An-Najah National University
Faculty of Graduate Studies

Linear Fredholm Integro-Differential Equation of the Second Kind

By
Khulood Nedal Iseed Thaher

Supervised
Prof. Naji Qatanani

This Thesis is Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science of Mathematics, Faculty of Graduate Studies, An-Najah National University, Nablus, Palestine.
2016

Linear Fredholm Integro-Differential Equation of the Second Kind

By

Khulood Nedal Iseed Thaher

This Thesis was Defended Successfully on 23/5/2016 and approved by:

<u>Defense Committee Members</u>	**<u>Signature</u>**
1. Prof. Naji Qatanani /Supervisor	N. Qatanani ………………
2. Dr. Maher Qarawani / External Examiner	[illegible] ………………
3. Dr. Hadi Hamad / Internal Examiner	[illegible] ………………

Dedication

I dedicate this thesis to my beloved Palestine, my parents, my love my husband Ayman, my children Sandra and Kareem, my brother Khaled and my sisters Amany and Farah. Without their patience, understanding, support and most of all love, this work would not have been possible.

ACKNOWLEDGMENT

First of all, I thank my God for all the blessing he bestowed on me and continues to bestow on me.

I would sincerely like to thank and deeply greatful to my supervisor Prof. Dr. Naji Qatanani who without his support, kind supervision, helpful suggestions and valuable remarks, my work would have been more difficult. My thanks also to my external examiner Dr. Maher Qarawani and to my internal examiner Dr. Hadi Hamad for their useful and valuable comments.

Also, my great thanks are due to my family for their support, encouragement and great efforts for me.

Finally, I would also like to acknowledge to all my teachers in An-Najah National University department of mathematics.

الإقرار

أنا الموقع أدناه ، مقدم الرسالة التي تحمل العنوان :

Linear Fredholm Integro-Differential Equation of the Second Kind

أقر بان ما شملت عليه هذه الرسالة إنما هو نتاج جهدي الخاص، بإستثناء ما تمت الإشارة اليه حيثما ورد، و أن هذه الرسالة كاملة، أو أي جزء منها لم يقدم من قبل لنيل أي درجة أو لقب علمي أو بحثي لدى 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 :	خلود نضال اسعيد ظاهر	**أسم الطالب :**
Signature:	[illegible]	**التوقيع :**
Date :	٢٣/٥/٢٠١٦	**التاريخ :**

Table of Contents

List of Figures

List of Tables

Linear Fredholm Integro-Differential Equation of the Second Kind

By

Khulood Nedal Iseed Thaher

Supervisor

Prof. Naji Qatanani

Abstract

In this thesis we focus on solving linear Fredholm integro-differential equation of the second kind due to it's wide range of physical applications. We will investigate some analytical and numerical methods to solve this equation. The discussed analytical methods include: Direct computation method, variational iteration method, Adomian decomposition method, modified decomposition method, noise terms phenomenon and series solution method.

The numerical methods that will be presented here are: B-spline scaling function and wavelet method, Homotopy perturbation method, Legendre polynomial method and Taylor collocation method. Particular numerical examples demonstrating these numerical methods have been implemented for solving linear Fredholm integro-differential of the second kind. A comparison has been drawn between these numerical methods. Our numerical results show that the Homotopy perturbation method and Legendre polynomial method have proved to be the most efficient in comparison to the other numerical methods regarding their performance on the used examples.

Introduction

The subject of integro-differential equations is one of the most important mathematical tools in both pure and applied mathematics. Integro-differential equations play a very important role in modern science and technology such as heat transfer, diffusion processes, neutron diffusion and biological species. More details about the sources where these equations a rise can be found in physics, biology and engineering applications as well as in advanced integral equations books. (see [2, 4, 5, 8, 18, 19, 20, 22]).

Some valid numerical methods, for solving Fredholm integro-differential equations have been developed by many researchers. In [7] Behiry and Hashish used wavelet methods for the numerical solution of Fredholm integro-differential equation. Lakestani, Razzaghi and Dehghan [25] applied linear semiorthogonal B-spline wavelets, specially constructed for the bounded interval to solve linear Fredholm integro-differential equation. In [16] Ji-Huan He solved linear Fredholm integro-differential equation by a Homotopy perturbation method. This method yields a very rapid convergence of the solution series in most cases, usually only few iterations leading to very accurate solutions. A Legendre collocation matrix method is presented to solve high-order linear Fredholm integro-differential equations under the mixed conditions in terms of Legendre polynomials were used in [42] by Yalcinbas, Sezer and Sorkan. In [23] Karamete and Sezer solved Fredholm integro-differential equation by a truncated Taylor series. Using the Taylor collocation points, this method transforms the integro-differential equation to a matrix equation which corresponds to a system of linear algebraic equations with unknown Taylor coefficients.

However many approaches for solving the linear and nonlinear kind of these equations may be found in [10], [11], [13], [14], [15], [29], [30], [32], [36], [39] and [43].

In this work, some analytical methods have been used to solve the Fredholm integro-differential equation of the second kind. These methods are: direct computation method, variational iteration method, Adomian decomposition method, modified decomposition method, noise terms phenomenon and series solution method.

For the numerical treatment of the Fredholm integro-differential equation of the second kind, we have implemented the following methods: B-spline scaling functions and wavelets, Homotopy perturbation method, Legendre polynomial method and Taylor collocation method.

This thesis is organized as follow: In chapter one, we introduce some basic concepts of integro-differential equations. In chapter two, we investigate some analytical methods to solve the Fredholm integro-differential equation. These have been mentioned before. In chapter three, we implement some numerical methods for solving Fredholm integro-differential equation. These have mentioned before. Numerical examples and results are presented in chapter four and conclusions have been drawn.

Chapter One
Mathematical Preliminaries

Definition 1.1 [39]

An integro-differential equation is the equation in which the unknown function appears inside an integral sign and contains ordinary derivatives.

A standard integro-differential equation is of the form:

$$v^{(n)}(x) = f(x) + \lambda \int_{g(x)}^{h(x)} G(x,y)\, v(y)\, dy, \tag{1.1}$$

where $v^{(n)}(x) = \dfrac{d^n v}{dx^n}$, n integer, λ is a constant, $g(x)$ and $h(x)$ are limits of integration that may be both variables, constants, or mixed,$G(x,y)$ is a known function of two variables x and y called the kernel or the nucleus of the integro-differential equation. The function v that will be determined appears under the integral sign, and sometimes outside the integral sign. The function $f(x)$ and $G(x,y)$ are given .

1.1 Classification of Integro-Differential Equations

1.1.1 Types of integro-differential equations

There are three types of integro-differential equations**:**

1. Fredholm integro- differential equation

The Fredholm integro-differential equation of the second kind appears in the form:

$$v^{(n)}(x)=f(x)+\lambda\int_a^b G(x,y)\,v(y)\,dy. \tag{1.2}$$

2. Volterra integro-differential equation

The Volterra integro-differential equation of the second kind appears in the form:

$$v^{(n)}(x)=f(x)+\lambda\int_a^x G(x,y)\,v(y)\,dy, \tag{1.3}$$

where the upper limit of integration is variable.

3. Volterra-Fredholm integro-differential equation

The Volterra-Fredholm integro-differential equation of the second kind appear in two forms, namely:

$$v^{(n)}(x)=f(x)+\lambda_1\int_a^x G_1(x,y)\,v(y)\,dy+\lambda_2\int_a^b G_2(x,y)\,v(y)\,dy \tag{1.4}$$

and

$$v^{(n)}(x,y)=f(x,y)+\lambda\int_0^y\int_\Omega F(x,y,\xi,\tau,v(\xi,\tau))\,d\xi\,d\tau,\quad x,y\in\Omega\times[0,Y]$$

(1.5)

where $f(x,y)$ and $F(x,y,\xi,\tau,v(\xi,\tau))$ are analytic functions,

$D=\Omega\times[0,Y]$and Ω is a closed subset of R^n, $n=1,2,3$. It is interesting to note that (1.4) contains disjoint Volterra and Fredholm integrals, whereas (1.5) contains mixed integrals. Other derivatives of less order may appear as well.

1.1.2 Linearity of integro-differential equation

Definition 1.2 [39]

The integro- differential equation

$$v^{(n)}(x)=f(x)+\lambda\int_{g(x)}^{h(x)}G(x,y)v(y)\,dy,$$

is said to be linear if the exponent of the unknown function $v(x)$ under the integral sign is one and the equation does not contain nonlinear functions of $v(x)$, otherwise, the equation is called nonlinear.

1.1.3 Homogeneity of integro-differential equation

Definition 1.3 [39]

The integro-differential equation

$$v^{(n)}(x)=f(x)+\lambda\int_{g(x)}^{h(x)}G(x,y)v(y)\,dy,$$

is said to be homogeneous if $f(x)$ is identically zero, otherwise it is called nonhomogeneous.

1.1.4 Singularity of integro-differential equation

When one or both limits of integration become infinite or when the kernel becomes infinite at one or more points within the range of integration. For example, the integro-differential equation.

$$dy \ \ v(y) \ v^{(n)}(x) = f(x) + \lambda \int_{-\infty}^{\infty} G(x,y)$$

is a singular integro-differential equation of the second kind.

(i) Singular integro-differential equation

If the kernel is of the form

$$G(x,y) = \frac{H(x,y)}{x-y} \tag{1.6}$$

where $H(x,y)$ is differentiable function $a \le x \le b$, $a \le y \le b$ with $H(x,y) \ne 0$, then the integro-differential equation is said to be a singular equation with Cauchy kernel where the $G(x,y) = \int_a^b \frac{H(x,y)}{x-y} \ f(y)\,dy$ is understood in the sence of Cauchy Principal value (CPV) and the notation P.V $\int_a^b \frac{H(x,y)}{x-y} dy$ is usually used to denote this. Thus

$$\text{P.V.} \int_a^b \frac{H(x,y)}{x-y} dy = \lim_{x \to \varepsilon} \left\{ \int_a^{x-\varepsilon} \frac{H(x,y)}{x-y} dy + \int_{x+\varepsilon}^{b} \frac{H(x,y)}{x-y} dy \right\},$$

for example

$$v'(x)=\frac{2}{3}x^2+7x^4+x^5\ln\left(\frac{1-x}{1+x}\right)+\frac{2}{5}+\int_{-1}^{1}\frac{v(y)}{x-y}dy,\quad |x|<1.$$

(ii) Weakly singular integro-differential equation

The kernel is of the form

$$G(x,y)=\frac{H(x,y)}{|x-y|^{\alpha}} \tag{1.7}$$

where $H(x,y)$ is bounded $i.e.$several times continuously differentiable

$a\le x\le b$ and $a\le y\le b$ with $H(x,y)\ne 0$ and α is a constant such that $0<\alpha<1$. For example, the equation

$$v^{(n)}(x)=\lambda\int_0^x\frac{1}{|x-y|^{\alpha}}v(y)\,dy,\qquad 0<\alpha<1$$

is a singular integro-differential equation with a weakly singular kernel.

1.2 Systems of Fredholm Integro-Differential Equations

A system of Fredholm integro-differential equations of the second kind can be written as:

$$u^{(i)}(x)=f_1(x)+\int_a^b(G_1(x,y)\,u(y)+\tilde{G}_1(x,y)\,v(y))\,dy \tag{1.8}$$

$$v^{(i)}(x)=f_2(x)+\int_a^b(G_2(x,y)\,u(y)+\tilde{G}_2(x,y)\,v(y))\,dy.$$

The unknown functions $u(x)$ and $v(x)$, that will be determined, occur inside the integral sign whereas their derivatives appear mostly outside the integral sign. The kernels $G_i(x,y)$, $\tilde{G}_i(x,y)$ and the functions $f_i(x)$ are given as real-valued functions [39].

1.3 Systems of Volterra Integro-Differential Equations

A system of Volterra integro-differential equation of the second kind has the form:

$$u^{(i)}(x)=f_1(x)+\int_0^x (G_1(x,y)\,u(y)+\tilde{G}_1(x,y)\,v(y))\,dy \qquad (1.9)$$

$$v^{(i)}(x)=f_2(x)+\int_0^x (G_2(x,y)\,u(y)+\tilde{G}_2(x,y)\,v(y))\,dy\,.$$

The unknown functions $u(x)$ and $v(x)$ that will be determined, occur

inside the integral sign whereas their derivatives appear mostly outside the integral sign. The kernels $G_i(x,y)$, $\tilde{G}_i(x,y)$ and the functions $f_i(x)$ are given as real-valued functions [39].

1.4 Kinds of Kernels

1. Separable kernel

A kernel $G(x,y)$ is said to be separable or (degenerate) if it can be expressed in the form (1.10)

$$G(x,y)=\sum_{k=1}^{n} g_k(x)h_k(y),$$

where the functions $g_k(x)$ and the functions $h_k(y)$ are linearly indepe-ndent. (see [39]).

2. Symmetric (or Hermitian) kernel

A complex-valued function $G(x,y)$ is called symmetric (or Hermitian) if

$$G(x,y) = G^*(x,y), \tag{1. 11}$$

where the asterisk denotes the complex conjugate. For a real kernel, we have

$$G(x,y) = G(y,x). \tag{1.12}$$

Chapter Two
Analytical Methods for Solving Fredholm Integro-Differential Equation of the Second Kind

There are many analytical methods for solving Fredholm integro-differential equation of the second kind. In this chapter we will focus on the following methods: direct computation method, variational iteration method, Adomian decomposition method, modified decomposition method, noise terms phenomenon and series solution method.

2.1 Direct Computation Method

This method can be used to solve the Fredholm integro-differential equation of the second kind directly instead of a series form.

For the application of this method we consider the separable kernel of the form

$$G(x,y)=g(x)\,h(y), \tag{2.1}$$

We consider the Fredholm integro-differential equation of the general form

$$v^{(n)}(x)=f(x)+\int_a^b G(x,y)\,v(y)\,dy, \tag{2.2}$$

with the initial conditions $v^{(k)}(0)=b_k$, $0\le k\le n-1$.

Substituting (2.1) into (2.2) gives

$$v^{(n)}(x)=f(x)+g(x)\int_a^b h(y)\,v(y)\,dy. \tag{2.3}$$

Since the integral in equation (2.3) is a bounded integral and dependent only on one variable y, then we can assign this integral by a constant β, That is

$$\int_a^b h(y)v(y)\,dy = \beta. \tag{2.4}$$

Thus equation (2.3) becomes

$$v^{(n)}(x) = f(x) + \beta g(x). \tag{2.5}$$

Integrating both sides of (2.5) n times from 0 to x, also using the initial conditions, we can find formula for $v(x)$ that depends on β and x. This means we can write

$$v(x) = u(x,\beta). \tag{2.6}$$

Substituting (2.6) into the right side of (2.4), calculating the integral, also solving the resulting equation, we determine β. We obtain the exact solution $v(x)$ after substituting β into (2.6).

Example 2.1

Consider the Fredholm integro-differential equation

$$v'(x) = 12x + \int_0^1 v(y)\,dy \quad \text{with} \quad v(0) = 0. \tag{2.7}$$

This equation may be written as

$$v'(x) = 12x + \beta \;, \; v(0) = 0. \tag{2.8}$$

obtained by setting

$$\int_0^1 v(y)\,dy = \beta. \tag{2.9}$$

Integrating both sides of (2.8) from 0 to x, and by using the initial condition we obtain

$$v(x) = 6x^2 + \beta x. \tag{2.10}$$

Substituting (2.10) into (2.9) and evaluating the integral yield

$$\beta = \int_0^1 v(y)\,dy = 2 + \frac{1}{2}\beta \tag{2.11}$$

hence we find

$$\beta = 4. \tag{2.12}$$

The exact solution is given by

$$v(x) = 6x^2 + 4x. \tag{2.13}$$

2.2 Variational Iteration Method (VIM)

The variational iteration method (VIM) was established by Ji-Huan He [17]. The method provides rapidly convergent successive approximations of the exact solution only if such a closed form solution exists.

We consider the Fredholm integro-differential equation of the general form

$$v^{(i)}(x) = f(x) + \lambda \int_a^b G(x,y)\, v(y)\, dy. \tag{2.14}$$

The correction functional for this equation is given by

$$v_{n+1}(x) = v_n(x) + \int_0^x \lambda(\tau)\, [v_n^{(i)}(\tau) - f(\tau) - \int_a^b G(\tau,z)\tilde{v}_n(z)\, dz\,]\, d\tau. \tag{2.15}$$

The variational iteration method needs to determine the Lagrange multiplier $\lambda(\tau)$ via integration by parts and by using a restricted variation. The variational iteration formula, without restricted variation, should be used for the determination of the successive approximations $v_{n+1}(x),\, n \geq 0$ of the solution $v(x)$. The zeroth approximation v_0 can be any selective function. However, using the given initial values $v(0), v'(0), \ldots$are preferably used for the selective zeroth approximation v_0.

Consequently, the solution is given by

$$v(x) = \lim_{n\to\infty} v_n(x). \tag{2.16}$$

Example 2.2

Consider the Fredholm integro-differential equation

$$v'(x) = 12x + \int_0^1 v(y)\, dy \quad \text{with} \quad v(0) = 0. \tag{2.17}$$

The correction functional for this equation is given by

$$v_{n+1}(x) = v_n(x) - \int_0^x [v_n'(\tau) - 12\tau - \int_0^1 v_n(z)\, dz\,]\, d\tau, \tag{2.18}$$

as $\lambda = -1$ for first-order Fredholm integro-differential equation.

This gives:

$$v_0(x) = 0$$

$$v_1(x) = v_0(x) - \int_0^x [v_0'(\tau) - 12\tau - \int_0^1 v_0(z)\, dz]\, d\tau = 6x^2$$

$$v_2(x) = v_1(x) - \int_0^x [v_1'(\tau) - 12\tau - \int_0^1 v_1(z)\, dz]\, d\tau = 6x^2 + 2x$$

$$v_3(x) = v_2(x) - \int_0^x [v_2'(\tau) - 12\tau - \int_0^1 v_2(z)\, dz]\, d\tau = 6x^2 + 2x + x \qquad (2.19)$$

$$v_4(x) = v_3(x) - \int_0^x [v_3'(\tau) - 12\tau - \int_0^1 v_3(z)\, dz]\, d\tau = 6x^2 + 2x + x + \frac{1}{2}x$$

$$v_5(x) = v_4(x) - \int_0^x [v_4'(\tau) - 12\tau - \int_0^1 v_4(z)\, dz]\, d\tau = 6x^2 + 2x + x + \frac{1}{2}x + \frac{1}{4}x$$

$$v_n(x) = v_{n-1}(x) - \int_0^x [v_{n-1}'(\tau) - 12\tau - \int_0^1 v_{n-1}(z)\, dz]\, d\tau$$
$$= 6x^2 + 3x + (\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots)x$$

and so on. The VIM admits the use of

$$v(x) = \lim_{n\to\infty} v_n(x). \qquad (2.20)$$

We obtain the exact solution

$$v(x) = 6x^2 + 4x. \qquad (2.21)$$

2.3 Adomian Decomposition Method

The Adomian decomposition method (ADM) was introduced and developed by George Adomian [1-3]. ADM gives the solution in an infinite series of components. The idea of Adomian decomposition method is transforming Fredholm integro-differential equation to an integral equation.

We consider the second order Fredholm integro-differential equation of the second kind of the form .

$$v''(x) = f(x) + \int_a^b G(x,y)v(y)\,dy, \tag{2.22}$$

with the initial conditions $v(0) = b_0$, $v'(0) = b_1$.

