));
    else u(j,k)=0;
    end
end
end
u(9,8)=(1-(h/x(9)));
u(9,9)=-((2*(h)^(2)/(x(9))^(2))+2);
u(8,9)=(1+(h/x(8)));
u;
for oo=2:8
    ll(oo)=log(x(oo));
    bb(oo)=((h)^(2)*sin(ll(oo)))/(x(oo))^(2);
end
bb(9)=((h)^2)*sin(log(x(9)))/(x(9))^(2)-(2*(1+(h/x(9))));
[L U]=lu(u);
z=inv(L)*bb;
y=inv(U)*z;
y(10)=2;

```

```

for rr=1:10
    ee(rr)=abs(y(rr)-yy(rr));
end
[x' y yy' ee']
plot(x,y,x,yy);
xlabel('Time');
legend('Approx. Sol.','Exact Sol.');
```

### **program 5**

```

clc
clear
format long
a=0;
b=1;
n=19;
h=1/20;
y=zeros(20,1);
%y(1)=1;
for i=1:20
    x(i)=a+(i)*h;
end
for jj=1:20
    yy(jj)=(exp(1))^2*(((exp(1))^4-1)^(-1))*((exp(1))^(2*x(jj))-
((exp(1))^(2*x(jj))))+x(jj);
```

```
end
[x' yy'];
u=zeros(19,19);
bb=zeros(19,1);
u(1,1)=-2+(4*(h)^2);
u(1,2)=(1);
u(2,1)=(1);
bb(1)=(-4*x(1)*(h)^2);
for j=2:18
    for k=2:18
        if j==k
            u(j,k)=-1*(2+(4*(h)^2));
        elseif j==k+1
            u(j,k)=(1);
        elseif j==k-1
            u(j,k)=(1);
        else u(j,k)=0;
        end
    end
end
end
u(19,18)=(1);
u(19,19)=-2+(4*(h)^2);
u(18,19)=(1);
u;
```

```

for oo=2:18
    bb(oo)=(-4*(x(oo))*(h)^(2));
end
bb(19)=(-4*(x(oo))*(h)^(2))-2;
bb;
[L U]=lu(u);
z=inv(L)*bb;
y=inv(U)*z;
y(20)=2;
for rr=1:20
    ee(rr)=abs(y(rr)-yy(rr));
end
[x' y yy' ee']
plot(x,y,x,yy);
xlabel('Time');
legend('Approx. Sol.','Exact Sol.');
```

### **program 6**

```

clc
format long
clear
a=1;
b=3;
y0=17;
```

```

h=0.1;
n=(b-a)/h-1;
yn=(43/3);

x=[a+h:h:b-h];
y=17*ones(n,1);k=1;
c=zeros(n,1);j=zeros(19,19);m=50;
while norm (c-y ,inf)>0.000001
    %finding he jacobian matrix
    j(1,1)=-2+(1/16)*h*(y(2)-y0);
    j(1,2)=1+(1/16)*h*y(1);
    for i=2:n-1
        for t=2:n-1
            if i==t
                j(i,t)=-2+(1/16)*h*(y(i+1)-y(i-1));
            elseif i==t+1
                j(i,t)=1-(1/16)*h*y(i);
            elseif i==t-1
                j(i,t)=1+(1/16)*h*y(i);
            end
        end
    end
end
j(n-1,n)=1+(1/16)*h*y(n-1);
j(n,n-1)=1-(1/16)*h*yn;

```

```

j(n,n)=-2+(1/16)*h*yn-(1/8)*h*y(n-1);

%=====

f(1,1)=(y(2)-2*y(1)+y0)-((h)^2/8)*(32+(2*(x(1).^3))-(y(1)*((y(2)-
y0)/(2*h))));

    for i=2:n-1

        f(i,1)= (y(i+1)-2*y(i)+y(i-1))-((h)^2/8)*(32+(2*(x(i).^3))-
(y(i)*((y(i+1)-y(i-1))/(2*h))));

    end

    f(n,1)=(yn-2*y(n)+y(n-1))-((h)^2/8)*(32+(2*(x(n)^3))-(y(n)*((yn-y(n-
1))/(2*h))));

    f;

    z=inv(j)*(-1*f);

    c=y;

    y=y+z;

    y

    end

xx=[a,x,b];

y1=[y0,y',yn];

for tt=1:n+2

    yy(tt)=(xx(tt))^2+(16/xx(tt));

    e(tt)=abs(y1(tt)-yy(tt));

end

[xx;y1;yy;e]'

```



```

    plot(xx,y1,xx,yy);
xlabel('Time');
legend('Approx. Sol.','Exact Sol.');
```

### **program 7**

