An-Najah National University

Faculty of Graduated Studies

Numerical Methods for Solving Volterra-Fredholm Integral Equation of the Second Kind

By

Ruba Nasser Ali Salman

Supervisor

Prof.Dr. Naji Qatanani

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

2019

Numerical Methods for Solving Volterra-Fredholm Integral Equation of the Second Kind

By

Ruba Nasser Ali Salman

This Thesis was Defended Successfully on 8\1\2019 approved by:

Defense Committee Members	**Signature**
– **Prof. Naji Qatanani/ Supervisor**	**…….**
– **Prof. Maher Qarawani/ External Examiner**	**…….**
– **Dr. Anwar Saleh/ Internal Examiner**	**…….**

Dedication

I dedicate this thesis to my parents, my husband, my sons Ameer and Kareem, my sisters and my brothers, without their patience, understanding, support and most of all love, this work would not have been possible.

Acknowledgment

First and foremost I am grateful to Allah (swt) for giving me the strength to complete this thesis.

I am heartily thankful to my supervisor, Prof. Dr. Naji Qatanani, whose encouragement, guidance and support from the initial to the final levele nabled me to develop and understanding the subject.

My thanks and appreciation goes to my thesis committee members Prof. Dr. Maher Qarawani and Dr. Anwar Saleh for their encouragement, support, interest and valuable hints.

I acknowledge An-Najah National University for supporting this work, and I wish to pay my great appreciation to all respected teachers and staff in department of mathematics.

Lastly, I offer my regards and blessings to all of those who supported me in any respect during the completion of this thesis.

الإقرار

أنا الموقع أدناه مقدم الرسالة التي تحمل العنوان:

Numerical Methods for Solving Volterra-Fredholm Integral Equation of the Second Kind

أقر بأن ما اشتملت عليه الرسالة هو نتاج جهدي الخاص, باستثناء ما تمت الإشارة اليه حيثما ورد, وأن هذه الرسالة ككل أو جزء منها لم يقدم من قبل لنيل اي درجة علمية أو بحث علمي أو بحثي لدى مؤسسة تعليمية أخرى.

Declaration

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

Student's name: **اسم الطالبة:**

Signature: **التوقيع:**

Date: **التاريخ:**

Table of contents

List of Figures

List of Tables

Numerical Methods for Solving Volterra-Fredholm Integral Equation of the Second Kind

By
Ruba Nasser Ali Salman
Supervisor
Prof. Dr. Naji Qatanani

Abstract

In this thesis we focus on the numerical handling of the Volterra-Fredholm integral equation of the second kind due to their wide range of physical applications such as theory of elasticity, airfoil theory, elastic constant problems and molecular conduction.

After the classification of integral equations, we will investigate some numerical methods for solving the Volterra-Fredholm integral equation of the second kind. These include: Taylor collocation method, least squares approximation method, Legendre collocation method and Lagrange interpolation method.

The mathematical framework of these numerical methods together with their convergence properties will be analyzed. Some numerical examples implementing these numerical methods have been illustrated.

Numerical results clearly show that the Lagrange interpolation method is one of the most powerful and efficient numerical methods for solving Volterra-Fredholm integral equation of the second kind in comparison with the other numerical methods used in this thesis. On the other hand, the Taylor collocation method is one of the fastest methods in terms of CPU-time for solving Volterra-Fredholm integral equation of the second kind.

Introduction

Many physical phenomena in the domain of elasticity, airfoil theory, contact problems, elastic constant problems and molecular conduction can be described by Volterra-Fredholm integral equations (for more details see [1, 2, 10, 25]).

In fact, Volterra-Fredholm integral equations have attracted considerable attention in the past few years due to their wide range of applications.