Integrating both sides of (2.22) from 0 to x twice, we get

$$v(x) = b_0 + b_1 x + L^{-1}(f(x)) + L^{-1}\left(\int_a^b G(x,y)v(y)\,dy\right) \tag{2.23}$$

where the initial conditions are used, and L^{-1} is a two-fold integral operator. $v(x)$ expressed as

$$v(x) = \sum_{n=0}^{\infty} v_n(x). \tag{2.24}$$

Setting (2.24) into (2.23) gives:

$$\sum_{n=0}^{\infty} v_n(x) = b_0 + b_1 x + L^{-1}(f(x)) + L^{-1}\left(\int_0^x G(x,y)\sum_{n=0}^{\infty} v_n(y)dy\right) \tag{2.25}$$

or equivalently,

$$v_0(x)+v_1(x)+v_2(x)+\ldots=b_0+b_1(x)+L^{-1}(f(x))$$

$$+L^{-1}\left(\int_0^x G(x,y)v_0(y)\,dy\right)+L^{-1}\left(\int_0^x G(x,y)v_1(y)\,dy\right)$$

$$+L^{-1}\left(\int_0^x G(x,y)v_2(y)\,dy\right)+\ldots \qquad (2.26)$$

Note that, $v_0(x)$ is defined by all the terms that are not included under the integral sign, that is

$$v_0(x)=b_0+b_1(x)+L^{-1}(f(x))$$

$$v_{k+1}(x)=L^{-1}\left(\int_0^x G(x,y)v_k(y)\,dy\right),\quad k\geq 0. \qquad (2.27)$$

Using (2.24), the obtained series converges to the exact solution if such a solution exists, see ([12], [39]).

Example 2.3

Consider the Fredholm integro-differential equation

$$v'(x)=-1+24x+\int_0^1 v(y)\,dy \quad \text{with } v(0)=0. \qquad (2.28)$$

Note that, the integral at the right side is equivalent to a constant.

Integrating both sides of Eq. (2.28) from 0 to x, we get

$$v(x) - v(0) = -x + 12x^2 + x\left(\int_0^1 v(y)dy\right) \tag{2.29}$$

using the initial condition

$$v(x) = -x + 12x^2 + x\left(\int_0^1 v(y)dy\right). \tag{2.30}$$

Let

$$v(x) = \sum_{n=0}^{\infty} v_n(x). \tag{2.31}$$

Setting (2.31) into (2.30) gives

$$\sum_{n=0}^{\infty} v_n(x) = -x + 12x^2 + x\left(\int_0^1 v_n(y)dy\right). \tag{2.32}$$

The components $v_j,\ j \ge 0$ of $v(x)$ can be determined by using the recu-rrence relation

$$v_0(x) = -x + 12x^2 \quad , \quad v_{k+1}(x) = x\left(\int_0^1 v_k(y)dy\right), \ k \ge 0. \tag{2.33}$$

This gives

$$v_0(x) = -x + 12x^2 \ , \quad v_1(x) = x\left(\int_0^1 v_0(y)dy\right) = \frac{7}{2}x \quad ,$$

$$v_2(x) = x\left(\int_0^1 v_1(y)dy\right) = \frac{7}{4}x \quad , \quad v_3(x) = x\left(\int_0^1 v_2(y)dy\right) = \frac{7}{8}x \tag{2.34}$$

Hence, the solution is

$$v(x) = -x + 12x^2 + \frac{7}{2}x\left(1 + \frac{1}{2} + \frac{1}{4} + \cdots\right), \tag{2.35}$$

which gives the exact solution

$$v(x) = 6x + 12x^2. \tag{2.36}$$

2.4 Modified Decomposition Method

As shown before, The Adomian decomposition method provides the solution in an infinite series of components. The method substitutes the decomposition series of $v(x)$, given by

$$v(x) = \sum_{n=0}^{\infty} v_n(x). \tag{2.37}$$

The process of converting Fredholm integro-differential equation to an integral equation can be achieved by integrating both sides of the Fredholm integro-differential equation from 0 to x as many times is needed to convert it into the integral equation

$$v(x) = f(x) + \lambda \int_a^b G(x,y)\, v(y)\, dy. \tag{2.38}$$

The standard Adomian decomposition method introduces the recurrence relation

$$\begin{aligned} v_0(x) &= f(x) \\ v_{k+1}(x) &= \lambda \int_a^b G(x,y)\, v_k(y)\, dy, \quad k \geq 0. \end{aligned} \tag{2.39}$$

The modified decomposition method presents a slight variation to the recurrence relation (2.38) to determine the components of $v(x)$ in an easier and faster manner. For many cases, the function $f(x)$ can be set as the sum of two partial functions, namely $f_1(x)$ and $f_2(x)$. In other words

$$f(x) = f_1(x) + f_2(x). \tag{2.40}$$

The modified decomposition method admits the use of the modified recurrence relation

$$\begin{aligned} &v_0(x) = f_1(x) \\ &v_1(x) = f_2(x) + \lambda \int_a^b G(x,y)\, v_0(y)\, dy \\ &v_{k+1}(x) = \lambda \int_a^b G(x,y)\, v_k(y)\, dy \ , \ k \geq 1. \end{aligned} \tag{2.41}$$

The difference between the recurrence relation (2.39) and the modified recurrence relation (2.41) is in the formation of $v_0(x)$ and $v_1(x)$ only. The other components $v_j(x)$, $j \geq 2$ remain the same in the two recurrence relations [39].

Example 2.4

Consider the Fredholm integro-differential equation

$$v''(x) = -\sin(x) + x - x\int_0^{\frac{\pi}{2}} y\ v(y)\, dy \ \text{ with } v(0) = 0,\ v'(0) = 1. \tag{2.42}$$

Using the modified decomposition method. First, integrate both sides of (2.42) from 0 to x twice, and using the given conditions, we obtain the Fredholm integral equation

$$v(x) = \sin(x) + \frac{1}{3!}x^3 - \frac{1}{3!}x^3 \int_0^{\frac{\pi}{2}} y\ v(y)\, dy. \tag{2.43}$$

$$f(x) = \sin(x) + \frac{1}{3!}x^3 \tag{2.44}$$

We split $f(x)$ into two parts, namely

$$f_1(x) = \sin(x)$$

$$f_2(x) = +\frac{1}{3!}x^3, \tag{2.45}$$

$f_1(x)$ related to the zeroth component, $v_0(x)$, and add $f_2(x)$ to the component $v_1(x)$. Therefore, we obtain the modified recurrence relation

$$v_0(x) = \sin(x)$$

$$v_1(x) = \frac{1}{3!}x^3 - \frac{1}{3!}x^3 \int_0^{\frac{\pi}{2}} y\ v_0(y)\, dy = 0, \tag{2.46}$$

all remaining components $v_2(x), v_3(x), \ldots$ are zeros. This gives the exact solution as

$$v(x) = \sin(x) \tag{2.47}$$

2.5 The Noise Terms Phenomenon

The idea behind solving equations with noise terms if the noise terms appear between components of $v(x)$, then the exact solution can be obtained by considering only the first two components $v_0(x)$ and $v_1(x)$. The noise terms are defined as the identical terms, with opposite signs, that may appear between the components of the solution $v(x)$. The conclusion made by [1], [3], and [41] suggests that if we observe the appearance of identical terms in both components with opposite signs, then by canceling these terms, the remaining non-canceled terms of v_0 may in some cases provide the exact solution, that should be justified through substitution.

It was formally proved that other terms in other components will vanish in the limit if the noise terms occurred in $v_0(x)$ and $v_1(x)$.

Example 2.5

Consider the Fredholm integro-differential equation

$$v'(x) = 1 - \frac{1}{3}x + \int_0^1 xy\, v(y)\, dy \quad \text{with} \quad v(0) = 0. \tag{2.48}$$

Using the noise terms phenomenon. By integrating both sides of the equation (2.48) from 0 to x gives

$$v(x) - v(0) = x - \frac{1}{6}x^2 + \frac{1}{2}x^2 \int_0^1 y\, v(y)\, dy\ ,\ v(0) = 0. \tag{2.49}$$

by using the initial condition

$$v(x) = x - \frac{1}{6}x^2 + \frac{1}{2}x^2 \int_0^1 y\, v(y)\, dy\,. \tag{2.50}$$

Let

$$v(x) = \sum_{n=0}^{\infty} v_n(x). \tag{2.51}$$

Substituting (2.51) into both sides of (2.50) yields

$$\sum_{n=0}^{\infty} v_n(x) = x - \frac{1}{6}x^2 + \frac{1}{2}x^2 \int_0^1 y\,(\sum_{n=0}^{\infty} v_n(y))dy\,. \tag{2.52}$$

or equivalently

$$\begin{aligned} v_0(x) &= x - \frac{1}{6}x^2 \\ v_1(x) &= \frac{7}{48}x^2. \end{aligned} \tag{2.53}$$

Notice that $v_1(x)$ can be written as

$$v_1(x) = \frac{1}{6}x^2 - \frac{1}{48}x^2. \tag{2.54}$$

The noise terms $\pm\frac{1}{6}x^2$ appear between the components $v_0(x)$ and $v_1(x)$.

By canceling the noise term form $v_0(x)$, we obtain the exact solution

$$v(x) = x\,. \tag{2.55}$$

2.6 The Series Solution Method

The series method is useful method that stems mainly form the Taylor series for analytic functions for Fredholm integro-differential equation.

Definition 2.1 [39]

A real function $v(x)$ has the Taylor series representation

$$v(x) = \sum_{k=0}^{n} \frac{v^{(k)}(b)}{k!}(x-b)^k, \tag{2.56}$$

is called analytic if it has derivatives of all orders such that the Taylor series at any point b in it is domain converges to $v(x)$ in a neighborhood of b.

For simplicity, the generic form of the Taylor series at $b=0$ is given as

$$v(x) = \sum_{n=0}^{\infty} a_n x^n. \tag{2.57}$$

We will assume that the Fredholm integro-differential equation of the second kind

$$v^{(k)}(x) = f(x) + \lambda \int_a^b G(x,y)\, v(y)\, dy, v_j(0) = a_j,\ 0 \le j \le (k-1) \tag{2.58}$$

is analytic, and therefore possesses a Taylor series of the form given in (2.57), where the coefficients a_n will be determined recurrently.

Setting (2.57) into (2.58) yields

$$\left(\sum_{n=0}^{\infty} a_n x^n\right)^{(k)} = T(f(x)) + \lambda \int_a^b G(x,y) \left(\sum_{n=0}^{\infty} a_n y^n\right) dy, \tag{2.59}$$

This is equivalent to

$$(a_0+a_1x+a_2x^2+\ldots)^{(k)}=T(f(x))+\lambda\int_a^b G(x,y)\,(a_0+a_1y+a_2y^2+\cdots)dy$$

(2.60)

where $T(f(x))$ is the Taylor series for $f(x)$.

After performing integration in the right hand side of (2.60), we compare coefficients of power of x on both sides.

Example 2.6

Consider the Fredholm integro-differential equation of the second kind

$$v'(x)=4x+\int_{-1}^{1}(x-y)v(y)\,dy \quad \text{with } v(0)=2, \tag{2.61}$$

using the series method.

Let

$$v'(x)=(\sum_{n=0}^{\infty}a_nx^n)'. \tag{2.62}$$

Inserting (2.62) into (2.61), yields

$$\sum_{n=0}^{\infty}na_nx^{n-1}=4x+\int_{-1}^{1}((x-y)\sum_{n=0}^{\infty}a_ny^n)dy. \tag{2.63}$$

$a_0=2$ from the given initial conditions. Evaluating the integral at the right side gives

$$\begin{aligned} &a_1+2a_2x+3a_3x^2+\cdots= \\ &-\frac{2}{3}a_1-\frac{2}{5}a_3-\frac{2}{7}a_5-\frac{2}{9}a_7-\cdots+(4+2a_0+\frac{2}{3}a_2+\frac{2}{5}a_4+\frac{2}{7}a_6+\cdots)x\,. \end{aligned} \quad (2.64)$$

Comparing coefficients of like powers of x in both sides of (2.64) gives

$$a_1=0, \qquad a_2=6, \qquad a_n=0\ ,\ n\geq 2. \qquad (2.65)$$

The exact solution is given by

$$v(x)=2+6x^2. \qquad (2.66)$$

where we used $a_0=2$ from the initial condition.

Chapter Three

Numerical Methods for Solving Fredholm Integro-Differential Equation of the Second Kind

There are many numerical methods available for solving Fredholm integro-differential equation of the second kind . In this chapter we will discuss the following methods: B-spline scaling functions and wavelets, Homotopy perturbation method, Legendre polynomial method and Taylor collocation method.

3.1 B-Spline Scaling Functions and Wavelets on [0,1]

B-spline scaling functions and wavelets which are presented to approximate the solution of linear second order Fredholm integro-differential equation [6], [7] and [9]. Their properties and the operational matrices of derivative for these functions are presented to reduce the solution of linear Fredholm integro-differential equations to a solution of algebraic equations.

Consider the linear second-order Fredholm integro-differential equation of the form:

$$\sum_{i=0}^{2} \tau_i(x) y^{(i)}(x) = f(x) + \int_0^1 K(x,t)\, y(t)\, dt, \quad 0 \le x \le 1 \qquad (3.1)$$

with

$$y(0) = y_0 \quad , \quad y(1) = y_1 \qquad (3.2)$$

where τ_i $(i = 0,1,2)$, f, and K, are given functions in $L^2[0,1]$, y_0 and y_1 are given real numbers and y is the unknown function to be found.

The second-order B-splines scaling functions are defined as:

$$\chi_{i,j}(x) = \begin{cases} x_i - j, & j \le x_i < j+1 \\ 2-(x_i - j), & j+1 \le x_i < j+2, j = 0,\dots,2^i - 2 \\ 0, & otherwise \end{cases} \tag{3.4}$$

with the respective left- and right-hand side boundary scaling functions

$$\chi_{i,j}(x) = \begin{cases} 2-(x_i - j), & 0 \le x_i < 1, j = -1 \\ 0, & otherwise \end{cases} \tag{3.5}$$

$$\chi_{i,j}(x) = \begin{cases} x_i - j, & j \le x_i < j+1, j = 2^i - 1 \\ 0, & otherwise \end{cases} \tag{3.6}$$

the actual coordinate position x is related to x_i according to $x_i = 2^i x$.

The second-order B-spline wavelets are given by

$$\mu_{i,j}(x) = \frac{1}{6}\begin{cases} x_i - j, & j \le x_i < j + 1/2 \\ 4 - 7(x_i - j), & j + 1/2 \le x_i < j+1 \\ -19 + 16(x_i - j), & j+1 \le x_i < j + 3/2 \\ 29 - 16(x_i - j), & j + 3/2 \le x_i < j+2 \\ -17 + 7(x_i - j), & j+2 \le x_i < j + 5/2 \\ 3 - (x_i - j), & j + 5/2 \le x_i < j+3 \\ 0, & otherwise \end{cases} \tag{3.7}$$

with the respective left- and right-hand side boundary wavelets:

$$\mu_{i,j}(x) = \frac{1}{6}\begin{cases} -6+23x_i, & 0 \le x_i < 1/2 \\ 14-17x_i, & 1/2 \le x_i < 1 \\ -10+7x_i, & 1 \le x_i < 3/2 \\ 2-x_i, & j+3/2 \le x_i < 2, j=-1 \\ 0, & otherwise \end{cases} \tag{3.8}$$

$$\mu_{i,j}(x) = \frac{1}{6}\begin{cases} 2-(j+2-x_i), & j \le x_i < j+1/2 \\ -10+7(j+2-x_i), & j+1/2 \le x_i < j+1 \\ 14-17(j+2-x_i), & j+1 \le x_i < j+3/2, j = 2^i-2 \\ -6+23(j+2-x_i), & j+3/2 \le x_i < j+2 \\ 0, & otherwise. \end{cases} \tag{3.9}$$

From (3.4)-(3.9), we get more clear description for these two sets of equations

$$\chi_{i,j}(x) = \frac{1}{2}\chi_{i+1,2j}(x) + \chi_{i+1,2j+1}(x) + \frac{1}{2}\chi_{i+1,2j+2}(x), \quad i = 2,3,\ldots$$

$$j = 0,1,2,\ldots,2^j-2$$

$$\chi_{i,-1}(x) = \frac{1}{2}\chi_{i+1,-1}(x) + \frac{1}{2}\chi_{i+1,0}(x), \ i = 2,3,\ldots \tag{3.10}$$

$$\chi_{i,2^i-1}(x) = \chi_{i+1,2^{i+1}-1}(x) + \frac{1}{2}\chi_{i+1,2^{i+1}}(x), \ i = 2,3,\ldots$$

and

$$\mu_{i,j}(x) = \frac{1}{12}\chi_{i+1,2j}(x) - \frac{1}{2}\chi_{i+1,2j+1}(x) + \frac{5}{6}\chi_{i+1,2j+2}(x)$$

$$-\frac{1}{2}\chi_{i+1,2j+3}(x) + \frac{1}{12}\chi_{i+1,2j+4}(x), \ i = 2,3,\ldots, \ j = 0,1,2,\ldots,2^j-2$$

$$\mu_{i,-1}(x) = -\chi_{i+1,-1}(x) + \frac{11}{12}\chi_{i+1,0}(x) - \frac{1}{2}\chi_{i+1,1}(x) + \frac{1}{12}\chi_{i+1,2}(x),$$

$$\mu_{i,2^i-1}(x) = -\chi_{i+1,2^{i+1}-2}(x) + \frac{11}{12}\chi_{i+1,2^{i+1}-3}(x) - \frac{1}{2}\chi_{i+1,2^{i+1}-4}(x) + \frac{1}{12}\chi_{i+1,2^{i+1}-5}(x).$$

(3.11)

We define

$$\mathrm{X}_{M+1,k}(x)(k = -1,...,2^{M+1}-1) = \left[\chi_{M+1,-1}, \chi_{M+1,0}, \dots, \chi_{M+1,2^{M+1}-1}\right]^T \quad (3.12)$$

let

$$Q_{M+1} = \int_0^1 \mathrm{X}_{M+1}\mathrm{X}^T{}_{M+1}dx. \quad (3.13)$$

The entry $(Q_{M+1})_{ij}$ of the matrix Q_{M+1} in (3.13) is calculated from

$$\int_0^1 \chi_{M+1,i}(x)\chi_{M+1,j}(x)dx \quad (3.14)$$

using (3.4)-(3.6), (3.12) and (3.14) , we get a symmetric

$(2^{M+1}+1)\times(2^{M+1}+1)$ matrix for Q_{M+1} which is given by:

$$Q_{M+1} = \frac{1}{2^{M-1}}\begin{bmatrix} \frac{1}{12} & \frac{1}{24} & 0 & \dots & 0 \\ \frac{1}{24} & \frac{1}{6} & \frac{1}{24} & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & \frac{1}{24} & \frac{1}{6} & \frac{1}{24} \\ 0 & \dots & 0 & \frac{1}{24} & \frac{1}{12} \end{bmatrix}. \quad (3.15)$$

Finally, we want to define a vector M of order $(2^{(M+1)}+1)\times 1$ as:

$$\mathrm{M} = [\chi_{2,-1}, \chi_{2,0}, \ldots, \chi_{2,3}, \mu_{2,-1}, \ldots, \mu_{2,2}, \mu_{3,-1}, \ldots, \mu_{3,6}, \ldots, \mu_{M,-1}, \ldots, \mu_{M,2^M-2}]^T .$$

(3.16)

The operational matrix of derivative

The differentiation of the vector X_{M+1} and M in (3.16) can be express-ed as:

$$\mathrm{X}'_{M+1} = D_{\mathrm{X}}\mathrm{X}_{M+1} \quad , \quad \mathrm{M}' = D_{\mathrm{M}}\mathrm{M} \tag{3.17}$$

where D_{X} and D_{M} are $(2^{M+1}+2)\times(2^{M+1}+2)$ operational matrices of derivative of B-spline scaling functions and wavelets respectively. The matrix D_{X}

$$D_{\mathrm{X}} = \int_0^1 \mathrm{X}'_{M+1}(t)\tilde{\mathrm{X}}^T_{M+1}(t)dt = \left(\int_0^1 \mathrm{X}'_{M+1}(t)\mathrm{X}^T{}_{M+1}(t)dt\right)\left(Q_{M+1}\right)^{-1} = R\left(Q_{M+1}\right)^{-1}$$