```

clc
format long
clear
a=1;
b=2;
y0=2;
h=0.1;
n=(b-a)/h-1;
yn=(5/2);
x=[a+h:h:b-h];
y=2*ones(n,1);k=1;
c=zeros(n,1);j=zeros(n,n);m=50;
while norm (c-y ,inf)>0.000001
    %finding he jacobian matrix
    j(1,1)=((-2/(h)^2)-2*(y(1))^2+6);
    j(1,2)=1/(h)^2;
    for i=2:n-1
        for t=2:n-1
            if i==t
```

```

    j(i,t)=((-2/(h)^2)-2*(y(i))^2+6);
elseif i==t+1
    j(i,t)=1/(h)^2;
elseif i==t-1
    j(i,t)=1/(h)^2;
end
end
end
j(n-1,n)=1/(h)^2;
j(n,n-1)=1/(h)^2;
j(n,n)=((-2/(h)^2)-2*(y(n))^2+6);
%=====
f(1,1)=(y(2)-2*y(1)+y0)+((h)^2)*(-2*(y(1))^3+6*(y(1))+2*(x(1))^3);
for i=2:n-1
    f(i,1)=(y(i+1)-2*y(i)+y(i-1))+((h)^2)*(-
2*(y(i)^3)+6*(y(i))+2*(x(i))^3);
end
f(n,1)=(yn-2*y(n)+y(n-1))+((h)^2)*(-2*(y(n)^3)+6*(y(n))+2*(x(n))^3);
f;
z=inv(j)*(-1*f);
c=y;
y=y+z;
y;
end

```

```
xx=[a,x,b];
y1=[y0,y',yn];
for tt=1:n+2
    yy(tt)=(xx(tt))+xx(tt)^(-1);
    e(tt)=abs(y1(tt)-yy(tt));
end
[xx;y1;yy;e]'
plot(xx,y1,xx,yy);
xlabel('Time');
legend('Approx. Sol.','Exact Sol.');
```

### **program 8**

```
clc
clear
ba=0;
bb=1;
n=15;
h=1/16;
alpha=1;
beta=2.718281828;
aa=0.5;
k=4;
for i=1:n
    t(i)=ba+i*h;
```

```

end

t

for j=1:n
p1((j))=aa/t(j);
q1((j))=(k*(t(j))^(aa+k-2)*(aa+k-1+k*(t(j))^(k)))/(t(j)^(aa));
end

[t' p1' q1']

a(1)=(-4-2*((h)^2)*q1((1)));
c(1)=(2+h*p1((1)));
for i=2:n-1
    a(i)=(-4-2*((h)^2)*q1((i)));
    c(i)=(2+h*p1((i)));
    d(i)=(2-h*p1((i)));
end

a(n)=-4-2*((h)^2)*q1((n));
d(n)=2-h*p1((n));
l(1)=a(1);u(1)=c(1)/l(1);
for i=2:n-1
    l(i)=a(i)-d(i)*u(i-1);
    u(i)=c(i)/l(i);
end

l(n)=a(n)-d(n)*u(n-1);
dd(1)=-((2-h*p1(1))*alpha);
for i=2:n-1

```

```
    dd(i)=0;
end
dd(n)=-(2+h*p1(n))*beta;
z(1)=dd(1)/l(1);
for i=2:n
    z(i)=(dd(i)-d(i)*z(i-1))/l(i);
end
y(n)=z(n);
for i=n-1:-1:1
    y(i)=z(i)-u(i)*y(i+1);
end
y
for i=1:n
    yy(i)=(2.718281828)^((t(i))^k);
end
yy
for i=1:n
    ee(i)=y(i)-yy(i);
end
ee
[t' y' yy' ee']
plot(t',y',t',yy')
legend('Approx. Sol.','Exact Sol.');
```

**program 9**