Majeed et al. [21] used some numerical methods to solve linear Volterra-Fredholm integral equation of the first and second kinds; namely the repeated trapezoidal method and the repeated Simpson's 1/3 method, Al-Jarrah et al. [31] used Scaling function interpolation method to solve linear Volterra-Fredholm integral equation. Lagrange interpolation polynomial used to solve integral equations by Mustafa [26], Adibi et al. [3] applied Legendre-spectral method to solve functional integral equations where the Legendre Gauss points are used as collocation nodes and Lagrange scheme is employed to interpolate the quantities needed. In [10], Abdou used orthogonal polynomials to solve Volterra-Fredholm integral equations. Also, Yusufoglu et al. [41] presented the method based on interpolation in solving linear Volterra-Fredholm integral equations. Maleknejad et al. [23] solved nonlinear Volterra-Fredholm Hammerestein integral equations in terms of Legendre polynomials. Maleknejad et al. [22] solved the system of Volterra-Fredholm integral equations by Adomian decomposition method. Kauthen in [18], used continuous time collocation method Volterra-Fredholm integral

equations. Legendre wavelets were applied for solving Volterra-Fredholm integral equations [40].

Other methods for linear and nonlinear equation of Volterra-Fredholm integral equations can be found in [38]. Yalsinbas developed numerical solution of nonlinear Volterra-Fredholm integral equation by using Taylor polynomial. Also, Mirzaee et al. [24] used modified block-pulse functions to find approximate solutions for mixed nonlinear Volterra-Fredholm integral equation.

Daftardar-Gejji et al. [11] proposed a new technique for solving linear and nonlinear functional equations known as the New Iterative Method (NIM). This method has proven useful for solving a variety of nonlinear equations such as algebraic equations, integral equations, ordinary and partial differential equations of integer and fractional order and systems of equations as well. Bhalekar et al. [6] applied the new iterative method to Fractional-order logistic Eequation. The obtained results were compared with the Adomian decomposition and Homotopy perturbation method. Ambreen et al. [4] solved time-fractional Schrödinger Equations using the New Iterative Method (NIM). The technique is fully compatible with the complexity of these problems and obtained results are highly encouraging. Numerical results coupled with graphical representations explicitly reveal the complete reliability of the suggested algorithm. Daftardar-Gejji et al. [12] employed the NIM to find solutions of linear and nonlinear fractional

diffusion-wave equations. The results obtained were free from rounding off errors since it does not involve discretization.

Ramadan et al. [28] applied the NIM to find approximate analytical solution of the Fornberg-Whitham equation. A comparison is made between the NIM results, homotopy perturbation transform method (HPTM) and Adomian decomposition method (ADM). It was discovered that the NIM solved nonlinear problems without using Adomian's polynomials 2 Hassan Ibrahim et al. On the solution of Volterra-Fredholm and Mixed Volterra-Fredholm Integral Equations using the NIM. Ibrahim et al. [16] Srivastava et al. [31] proposed a new mathematical model, namely a multi-term fractional diffusion equation, for oxygen delivery through a capillary of tissues. They used the NIM and modified Adomian decomposition method (MADM) to solve the multi-term fractional diffusion equation for different conditions. The results thus obtained are compared and presented graphically. It was observed that the order of the diffusion equation affects the delivery of oxygen significantly. The basic difference between the methods is that, the NIM is direct and straightforward and it avoids the volume of calculations resulting from computing the Adomain polynomials. Hemeda [14] obtained the solution of fractional difference equations using the new iterative method (NIM). The results obtained, confirmed the power of the method in reducing the size of calculations compared with other traditional methods. Kocak et al. [19] applied new iterative method in finding the exact solutions of nonlinear time-fractional partial differential equations. The fractional derivatives are described in the Caputo sense. Yaseen et al. [39] used the

iterative method to find an analytic treatment for Laplace equation with Dirichlet and Neumann boundary conditions. The results obtained, show that the present approach is highly accurate and requires reduced amount of calculations compared with the existing iterative methods. It can be concluded that the New Iterative method NIM is a useful technique for solving both linear and nonlinear problems, most especially, in sciences and engineering.

In this thesis, some numerical methods are used to solve the linear Volterra-Fredholm integral equation of the second kind. These methods are Taylor collocation method, Least squares approximation method, Legendre collocation method and Lagrange interpolation method.

This thesis is organized as follows: in chapter one, we introduce some basic concepts of integral equations and the existence theorem. In chapter two, we implement some numerical methods for solving the linear Volterra-Fredholm integral equation of the second kind. These are Taylor collocation method, Least squares approximation method, Legendre collocation method and Lagrange interpolation method. Numerical examples and results are presented in chapter three and the conclusion has been drawn.