(3.18)

where

$$R = \int_0^1 \mathrm{X}'_{M+1}(t)\mathrm{X}^T{}_{M+1}(t)dt. \tag{3.19}$$

In (3.19), R is $(2^{M+1}+1)\times(2^{M+1}+1)$ matrix given by

$$R = \begin{bmatrix} \int_0^1 \chi'_{M+1,-2}(t)\chi_{M+1,-2}(t)dt & \cdots & \int_0^1 \chi'_{M+1,-2}(t)\chi_{M+1,2^M-1}(t)dt \\ \vdots & \ddots & \vdots \\ \int_0^1 \chi'_{M+1,2^{M+1}-1}(t)\chi_{M+1,-2}(t)dt & \cdots & \int_0^1 \chi'_{M+1,2^{M+1}-1}(t)\chi_{M+1,2^{M+1}-1}(t)dt \end{bmatrix}$$

(3.20)

since the elements in the vector given in (3.12) are nonzero between $k/2^{M+1}$ and $(k+2)/2^{M+1}$, then for any entries of $R_{j,k}$, we have

$$\left.\begin{aligned} R_{j,k} &= \int_0^1 \chi'_{M+1,j}(t)\chi_{M+1,k}(t)dt = \int_{k/2^{M+1}}^{(k+2)/2^{M+1}} \chi'_{M+1,j}(t)\chi_{M+1,k}(t)dt \\ &= \int_{k/2^{M+1}}^{(k+1)/2^{M+1}} \chi'_{M+1,j}(t)\chi_{M+1,k}(t)dt + \int_{(k+1)/2^{M+1}}^{(k+2)/2^{M+1}} \chi'_{M+1,j}(t)\chi_{M+1,k}(t)dt \\ &= \int_{k/2^{M+1}}^{(k+1)/2^{M+1}} 2^{M+1}\chi_{M+1,k}(t)dt + \int_{(k+1)/2^{M+1}}^{(k+2)/2^{M+1}} -2^{M+1}\chi_{M+1,k}(t)dt \end{aligned}\right\} . \quad (3.21)$$

From (3.21), we get

$$R = \begin{bmatrix} \frac{-1}{2} & \frac{-1}{2} & & & \\ \frac{1}{2} & 0 & \frac{-1}{2} & & \\ & \ddots & \ddots & \ddots & \\ & & \frac{1}{2} & 0 & \frac{-1}{2} \\ & & & \frac{1}{2} & \frac{1}{2} \end{bmatrix}. \qquad (3.22)$$

The matrix D_{M} can be obtained by considering

$$\mathrm{M} = G\,\mathrm{X}_{M+1} \tag{3.23}$$

where G is a $(2^{M+1}+1)\times(2^{M+1}+1)$ matrix, which can be calculated as follows. Let X and M be defined as:

$$\mathrm{X} = \left[\chi_{j,-1}, \chi_{j,0}, \dots, \chi_{j,2^j-1}\right], \tag{3.24}$$

$$\mathrm{M} = \left[\mu_{j,-1}, \mu_{j,0}, \dots, \mu_{j,2^j-1}\right]. \tag{3.25}$$

Using (3.10) and (3.24) , we get

$$\mathrm{X}_j = \alpha_j \mathrm{X}_{j+1} \tag{3.26}$$

with

$$\alpha_j = \begin{bmatrix} 1 & \frac{1}{2} & & & & & \\ & \frac{1}{2} & 1 & \frac{1}{2} & & & \\ & & & \frac{1}{2} & 1 & \frac{1}{2} & & \\ & & & & \ddots & \ddots & \ddots & \\ & & & & & \frac{1}{2} & 1 & \frac{1}{2} & \\ & & & & & & & \frac{1}{2} & 1 \end{bmatrix} \tag{3.27}$$

where $\alpha_j\,(j=2,3,\dots),$ is $(2^j+1)\times(2^{j+1}+1)$ matrix. From (3.11) and (3.25), we have

$$\mathrm{M}_j = \beta_j \mathrm{X}_{j+1} \tag{3.28}$$

with β_j

$$= \begin{bmatrix} -1 & \frac{11}{12} & \frac{-1}{2} & \frac{1}{12} & & & & & & & & & & & \\ & \frac{1}{12} & \frac{-1}{2} & \frac{5}{6} & \frac{-1}{2} & \frac{1}{12} & & & & & & & & & \\ & & & \frac{1}{12} & \frac{-1}{2} & \frac{5}{6} & \frac{-1}{2} & \frac{1}{12} & & & & & & & \\ & & & & & \frac{1}{12} & \frac{-1}{2} & \frac{5}{6} & \frac{-1}{2} & \frac{1}{12} & & & & & \\ & & & & & & \ddots & \ddots & \ddots & \ddots & \ddots & & & & \\ & & & & & & & \frac{1}{12} & \frac{-1}{2} & \frac{5}{6} & \frac{-1}{2} & \frac{1}{12} & & & \\ & & & & & & & & & \frac{1}{12} & \frac{-1}{2} & \frac{5}{6} & \frac{-1}{2} & \frac{1}{12} & \\ & & & & & & & & & & & \frac{1}{12} & \frac{-1}{2} & \frac{11}{12} & -1 \end{bmatrix} \tag{3.29}$$

where $\beta_j\ (j = 2,3,\ldots)$, is $2^j \times (2^{j+1} + 1)$ matrix.

From (3.26) and (3.28), we get

$$\begin{aligned} \mathrm{X}_j &= \alpha_j \times \alpha_{j+1} \times ... \times \alpha_M \, \mathrm{X}_{M+1}, \\ \mathrm{M}_j &= \beta_j \times \alpha_{j+1} \times \ldots \times \alpha_M \, \mathrm{X}_{M+1}. \end{aligned} \tag{3.30}$$

Using (3.24) and (3.16), we have

$$G = \begin{bmatrix} \alpha_2 \times \alpha_3 \times \ldots \times \alpha_M \\ ---------- \\ \beta_2 \times \alpha_3 \times \ldots \times \alpha_M \\ ---------- \\ \vdots \\ \beta_{M-2} \times \alpha_{M-1} \times \ldots \times \alpha_M \\ ---------- \\ \beta_{M-1} \alpha_M \\ ---------- \\ \beta_M \end{bmatrix}. \quad . \tag{3.31}$$

From (3.17),(3.18), and (3.31), we get

$$\mathrm{M}' = G\,\mathrm{X}'_{M+1} = GD_{\mathrm{X}}\mathrm{X}_{M+1} = GR\,(Q_{M+1})^{-1}\mathrm{X}_{M+1} = GR\,(Q_{M+1})^{-1}G^{-1}\mathrm{M} = D_{\mathrm{M}}\mathrm{M}, \tag{3.32}$$

where

$$D_{\mathrm{M}} = GR\,(Q_{M+1})^{-1}G^{-1}. \tag{3.33}$$

we want to explain the technique. We solve linear Fredholm integro-differential equation by using B-spline wavelets. For this purpose we write (3.1) as

$$\sum_{i=0}^{2}\tau_i(x)y^{(i)}(x) = f(x) + z(x) \quad 0 \le x \le 1, \tag{3.34}$$

where

$$z(x) = \int_0^1 K(x,t)y(t)dt. \tag{3.35}$$

We define

$$y(x) = C^T\,\mathrm{M}(x) \tag{3.36}$$

$$z(x) = \int_0^1 K(x,t)\,C^T\ \mathrm{M}(t)\,dt, \tag{3.37}$$

where $\mathrm{M}(x)$ is defined in (3.16), and C is $(2^{M+1}+1)\times 1$ unknown vector defined as

$$\left[c_{-1}, c_0, \ldots, c_3, d_{2,-1}, \ldots, d_{2,2}, d_{3,1}, \ldots, d_{3,6}, \ldots, d_{M,-1}, \ldots, d_{M,2^M-2}\right]^T$$

We can approximate (3.37), using Newton-Cotes techniques as

$$z(x) = \int_0^1 K(x,t)\,C^T\ \mathrm{M}(t)\,dt = \sum_{i=1}^{n} w_i K(x,t_i)\,C^T\ \mathrm{M}(t_i) = F(x,C), \tag{3.38}$$

where w_i and t_i are weight and nodes of Newton-Cotes integration techniques respectively. Using (3.32) and (3.36), we get

$$\begin{aligned} y'(x) &= C^T \mathrm{M}'(x) = C^T D_{\mathrm{M}} \mathrm{M}(x), \\ y''(x) &= C^T D''_{\mathrm{M}} \mathrm{M}(x), \end{aligned} \qquad . \tag{3.39}$$

From (3.30), (3.34), and (3.35), we get

$$\tau_0(x) C^T \mathrm{M}(x) + \tau_1(x) C^T D_w \mathrm{M}(x) + \tau_2(x) D^2{}_w \mathrm{M}(x) = f(x) + F(x, C). \tag{3.40}$$

Also, using (3.2) and (3.36), we have

$$C^T \Psi(0) = y_0, \quad C^T \Psi(1) = y_1. \tag{3.41}$$

To find the solution $y(x)$, we first collocate (3.40) in $x_i = (2i-1)/(2^{M+2}-2),\ i = 1, 2, \ldots, 2^{M+1}-1,$ the resulting equation generrates $2^{M+1}-1$ linear equations which can be solved using Newton's iterative method. The initial values required to start Newton's method have been chosen by taking $y(x)$ as linear function between $y(0) = y_0$ and $y(1) = y_1$. The total unknown for vector C is $2^{M+1}+1$. These can be obtained by using (3.40) and (3.41).

3.2 Homotopy Perturbation Method (HPM)

The Homotopy perturbation method was introduced and developed by Jihvan He [16]. The Homotopy perturbation method couples a Homotopy technique of topology and a perturbation technique.

Consider the following Fredholm integro-differential equation of the second kind of the form

$$v'(x) = f(x) + \lambda \int_0^1 G(x,y)\, v(y)\, dy. \qquad (3.50)$$

We define the operator

$$L(u) = u'(x) - \lambda \int_0^1 G(x,y)\, u(y)\, dy - f(x) = 0 \qquad (3.51)$$

where $u'(x) = v'(x)$. Next we define the Homotopy $H(u,m)$, $m \in [0,1]$ by

$$H(u,0) = F(u) \quad , \quad H(u,1) = L(u), \qquad (3.52)$$

where $F(u)$ is a functional operator. We construct a convex Homotopy of the form

$$H(u,m) = (1-m)F(u) + mL(u). \qquad (3.53)$$

This Homotopy satisfies (3.52) for $m = 0$ and $m = 1$ respectively. The embedding parameter m monotonously increases from zero to unity as the trivial problem $F(u) = 0$ is continuously deformed to the original problem $L(u) = 0.$

HPM uses the Homotopy parameter m as an expanding parameter to obtain

$$u = w_0 + mw_1 + m^2 w_2 + m^3 w_3 + \ldots. \qquad (3.54)$$

when $m \to 1$, (3.54) corresponds to (3.53) and becomes the approximate solution of (3.51), i.e.,

$$v = \lim_{m \to 1} u = w_0 + w_1 + w_2 + w_3 + \dots \tag{3.55}$$

The series (3.55) is convergent for most cases and the rate of convergence depends on $L(u)$.

Assume $F(u) = u(x) - f(x)$, and substituting (3.54) in (3.51) and equating the terms with identical power of m, we have

$$m^0 : w_0'(x) = f(x) \tag{3.56}$$

$$m^n : w_n'(x) = \int_0^1 G(x, y) w_{n-1}(y) dy, n = 1, 2, ... \tag{3.57}$$

Notice that the recurrence relation (3.57) is the same standard Adomian decomposition method as presented before in chapter two.

3.3 Legendre Polynomial Method

Orthogonal polynomials play a very important role in applications of mathematics, mathematical physics, engineering and computer science. One of the most common set of the Legendre polynomials $\{P_0(x), P_1(x), \dots, P_N(x)\}$ which are orthogonal on $[-1,1]$ and satisfy the Legendre differential equation

$$(1 - x^2) y''(x) - 2xy'(x) + n(n+1)y(x) = 0, \; -1 < x < 1, \; n \geq 0$$

and are given by the form

$$P_n(x) = \frac{1}{2^n}\sum_{k=0}^{[n/2]}(-1)^k \binom{n}{k}\binom{2n-2k}{n} x^{n-2k} \quad n = 0,1,\ldots \tag{3.58.a }$$

Moreover, recurrence formulas associated with derivates of Legendre polynomials are given by the relation

$$P'_{n+1}(x) - P'_{n-1}(x) = (2n+1)P_n(x), \ \ n \geq 1. \tag{3.58.b}$$

Consider the following m^{th} order linear Fredholm Integro-differential equation with variable coefficients

$$\sum_{k=0}^{m} F_k(x) y^{(k)}(x) = g(x) + \lambda \int_{-1}^{1} K(x,t) y(t)\, dt, \ -1 \leq x, \ t \leq 1 \tag{3.59}$$

with conditions

$$\sum_{k=0}^{m-1} a_{jk} y^{(k)}(-1) + b_{jk} y^{(k)}(1) + c_{jk} y^{(k)}(0) = \mu_j, \ j = 0,1,2,\ldots,m-1 \tag{3.60}$$

where the constants $a_{jk}, b_{jk}, c_{jk}, \lambda$ and μ_j are stable constants.

Our aim is to obtain a solution expressed in the form:

$$y(x) = \sum_{n=0}^{N} a_n P_n(x), \quad -1 \leq x \leq 1, \tag{3. 61}$$

so that the Legendre coefficients to be determined are the a_n's where $n = 0,1,2,\ldots N$ and the functions $P_n(x)$ $(n = 0,1,2,\ldots,N)$ are the Legendre polynomials defined by the formula (3.58.a). Here $F_k(x)$, $g(x)$ and $K(x,t)$ are functions defined in the interval $-1 \leq x, t \leq 1$.

Fundamental matrix relations

Equation (3.59) can be written as

$$D(x) = g(x) + \lambda I_f(x) \tag{3.62}$$

where

$$D(x) = \sum_{k=0}^{m} F_k(x) y^{(k)}(x)$$

and

$$I_f(x) = \int_{-1}^{1} K(x,t) y(t) dt \, .$$

We convert the solution $y(x)$ and its derivative $y^{(k)}(x),$ the parts $D(x)$ and $I_f(x)$, and the mixed conditions in (3.60) to matrix form.

Matrix relations for $y(x)$ and $y^{(k)}(x)$

The function $y(x)$ defined by (3.61) can be expanded to the truncated Legendre series in the form

$$y(x) = \sum_{n=0}^{N} a_n P_n(x), \quad -1 \le x \le 1, \tag{3.63}$$

formula (3.63) and its derivative can be written in the matrix forms,

$$[y(x)] = \mathrm{P}(x)\mathrm{A} \quad \text{and} \quad [y^{(k)}(x)] = \mathrm{P}^{(k)}(x)\mathrm{A}, \tag{3.64}$$

where

$$\mathrm{P}(x) = [P_0(x) \quad P_1(x) \;\ldots\ldots P_N(x)]$$

$\mathrm{P}^{(k)}(x)=[P^{(k)}(x)\quad P_1^{(k)}(x)\ \ldots.\ P_N{}^{(k)}(x)]$

and $\qquad \mathrm{A}=[a_0\quad a_1\ldots.\ a_N\ ],$

which A is the coefficient matrix.

On the other hand, by using the Legendre recursive formulas (3.58.a) and (3.58.b) and taking $n=0,1,\ldots,N$ we can obtain the matrix equation

$$\mathrm{P}^{(1)}(x)=\mathrm{P}(x)\prod^{T}, \qquad (3.65)$$

where, for odd values of N

$$\begin{bmatrix} 0 & 0 & 0 & 0 & \ldots. & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & \ldots. & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & \ldots. & 0 & 0 & 0 \\ 1 & 0 & 5 & 0 & \ldots. & 0 & 0 & 0 \\ : & : & : & : & & : & : & : \\ 0 & 3 & 0 & 7 & \ldots. & 2N-3 & 0 & 0 \\ 1 & 0 & 5 & 0 & \ldots. & 0 & 2N-1 & 0 \end{bmatrix}$$

for even values of N

$$\begin{bmatrix} 0 & 0 & 0 & 0 & \ldots. & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & \ldots. & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & \ldots. & 0 & 0 & 0 \\ 1 & 0 & 5 & 0 & \ldots. & 0 & 0 & 0 \\ : & : & : & : & & : & : & : \\ 1 & 0 & 5 & 0 & \ldots. & 2N-3 & 0 & 0 \\ 0 & 3 & 0 & 7 & \ldots. & 0 & 2N-1 & 0 \end{bmatrix}$$

Also, it is clearly seen that the relation between the matrix $P(x)$ and its derivative $P^{(k)}(x),$ from (3.65), is

$$\begin{aligned} &P^{(1)}(x) = P(x)\prod^{T} \\ &P^{(2)}(x) = P^{(1)}(x)\prod^{T} = P(x)(\prod^{T})^{2} \qquad (3.66) \\ &P^{(k)}(x) = P^{(\mathrm{k}-1)}(x)(\prod^{T})^{k-1} = P(x)(\prod^{T})^{k}, k = 0,1,2,.... \end{aligned}$$

Consequently, by substituting the matrix relations (3.66) into Eq. (3.64), we obtain the matrix relations for $y(x)$ and $P^{(k)}(x)$ as

$$[y^{(k)}(x)] = \mathrm{P}(x)(\prod^{T})^{k}\,\mathrm{A}. \qquad (3.67)$$

Matrix representations based on collocation points

The Legendre collocation points defined by

$$x_i = -1 + \frac{2}{N}i, \quad i = 0,1,...,N, \qquad (3.68)$$

we substitute the collocation points (3.68) into Eq. (3.62) to obtain the system

$$D(x_i) = g(x_i) + \lambda I_f(x_i)$$

Represented in matrix equation as

$$\mathrm{D} = \mathrm{G} + \lambda\,\mathrm{I}_f \qquad (3.69)$$

where

$$\mathrm{D} = \begin{bmatrix} D(x_0) \\ D(x_1) \\ \vdots \\ D(x_N) \end{bmatrix}, \mathrm{G} = \begin{bmatrix} g(x_0) \\ g(x_1) \\ \vdots \\ g(x_N) \end{bmatrix}, \mathrm{I} = \begin{bmatrix} I_f(x_0) \\ I_f(x_1) \\ \vdots \\ I_f(x_N) \end{bmatrix}$$

Matrix relation for the differential part $D(x)$

To reduce the $D(x)$ part to the matrix form by means of the collocation points, we first write the matrix D defined in eq.(3.69) as

$$\mathrm{D} = \sum_{k=1}^{m} \mathrm{F}_k Y^{(k)} \tag{3.70}$$

where

$$\mathrm{F}_k = \begin{bmatrix} F_k(x_0) & 0 & \cdots & 0 \\ 0 & F_k(x_1) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & F_k(x_N) \end{bmatrix},$$

$$Y^{(k)} = \begin{bmatrix} y^{(k)}(x_0) \\ y^{(k)}(x_1) \\ \vdots \\ y^{(k)}(x_N) \end{bmatrix}.$$

Using the collocation points $x_i\ (i = 0, 1, \ldots N)$ in (3.67) we have the system of matrix equations

$$[y^{(k)}(x_i)] = \mathrm{P}(x_i)(\Pi^T)^k\,\mathrm{A}, \quad k = 0,1,\ldots,m$$

or

$$Y^{(k)} = \begin{bmatrix} y^{(k)}(x_0) \\ y^{(k)}(x_1) \\ \vdots \\ y^{(k)}(x_N) \end{bmatrix} = \begin{bmatrix} P(x_0) \\ P(x_1) \\ \vdots \\ P(x_N) \end{bmatrix} [(\Pi^T)^k\,\mathrm{A}] = \mathrm{P}(\Pi^T)^k\,\mathrm{A}, \tag{3.71}$$

where

$$\mathrm{P} = \begin{bmatrix} P_0(x_0) & P_1(x_0) & \dots & P_N(x_0) \\ P_0(x_1) & P_1(x_1) & \dots & P_N(x_1) \\ \vdots & \vdots & & \vdots \\ P_0(x_N) & P_1(x_N) & \dots & P_N(x_N) \end{bmatrix}$$

Thus, from the matrix forms (3.70) and (3.71), we get the fundamental matrix relation for the differential part $D(x)$

$$\mathrm{D} = \sum_{k=0}^{m} \mathrm{F}_k \, \mathrm{P}(\Pi^T)^k \, \mathrm{A}. \qquad (3.72)$$

Matrix relation for the Fredholm integral part $I_f\ (x)$

The kernel function $K(x,t)$ can be approximated by the truncated

Legendre series

$$K(x,t) = \sum_{m=0}^{N} \sum_{n=0}^{N} k_{mn}^{t} P_m(x) P_n(t) \qquad (3.73)$$

or

$$K(x,t) = \sum_{m=0}^{N} \sum_{n=0}^{N} k_{mn}^{t} x^m t^n \qquad (3.74)$$

where

$$k_{mn}^{t} = \frac{1}{m!n!} \frac{\partial^{m+n} K(0,0)}{\partial x^m \partial t^n}; \ m,n = 0,1,\dots,N.$$

We convert (3.73) and (3.74) to matrix forms and then equalize

$$[K(x,t)] = \mathrm{P}(x)\mathrm{K}_l \mathrm{P}^T(t) = \mathrm{X}(x)\mathrm{K}_t \mathrm{X}^T(t) \qquad (3.75)$$

where

$$\mathrm{P}(x)=[P_0(x)\quad P_1(x)\ \dots\ P_N(x)]$$

$$\mathrm{X}(x)=[1\quad x\ \dots\ x^N]$$

$$\mathrm{K}_l=[k^l_{mn}]\ ,\quad \mathrm{K}_t=[k^t_{mn}];\ m,n=0,1,\dots,N\,.$$

On the other hand, by using the Legendre recursive formula (3.58.a) for $n=0,1,\dots,N$ we can obtain the matrix equation

$$\mathrm{P}^T(x)=D\mathrm{X}^T(x)\quad \text{or}\qquad \mathrm{P}(x)=\mathrm{X}(x)D^T \tag{3.76}$$

where for odd values of N and for even values of N see [42].