```
format long
```

```
clear
```

```
clc
```

```
a=0;
```

```
b=1;
```

```
aa=0.5;
```

```
k1=4;
```

```
h=1/10;
```

```
n=10;
```

```
x(1)=0.1;
```

```
m(1)=1;
```

```
w(1)=0;
```

```
m1(1)=0;
```

```
w1(1)=1;
```

```
k=zeros(4,2);
```

```
for i=1:n
```

```
    k(1,1)=h*w(i);
```

```
    k(1,2)=h*[(-aa/x(i))*w(i)+((((k1*(x(i))^(aa+k1-2))*(aa+k1-1+k1*(x(i))^(k1)))/(x(i)^(aa))*m(i)+(0*(x(i))))];
```

```
    k(2,1)=h*[w(i)+(1/2)*k(1,2)];
```

```
    k(2,2)=h*[(-aa/(x(i)+(h/2)))*(w(i)+(1/2)*k(1,2))+((((k1*((x(i)+(h/2))^(aa+k1-2))*(aa+k1-1+k1*(x(i)+(h/2))^(k1)))/(x(i)+(h/2))^(aa))*m(i)+(1/2)*k(1,1)))-(0*(x(i)+(h/2)))];
```

```

k(3,1)=h*[w(i)+(1/2)*k(2,2)];
k(3,2)=h*[(-aa/(x(i)+(h/2)))*(w(i)+(1/2)*k(2,2))+
((((k1*((x(i)+(h/2))^(aa+k1-2))*(aa+k1-1+k1*(x(i)+(h/2))^(k1)))/(x(i)
+(h/2))^(aa)))*(m(i)+(1/2)*k(2,1)))-(0*(x(i)+(h/2)))]);
k(4,1)=h*[w(i)+(1/2)*k(3,2)];
k(4,2)=h*[(-aa/(x(i)+h))*(w(i)+k(3,2))+((((k1*((x(i)+h)^(aa+k1-
2))*(aa+k1-1+k1*(x(i)+h)^(k1)))/(x(i)+h)^(aa)))*(m(i)+k(3,1)))-
(0*(x(i)+h)))]);
m(i+1)=m(i)+(1/6)*(k(1,1)+2*k(2,1)+2*k(3,1)+k(4,1));
w(i+1)=w(i)+(1/6)*(k(1,2)+2*k(2,2)+2*k(3,2)+k(4,2));
x(i+1)=a+h*i;
end
x
kk=zeros(4,2);
for i=1:n
    kk(1,1)=h*w1(i);
    kk(1,2)=h*[(-aa/x(i))*w1(i)+((((k1*(x(i))^(aa+k1-2))*(aa+k1-
1+k1*(x(i))^(k1)))/(x(i))^(aa))*m1(i)]);
    kk(2,1)=h*[w1(i)+(1/2)*kk(1,2)];
    kk(2,2)=h*[(-aa/(x(i)+(h/2)))*(w1(i)+(1/2)*kk(1,2))
+((((k1*(x(i)+(h/2))^(aa+k1-2))*(aa+k1-1+k1*(x(i)+(h/2))^(k1)))/
(x(i)+(h/2))^(aa)))*(m1(i)+(1/2)*kk(1,1)))]);
    kk(3,1)=h*[w1(i)+(1/2)*kk(2,2)];
    kk(3,2)=h*[(-aa/(x(i)+(h/2)))*(w1(i)+(1/2)*kk(2,2))

```