Chapter One

Mathematical Preliminaries

1. Some Basic Concepts of Integral Equations

Definition 1.1.1[17]

An integral equation is the equation in which the unknown function appears under one or more integral signs.

The most general type of integral equation is of the form:

$$\psi(x)u(x) = g(x) + \gamma \int_{a(x)}^{b(x)} R(x,t)u(t)dt, \qquad (1.1)$$

where $g(x)$ and $R(x,t)$ are analytic functions.. The functions $a(x)$, $b(x)$, $\psi(x)$ and $R(x,t)$ are given functions, and $\gamma \neq 0$ is a real or complex parameter. The function $R(x,t)$ is called the kernel.

1.1 Classification of Integral Equations

There are many classes of integral equations:

1. **Fredholm Integral Equation**

In Fredholm integral equations, the limits $a(x)$ and $b(x)$are constant.

The standard form:

$$\psi(x)u(x) = g(x) + \gamma \int_a^b R(x,t)u(t)dt, \quad a \le x,t \le b \tag{1.2}$$

In addition, Fredholm integral equations comes in tow types [36]:

a. Fredholm integral equation of the first kind, when $\psi(x) = 0$,

$$g(x) + \gamma \int_a^b R(x,t)u(t)dt = 0 \qquad a \le x,t \le b \tag{1.3}$$

b. Fredholm integral equations of the second kind, when $\psi(x) = 1$,

$$u(x) = g(x) + \gamma \int_a^b R(x,t)u(t)dt\,, \quad a \le x,t \le b \tag{1.4}$$

2. Volterra Integral Equations

In Volterra integral equations, the upper limit of Integral sign is variable $b(x) = x$ and the lower limit $b(x)$ is constant. The standard form is:

$$\psi(x)u(x) = g(x) + \gamma \int_a^x R(x,t)u(t)dt \tag{1.5}$$

Also, Volterra integral equations comes in types [15]:

a. Volterra integral equation of the first kind, when $\psi(x) = 0$,

$$g(x) = \gamma \int_a^x R(x,t)u(t)dt \tag{1.6}$$

b. Volterra integral equations of the second kind, when $\psi(x) = 1$,

$$u(x) = g(x) + \lambda \int_a^x R(x,t)u(t)dt \tag{1.7}$$

3. Volterra-Fredholm Integral Equation

The Volterra-Fredholm integral equation of the second kind is a mixed of Fredholm integral equation and Volterra integral equation. It appears in two forms; the mixed form:

$$u(x) = g(x) + \int_0^x \int_a^b R(s,t)u(t)dsdt \tag{1.8}$$

and the disjoint form,

$$u(x) = g(x) + \gamma_1 \int_a^x R_1(x,t)u(t)dt + \gamma_2 \int_a^b R_2(x,t)u(t)dt \tag{1.9}$$

where $g(x)$ and $R(x,t)$ are continuous and differentiable functions. see [37].

4. Integro-Differential Equations

In this type of equation, the unknown function appears as a combination of ordinary derivative and under integral sign. There are two types:

a. Fredholm-integro-differential equation

$$u^{(i)}(x) = g(x) + \gamma \int_a^b R(x,t)u(t)dt, \text{for some positive integer } i \tag{1.10}$$

b. Volterra-integro-differential equation

$$u^{(i)}(x) = g(x) + \gamma \int_a^x R(x,t)u(t)dt, \tag{1.11}$$

where $u^{(i)}(x) = \frac{d^i u}{dx^i}$, for some positive integer i

5. Singular Integral Equations

The integral equation is called singular when one or both limits of integration become infinite or when the kernel $R(x,t)$ becomes infinite at one or more points in the domain of the integration.

1.2 Further Concepts of Integral Equations

Definition 1.2.1[17]:

An integral equation is called linear integral equation if the exponent of the unknown function $u(x)$ inside the integral sign is one. If the unknown function $u(x)$ has exponent other than one, or if the equation contains nonlinear functions of $u(x)$, such as e^u, $\sinh u$, $\cos u$, $\ln(1+u)$, the integral equation or the integro-differential equation is called nonlinear.