Therefore, substituting the matrix $P(x)$ in (3.76) into the relation (3.75) and simplifying the result equation, we find the matrix relation

$$\mathrm{K}_l=(D^{-1})^T\,\mathrm{K}_t D^{-1} \tag{3.77}$$

which is the relation between the Legendre and Taylor coefficients of $K(x,t)$. Substituting the matrix forms (3.75) and (3.64) corresponding to the functions $K(x,t)$ and $y(t)$ into the Fredholm integral part $I_f(x)$, we obtain

$$[I_f(x)]=\int_{-1}^{1}\mathrm{P}(x)\mathrm{K}_l\mathrm{P}^T(t)\mathrm{P}(t)\mathrm{A}dt=\mathrm{P}(x)\mathrm{K}_l\mathrm{Q}\mathrm{A}, \tag{3.78}$$

where

$$\mathrm{Q}=\int_{-1}^{1}\mathrm{P}^T(t)\mathrm{P}(t)dt=[q_{mn}];$$

$$q_{mn}=\begin{cases}\dfrac{2N}{2m+1} & : \quad m=n \\ 0 & : \quad m\neq n\end{cases} \; ; \; m,n \;=0,1,\ldots,N$$

and the matrix K_l is defined in (3.77).

Using the collocation points $x_i\,(i=0,1,\ldots,N)$ defined in (3.68) in the relation (3.78) we obtain the system of the matrix equations

$$\mathrm{I}_f(x_i)=\mathrm{P}(x_i)_l\,\mathrm{K}_l\mathrm{QA};\; i=0,1,\ldots,N$$

or briefly
$$\mathrm{I}_f=\mathrm{PK}_l\mathrm{QA}; \tag{3.79}$$

which is the fundamental matrix relation for $I_f(x)$.

Matrix relation for the mixed conditions

We can obtain the corresponding matrix form for the conditions (3.60), by means of the relation (3.66), as

$$\sum_{k=0}^{m-1}[a_{jk}P(-1)+b_{jk}P(1)+c_{jk}P(0)](\Pi^T)^k\,\mathrm{A}=\mu_j,\; j=0,1,\ldots,m-1. \tag{3.80}$$

Method of solution

Now, we are ready to construct the fundamental matrix equation corresponding to (3.59). For this purpose, substituting the matrix relations (3.72) and (3.79) into Eq. (3.69) then simplifying it, we obtain the matrix equation

$$\sum_{k=0}^{m}\{\mathrm{F}_k\mathrm{P}(\prod^T)^k-\lambda\mathrm{PK}_l\mathrm{Q}\}\mathrm{A}=\mathrm{G}, \tag{3.81}$$

which corresponds to a system of $(N+1)$ algebraic equations for the $(N+1)$ unknown Legendre coefficients $a_0,\ a_1,\dots a_N$. Briefly, we can write Eq. (3.81) in the form

$$\mathrm{WA}=\mathrm{G} \quad \text{or} \quad [\mathrm{W}\,;\mathrm{G}] \tag{3.82}$$

where

$$\mathrm{W}=[w_{pq}]=\sum_{k=0}^{m}\mathrm{F}_k\,\mathrm{P}(\prod^{T})^{k}-\lambda \mathrm{PK}_l\mathrm{Q},\ \ p,q=0,1,\dots,N$$

$$\mathrm{G}=[g(x_0)\quad g(x_1)\quad \dots\ g(x^{N})]^{T}.$$

On the other hand, the matrix form (3.80) for the conditions (3.60) can be written as

$$\mathrm{U}_j\mathrm{A}=[\mu_j] \quad \text{or} \quad [\mathrm{U}_j\,;\mu_j]\ ,\ \ j=0,1,\dots,m-1 \tag{3.83}$$

where

$$\mathrm{U}_j=\sum_{k=0}^{m-1}[a_{jk}P(-1)+b_{jk}P(1)+c_{jk}P(0)](\prod^{T})^{k}$$

$$\equiv[u_{j0}\quad u_{j1}\quad \dots.\ u_{jN}].$$

To obtain the solution of Eq. (3.82) under the conditions (3.83), by replacing the rows matrices (3.83) by the last m rows of the matrix (3.81) we have the new augmented matrix

$$[\tilde{W};\tilde{G}] = \begin{bmatrix} w_{00} & w_{01} & \cdots & w_{0N} & ; & g(x_0) \\ w_{10} & w_{11} & \cdots & w_{1N} & ; & g(x_1) \\ \vdots & \vdots & \ddots & \vdots & ; & \vdots \\ w_{N-m,0} & w_{N-m,1} & \cdots & w_{N-m,N} & ; & g(x_{N-m}) \\ u_{00} & u_{01} & \cdots & u_{0N} & ; & \mu_0 \\ u_{10} & u_{11} & \cdots & u_{1N} & ; & \mu_1 \\ \vdots & \vdots & \ddots & \vdots & ; & \vdots \\ u_{m-1,0} & u_{m-1,1} & \cdots & u_{m-1,N} & ; & \mu_{m-1} \end{bmatrix} \tag{3.84}$$

If rank $\tilde{W} = \text{rank}[\tilde{W};\tilde{G}] = N+1$, then we can write

$$A = (\tilde{W})^{-1}\tilde{G}.$$

Thus the coefficients $a_n (n = 0,1,...,N)$ are uniquely determined by Eq. (3.84).

3.4 Taylor Collocation Method

Consider the mth -order linear Fredholm integro-differential equation

$$\sum_{k=0}^{m} P_k(x) y^{(k)}(x) = f(x) + \lambda \int_a^b K(x,t) y(t)\, dt \tag{3.85}$$

where the unknown functions $P_k(x)$, $f(x)$, $K(x,t)$ are defined on $a \le x,t \le b$ and λ is a real parameter where $y(x)$ is the unknown function.

With conditions

$$\sum_{j=0}^{m-1} [a_{ij} y^{(j)}(a) + b_{ij} y^{(j)}(b) + c_{ij} y^{(j)}(c)] = \lambda_i, i = 0,1,...,m-1 \tag{3.86}$$

where $a \le c \le b$, provided that the real coefficients a_{ij}, b_{ij}, c_{ij} and λ_i are appropriate constants.

We assume that the solution of (3.85) be the truncated Taylor series

$$y(x) = \sum_{n=0}^{N} \frac{y^{(n)}(c)}{n!}(x-c)^n; \qquad a \le x \le b \tag{3.87}$$

where N is chosen any positive integer such that $N \ge m$. Then the solution (3.87) of Eq. (3.85) can be expressed in the matrix form

$$[y(x)] = \mathrm{X}\,\mathrm{M}_0\,\mathrm{A}$$

where

$$\mathrm{X} = [1 \quad x-c \quad (x-c)^2 \quad \ldots (x-c)^N\,]$$

$$\mathrm{A} = [y^{(0)}(c) \quad y^{(1)}(c) \quad y^{(2)}(c) \quad \ldots y^{(N)}(c)]$$

and

$$\mathrm{M}_0 = \begin{bmatrix} \frac{1}{0!} & 0 & 0 & \cdots & 0 \\ 0 & \frac{1}{1!} & 0 & \cdots & 0 \\ 0 & 0 & \frac{1}{2!} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \frac{1}{N!} \end{bmatrix}.$$

To obtain a solution, we can use the following matrix method, which is a Taylor Collocation method. This method is based on computing the Taylor coefficients by means of the Taylor collocation points are thereby finding the matrix A containing the unknown Taylor coefficients.

We substitute the Taylor collocation points

$$x_i = a + i\,\frac{b-a}{N}; \quad i = 0,1,...,N; \ \ x_0 = a,\ x_N = b \tag{3.88}$$

into (3.85) to obtain

$$\sum_{k=0}^{m} P_k(x_i)y^{(k)}(x_i) = f(x_i) + \lambda I(x_i) \tag{3.89}$$

such that

$$I(x_i) = \int_a^b K(x,t)y(t)dt$$

then we can write the system (3.81) in the matrix form

$$\mathrm{P}_0\mathrm{Y}^{(0)} + \mathrm{P}_1\mathrm{Y}^{(1)} + ... + \mathrm{P}_m\mathrm{Y}^{(m)} = \sum_{k=0}^{m}\mathrm{P}_k\mathrm{Y}^{(k)} = \mathrm{F} + \lambda\mathrm{I} \tag{3.90}$$

where

$$\mathrm{P}_k = \begin{bmatrix} P_k(x_0) & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & P_k(x_N) \end{bmatrix}$$

$$\mathrm{F}=\begin{bmatrix} f(x_0) \\ f(x_1) \\ \vdots \\ f(x_N) \end{bmatrix},\ \mathrm{Y}^{(k)}=\begin{bmatrix} y^{(k)}(x_0) \\ y^{(k)}(x_1) \\ \vdots \\ y^{(k)}(x_N) \end{bmatrix},\ \mathrm{I}=\begin{bmatrix} I(x_0) \\ I(x_1) \\ \vdots \\ I(x_N) \end{bmatrix}.$$

Assume the k^{th} derivative of the function (3.87) with respect to x has the truncated Taylor series expansion defined by (3.87):

$$y^{(k)}(x_i)=\sum_{n=k}^{N}\frac{y^{(n)}(c)}{(n-k)!}(x_i-c)^{n-k}; \quad a\le x\le b$$

where $y^{(k)}(x)$ $(k=0,\ldots,N)$ are Taylor coefficients; $y^{(0)}(x)=y(x)$.

Then substituting the Taylor collocation points, we obtain

$$[y^{(k)}(x_i)]=\mathrm{X}_{x_i}\mathrm{M}_k\mathrm{A},\ k=0,1,\ldots,N \tag{3.91}$$

or the matrix equation

$$y^{(k)}=\mathrm{C}\mathrm{M}_k\mathrm{A} \tag{3.92}$$

where

$$\mathrm{C}=\begin{bmatrix} \mathrm{X}_{x_0} \\ \mathrm{X}_{x_1} \\ \vdots \\ \mathrm{X}_{x_N} \end{bmatrix}=\begin{bmatrix} (x_0-c)^0 & (x_0-c)^1 & \ldots & (x_0-c)^N \\ (x_1-c)^0 & (x_1-c)^1 & \cdots & (x_0-c)^N \\ \vdots & \vdots & \ddots & \vdots \\ (x_N-c)^0 & (x_N-c)^1 & \ldots & (x_N-c)^N \end{bmatrix}$$

$$\mathrm{M}_k = \begin{bmatrix} 0 & 0 & \cdots & \frac{1}{0!} & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & \frac{1}{1!} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & \frac{1}{(N-k)!} \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \end{bmatrix}.$$

Then Eq. (3.88) can be written as

$$(\sum_{k=0}^{m} \mathrm{P}_k \mathrm{C} \mathrm{M}_k)\mathrm{A} = \mathrm{F} + \lambda \mathrm{I} \tag{3.93}$$

The kernel $K(x,t)$ is expanded to construct Taylor series

$$K(x,t) = \sum_{n=0}^{N} \sum_{m=0}^{N} k_{nm} (x-c)^n (t-c)^m$$

$$k_{nm} = \frac{1}{n!m!} \frac{\partial^{n+m} K(0,0)}{\partial x^n \partial t^m}\bigg|_{(x=c,t=c)}.$$

The matrix representation of $K(x,t)$ defined by

$$[K(x,t)] = \mathrm{X}\ \ \mathrm{K}\ \ \mathrm{T}^T \tag{3.94}$$

where

$$\mathrm{X} = [1 \quad x-c \quad (x-c)^2 \quad ....(x-c)^N\,]$$

$$\mathrm{T} = [1 \quad t-c \quad (t-c)^2 \quad .... \quad (t-c)^N\,]$$

$$\mathrm{K} = \begin{bmatrix} k_{00} & k_{01} & \cdots & k_{0N} \\ k_{10} & k_{11} & \cdots & k_{1N} \\ \vdots & \vdots & \ddots & \vdots \\ k_{N0} & k_{N1} & \cdots & k_{NN} \end{bmatrix}$$

In addition, the matrix representation of $y(x)$ and $y(t)$ are

$$[y(x)] = \mathrm{X}\ \mathrm{M}_0\ \mathrm{A}, \quad [y(t)] = \mathrm{T}\ \mathrm{M}_0\ \mathrm{A}. \tag{3.95}$$

Setting (3.94) and (3.95) into $I(x_i)$ defined in (3.89), we have

$$[I(x)] = \int_a^b \{\mathrm{XKT}^T\ \mathrm{TM}_0\mathrm{A}\}dt = \mathrm{XKHM}_0\mathrm{A} \tag{3.96}$$

$$\mathrm{H} = [\mathrm{h}_{nm}] = \int_a^b \mathrm{T}^T\ \mathrm{T}dt, \ \ \mathrm{h}_{nm} = \left.\frac{(b-c)^{n+m+1} - (a-c)^{n+m+1}}{n+m+1}\right|_{n,m=0,1,\ldots,N}$$

from (3.96) we get the matrix I in the form

$$\mathrm{I} = \mathrm{C\ K\ H\ M}_0\ \mathrm{A}. \tag{3.97}$$

Finally, substituting (3.97) in (3.93), we get the matrix equation

$$(\sum_{k=0}^{m} \mathrm{P}_k \mathrm{CM}_k \text{ - } \lambda \mathrm{CKHM}_0)\mathrm{A} = \mathrm{F} \tag{3.98}$$

which is the fundamental relation for solving of Fredholm integro-differential equation defined in $a \le x \le b$.

Equation (3.98) can be written as

$$\mathrm{W\ \ A\ \ =\ \ F} \tag{3.99}$$

which corresponds to a system of $(N+1)$ algebraic equations with the unknown Taylor coefficients

$$\mathrm{W}=[w_{ij}]=\sum_{k=0}^{m}\mathrm{P}_k\,\mathrm{CM}_k-\lambda\mathrm{CKHM}_0,\quad i,j=0,1,\ldots,N.$$

Also, for the conditions in (3.86). We assume

$$\begin{aligned}\left[y^{(j)}(a)\right]&=\mathrm{PM}_j\mathrm{A}\\ \left[y^{(j)}(b)\right]&=\mathrm{QM}_j\mathrm{A}\\ \left[y^{(j)}(c)\right]&=\mathrm{RM}_j\mathrm{A}\end{aligned}\qquad(3.100)$$

where

$$\mathrm{P}=[1\ \ (a-c)\quad (a-c)^2\ \ldots.(a-c)^N\,]$$

$$\mathrm{Q}=[1\ \ (b-c)\quad (b-c)^2\ \ldots.(b-c)^N\,]$$

$$\mathrm{R}=[1\quad 0\quad 0\ \cdots 0].$$

Setting (3.100) in (3.86), we get

$$\sum_{j=0}^{m-1}\{a_{ij}\mathrm{P}+b_{ij}\mathrm{Q}+c_{ij}\mathrm{R}\}\mathrm{M}_j\mathrm{A}=[\lambda_i]$$

Let

$$\mathrm{U}_i=\sum_{j=0}^{m-1}\{a_{ij}\mathrm{P}+b_{ij}\mathrm{Q}+c_{ij}\mathrm{R}\}\mathrm{M}_j=[u_{i0}\quad u_{i1}\ \ldots.u_{iN}\,],\ \ i=0,1,\ldots.,m-1.\quad(3.101)$$

Thus, the conditions (3.86) becomes

$$\mathrm{U}_i\ \mathrm{A} = [\lambda_i] \tag{3.102}$$

also the augmented matrices representing them are

$$[\mathrm{U}_i ; \lambda_i] = [u_{i0} \quad u_{i1} \ \ldots \ u_{iN} \ ; \ \lambda_i]. \tag{3.103}$$

Finally, the augmented matrix $[\mathrm{W};\mathrm{F}]$ is defined by the matrices:

$$\mathrm{W} = \begin{bmatrix} w_{00} & w_{01} & \cdots & w_{0N} \\ w_{10} & w_{11} & \cdots & w_{1N} \\ \vdots & \vdots & \ddots & \vdots \\ w_{N-m,0} & w_{N-m,1} & \cdots & w_{N-m,N} \\ u_{00} & u_{01} & \cdots & u_{0N} \\ \vdots & \vdots & \ddots & \vdots \\ u_{m-1,0} & u_{m-1,1} & \cdots & u_{m-1,N} \end{bmatrix}$$

$$\mathrm{F} = \begin{bmatrix} f\,(x_0) \\ f\,(x_1) \\ \vdots \\ f\,(x_{N-m}) \\ \lambda_0 \\ \vdots \\ \lambda_{m-1} \end{bmatrix}.$$

If, $|\mathrm{W}| \neq 0$ we can write

$$\mathrm{A} = \mathrm{W}^{-1}\ \mathrm{F}$$

Matrix A is uniquely determined. Then (3.85) has a unique solution in the form (3.87).

Chapter Four

Numerical Examples and Results

To test the efficiency and accuracy of the numerical methods represented in chapter three we will consider the following numerical examples:

Example 4.1

Consider the Fredholm integro-differential equation of the second kind

$$u'(x) = 3e^{3x} - \frac{1}{3}(2e^{3}+1)x + \int_0^1 3xt\, u(t)\,dt \quad \text{with} \quad u(0)=1. \tag{4.1}$$

Equation (4.1) has the exact solution $u(x) = e^{3x}$.