```

+((((k1*((x(i)+(h/2))^(aa+k1-2))*(aa+k1-1+k1*(x(i)+(h/2))^(k1)))/
(x(i)+(h/2))^(aa)))*(m1(i)+(1/2)*kk(2,1))]);
kk(4,1)=h*[w1(i)+(1/2)*kk(3,2)];
kk(4,2)=h*[(-aa/(x(i)+h))*(w1(i)+kk(3,2))+((((k1*((x(i)+h)^(aa+k1-
2))*(aa+k1-1+k1*(x(i)+h)^(k1)))/(x(i)+h)^(aa)))*(m1(i)+kk(3,1)))]];
m1(i+1)=m1(i)+(1/6)*(kk(1,1)+2*kk(2,1)+2*kk(3,1)+kk(4,1));
w1(i+1)=w1(i)+(1/6)*(kk(1,2)+2*kk(2,2)+2*kk(3,2)+kk(4,2));
end
y=m+(((exp(1))-m(n+1))/m1(n+1))*m1;
for j=1:n+1
yy(j)=(2.718281828)^(x(j))^k1;
ee(j)=abs(y(j)-yy(j));
end
[x' y' yy' ee']
plot (x,y,x,yy)
legend('Approx. Sol.','Exact Sol.');
```

### program 10

```

%PADE APPROXIMATE METHOD (2,0) TO SOLVE BVP
%((x)^(aa)*y)'=k*x(aa+k-2)*(aa+k-1+k*(x)^k)*y
%0<x<1      y(0)=1   y(1)=e(1)
%=====
clc
clear
```



format long

D=[0 1];

n=15;

a=.5;

b=4;

h=(D(2)-D(1))/(n+1);

xx=D(1)+h+[0:h:D(2)-h];

xx

P=inline('-a/x','x','a');

Q=inline('(b\*x^(a+b-2)\*(a+b-1+b\*x^b))/(x^a)','x','a','b');

PP=inline('a/(x^2)','x','a');

QQ=inline('28\*x+96\*x^5','x');

PPP=inline('-1/(x)','x');

QQQ=inline('28+480\*x^4','x');

f=[-1 0;0 -1]

for i=1:1:n+1

aa(1,1)=h\*P(xx(i),a)+(h^2\*(-PP(xx(i),a)+(P(xx(i),a))^2-  
 Q(xx(i),a,b)))/2+h^3\*(PPP(  
 xx(i))-3\*P(xx(i),a)\*PP(xx(i),a)+2\*QQ(xx(i))+(P(xx(i),a))^3-  
 2\*P(xx(i),a)\*Q(xx(i),a,b))/6;

aa(1,2)=h\*Q(xx(i),a,b)+h^2\*(-  
 QQ(xx(i))+P(xx(i),a)\*Q(xx(i),a,b))/2+(h^3\*(QQQ(xx(i))-2\*

```

PP(xx(i),a)*Q(xx(i),a,b)-
P(xx(i),a)*QQ(xx(i))+(P(xx(i),a))^2*Q(xx(i),a,b)-(Q(xx(i),a,b))^2))/6;
aa(2,1)=-h-(h^2*P(xx(i),a))/2+h^3((PP(xx(i),a)-
(P(xx(i),a))^2+Q(xx(i),a,b)))/6;
aa(2,2)=-1*(h^2*Q(xx(i),a,b))/2+h^3(QQ(xx(i))-
P(xx(i),a)*Q(xx(i),a,b))/6;
AA(2*i-1,2*i-1)=aa(1,1);
AA(2*i-1,2*i)=aa(1,2);
AA(2*i,2*i-1)=aa(2,1);
AA(2*i,2*i)=aa(2,2);
end
for r=1:1:n
AA(2*r+1:2*r+2,2*r-1:2*r)=f;
end
AA(1,end)=-1;
AA(end-1,end)=0;
AA(end,end)=0;
AA
G=zeros(2*n+2,1);
G(2)=1;
G(2*n+1)=-1.6035*exp(1);
G(2*n+2)=-0.9414*exp(1);
G;
[l,u]=lu(AA);

```

```

t=inv(l)*G;
y=inv(u)*t;
y;
y1(1)=y(2)
for jj=4:2: 32      %(2*(n+1))
    y1(jj/2)=y(jj);
end
y1
for i=1:n+1
    yy(i)=(2.718281828)^((xx(i))^b);
end
yy
for i=1:n+1
    ee(i)=abs(yy(i)-y1(i));
end
[xx' y1' yy' ee']

```

### Program 11