Definition 1.2.2[33]

If the function $g(x)$ in the Volterra or Fredholm integral equations or integro-differential equations is identically zero, the equation is called homogeneous. Otherwise it is called inhomogeneous.

Special Kinds of Kernels [17]

1. Symmetric Kernel

The kernel $R(x,t)$ is said to be symmetric if $R(x,t) = R(t,x)$.

2. Separable Kernel

The kernel $R(x,t)$ is said to be separable if it can be written as the sum of a finite number of terms, each of them is the product of a function of x only and function of t only, i.e linearly independent.

$$R(x,t) = \sum_{j=1}^{m} a_j(x)b_j(t)\text{, for some positive integer } m$$

1.3 The Existence and Uniqueness Theorem

Preliminaries

Let (X,d) be a metric space and $T: X \to X$ is operator. We shall use the following notations:

a. The power set of X denoted by $P(X) = \{A \subseteq X | A \neq \emptyset\}$;

b. The fixed point set of T $F_P = \{x \in X | T(x) = x\}$.

c. $\| . \|_B : C[a,b] \to \mathbb{R}_+, \|y\|_B = \max_{x\in[a,b]} |y(x)|\, e^{-\tau(x-a)}, \tau > 0$, a Bielecki norm on $C[a,b]$.

d. $d_B: C[a,b] \times C[a,b] \to \mathbb{R}_+, d_B(y,z) = \|y-z\|_B$ -the corresponding metric of $\|.\|_B$.

Definition 1.3.1 [8]

Let D be a metric space, the set of all continuous, real-valued functions $f: D \to \mathbb{R}$ is the space of continuous functions denoted by $C(D)$. The set $C(D)$ is a real linear space under the pointwise addition of functions and the scaler multiplication of functions by real numbers. That is, for $f, g \in C(D)$ and $a \in \mathbb{R}$, we define: $(f+g)(z) = f(z) + g(z)$, $(af)(z) = a\big(f(z)\big)$.

Definition 1.3.2 [30]:

Let (X,d) be a metric space. An operator $T: X \to X$ is a Picard operator if there exists $x^* \in X$ such that $F_T = \{x^*\}$ and the sequence $(T_n(x_0))_{n\in\mathbb{N}}$ converges to x^* for all $x_0 \in X$.

Theorem (Contraction Principle) 1.3.1 [8]

Let (X,d) be a complete metric space and $T: X \to X$ is contraction. Then P is a Picard operator and

$$d\left(\{x^*{}_T, T^n(x_0)\right) \leq \frac{\beta^n}{1-\beta} d\big(x_0, T(x_0)\big), for\ all\ n \in \mathbb{N}.$$

Theorem 1.3.2 [29]

Let (X, d) be a complete metric space and let T and Q be two operators on X. Suppose that:

a. T is β −contraction and $F_T = \{x^*\}$;

b. Q has fixed points and ${x_Q}^* \in F_Q$;

c. There exists $\omega > 0$ such that $d\big(T(x), Q(x)\big) \leq \omega$, for all $x \in X$;

Then

$$d\big({x_T}^*, {x_Q}^*\big) \leq \frac{\omega}{1-\beta}.$$

Now, Consider the following Volterra-Fredholm integral equation:

$$u(x) = g(x) + \int_a^x R_1(x,t)u(t)dt + \int_a^b R_2(x,r)u(r)dr, \quad x\epsilon[a,b] \quad (1.14)$$

where $g \in C[a,b], R_1 \in C(D_1)$ and $R_2 \in C(D_2)$

Here $D_1 = \{(x,t)\epsilon\mathbb{R}^2 | a \leq t \leq x \leq b$ and $D_2 = [a,b] \times [a,b]$

Let $M_1 = \max_{(x,t)\epsilon D_1} |R_1(x,t)|$ and $M_2 = \max_{(x,r)\epsilon D_2} |R_2(x,r)|$

Then we have the following theorem

Theorem 1.3.3 [8]: In virtue of the above continuity conditions, suppose that there exists $\tau > 0$ such that:

$$\frac{1}{\tau}\left[M_1 + M_2 e^{\tau(b-a)}\right] < 1$$

then equation (1.14) has a unique solution $u^* \in C[a,b]$, and this solution can be obtained by the successive approximation method, starting from any element of $C[a,b]$.