4.1 The numerical realization of equation (4.1) using the B-spline scaling functions and wavelets on [0,1]

The following algorithm implements the **B-spline scaling functions and wavelets on [0,1]** using the **Maple** software

Algorithm 4.1 [25]

1. Input the fixed positive integer M
2. Input the values of the matrix Q_{M+1}
3. Input the values of the matrix R
4. Input the values of the matrix $\alpha(j)$
5. Input the values of the matrix $\beta(j)$

6. Calculate the matrix G, by using (3.24) and (3.38).

7. Calculate the matrix D_1

8. For i from 2 to $2^{M+2}-2$, $X(1)=0,\ X(2^{M+1}+1)=1$

   $X(i)=(2i-1)/(2^{M+2}-2)$

9. Defined $\chi_{i,j}(x),\ \chi_{i,-1}(x),\ \chi_{i,2^i-1}(x)$

10. Defined $\mu_{i,j}(x),\ \mu_{i,-1}(x),\ \mu_{i,2^i-1}(x)$

11. Input $h=\dfrac{(b-a)}{M},\ t(i)=i*h$

12. For i from 1 to 5, for j from 1 to $2^{M+1}+1$, for k from 2 to M, for I from 1 to 2^K, define $mut(i,j)=(mu(2,i-2,t(j-1))$, and $mut_1(i,j)=(mu(k,I-2,(j-1))$ to calculate $mutt=\langle mut, mut_1\rangle$

13. Define the kernel

14. For i from 1 to 5, for j from 1 to $2^{M+1}+1$, for k from 2 to M, for I from 1 to 2^K, define $muX(i,j)=(chi(2,i-2,X(j))$, and $muX_1(i,j)=(mu(K,I-2,\mathrm{X}(j))$

15. Calculate $muXX=\langle muX, muX_1\rangle$

16. Calculate $zz=z-$ integration

17. $F(1,1)=1$, and calculate $F(1,i)$

18. Calculate $\mathrm{C}=F*$ Matrix Inverse zz

19. The solution is $U=C*muXX$

20. Input the exact solution

$u = Transpose(U)$

$u_1 = convert(u, vector)$

$r := plot(X, u_1)$

$r_2 := plot(u_2(x))$

21. Display (r, r_2)

22. Absolute error = $|u - u_M|$

Table 4.1 shows the exact and numerical solutions when applying algorithm 4.1 on equation (4.1), and showing the resulting error of using the numerical solution.

Table 4.1: The exact and numerical solutions of applying Algorithm 4.1 on. equation (4.1).

x	Exact solution $u(x) = \exp(3x)$	Numerical solutionu_M	Absolute error $= \|u - u_M\|$
0.1	1.349858808	1.36616469332652	0.0163058853265190
0.2	1.822118800	1.85217464520521	0.0300558452052084
0.3	2.459603111	2.48804090541187	0.0284377944118681
0.4	3.320116923	3.33272256083022	0.012605637830226
0.5	4.4816890701	4.51948395416716	0.0377948841671625
0.6	6.049647464	6.06159218771469	0.0119447237146941
0.7	8.166169913	8.19415390871779	0.0279839957177916
0.8	11.02317638	11.0673029532516	0.0441265732516403
0.9	14.87993172	14.8583456549649	0.0213860650350881
1.0	20.08553692	19.7454628997592	0.340074020240781

These results show the accuracy of the B-spline scaling functions and wavelets to solve equation (4.1) with the max error = 0.34007402024078.

Figure 4.1 compares the exact solution $u(x) = e^{3x}$ with the approximate solution with $M = 4$.

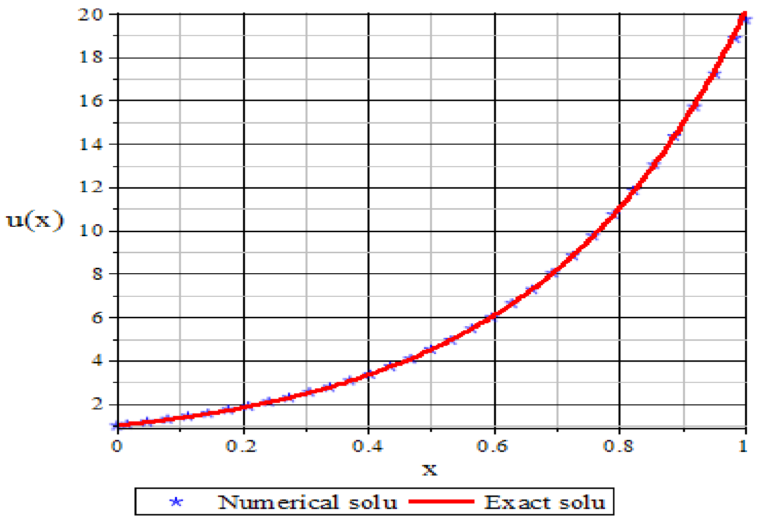

Figure 4.1: The exact and numerical solutions of applying algorithm 4.1 for equation (4.1).

Figure 4.2 shows the absolute error resulting of applying algorithm 4.1 on equation (4.1).

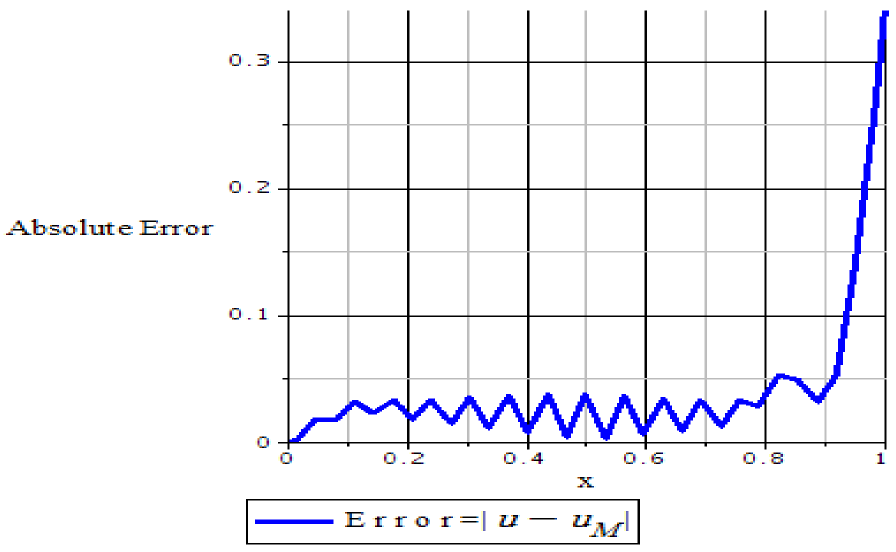

Figure 4.2: The error resulting of applying algorithm 4.1 on equation (4.1).

4.2 The numerical realization of equation (4.1) using the Homotopy perturbation method

The following algorithm implements the **Homotopy perturbation method** using the **Maple** software

Algorithm 4.2

```
(1) Input $a, \lambda, n, f(x), G(x,y)$
(2) Calculate $wm1[0] = int((f, x))$
(3) Calculate $wm[0] := y \to sub(x = y, wm1[0])$
(4) Calculate $c = a - eval(wm[0](0))$
(5) Define $ws[0] = wm1[0] + c$
(6) Calculate $w[0] = subs(x = y, ws[0])$
(7) For $i = 1$ to $n$, calculate
    $wm[i] = (int(G * w[i-1], y = 0...1))$
    $ws[i] = (int(wm[i], x))$
    $w[i] = subs(x = y, ws[i])$
(9) Define $u_n = combine(add(ws[k], k = 0..1)$ ,
(10) Define
    $u := x \to \exp(x)$
    $r := plot(u_n(x))$
    $r_2 := plot(u(x))$
(11) Absolute Error = $|u - u_n|$
```

For more details see [21, 43].

Table (4.2) shows the exact and numerical results when $n = 4$, and showing the error resulting of using the numerical solution.

Table (4.2): The exact and numerical solution of applying Algorithm 4.2 on equation (4.1).

x	Exact solution $u(x) = e^{3x}$	Numerical solution $u_n(x)$	Absolute Error $= \|u - u_n\|$
0.1	1.349858808	1.348501851	0.001356957
0.2	1.822118800	1.816690973	0.005427827
0.3	2.459603111	2.447390500	0.012212611
0.4	3.320116923	3.298405614	0.021711309
0.5	4.4816890704	4.447765150	0.033923920
0.6	6.049647463	6.000797020	0.048850444
0.7	8.166169918	8.099679031	0.066490882
0.8	11.023176388	10.936331144	0.08684524
0.9	14.879731722	14.679818222	0.10991350
1	20.085536922	19.949841244	0.13569568

These results show the accuracy of the Homotopy perturbation method to solve equation (4.1) with the max error = 0.13569568.

Figure 4.3: shows both exact and numerical solutions with $n = 4$.

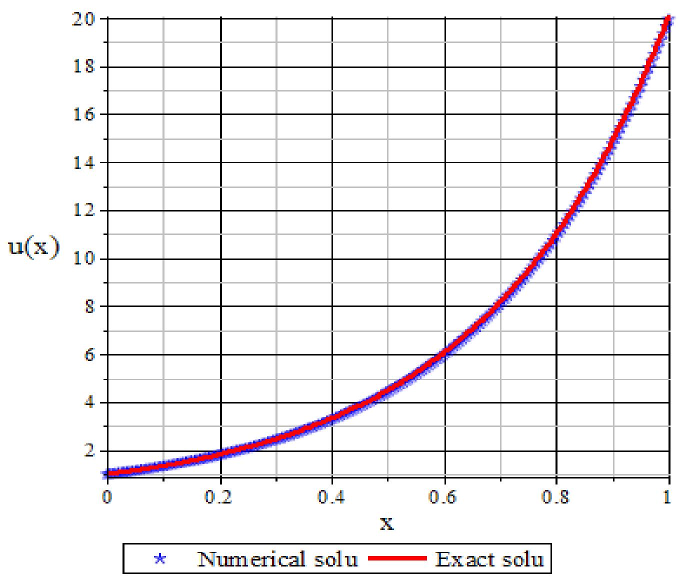

Figure 4.3: The exact and numerical solutions of applying algorithm 4.2

for equation (4.1)

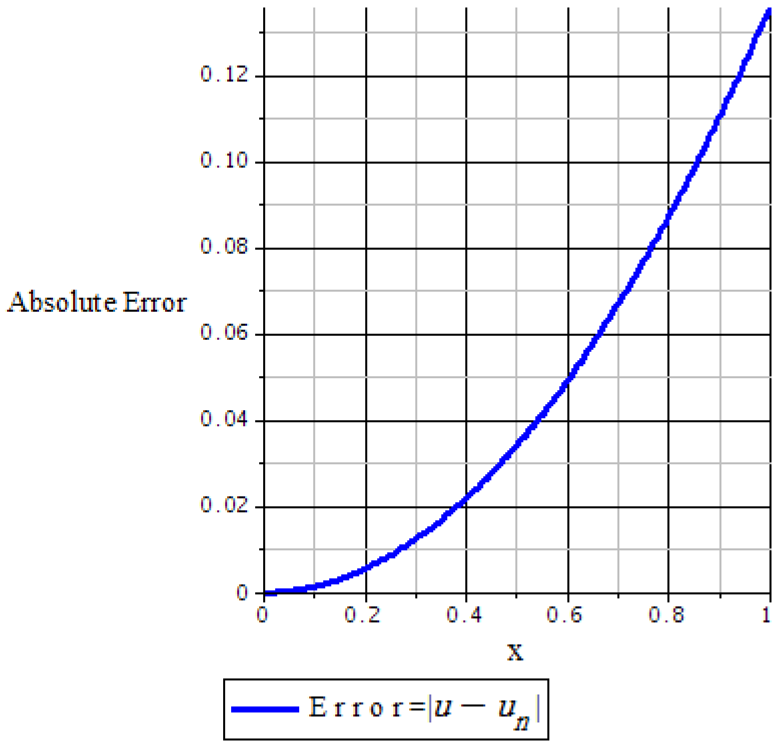

Figure 4.4: The error resulting of applying algorithm 4.2 on equation (4.1).

Example 4.2

Consider the Fredhom integro-differential equation

$$u'(x) = -2\pi\sin(2\pi x) - \pi\sin(4\pi x) + 2\pi\int_0^1 \sin(4\pi x + 2\pi t)u(t)dt, \qquad (4.2)$$
$$u(0) = 1.$$

Equation (4.2) has the exact solution $u(x) = \cos(2\pi x)$.

4.3 The numerical realization of equation (4.2) using the B-spline scaling functions and wavelets on [0,1]

Table (4.3) shows the exact and numerical results when $M = 8$, and showing the error resulting of using the numerical solution.

Table (4.3): The exact and numerical solution of applying Algorithm 4.1 on equation (4.2).

x	Exact solution $u(x)=\cos(2\pi x)$	Approximate solution u_M	Aboslute Error $\lvert u-u_M \rvert$
0.1	0.8090169943	0.716839678443304	0.0921773158566965
0.2	0.3090169938	0.198368704577070	0.11064828932222930
0.3	-0.3090169938	-0.414317811956477	0.105300818156477
0.4	-0.8090169943	-0.909564138036594	0.100547143736594
0.5	-1	-1.09848238174981	0.0984823817498084
0.6	0.8090169943-	-0.910024012119846	0.101007017819846
0.7	0.3090169938-	-0.414601813875286	0.105584820075286
0.8	0.3090169938	0.198637238538349	0.1110379755261651
0.9	0.8090169943	0.717312752695359	0.0917042416046411
1	1	1.00027574181029	0.000275741810288777

These results show the accuracy of the B-spline scaling functions and wavelets to solve equation (4.2) with the max error =0.11064828932222930.

Figure 4.5 shows both the exact and the numerical solutions with $M=8$.

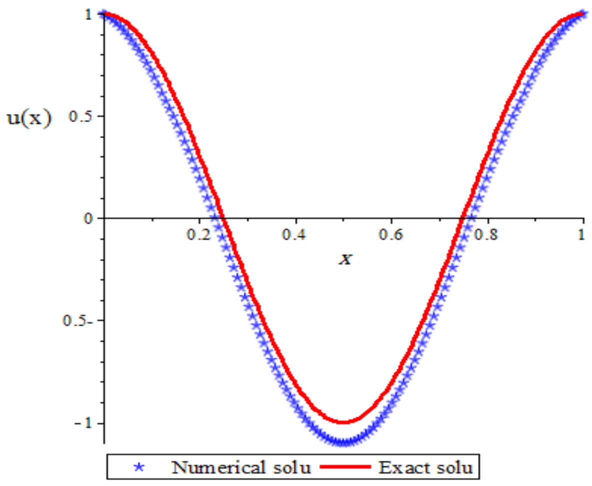

Figure 4.5: The exact and numerical solutions of applying algorithm 4.1 on equation (4.2).

Figure 4.6 shows the absolute error resulting of applying algorithm 4.1 on equation (4.2).

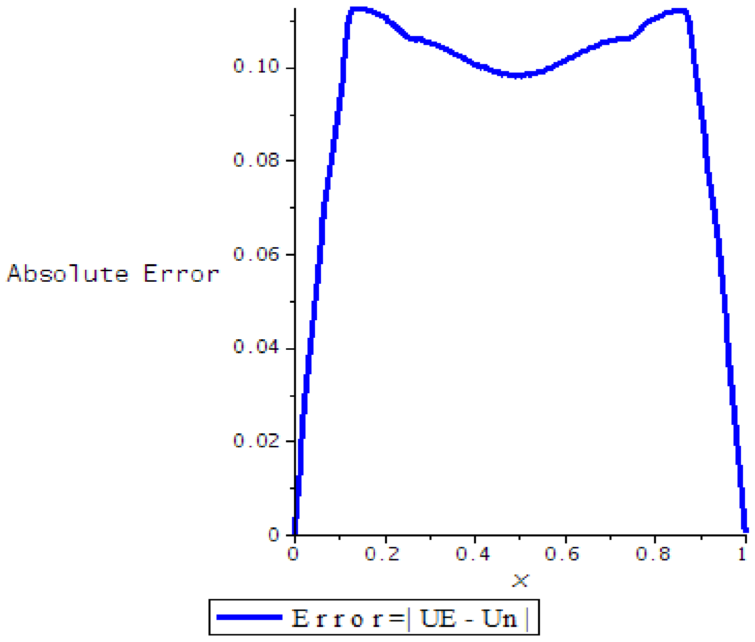

Figure 4.6 : The error resulting of applying algorithm 4.1 on equation (4.2).

4.4 The numerical realization of equation 4.2 using the Homotopy perturbation method

Table (4.4) shows the exact and numerical solutions when applying algorithm 4.2 on equation (4.2) when $n = 8$, and showing the error resulting of using the numerical solution.

Table (4.4): The exact and numerical solution of applying Algorithm 4.2 on equation (4.2).

x	Exact solution $u(x)=\cos(2\pi x)$	Approximate solution $u_n(x)$	Error $=\lvert u-u_n\rvert$
0	1	1	0
0.1	0.8090169943	0.8090169943	0
0.2	0.3090169938	0.3090169938	0
0.3	0.3090169938-	0.3090169942-	4×10^{-10}
0.4	0.8090169943-	0.8090169945-	2×10^{-10}
0.5	-1	-1	0
0.6	0.8090169943-	0.8090169940-	3×10^{-10}
0.7	0.3090169938-	0.3090169934-	4×10^{-10}
0.8	0.3090169938	0.3090169946	8×10^{-10}
0.9	0.8090169943	0.8090169947	4×10^{-10}
1	1	1	0

These results show the accuracy of the Homotopy perturbation method to solve equation (4.2) since the max error = 8×10^{-10}.

Figure 4.7: shows both exact and the numerical solutions with $n=8$

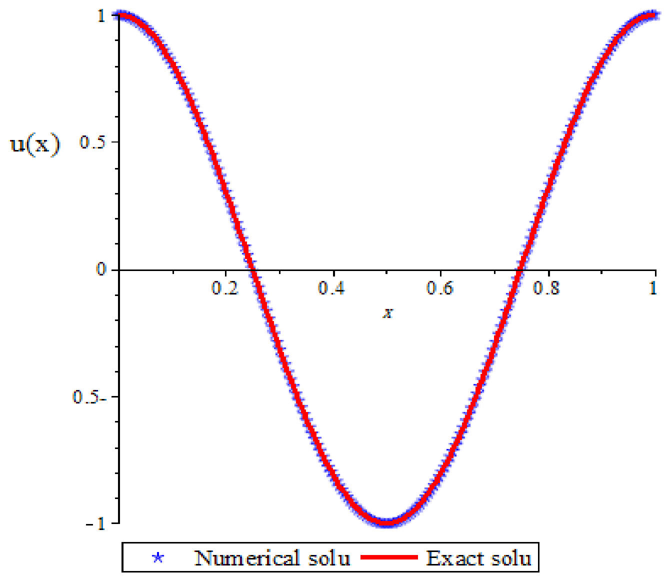

Figure 4.7: The exact and numerical solutions of applying algorithm 4.2 on equation (4.2).

Figure 4.8 shows the absolute error resulting of applying algorithm 4.2 on equation (4.2)

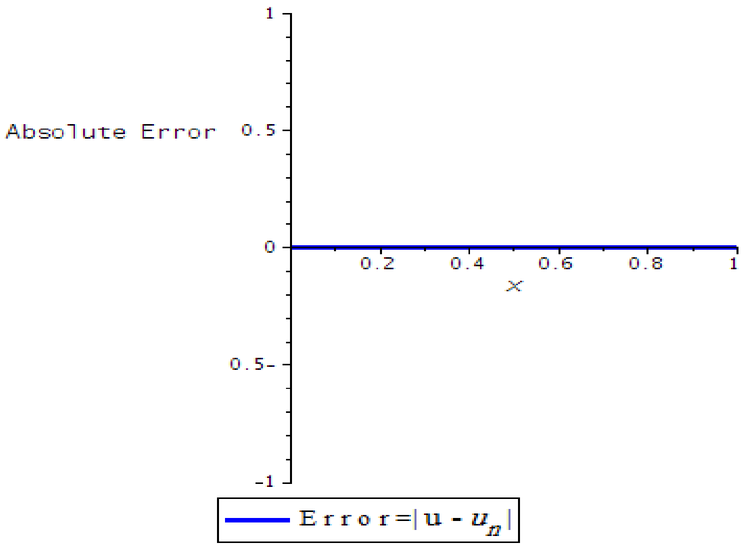

Figure 4.8 : The error resulting of applying algorithm 4.2 on equation (4.2).

Example 4.3

Consider the Fredholm integro-differential equation

$$y''+xy'-xy=e^{x}-2\sin(x)+\int_{-1}^{1}\sin(x)e^{-t}y(t)dt\ ,y(0)=1\ ,y'(0)=1$$

(4.3)

Equation (4.3) has the exact solution $y(x)=\exp(x)$.

We will find an approximate solution to equation (4.3) by the Legendre polynomial method, the Taylor collocation method.