```

%PADE APPROXIMATE METHOD (3,0) TO SOLVE BVP
%((x)^(aa)*y)'=k*x(aa+k-2)*(aa+k-1+k*(x)^k)*y
%0<x<1      y(0)=1   y(1)=e(1)
%=====
clc
clear

```

format long

D=[0 1];

n=15;

a=.5;

b=4;

h=(D(2)-D(1))/(n+1);

xx=D(1)+h+[0:h:D(2)-h];

xx

P=inline('-a/x','x','a');

Q=inline('(b\*x^(a+b-2)\*(a+b-1+b\*x^b))/(x^a)','x','a','b');

PP=inline('a/(x^2)','x','a');

QQ=inline('28\*x+96\*x^5','x');

PPP=inline('-1/(x)','x');

QQQ=inline('28+480\*x^4','x');

f=[-1 0;0 -1]

for i=1:1:n+1

aa(1,1)=h\*P(xx(i),a)+(h^2\*(-PP(xx(i),a)+(P(xx(i),a))^2-  
Q(xx(i),a,b)))/2+h^3\*(PPP(xx(i))-  
3\*P(xx(i),a)\*PP(xx(i),a)+2\*QQ(xx(i))+P(xx(i),a))^3-  
2\*P(xx(i),a)\*Q(xx(i),a,b))/6;

aa(1,2)=h\*Q(xx(i),a,b)+h^2\*(-  
QQ(xx(i))+P(xx(i),a)\*Q(xx(i),a,b))/2+(h^3\*(QQQ(xx(i))-  
2\*PP(xx(i),a)\*Q(xx(i),a,b)-  
P(xx(i),a)\*QQ(xx(i))+P(xx(i),a))^2\*Q(xx(i),a,b)-(Q(xx(i),a,b))^2))/6;

```

aa(2,1)=-h-(h^2*P(xx(i),a))/2+h^3*((PP(xx(i),a)-
(P(xx(i),a))^2+Q(xx(i),a,b)))/6;

aa(2,2)=-1*(h^2*Q(xx(i),a,b))/2+h^3*(QQ(xx(i))-
P(xx(i),a)*Q(xx(i),a,b))/6;

AA(2*i-1,2*i-1)=aa(1,1);

AA(2*i-1,2*i)=aa(1,2);

AA(2*i,2*i-1)=aa(2,1);

AA(2*i,2*i)=aa(2,2);

end

for r=1:1:n

    AA(2*r+1:2*r+2,2*r-1:2*r)=f;

end

AA(1,end)=-1;

AA(end-1,end)=0;

AA(end,end)=0;

AA

G=zeros(2*n+2,1);

G(2)=1;

G(2*n+1)=1.5891723*exp(1);

G(2*n+2)=-0.0529378*exp(1);

G;

[l,u]=lu(AA);

t=inv(l)*G;

y=inv(u)*t;

```

```

y;
y1(1)=y(2)
for jj=4:2: 32      %(2*(n+1))
    y1(jj/2)=y(jj);
end
y1
for i=1:n+1
    yy(i)=(2.718281828)^((xx(i))^b);
end
yy
for i=1:n+1
    ee(i)=abs(yy(i)-y1(i));
end
[xx' y1' yy' ee']

```

### Example

Find pade approximations for  $f(x) = \frac{1}{\sqrt{1-x}}$  expanded about  $x_0 = 0$

$$f(x) = \frac{1}{\sqrt{1-x}} :$$

$$x_0 = 0;$$

$$a = -1.5;$$

$$b = 1.5;$$

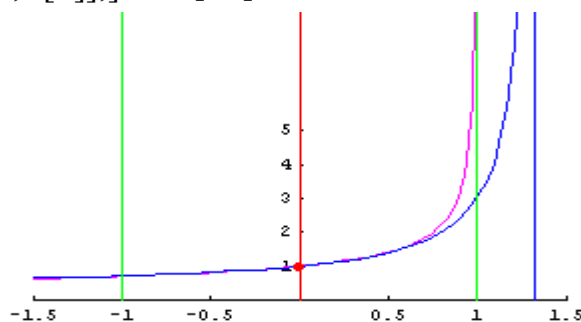
$$c = 0.0;$$

$$d = 8.5;$$