Proof:

Define the operator $T: (C[a,b], d_B) \to (C[a,b], d_B)$:

$$T(u)(x) = g(x) + \int_a^x R_1(x,t)u(t)dt + \int_a^b R_2(x,r)u(r)dr$$

To show that T satisfies Lipschitz condition:

$$
\begin{aligned}
&|T(u)(x) - T(z)(x)| \\
&\qquad = \left| \int_a^x R_1(x,t)\big(u(t) - z(t)\big)dt \right. \\
&\qquad \left. + \int_a^b R_2(x,r)\big(u(r) - z(r)\big)dr \right| \\
&\qquad \leq \int_a^x |R_1(x,t)||u(t) - z(t)|dt \\
&\qquad + \int_a^b |R_2(x,r)||u(r) - z(r)|dr \\
&\qquad \leq M_1 \int_a^x |u(t) - z(t)| e^{-\tau(t-a)} e^{\tau(t-a)} dt \\
&\qquad + M_2 \int_a^b |u(r) - z(r)| e^{-\tau(r-a)} e^{\tau(r-a)} du \\
&\qquad \leq \left[\frac{M_1}{\tau}(e^{\tau(x-a)} - 1) + \frac{M_2}{\tau}(e^{\tau(b-a)} - 1) \right] \|u - z\|_B \\
&\qquad \leq \frac{1}{\tau}(M_1 e^{\tau(x-a)} + M_2 e^{\tau(x-a+b-x)}) \|u - z\|_B \\
&\qquad = \frac{e^{\tau(x-a)}}{\tau}(M_1 + M_2 e^{\tau(b-x)}) \|u - z\|_B \\
&\qquad \leq \frac{e^{\tau(x-a)}}{\tau}(M_1 + M_2 e^{\tau(b-a)}) \|u - z\|_B .
\end{aligned}
$$

By using Bielecki norm it follows that:

$$
|T(u)(x) - T(z)(x)| e^{-\tau(x-a)} \leq \frac{1}{\tau}(M_1 + M_2 e^{\tau(b-a)}) \|u - z\|_B
$$

For all $x \epsilon [a, b]$. Therefore,

$$
\|T(u) - T(z)\|_B \leq \frac{1}{\tau}(M_1 + M_2 e^{\tau(b-a)}) \|u - z\|_B
$$

or

$$
d_B(T(u), T(z)) \leq \frac{1}{\tau}(M_1 + M_2 e^{\tau(b-a)}) d_B(u, z)
$$

The operator T satisfies Lipschitz condition with Lipschitz constant

$$L_T = \frac{1}{\tau}(M_1 + M_2 e^{\tau(b-a)})$$

The supposed condition ensured that T is a contraction. So, we apply contraction principle.

Chapter Two

Numerical Methods for Solving Volterra-Fredholm Integral Equation of the Second Kind

In this chapter we consider the following numerical methods that can be used to solve Volterra-Fredholm integral equation of the second kind. These are:

a. Taylor collocation method

b. Least squares approximation method

c. Legendre collocation method

d. Lagrange interpolation method

We consider the Volterra-Fredholm integral equation of the second kind of the form:

$$u(x) = g(x) + \lambda_1 \int_a^x R_1(x,t)u(t)dt + \lambda_2 \int_a^b R_2(x,t)u(t)dt \qquad (2.1)$$

where the functions $R_1(x,t), R_2(x,t)$ and $g(x)$ are known functions defined on the region $D = \{(x,t): a \leq x \leq \infty, t \leq b, a, b \in \mathbb{R}\}$, $u(x)$ is the unknown function to be determined, $\gamma_1, \gamma_2 \in \mathbb{R}$ and ${\gamma_1}^2 + {\gamma_2}^2 \neq 0$.