4.5 The numerical realization of equation 4.3 using the Legendre polytnomial method

The following algorithm implements the **Legendre polynomial method** using the **Matlab** software

Algorithm 4.3 [42]

```
1. Input $N, a, b, m, P_0(x), P_1(x), P_2(x), f(x), G(x,y)$

2. Let $x_i = zeros(1,N+1)$

  for $i=0:N$

    $x_i(1,i+1)=a+i*((b-a)/N)$

  end

3. Let $F=zeros(N+1,N+1,N)$

      for $k=1:length(f)$

           for $i=0:N$
```

```
            $F(i+1, i+1, k) = subs(f(k), \{x\}, \{x_i(i+1)\})$
        end
    end
4.  $Pk = zeros(N)$
    for $j = 0:N$
        for $i = 0:N$
            $P = Legendrep(i, 'X')$
            $X = x_i(j+1)$
            $Pk(j+1, i+1) = evalf(P)$
        end
    end
5. Define $PI = LegendrePi(N+1)$
6. Define $K_l = LegendreK_t(N, K(x,t))$
7. Calculate $Q = diag(2/(2*n+1))$
8. Calculate $G = subs(g(x), x, x_i)$
9. Let $WD = zeros(N+1, N+1), N-1)$
    for $k = 0:N-1$
```

```
        $WD(:,:,k+1) = F(:,:,k+1) * Pk * ((PI')^k)$
    end
10. Let $WX = zeros(N+1)$
    for $k = 0:N-1$
        $WD(:,:) = WX(:,:) + WD(:,:,k+1)$
    end
$W = WX - \lambda * Pk * K_l * Q$
11. Let $U_0 = zeros(1,N)$
    for $i = 0:N$
        $P = Legendre(i,'X')$
        $X = 0$
        $U_0(1,i+1) = evalf(P)$
    end
12. Let $U_1 = zeros(1,N)$
    $U(1,1) = 0$
    for $i = 0:N$
        $P = Legendrep(i,'X')$
        $P = evalf\ (P)$
        $P = diff\ (P)$
        $U_1(1, i+1) = subs(P, x, 0)$
    end
13. Put $WU = [W(1:N-1,:);U_0,U_1]$
14. $GU = [G(1:N-1),1,1]$
15. $A = WU/GU$
16. $Y = '0'$
    for $i = 0:N$
        $aux = ['('num2str(A(i+1))$
        $Legendrep(i,'x')')']$
        $Y = [Y\ '+'aux]$
    end
$Y = evalf\ (Y)$
```

Table 4.5 shows the exact and numerical solutions when applying algorithm 4.3 on equation (4.3), and showing the error resulting of using the numerical solution.

Table 4.5: The exact and numerical solution of applying Algorithm 4.3 on equation (4.3).

x_i	Exact solution $y(x)=\exp(x)$	Approximate solution $y_N(x)$	Absolute Error = $\lvert y-y_N\rvert$
-1	0.36787944	0.36787945	1×10^{-8}
-0.8	0.44932896	0.44932895	1×10^{-8}
-0.6	0.54881163	0.54881163	0
-0.4	0.67032004	0.67032005	1×10^{-8}
-0.2	0.81873075	0.81873075	0
0	1	1	0
0.2	1.22140275	1.22140276	1×10^{-8}
0.4	1.49182469	1.49182470	0
0.6	1.82211880	1.82211882	2×10^{-8}
0.8	2.22554092	2.22554046	4.6×10^{-7}
1	2.71828182	2.71827657	5.25×10^{-6}

These results show the accuracy of the Legendre polynomial method to solve equation (4.3) with the max error $=5.25\times10^{-6}$.

Figure 4.9 compares the exact solution $y(x)=\exp(x)$ with the approximate solution

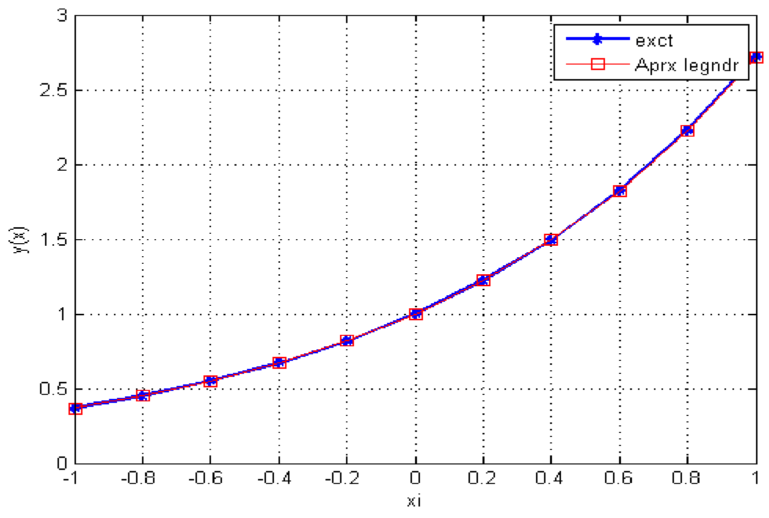

Figure 4.9: The exact and numerical solutions of applying algorithm 4.3 on equation (4.3).

Figure 4.10 shows the absolute error resulting of applying algorithm 4.3 on equation (4.3)

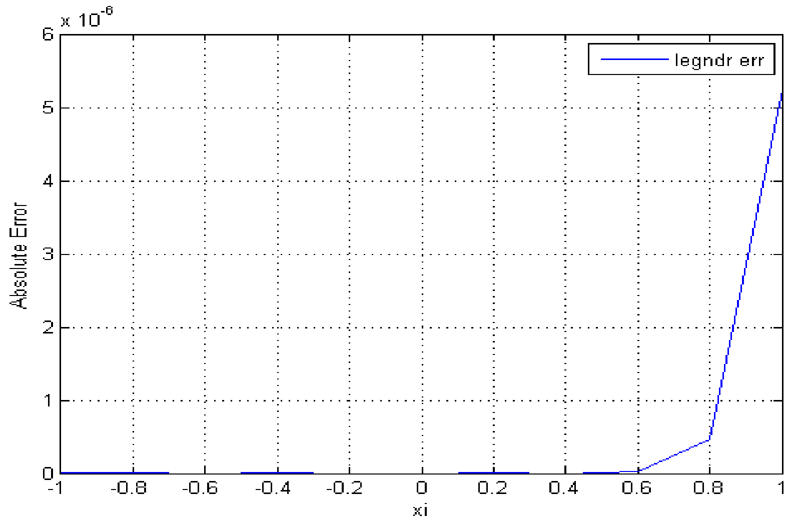

Figure 4.10: The error resulting of applying algorithm 4.3 on equation (4.3).

4.6 The numerical realization of equation 4.3 using the Taylor collocation method

The following algorithm implements the **Taylor collocation method** using the **Matlab** software

Algorithm 4.4

```
1. Input $a, b, c, K(x,t), f(x), P_0, P_1, P_2$

2. Let $x_i = zeros(1, N+1)$

   for $i = 0:N$

       $x_i(1, i+1) = a + i*((b-a)/N)$

   end

3. Let $H = zeros(N+1)$

   for $n = 0:N$

       for $m = 0:N$

           $expnt = n + m + 1$

           $H(n+1, m+1) = ((b-c)^\wedge expnt - (a-c)^\wedge expnt)/expnt$

       end

   end
```

```
4. Let $C = zeros(N+1)$
   for $n = 0:N$
       for $m = 0:N$
           $C(n+1, m+1) = (x_i(1, n+10)\wedge m$
       end
   end
5. Let $F = zeros(N+1, 1)$
   for $i = 0:N$
       $F(i = 1,1) = subs(FX, \{x\}, \{x_i(i+1)\})$
   end
6. Let $K_l = zeros(N+1)$
   for $n = 0:N$
       for $m = 0:N$
           for $td = 0:m$
               if $td > 0$
                   $PK = diff(K(x,t), t)$
                   $K(x,t) = Pk$
```

```
            end
        end
        for $xd = 0:n$
            if $xd > 0$
                $PK = diff(K(x,t),x)$
                $K(x,t) = Pk$
            end
        end
7. Calculate $K_l(n+1,m+1) = (1/(factorial(n)*factorial(m))$
                                   $*subs(K(x,t),\{x,t\},\{0,0\})$
8. Input the values of the matrix M
9. Input $P_0 = -x$ , $P_1 = x$ , $P_2 = 1$, $P_S = [1,x,-x]$
   $P = zeros(N+1,N+1,K)$
   for $k = 0:K$
       for $n = 0:N$
           for $m = 0:N$
               if $n = m$
```

```
    $P(n+1, m+1, (k=1)) = subs(P_S(k+1), \{x\}, \{x_i(m+1)\})$

            end

         end

       end

   end

10. Let $YS = zeros(N+1,1)$

    $YS(1) = 1$

    $YS(2) = 1$

11. $FU = [F; YS]$

12. Let $U = zeros(N+1)$

    for $n = 0:N$

       for $m = 0:N$

          if $n = m$ & $(YS(m+1) = 1))$

          end

       end

     end

13. Let $WD = zeros(N+1, N+1, K)$
```

```
    for $k = 0:K$
        $WD(:,:,k+1) = P(:,:,k+1)*C*M(:,:,k+1)$
    end
14. Let $WX = zeros(N+1)$
    for $k = 0:K$
        $WD(:,:) = WX(:,:) + WD(:,:,k+1)$
    end
15. Calculate $W = WX - \lambda * C * K_l * H * M(:,:,1)$
    $WU = [W;U]$
16. $A = WU / FU$
17. Put $Y = 0$
    for $n = 0:N$
        $Y = Y + (A(n+1)*/factorial(n))*(x-c)^\wedge n$
    end
```

Table 4.6 shows the exact and numerical results , and showing the error resulting of using the numerical solution.

Table 4.6: The exact and numerical solution of applying Algorithm 4.4 on equation (4.3).

x_i	Exact solution $y(x)=\exp(x)$	Approximate solution $y_N(x)$	Error $=\lvert y-y_N\rvert$
-1	0.367879	0.368019	1.4×10^{-4}
-0.8	0.449328	0.449403	7.5×10^{-5}
-0.6	0.548811	0.548854	4.3×10^{-5}
-0.4	0.670320	0.670347	2.7×10^{-5}
-0.2	0.818730	0.818741	1.1×10^{-5}
0	1	1	0
0.2	1.221402	1.221415	1.3×10^{-5}
0.4	1.491824	1.491827	3.0×10^{-6}
0.6	1.822118	1.821847	2.7×10^{-4}
0.8	2.225540	2.224079	1.5×10^{-3}
1	2.718281	2.713341	4.9×10^{-3}

These results show the accuracy of the Taylor collocation method to solve equation (4.3) with a maximum error $=4.9\times10^{-3}$.

Figure 4.11 compares the exact solution $y(x) = \exp(x)$ with the approximate solution.

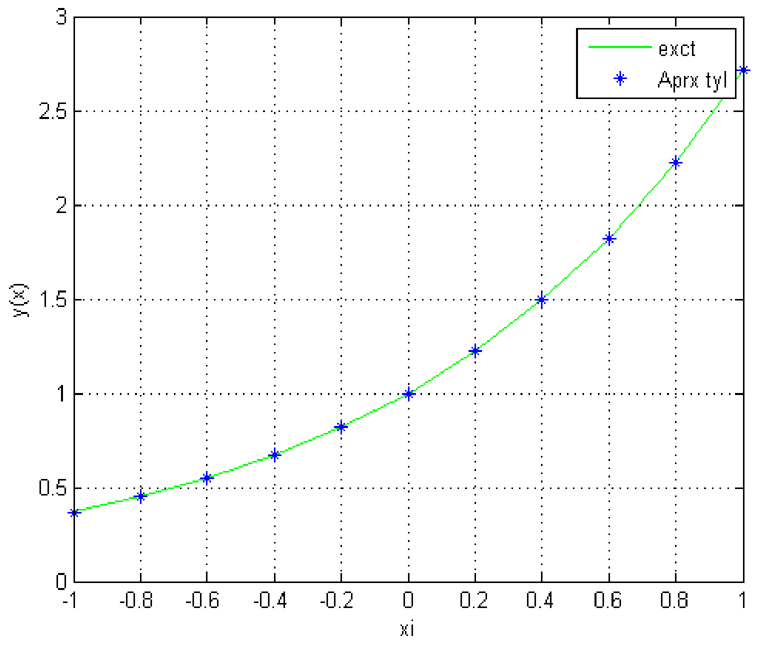

Figure 4.11: The exact and numerical solutions of applying algorithm 4.4 on equation (4.3).

Figure 4.12 shows the absolute error resulting of applying algorithm 4.4 on equation (4.3)

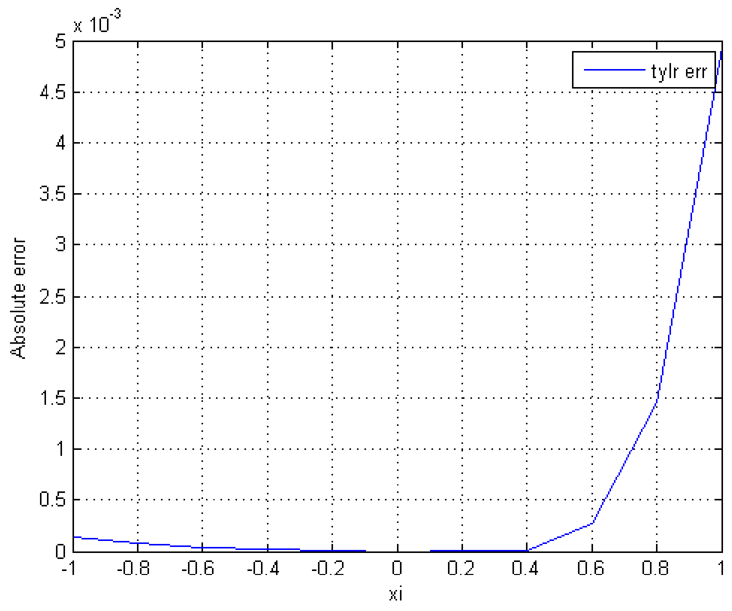

Figure 4.12: The error resulting of applying algorithm 4.4 on equation (4.3).

Conclusions

Analytical and numerical methods have been used to solve linear Fredholm integro-differential equation of the second kind. The numerical methods are implemented in a form of algorithms to solve some numeri- cal examples using Maple and Matlab software.

The numerical results show the following observations:

(1) Numerical results for examples 4.1 and 4.2 show clearly that the Homotopy Perturbation Method (HPM) is very efficient in comparison with the B-spline scaling functions and wavelets method. This is because the HPM is very well known for it's fast convergence and consequently requires less CPU time and therefore less error. Moreover, the HPM introduces less complexity and can easily be implemented.

(2) Example 4.2 has been solved numerically by both the Legendre polynomial method and the Taylor collocation method. In fact one of the most important feature of the Legendre polynomial method is that the error introduced is considerably small in comparison with the Taylor collocation method.

References

[1] G. Adomian, **Nonlinear Stochastic Operator Equations**, Academic Press, SanDiego, (1986).

[2] G. Adomian, **Solving Froutier Problems of Physics, The Decomposition Method**, Kluwer, Boston, (1994).

[3] G. Adomian and R. Rach, **Noise terms in Decomposition Series Solution**, Comput. Math. Appl. 24 (1992), 61-64.

[4] A. Akyuz and M. Sezer, **A Chebyshev Collocation Method for the Solution of Linear Integro-Differential Equations**, Int. J. Comput Math. Vol. 27, No. 4, (1999).

[5] K. Atkinson, **The Numerical Solution of Integral Equations of the Second Kind**, The Press Syndicate of the University of Cambridge, United Kingdom, (1997).

[6] A. Ayad, **Spline Approximation for First Order Fredholm Integro-Differential Equations**, Universitatis Babies-Bolyai. Studia. Mathmatica 41 (1996), No.3, 1-8.

[7] S.H. Behiry and H. Hashish, ***Wavelet Methods for the Numerical Solution of Fredholm Integro-Differential Equations***, **International Journal of Applied Math**. 11(2002), No.1,27-35.

[8] J. Biazar and M. Pourabd, **A Maple Program for Solving Systems of Linear and Nonlinear Integral Equations by Adomian Decompo-**

sition method, Int. J. Contemp Math. Science, Vol. 2, (2007), No.29, 1425-1432.

[9] C.K. Chui, **An Introduction to Wavelets, Wavelet Analysis and Its Applications**, Vol.1, Academic Press, Massachusetts, (1992).

[10] H.T. Davis, **Introduction to Nonlinear Differential and Integral Equations**, Dover, New York (1962).

[11] W.N. Everitt, L.L. Litlejohn and R.Wellman, **Legendre Polynomials, Legendre- Stirling Numbers and the Left-definite Spectral Analysis of the Legendre Differential Expression**, J. Comput, App. Math. 148(2002).

[12] L. Gabet, **The Theoretical Foundation of the Adomian Method**, Computers Math. Applic. Vol.27, No. 12, (1994), 41-52.
[13] M. Gorji, D. Gangi, S. Soleiman, **New Application of He's Homotopy Perturbation**, Int. J. Nonlinear Sci Numer Simulate (2007),19-28.

[14] J.C. Goswami, A.K. Chan and C.K. Chui, **On Solving First Kind Integral Equations Using Wavelets on a Bounded Interval**, IEEE Transactions on Antennas and Propagation 43(1995), no.6, 614-622.

[15] J.H. He, **A Symmptotology of Homotopy Perturbation Method**, Appl. Math. Comput. 156(2004) 591-596. [16]
J.H. He, **Homotopy perturbation technique**, Comput, Methods Appl. Mech. Engrg., 178 (1999) 257-262.

[17] J.H. He, **Variational Iteration Method for Autonomous Ordinary Differential System**, Appl. Math. Comput., 114 (2/3) 2000), 115-123.

[18] A. Hossein, ***A New Analytical Method for Solving Systems of Linear Integro-Differential Equations***, **Journal of King Saud University- Science** (2011), 23, 349-353.

[19] S. M. Hosseini and S. Shamord**, Numerical Piecewise Approximate Solution of Fredholm Integro-Differential Equations by the Tau Method**, App. Math. Model 29(2005)1005-1021.

[20] A. J. Jerri, **Introduction to Integral Equations with Applications**, John Wiley and Sow, INC (1999).

[21] A. Kadem and A. Kilic man , **The Approximate Solution of Fract-ional Fredholm Integro-Differential Equations by Variational Iteration and Homotopy Perturbation Methods**, " Abstract and Applied Analysis", Article ID 486193, (2012).

[22] R.P. Kanwal, **Linear Integral Equations**, Birkhanser, Boston, (1997).

[23] A. Karamete and M. Sezer, **A Taylor Collocation Method for the Solution of Linear Integro-Differential Equations**, International Journal of Computer Mathmatics, Vol. 79(9), pp.987-1000, (2002).

[24] M. Khanian and A. Davari, ***Solution of System of Fredholm Integ-ro-Differential Equations by Adomain Decomposition Method,***

Australian Journal of Basic and Applied Science, Vol. 5, No. 12, (2011), 2356-2361.

[25] M. Lakestani, M. Razzaghi, and M. Dehghan, **Semiorthogonal Spline Wavelets Approximation for Fredholm Integro-differential Equations**, Article ID 961848, 1-12, (2006).

[26] P. Linz, **A Method for the Approximate Solution of Linear Integro-Differential Equations**. SIAM Journal on Numerical Analysis 11 (1974), No.1, 137-144.

[27] K. Maleknejad and M. Tavassolikajani, **Solving Second Kind Integral Equations by Galerkin Methods with Hybrid Legendre and Block-Pulse Functions,** Appl. Math. Comput. 145(2003),623-629.

[28] H. R. Marzban and M. Razzaghi, **Optimal Control of Linear Delay Systems via Hyprid of Block-Pulse and Legendre Polynomials**, J. Franklin Inst. 341, (2004), 279-293.

[29] G. Micula and G. Fairweather, **Direct Numerical Spline Methods for First Order Fredholm Integro-Differential Equations**, Revue D'Analyse Numerique et de Theorie del' Approximation 22, (1993), No. 1, 59-66.