```

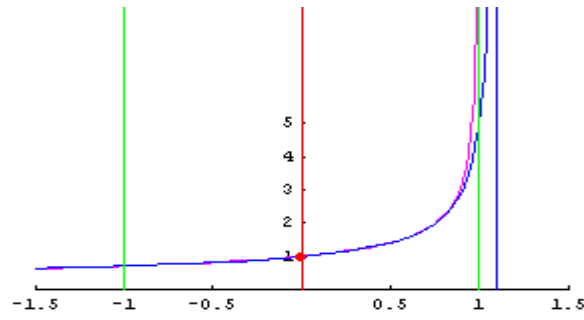
a0 = -1.0;
b0 = 1.0;
Needs ["Graphics' Colors"];
Needs ["Calculus `pade"];
Cdot = Graphics [{{Red, PointSize [0.02], Point [ {x0, f[x0]}]}}];
Rends=Graphics[{{Green,Line[{{a0,c},{a0,d}}]}, {Red,
Line[{{a0,c},{x0,d}}]}, {Green, Line[{{b0,c},{b0,d}}]}}];
For [ n=1, n<=5, n++,
P[x_] = Together [Pade[f[x], {x, x0, n, n}]];
graph1= plot[f[x],{x,-1.5,0.9999}, PlotStyle->Magenta,
DisplayFunction->Identity];
graph2= Plot[P[x],{x,-1.5,1.5},PlotStyle->Blue,DisplayFunction->Identity];
Show[graph1, graph2, Cdot, Rends, PlotRange -> {{a,b},{c,d}}, Tricks->
{Range[-1.5, 1.5, 0.5], Range[0, 5, 1]}, DisplayFunction->
$DisplayFunction];
Print ["f(x)=",f[x]];
Print["Pn,n", "[x]=",P[x]], See [15]

```



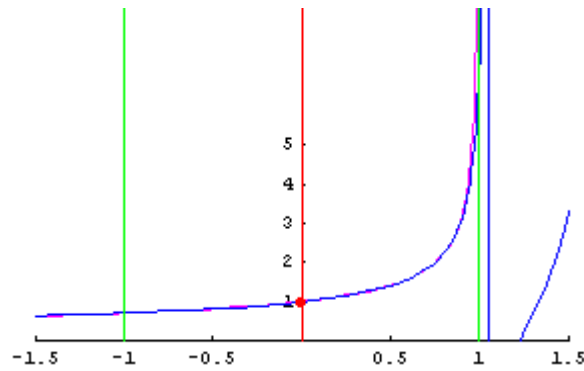
$$f[x] = \frac{1}{\sqrt{1-x}}$$

$$P_{1,1}[x] = \frac{-4+x}{-4+3x}$$



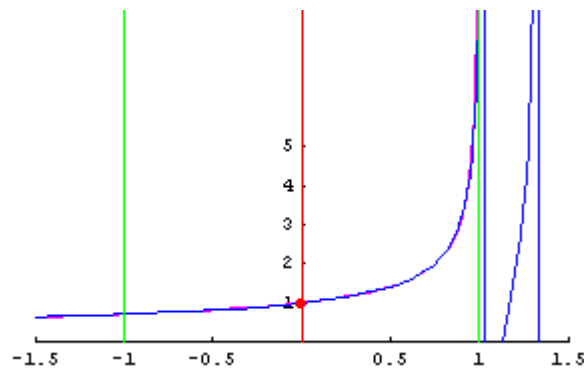
$$f[x] = \frac{1}{\sqrt{1-x}}$$

$$P_{2,2}[x] = \frac{16-12x+x^2}{16-20x+5x^2}$$



$$f[x] = \frac{1}{\sqrt{1-x}}$$

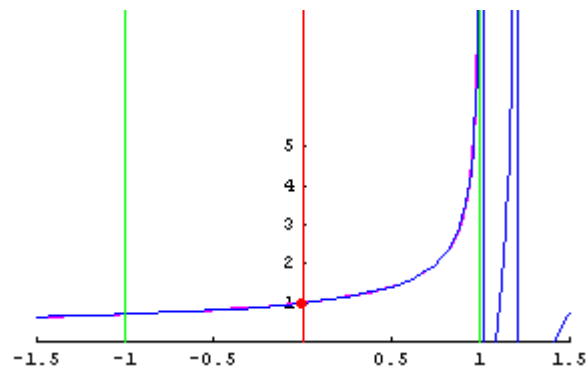
$$P_{3,3}[x] = \frac{-64+80x-24x^2+x^3}{-64+112x-56x^2+7x^3}$$



$$f[x] = \frac{1}{\sqrt{1-x}}$$



$$P_{4,4}[x] = \frac{256 - 448x + 240x^2 - 40x^3 + x^4}{256 - 576x + 432x^2 - 120x^3 + 9x^4}$$



$$f(x) = \frac{1}{\sqrt{1-x}}$$

$$P_{5,5}[x] = \frac{-1024 + 2304x - 1792x^2 + 560x^3 - 60x^4 + x^5}{-1024 + 2816x - 2816x^2 + 1232x^3 - 220x^4 + 11x^5}$$

## References

- [1] Ascher, Mattheij, R.M.M, RUSSELL,r.d, (1995). **Numerical Solution of Boundary Value Problems For Ordinary Dfferential Equations.**
- [2] Brabston, D.C. (1974). **Numerical Solution of Singular Endpoint Boundary Value Problems**, Pasadena, California.
- [3] Burden, R.L, Faires,J.D, (2001). **Numerical Method**, Youngstown State University, seventh edition.
- [4] Butcher, J.C. (2008). **Numerical methods for ordinary differential equations**, Wiley.
- [5] El-gebeily, M.A, Attili, B.S,(2003). **An Iterative Shooting Method For a certain Class of Singular Two-point Boundary Value Problems**, King Fahd University of Petroleum And Minerals.
- [6] Isaacson.E and keller.H.B,( 1960). **Analysis of Numerical Methods**, 1NC, Newyourk.
- [7] Keller,H.B, (1976). **Numerical Solution of Two-point Boundary Value Problems**, California Institute of Technologe.
- [8] Leveque,R.J,( 2006). **Finite Difference Methods For Differential Equations**, Washington University.
- [9] Shampine.L.F, Gladwell.I , Thompson.S, (2003). **Solving ODEs with MATLAB**, Cambridge, University Press.
- [10] Thompson.P.A.O, (1996).**Sequential And Parallel Numerical Methods For Second Order Two-point Boundary Value Problems;**

Brunel University.

[11] Twizell,E.I-J,( 1988). **Numerical Methods, With Applications In The Biomedical Sciences**, Brunel University.

[12] [www.physics.arizona.edu](http://www.physics.arizona.edu).

[13] [www.wikipedia.org](http://www.wikipedia.org).

[14] [www.mathworks.com](http://www.mathworks.com).

[15][www.math.fullerton.edu](http://www.math.fullerton.edu).

جامعة النجاح الوطنية

كلية الدراسات العليا

## المعادلات التفاضلية الحدودية غير محددة القيمة

إعداد

سوسن محمد أنيس حمدان

إشراف

د. سمير مطر

قدمت هذه الأطروحة استكمالاً لمتطلبات درجة الماجستير في الرياضيات المحوسبة بكلية الدراسات العليا في جامعة النجاح الوطنية في نابلس، فلسطين.

2010

ب

المعادلات التفاضلية الحدودية غير محددة القيمة

إعداد

سوسن محمد أنيس حمدان

إشراف

د. سمير مطر

الملخص

المعادلات التفاضلية الحدودية واسعة الاستخدام في مجالات متعددة في الحياة و في الدراسات العلمية، وتحدث المعادلات التفاضلية الحدودية غير محددة القيمة في النماذج الرياضية والمشاكل العملية.

لحل المعادلات التفاضلية الحدودية بالنسبة للمعادلات التفاضلية العادية التي تحوي معاملات منفردة، قمنا بدراسة العديد من الطرق منها: طريقة الفروق الدقيقة، وطريقة التقريب

(shooting method) وطريقة (pade approximation methods).