[30] S. Nas, S. Yalcinbas and M. Sezer, **A Taylor Polynomial Approach for Solving High Order Linear Fredholm Integro-Differential Equations**, Int. F. Math. Educ. Sci. Technol. 31, 491-507.

[31] A. Ralston and P. Rabinowitz, A **First Course in Numerical Analysis**, McGraw-Hill, New York, 1985.

[32] Y. Ren, B. Zhang and H. Qiao, **A Simple Taylor Series Expansion Method for A Class of Second Kind Integral Equations**, J. Comput. Appl. Math. 110, (1999).

[33] J. Saberi and A. Ghorbani**, He's Homotopy Perturbation Method. An Effective Tool for Solving Nonlinear Integral and Integro-Differential Equations**, Computers and Mathematical with Applications 58, (2009), 2379-2390.

[34] C. Schneider, **Regularity of the Solution to A Class of Weakly Singular Fredholm Integral Equations of the Second Kind**, Integral Equations Operator Theory 2 (1979), 62-68.

[35] M. Sezer, **A Method for the Approximate Solution of the Second-Order Linear Differential Equations in Terms of Taylor Polynomials**, Int. F. Math. Educ. Sci. Technol.27 (1996), 821-834.

[36] M. Tavassolikajani, M. Ghasemi and E. Babolian, **Comparison Between the Homotopy Perturbation Method and the Sine-Cosine Wavelet Method for Solving Linear Integro-Differential Equations**, Computers and Mathematics with applications 54 (2007), 1162-1168.

[37] A.M. Wazwaz, **A Reliable Modification of the Adomian Decomposition Method**, Appl. Math. Comput. 102(1999), 77-86.

[38] A.M. Wazwaz, **Partial Differential Equations and Soliatary Waves Theory**, HEP and Springer, Bijing and Berlin, (2009).

[39] A.M. Wazwaz, **Linear and Nonlinear Integral Equations: Methods and Application**, Springer Heidelberg, Dordrechi London, (2011).

[40] A.M. Wazwaz, **A First Course in Integral Equations**, (Second Edition) World Scientific Publishing Company, Singapore and New Jersey (2015).

[41] A.M. Wazwaz, **AFirst Course in Integral Equations**, World Scie-ntific, Singopore, (1997).

[42] S. Yalcinbas and M. Sezer, H. Sorkun, ***Legendre Polynomial Solut-ions of High-Order Linear Fredholm Integro-Differential Equations***, **International Journal of Applied Mathematics and Computation 210** (2009), 334-349.

[43] E. Yusufoglu, **Improved Homotopy Perturbation Method for Solving Fredholm Type Integro-differential Equations**, Chaos, Solitons and Fractals 41 (2009), 28-37.

Appendix

Maple Code for B-Spline Scaling Functions and Wavelets for Example 4.1

```
> restart:

> with(plots):with(LinearAlgebra):

> M:=4:

> 2^(M+1)+1;

> Q:=Matrix(2^(M+1)+1):R:=Matrix(2^(M+1)+1):

> for i from 1 to (2^(M+1)+1) do

> for j from 1 to (2^(M+1)+1) do

> if (i=j+1) then Q(i,j):= 1/24; R(i,j):= 1/2 elif
(j=i+1)then Q(i,j):= 1/24;R(i,j):= -1/2 elif (i=j) then
Q(i,j):= 1/6 end if

> od;

> od:

Q(1,1):=1/12:Q(2^(M+1)+1,2^(M+1)+1):=1/12:R(1,1):=-
1/2:R(2^(M+1)+1,2^(M+1)+1):=1/2:

> Qm:= 1/2^(M-1)*Q:

> for j from 2 to M do

> alpha(j):= Matrix(2^j+1,2^(j+1)+1):

> beta(j):=Matrix(2^j,2^(j+1)+1):

> od:

> for k from 2 to M do

> for i from 1 to 2^k+1 do

> for j from 1 to 2^(k+1)+1 do

> if (j=2*i-1) then alpha(k)(i,j):= 1 elif (j=2*(i-1))then
alpha(k)(i,j):= 1/2 elif (j=2*i) then alpha(k)(i,j):= 1/2
end if

> od:
```

```
> od:

> od:

> for k from 2 to M do

> for i from 2 to 2^k-1 do

> for j from 1 to 2^(k+1)+1 do

> if (j=2*i-1) then beta(k)(i,j):= -1/2 elif (j=2*(i-
1))then beta(k)(i,j):= 1/12 elif (j=2*i) then
beta(k)(i,j):= 5/6 elif (j=2*i+1) then beta(k)(i,j):= -1/2
elif (j=2*(i+1)) then beta(k)(i,j):=1/12 end if

> od:

> od:

> beta(k)(1,1):= -1:beta(k)(1,2):= 11/12:beta(k)(1,3):= -
1/2:beta(k)(1,4):= 1/12:beta(k)(2^k,2^(k+1)+1):= -
1:beta(k)(2^k,2^(k+1)):= 11/12:beta(k)(2^k,2^(k+1)-1):= -
1/2:beta(k)(2^k,2^(k+1)-2):= 1/12:

> od:

> for i from 2 to M do

> q(i):=alpha(i):

> for j from 2 to M do

> if (j+(i-2)=M) then g(i):=q(i); break;  else
g(i):=q(i).alpha(j+(i-2)+1);q(i):=g(i):  end if

> od:

> if (i-1=1) then G1(i-1):= g(i) else G1(i-1):= beta(i-
1).g(i) end if

> od:

> G1(M):=beta(M):

> G(1):=G1(1):

> for i from 1 to M-1 do

> G(i+1):= <G(i),G1(i+1)>:

> od:

> rtable_elems(G(M)):
```

```
> D1:= G(M).R.MatrixInverse(Qm).MatrixInverse(G(M)):

> X:= Vector(2^(M+1)+1):

> for i from 2 to 2^(M+1) do

> X(i):= (2*(i-1)-1)/(2^(M+2)-2):

> od:

> X(1):=0:X(2^(M+1)+1):=1:

> X;

> chi:=(i,j,x)-> piecewise ((j/2^i)<=x and x<=((j+1)/2^i),
2^i*x-j ,((j+1)/2^i)<= x and x<=  ((j+2)/2^i),2 -(2^i*x-
j),0  ):

> for i from 2 to M do

> chi(i,-1,x) :=piecewise(0 <= x and x <= 1/2^i,2 -(2^i*x-
(-1)),0 ):

> chi(i,(2^i)-1,x) :=  piecewise((((2^i)-1)/2^i)<=x and
x<=(((2^i)-1+1)/2^i),2^i*x-((2^i)-1) ,0)

> od:

> mu:= (i,j,x)-> 1/6* piecewise((j/2^i)<=x and x<=((j+
1/2)/2^i),2^i*x-j,((j+ 1/2)/2^i)<=x and x<= ((j+1)/2^i), 4-
7*(2^i*x-j), ((j+1)/2^i) <= x and x <= ((j+3/2)/2^i), -
19+16*(2^i*x-j),((j+3/2)/2^i)<= x and x <= ((j+2)/2^i), 29-
16*(2^i*x-j),((j+2)/2^i)<= x and x <= ((j+5/2)/2^i), -
17+7*(2^i*x-j), ((j+5/2)/2^i)<= x and x <= ((j+3)/2^i), 3-
(2^i*x-j),0 ):

> chi(2,2,9/14);

> for i from 2 to M do

> mu(i,-1,x) :=piecewise(0 <= x and x <= (1/2)/2^i,-6
+23*(2^i*x),((1/2)/2^i) <= x and x <= ((1)/2^i),14 -
17*(2^i*x),((1)/2^i) <= x and x <= ((3/2)/2^i),-10
+7*(2^i*x),((3/2)/2^i) <= x and x <= ((2)/2^i),2 -(2^i*x),0
):

> mu(i,((2^i)-2),x) :=  piecewise((((2^i)-2)/2^i)<=x and
x<=((((2^i)-2)+ 1/2)/2^i),2-(((2^i)-2)+2-2^i*x),((((2^i)-
2)+ 1/2)/2^i)<=x and x<= ((((2^i)-2)+1)/2^i), -
10+7*(((2^i)-2)+2-2^i*x), ((((2^i)-2)+1)/2^i) <= x and x <=
((((2^i)-2)+3/2)/2^i), -14-17*(((2^i)-2)+2-2^i*x),((((2^i)-
2)+3/2)/2^i)<= x and x <= ((((2^i)-2)+2)/2^i), -
6+23*(((2^i)-2)+2-2^i*x),0)
```

```
> od:
> mu(2,-1,1/14);
############################33 Integration
> h:=1/4:
> for i from 0 to 4 do
> t(i):= i*h:
> od:
> t(4);
> mut:=Matrix(5,5):
> Qs:= Vector(2^(M+1)+1):
> for i from 1 to 5 do
> for j from 1 to 5 do
> mut(i,j):= (chi(2,i-2,t(j-1))):
> od:
> mut1:=Matrix(2^(M+1)+1-5,5):
> od:
> for k from 2 to M do
> #for i from 0 to (2^(M)-4) do
> for l from 1 to 2^k do
> for j from 1 to 5 do
> mut1((2^k-4)+l,j):= (mu(k,l-2,t(j-1))):
> od:
> od:
> od:
> #od:
> mutt:=<mut,mut1>:
> for i from 1 to 5 do
> KE(i-1):=3*x*t(i-1)*mutt(1..2^(M+1)+1,i):

> od:

> KE(1):

intg:=(2*h/45)*(7*KE(0)+32*KE(1)+12*KE(2)+32*KE(3)+7*KE(4))
:

> intgr:=Matrix(2^(M+1)+1):

> for i from 1 to 2^(M+1)+1 do

> for j from 1 to 2^(M+1)+1 do

> intgr(i,j):=subs(x=X(j),intg(i)):

> od:

> od:

> intgr:

> muX:=Matrix(5,2^(M+1)+1):

> for i from 1 to 5 do

> for j from 1 to 2^(M+1)+1 do

> muX(i,j):= (chi(2,i-2,X(j))):

> od:

> od:

> muX:

 > muX1:=Matrix(2^(M+1)+1-5,2^(M+1)+1):

> for k from 2 to M do

> for l from 1 to 2^k do

> for j from 1 to 2^(M+1)+1 do

> muX1((2^k-4)+l,j):= (mu(k,l-2,X(j))):

> od:

> od:

> od:

> muX1:
```

```
> muXX:=<muX,muX1>:

> ##########################################3 The solution

> Z:=D1.muXX:

> ZZ:=Z-intgr:

> F:=Matrix(1,2^(M+1)+1): C:=Matrix(1,2^(M+1)+1):

> for i from 1 to 2^(M+1)+1 do

> F(1,i):= evalf(3*exp(3*X(i))-(1/3)*(2*exp(3)+1)*X(i));

> od:

> ZZ(1..2^(M+1)+1,1):=muXX(1..2^(M+1)+1,1):

> F(1,1):=1:

> C:=F.MatrixInverse(ZZ):

> U:=(C.muXX):

> u2:=x->evalf(exp(3*x)):

> u:=Transpose(U):

> u1:=convert(u,Vector):

> r:=plot( X,u1,color="Blue",style = point, symbol =
asterisk, symbolsize =15,legend = ["Numerical solu" ]):

r2:=plot(u2(x),x=0..1,color="Red",thickness=3,legend =
["Exact solu" ]):

> display(r,r2);

> ######## error

> Ue:=Vector(2^(M+1)+1):

> for i from 1 to 2^(M+1)+1 do

> Ue(i):=(u2(X(i)));

> od:

> Erroru:=Vector(2^(M+1)+1):

> Erroru:=abs(Ue-u1):
```

```
> A2:=plot( X, Erroru,x =
0..1,color="Blue",thickness=3,legend = ["E r r o r =| UE -
Un |" ]

> ):

> display(A2,titlefont = ["COURIER", 15],labelfont =
["COURIER", 10],axesfont = ["COURIER",  8]);

> X1:= Vector(11):

> for i from 1 to 11 do

> X1(i):= (i-1)*1/10:

> od:

> evalf(X1(11));

> muX2:=Matrix(5,11):

> for i from 1 to 5 do

> for j from 1 to 11 do

> muX2(i,j):= (chi(2,i-2,X1(j))):

> od:

> od:

> muX2:

> muX3:=Matrix(2^(M+1)+1-5,11):

> for k from 2 to M do

> for l from 1 to 2^k do

> for j from 1 to 11 do

> muX3((2^k-4)+l,j):= (mu(k,l-2,X1(j))):

> od:

> od:

> od:

> muX3:

> muXX2:=<muX2,muX3>:

> U2:=C.muXX2:
```

```
> rtable_elems(U2):

> NUM:= Matrix (10,1):Exact:= Matrix (10,1):Error:= Matrix
(10,1):

> for i from 1 to 10 do

> NUM(i,1):= U2(1,i+1):

> Exact(i,1):=u2(X1(i+1)):

> Error(i,1):=abs(Exact(i,1)-NUM(i,1))

> od:

evalf(X1(2..11));exact;Exact;Nume;(NUM);errorr;Error;
```

Maple Code for Homotopy Perturbation Method for Example 4.1

```
> restart;

> with(LinearAlgebra):with(plots):with(PDEtools):

> # This is a program to solve the integro-differential
equation on the form:

> Diff(v(x),x)=f(x)+ Int(G(x,y)*v(y),y=0..1);

> a:=1;

> f:=  3*exp(3*x)-(1/3)*(2*exp(3)+1)*x;

> G:= 3*x*y;

> n:=4;

> wm1[0]:= ( int(f,x)):

> wm[0]:=y->subs(x=y,wm1[0]):

> c:=a-eval(wm[0](0)):

 > ws[0]:= wm1[0]+c:

> w[0]:=subs(x=y,ws[0]):

> for i from 1 to n do

> wm[i]:= int(G*w[i-1],y=0..1);

> ws[i]:=(int(wm[i],x));

> w[i]:=subs(x=y,ws[i]);
> od:

> u:=combine(add(ws[k],k=0..n));

> um:= y-> subs(x=y,u):

> uexact:= x-> exp(3*x):

> r:=plot( um(x),x=0..1,color="Blue",style = point, symbol
= asterisk, symbolsize =15,legend = ["Numerical solu" ]):

r2:=plot(uexact(x),x=0..1,color="Red",thickness=3,legend =
["Exact solu" ]):

> display(r,r2);

> Erroru:=x->abs(uexact(x)-um(x)):

> Erroru(1);

> A2:=plot(  Erroru(x),x =
0..1,color="Blue",thickness=3,legend = ["E r r o r =| UE -
Un |" ]):

> display(A2,titlefont = ["COURIER", 15],labelfont =
["COURIER", 10],axesfont = ["COURIER",  8]);

> X1:= Vector(10):NUM:= Matrix (10,1):Exact:= Matrix
(10,1):Error:= Matrix (10,1):

> for i from 1 to 10 do

> X1(i):= i*1/10:

> od:

 > for i from 1 to 10 do

> NUM(i,1):= evalf(um(X1(i))):

> Exact(i,1):=evalf(uexact(X1(i))):

> Error(i,1):=evalf(abs(Exact(i,1)-NUM(i,1)))

> od:

evalf(X1(2..11));exact;Exact;Nume;(NUM);errorr;Error;
```

Maple Code for B-spline scaling functions and wavelets method for example 4.2

```
> restart:

> with(plots):with(LinearAlgebra):
```

```
> M:=6:

> 2^(M+1)+1;

> ########################3 Q & R

> Q:=Matrix(2^(M+1)+1):R:=Matrix(2^(M+1)+1):

> for i from 1 to (2^(M+1)+1) do

> for j from 1 to (2^(M+1)+1) do

> if (i=j+1) then Q(i,j):= 1/24; R(i,j):= 1/2 elif (j=i+1)then Q(i,j):= 1/24;R(i,j):= -1/2 elif
(i=j) then Q(i,j):= 1/6 end if

> od;

> od:

> Q(1,1):=1/12:Q(2^(M+1)+1,2^(M+1)+1):=1/12:R(1,1):=-
1/2:R(2^(M+1)+1,2^(M+1)+1):=1/2:

> Qm:= 1/2^(M-1)*Q:

> for j from 2 to M do

> alpha(j):= Matrix(2^j+1,2^(j+1)+1):

> beta(j):=Matrix(2^j,2^(j+1)+1):

> od:

> for k from 2 to M do

> for i from 1 to 2^k+1 do

> for j from 1 to 2^(k+1)+1 do

> if (j=2*i-1) then alpha(k)(i,j):= 1 elif (j=2*(i-1))then alpha(k)(i,j):= 1/2 elif (j=2*i)
then alpha(k)(i,j):= 1/2 end if

> od:

> od:

> od:

> for k from 2 to M do

> for i from 2 to 2^k-1 do

> for j from 1 to 2^(k+1)+1 do
> if (j=2*i-1) then beta(k)(i,j):= -1/2 elif (j=2*(i-1))then beta(k)(i,j):= 1/12 elif (j=2*i)
then beta(k)(i,j):= 5/6 elif (j=2*i+1) then beta(k)(i,j):= -1/2 elif (j=2*(i+1)) then
beta(k)(i,j):=1/12 end if

> od:

> od:

> beta(k)(1,1):= -1:beta(k)(1,2):= 11/12:beta(k)(1,3):= -1/2:beta(k)(1,4):=
1/12:beta(k)(2^k,2^(k+1)+1):= -1:beta(k)(2^k,2^(k+1)):= 11/12:beta(k)(2^k,2^(k+1)-
1):= -1/2:beta(k)(2^k,2^(k+1)-2):= 1/12:

> od:

> for i from 2 to M do

> q(i):=alpha(i):

> for j from 2 to M do

> if (j+(i-2)=M) then g(i):=q(i); break;  else g(i):=q(i).alpha(j+(i-2)+1);q(i):=g(i):  end if

> od:

> if (i-1=1) then G1(i-1):= g(i) else G1(i-1):= beta(i-1).g(i) end if

> od:

> G1(M):=beta(M):

> G(1):=G1(1):

> for i from 1 to M-1 do

> G(i+1):= <G(i),G1(i+1)>:

> od:

> rtable_elems(G(M)):

> ###########################3  D

> D1:= G(M).R.MatrixInverse(Qm).MatrixInverse(G(M)):

> X:= Vector(2^(M+1)+1):

> for i from 2 to 2^(M+1) do

> X(i):= (2*(i-1)-1)/(2^(M+2)-2):

> od:

> X(1):=0:X(2^(M+1)+1):=1:
```

```
> X;

> chi:=(i,j,x)-> piecewise ((j/2^i)<=x and x<=((j+1)/2^i), 2^i*x-j ,((j+1)/2^i)<= x and
x<=  ((j+2)/2^i),2 -(2^i*x-j),0  ):

> for i from 2 to M do

> chi(i,-1,x) :=piecewise(0 <= x and x <= 1/2^i,2 -(2^i*x-(-1)),0 ):

> chi(i,(2^i)-1,x) :=  piecewise((((2^i)-1)/2^i)<=x and x<=(((2^i)-1+1)/2^i),2^i*x-
((2^i)-1) ,0)

> od:

> mu:= (i,j,x)-> 1/6* piecewise((j/2^i)<=x and x<=((j+ 1/2)/2^i),2^i*x-j,((j+
1/2)/2^i)<=x and x<= ((j+1)/2^i), 4-7*(2^i*x-j), ((j+1)/2^i) <= x and x <= ((j+3/2)/2^i),
-19+16*(2^i*x-j),((j+3/2)/2^i)<= x and x <= ((j+2)/2^i), 29-16*(2^i*x-j),((j+2)/2^i)<=
x and x <= ((j+5/2)/2^i), -17+7*(2^i*x-j), ((j+5/2)/2^i)<= x and x <= ((j+3)/2^i), 3-
(2^i*x-j),0 ):

> chi(2,2,9/14);

> for i from 2 to M do

> mu(i,-1,x) :=piecewise(0 <= x and x <= (1/2)/2^i,-6 +23*(2^i*x),((1/2)/2^i) <= x and
x <= ((1)/2^i),14 -17*(2^i*x),((1)/2^i) <= x and x <= ((3/2)/2^i),-10
+7*(2^i*x),((3/2)/2^i) <= x and x <= ((2)/2^i),2 -(2^i*x),0 ):

> mu(i,((2^i)-2),x) :=  piecewise((((2^i)-2)/2^i)<=x and x<=((((2^i)-2)+ 1/2)/2^i),2-
(((2^i)-2)+2-2^i*x),((((2^i)-2)+ 1/2)/2^i)<=x and x<= ((((2^i)-2)+1)/2^i), -10+7*(((2^i)-
2)+2-2^i*x), ((((2^i)-2)+1)/2^i) <= x and x <= ((((2^i)-2)+3/2)/2^i), -14-17*(((2^i)-
2)+2-2^i*x),((((2^i)-2)+3/2)/2^i)<= x and x <= ((((2^i)-2)+2)/2^i), -6+23*(((2^i)-2)+2-
2^i*x),0)

> od:

> mu(2,-1,1/14);

> ############################33 Integration

> h:=1/8:

> for i from 0 to 8 do

> t(i):= i*h:

> od:

> t(8);

> mut:=Matrix(5,9):

> Qs:= Vector(2^(M+1)+1):
```

```
> for i from 1 to 5 do

> for j from 1 to 9 do

> mut(i,j):= (chi(2,i-2,t(j-1))):

> od:

> mut1:=Matrix(2^(M+1)+1-5,9):

> od:

> for k from 2 to M do

> #for i from 0 to (2^(M)-4) do

> for l from 1 to 2^k do

> for j from 1 to 9 do

> mut1((2^k-4)+l,j):= (mu(k,l-2,t(j-1))):

> od:

> od:

> od:

> #od:

> mutt:=<mut,mut1>:

> for i from 1 to 9 do

> KE(i-1):=evalf(2*Pi*sin(4*Pi*x+2*Pi*t(i-1))*mutt(1..2^(M+1)+1,i)):

> od:

> KE(1):

> intg:=(4*h/14175)*(989*KE(0)+5888*KE(1)-928*KE(2)+10496*KE(3)-
4540*KE(4)+10469*KE(5)-928*KE(6)+5888*KE(7)+989*KE(8)):

> intgr:=Matrix(2^(M+1)+1):

> for i from 1 to 2^(M+1)+1 do

> for j from 1 to 2^(M+1)+1 do

> intgr(i,j):=subs(x=X(j),intg(i)):

> od:
```

```
> od:
> intgr:
> muX:=Matrix(5,2^(M+1)+1):
> for i from 1 to 5 do
> for j from 1 to 2^(M+1)+1 do
> muX(i,j):= (chi(2,i-2,X(j))):
> od:
> od:
> muX:
> muX1:=Matrix(2^(M+1)+1-5,2^(M+1)+1):
> for k from 2 to M do
> for l from 1 to 2^k do
> for j from 1 to 2^(M+1)+1 do
> muX1((2^k-4)+l,j):= (mu(k,l-2,X(j))):
> od:
> od:
> od:
> muX1:
> muXX:=<muX,muX1>:
> ##########################################3 The solution
> Z:=D1.muXX:
> ZZ:=Z-intgr:
> F:=Matrix(1,2^(M+1)+1): C:=Matrix(1,2^(M+1)+1):
> for i from 1 to 2^(M+1)+1 do
> F(1,i):= evalf(-2*Pi*sin(2*Pi*X(i))-Pi*sin(4*Pi*X(i)));
> od:
> ZZ(1..2^(M+1)+1,1):=muXX(1..2^(M+1)+1,1):
```

```
> F(1,1):=1:

> C:=F.MatrixInverse(ZZ):

> U:=(C.muXX):

> u2:=x->evalf(cos(2*Pi*x)):

> u:=Transpose(U):

> u1:=convert(u,Vector):

> r:=plot( X,u1,color="Blue",style = point, symbol = asterisk, symbolsize =15,legend =
["Numerical solu" ]):

> r2:=plot(u2(x),x=0..1,color="Red",thickness=3,legend = ["Exact solu" ]):

> display(r,r2);

> ######## error

> Ue:=Vector(2^(M+1)+1):

> for i from 1 to 2^(M+1)+1 do

> Ue(i):=(u2(X(i)));

> od:

> Erroru:=Vector(2^(M+1)+1):

> Erroru:=abs(Ue-u1):

> A2:=plot( X, Erroru,x = 0..1,color="Blue",thickness=3,legend = ["E r r o r =| UE - Un
|" ]

> ):

> display(A2,titlefont = ["COURIER", 15],labelfont = ["COURIER", 10],axesfont =
["COURIER",  8]);

> X1:= Vector(11):

> for i from 1 to 11 do

> X1(i):= (i-1)*1/10:

> od:

> evalf(X1(11));

> muX2:=Matrix(5,11):
```

```
> for i from 1 to 5 do

> for j from 1 to 11 do

> muX2(i,j):= (chi(2,i-2,X1(j))):

> od:

> od:

> muX2:

> muX3:=Matrix(2^(M+1)+1-5,11):

> for k from 2 to M do

> for l from 1 to 2^k do

> for j from 1 to 11 do

> muX3((2^k-4)+l,j):= (mu(k,l-2,X1(j))):

> od:

> od:

> od:

> muX3:

> muXX2:=<muX2,muX3>:

> U2:=C.muXX2:

> rtable_elems(U2):

> NUM:= Matrix (10,1):Exact:= Matrix (10,1):Error:= Matrix (10,1):

> for i from 1 to 10 do

> NUM(i,1):= U2(1,i+1):

> Exact(i,1):=u2(X1(i+1)):

> Error(i,1):=abs(Exact(i,1)-NUM(i,1))>

> od:

> evalf(X1(2..11));exact;Exact;Nume;(NUM);errorr;Error;
```

Maple Code for Homotopy Perturbation Method for Example 4.2

```
> restart;
```

```
with(LinearAlgebra):with(plots):with(PDEtools):

> # This is a program to solve the integro-differential
equation on the form:

> Diff(v(x),x)=f(x)+ Int(G(x,y)*v(y),y=0..1);

> a:=1;

> f:=  -2*Pi*sin(2*Pi*x)-Pi*sin(4*Pi*x);

> G:= 2*Pi*sin(4*Pi*x+2*Pi*y);

> n:=8;

> wm1[0]:= ( int(f,x)):

> wm[0]:=y->subs(x=y,wm1[0]):

> c:=a-eval(wm[0](0)):

> ws[0]:= wm1[0]+c:

> w[0]:=subs(x=y,ws[0]):

> for i from 1 to n do

> wm[i]:= int(G*w[i-1],y=0..1);

> ws[i]:=(int(wm[i],x));

> w[i]:=subs(x=y,ws[i]);

> od:

> u:=combine(add(ws[k],k=0..n));

> um:= y-> subs(x=y,u):

> uexact:= x-> cos(2*Pi*x):

> r:=plot( um(x),x=0..1,color="Blue",style = point, symbol
= asterisk, symbolsize =15,legend = ["Numerical solu" ]):

r2:=plot(uexact(x),x=0..1,color="Red",thickness=3,legend =
["Exact solu" ]):

> display(r,r2);

> Erroru:=x->abs(uexact(x)-um(x)):

> A2:=plot(  Erroru(x),x =
0..1,color="Blue",thickness=5,legend = ["E r r o r =| u-uM
|" ]):
 > display(A2,titlefont = ["COURIER", 15],labelfont =
["COURIER", 10],axesfont = ["COURIER",  8]);

> X1:= Vector(11):NUM:= Matrix (10,1):Exact:= Matrix
(10,1):Error:= Matrix (10,1):

> for i from 1 to 11 do

> X1(i):= (i-1)*1/1:

> od:

 > for i from 1 to 10 do

> NUM(i,1):= evalf(um(X1(i))):

> Exact(i,1):=evalf(uexact(X1(i))):

> Error(i,1):=evalf(abs(Exact(i,1)-NUM(i,1)))

> od:

evalf(X1(2..11));exact;Exact;Nume;(NUM);errorr;Error;
```

Matlab Code for Legendre Polynomial Method

```
 %%

clc

clear

close

 %%

syms x t;

 a = -1;

b = 1;

N = 8;

Lambda = 1;

 f=[-x x 1];%  F0 F1 F2

 gx= exp(x)- 2*sin(x);

 kxt= sin(x)*exp(-t);

 %% xi
 xi=zeros(1,N+1);
for i=0:N
    xi(1,i+1) = a+i*((b-a)/N);
end
%% F
 F=zeros(N+1,N+1,N);
for k=1:length(f)
    for i=0:N
        F(i+1,i+1,k) = subs(f(k),{x},{xi(i+1)});
    end
end
%% P
 Pk=zeros(N);
for j = 0:N
    for i = 0:N
        P=legendrep(i, 'X');
        X=xi(j+1);
        Pk(j+1,i+1)=eval(P);
    end
end
%% PI
 PI = legendrePi(N+1);
%% KXT
 Kl = legendreKT(N,kxt);
%% Q
n=0:N;
Q = diag(2./(2*n+1));
```

```
%% G
 G = subs(gx,x,xi);
G=double(G);
%% W
WD=zeros(N+1,N+1,N-1);
for k=0:N-1
    WD(:,:,k+1)=F(:,:,k+1)*Pk*((PI')^k);
end
%% W
WX=zeros(N+1);
for k=0:N-1
    WX(:,:)=WX(:,:)+WD(:,:,k+1);
end
 W=WX-Lambda*Pk*Kl*Q;
 %% U
U0=zeros(1,N);
for i = 0:N
    P=legendrep(i, 'X');
    X=0;
    U0(1,i+1)=eval(P);
end
%%
U1=zeros(1,N);
U1(1,1)=0;
for i = 1:N
    P=legendrep(i,'x');
    P=eval(P);
    P=diff(P);

    U1(1,i+1)=subs(P,x,0);

end

%%

WU=[W(1:N-1,:);U0;U1];

GU=[G(1:N-1),1,1];

%%

A=WU\GU';

%%

Y='0';

for i = 0:N

    aux= ['(' num2str(A(i+1)) '*' legendrep(i,'x') ')'];

    Y=[Y '+' aux];

end

 Y=eval(Y);

 %% results

xu=-1:0.2:1;

Aprx_sol_legndr=double(subs(Y,{x},xu));

exct_sol=exp(xu);

legndr_err=abs(exct_sol-Aprx_sol_legndr);

 %%

load pd.mat

%% plot

figure(1)

% Create multiple lines using matrix input to plot

plot1 = plot(xu,exct_sol,xu,Aprx_sol_legndr);

set(plot1(1),'Marker','*','LineWidth',2);
set(plot1(2),'Marker','square','LineWidth',1,'Color',[1 0
0]);

grid

legend('exct','Aprx legndr')

figure(2)

plot(xu,legndr_err)

grid

legend('legndr err')
```

Note this part explains how to calculate legendre polynomials

```
function p=legendrep(m,x)

%   function which construct Legendre polynomial Pm(x)

%   where M is the degree of polynomial and X is the
variable.

%   Result - P is the char string that should be evaluated
EVAL(P)

%   Example:

%   P=legendrep(2,'cos(theta)') will produce

%   P='(3*cos(theta).^2 -1)/2' which then can be evaluated
as

%   theta=0.3; P=legendrep(2,'cos(theta)'); Lp=eval(P);
produce

%   Lp = 0.8690

%   For Matlab R14 the following example can be used:

%   x=-5:.1:5; p=legendrep(5,'x .*cos(x)'); xp = eval(p);

%   figure; plot(x, xp, 'r.-'); grid

 %% References:

%   Gradshteyn, Ryzhik "Table of Integrals Series and
Products", 6th ed., p.973

%

%___________________________________________________
```

```
%   Sergei Koptenko, Resonant Medical Inc., Toronto

%   sergei.koptenko@resonantmedical.com

%______________March/30/2004_______________________

%%

switch m

case 0

    p='1';     return

case 1

    p=x;     return

case 2

    p=['(3*' x '.^2 -1)/2'];     return

case 3

    p=['(5*' x '.^3 - 3 *' x ')/2'];     return

case 4

    p=['(35 *' x '.^4 - 30 * ' x '.^2 + 3)/8'];     return

case 5

    p=['( 63 * ' x '.^5 - 70 * ' x '.^3 + 15 *' x ' )/8'];
return

case 6

    p=['(231 *' x '.^6 - 315 * ' x '.^4 + 105 * ' x '.^2 -
5)/16'];     return

 case 7

   p=['(429 * ' x '.^7 - 693 * ' x '.^5 +315 * ' x '.^3 -35
*' x ' )/16'];     return

case 8

    p=['(6435 *' x '.^8 -12012 *' x '.^6 + 6930 * ' x '.^4
-1260 * ' x '.^2 +35)/128'];     return

case 9

    p=['(12155 * ' x '.^9 -25740 * ' x '.^7 +18018 * ' x
'.^5 -4620 * ' x '.^3 +315 *' x ' )/128'];
```

```
    return

case 10

       p=['(46189 *' x '.^10 -109395 *' x '.^8 +90090 *' x
'.^6 -30030 * ' x '.^4 +3465 * ' x '.^2 -63)/256'];

    return

case 11

   p=['(88179 * ' x '.^11 -230945 * ' x '.^9 +218790 * ' x
'.^7 -90090 * ' x '.^5 +15015 * ' x '.^3 -693 *' x '
)/256'];

    return

case 12

    p=['(676039 *' x '.^12 -1939938 *' x '.^10 +2078505 *'
x '.^8 -1021020 *' x '.^6 +225225 * ' x '.^4 -18018 * ' x
'.^2 +231)/1024'];

    return

otherwise

    iii=m-10;     %shift counter

   pp=strvcat(legendrep(11,x),legendrep(12,x)); % get last
two members

    for ii=3:1:iii,    % Begin construct from 13th member

        p_ii=[num2str(1/(ii)) ' * (' num2str(2*ii-1) ' * '
x '.*(' ...

        deblank(pp(ii-1,:)) ')-' num2str(ii-1) '.*('
deblank(pp(ii-2,:)) '))'];

        pp=strvcat(pp, p_ii);

    end

p=deblank(pp(iii,:)); % remove traiing blanks

   return

end

 return
```

Matlab Code for Taylor collocation method

```
clear all

clc

close all

format long

%%

%*******************************************************

%***  INPUT DATA

%*******************************************************

syms x t

%%

a=-1;

b=1;

c=0;

N=5;

K=2;

Lambda=1;

FX=exp(x)- 2*sin(x);

KXT=sin(x)*exp(-t);

PS=[1,-x, x];

%% xi

 xi=zeros(1,N+1);

for i=0:N

    xi(1,i+1)=a+i*((b-a)/N);

end

 %% H

 H=zeros(N+1);
for n=0:N
```

```
    for m=0:N
        expnt=n+m+1;
        H(n+1,m+1)=((b-c)^expnt-(a-c)^expnt)/expnt;
    end
end
 %% C
 C=zeros(N+1);
for n=0:N
    for m=0:N
        C(n+1,m+1)=(xi(1,n+1))^m;
    end
end
 %% F


F=zeros(N+1,1);
for i=0:N
    F(i+1,1)=subs(FX,{x},{xi(i+1)});
end
 %% K
 KL=zeros(N+1);
 for n=0:N
    for m=0:N
        for td=0:m
            if td > 0
                PK=diff(KXT,t);
                KXT=PK;
            end
        end
        for xd=0:n
            if xd > 0
                PK=diff(KXT,x);
                KXT=PK;
            end
        end

KL(n+1,m+1)=(1/(factorial(n)*factorial(m)))*subs(KXT,{x,t},
{0,0});
        KXT=sin(x)*exp(-t);
    end
end
 %% M's
 M=zeros(N+1,N+1,K);
 fctrl=0;
for k=0:K
    fctrl=0;
    for n=0:N
        for m=(k):N
            if n==(m-k)
                M(n+1,m+1,(k+1))=1/factorial(fctrl);
                fctrl=fctrl+1;
            end
        end
    end
end
 %% P's
 % P0=-x
% P1=x
% P2=1
% PS=[1,x,-x];
P=zeros(N+1,N+1,K);
for k=0:K
    for n=0:N
        for m=0:N
            if n==m
    P(n+1,m+1,(k+1))=subs(PS(k+1),{x},{xi(m+1)});
            end
        end
    end
end

%% YS
YS=zeros(N+1,1);
YS(1)=1;
YS(2)=1;
%% FU
FU=[F ; YS];
 %% U
 U=zeros(N+1);
for n=0:N
    for m=0:N
        if ((n==m) && (YS(m+1)==1))
            U(n+1,m+1)=1;
        end
    end
end
%% W
WD=zeros(N+1,N+1,K);
for k=0:K
    WD(:,:,k+1)=P(:,:,k+1)*C*M(:,:,k+1);
end
 WX=zeros(N+1);
for k=0:K
    WX(:,:)=WX(:,:)+WD(:,:,k+1);
end
 W=WX-Lambda*C*KL*H*M(:,:,1);
%% WU
WU=[W ; U];
 %% A
A=WU\FU;
%% paper
Co=zeros(1,N+1);
for n =0:N
    Co(1,n+1) = (A(n+1)/factorial(n));
end
%% Y
YT=0;
Co=[1 1 0.500343 0.166886 0.0403378 0.00577493];
for n =0:N
    YT = YT + Co(n+1)*(x - c)^n;
```

```
end

%% Plot

xu=-1:0.2:1;

Aprx_sol_tyl=double(subs(YT,{x},xu));

exct_sol=exp(xu);

figure(2)

plot(xu,exct_sol,':',xu,Aprx_sol_tyl,'g')

grid

legend('exct','Aprx tyl')

%% Table

% T =
table(xu',exct_sol',Aprx_sol_tyl','VariableNames',{'X'
'exct_sol' 'Aprx_sol'});

% filename = 'DataTable.xlsx';

% writetable(T,filename,'Sheet',1)

%% err

figure(4)

tyl_err=abs(exct_sol-Aprx_sol_tyl);

plot(xu,tyl_err)

grid

legend('tylr err')

%%tbl

tbl_tyl=[xu ; exct_sol ; Aprx_sol_tyl ; tyl_err]';
```

جامعة النجاح الوطنية

كلية الدراسات العليا

معادلة فريدهولم الخطية التكاملية التفاضلية من النوع الثانـي

إعداد

خلود نضال اسعيد ظاهر

إشراف

أ.د. ناجي قطناني

قدمت هذه الأطروحة استكمالاً لمتطلبات الحصول على درجة الماجستير في الرياضيات بكلية الدراسات العليا، في جامعة النجاح الوطنية ، نابلس – فلسطين.

2016

معادلة فريدهولم الخطية التكاملية التفاضلية من النوع الثاني

إعداد

خلود نضال اسعيد ظاهر

إشراف

أ.د. ناجي قطناني

الملخص

في هذه الاطروحة ركزنا على حل معادلة فريدهولم التكاملية التفاضلية من النوع الثاني لان لها مجال واسع في التطبيقات الفيزيائية. وقمنا باستقصاء بعض الطرق التحليلية والعددية لحل هذه المعادلة. الطرق التحليلية شملت: طريقة الحساب المباشر، طريقة التكرار التغييري، طريقة أدومين التحليلية، طريقة أدومين التحليلية المعدلة, ظاهرة الضوضاء وطريقة حل السلسلة.

الطرق العددية التي تناولناها هي: طريقة بي سبلاين و وظائف قياس المويجات، طريقة هوموتوبي الاضطرابية، طريقة كثيرات الحدود و دوال لجندر و طريقة تايلور التجميعية. وبشكل خاص الأمثلة العددية التي تناولناها نفذت باستخدام هذه الطرق العددية لحل معادلة فريدهولم التكاملية التفاضلية الخطية من النوع الثاني. تم وضع مقارنة بين هذه الطرق العددية حيث أظهرت لنا النتائج العددية أن طريقة هوموتوبي الاضطرابية وطريقة كثيرات الحدود ودوال لجندر أنها الأكثر كفاءة بالمقارنة مع الطرق العددية الأخرى وذلك بناء على الأمثلة التي استخدمناها.