2.1 Taylor collocation method [35]

Taylor collocation method with Taylor polynomials of degree N can be used to solve the Volterra-Fredholm integral equation of the second kind. This method uses collocation points to transform the integral equation to a matrix equation, here how we use Taylor polynomial for $u(x)$:

$$u_N(x) = \sum_{m=0}^{N} \frac{1}{m!} u^{(m)}(d)(x-d)^m, \quad a \le x, d \le b, \tag{2.2}$$

this is a Taylor polynomial of degree N at $x = d$, where $u^{(m)}(d)$ are the coefficients to be determined using collocation points that defined by:

$$x_n = a + hn, \qquad h = \frac{b-a}{N}, \quad n = 0,1,\dots,N, \tag{2.3}$$

It is clear that

$$a = x_0 < x_1 < \cdots < x_N = b.$$

We write equation (2.1) in the form:

$$U(x) = g(x) + \lambda_1 V(x) + \lambda_2 F(x) \tag{2.4}$$

where

$$U(x) = u(x)$$

$$V(x) = \int_a^x R_1(x,t)u(t)dt \tag{2.5}$$

and

$$F(x) = \lambda_2 \int_a^b R_2(x,t)u(t)dt \tag{2.6}$$

Substituting (2.2) and the collocation points (2.3) into equation (2.4) gives:

$$U(x_n) = f(x_n) + \lambda_1 V(x_n) + \lambda_2 F(x_n), \qquad n = 0,1,\dots,N \tag{2.7}$$

where

$$U(x_n) = \sum_{m=0}^{N} \frac{1}{m!} u^{(m)}(d)(x_n-d)^m \tag{2.8}$$

$$V(x_n) = u^{(m)}(d) \sum_{m=0}^{N} \frac{1}{m!} \int_a^{x_n} R_1(x_n,t)(x_n-d)^m dt \tag{2.9}$$

and

$$F(x_n) = u^{(m)}(d) \sum_{m=0}^{N} \frac{1}{m!} \int_a^b R_2(x_n, t)(x_n - d)^m dt, \tag{2.10}$$

or:

$$\sum_{m=0}^{N} \frac{1}{m!} u^{(m)}(d) \left[\widehat{U}(x_n) - \lambda_1 \hat{V}(x_n) - \lambda_2 \hat{F}(x_n)\right] = f(x_n) \tag{2.11}$$

where

$$\widehat{U}(x_n) = (x_n - d)^m, \tag{2.12}$$

$$\hat{V}(x) = \int_a^{x_n} R_1(x_n, t)(x_n - d)^m dt \tag{2.13}$$

and

$$\hat{F}(x_n) = \int_a^b R_2(x_n, t)(x_n - d)^m dt. \tag{2.14}$$

Take $m, n = 0,1, \dots, N$, then equation (2.6) yields the matrix form:

$$D^* U^* = F^* \tag{2.15}$$

where $D^* = (a_{nm})$ is a matrix of order $(N+1) \times (N+1)$ whose entries are defined as follows:

$$a_{nm} = \frac{1}{m!} \left[\widehat{U}(x_n) - \lambda_1 \hat{V}(x_n) - \lambda_2 \hat{F}(x_n)\right] \tag{2.16}$$

$$U^* = (\bar{u}^{(0)}(d), \bar{u}^{(1)}(d), \dots, \bar{u}^{(N)}(d))^T, \tag{2.17}$$

$$F^* = \left(g(x_0), g(x_1), \dots, g(x_N)\right)^T. \tag{2.18}$$

Solving system (2.15), we get the coefficients $\bar{U}^{(m)}(d), m = 0,1, \dots, N$ which are unique [7,18].

Hence, equation (2.1) has unique solution that can be obtained by Taylor polynomial called N-order Taylor collocation solution:

$$\bar{u}_N(x) \cong \sum_{m=0}^{N} \frac{1}{m!} \bar{u}^{(m)}(d)(x-d)^m. \tag{2.19}$$

2.2 Convergence Analysis

Here, we will show that the approximation solution from Taylor collocation method converges to the exact solution of the Volterra-Fredholm integral equation of the second kind.

Theorem (2.1) [35]

Assume that $g(x)$ is a function defined on $[\mathrm{a}, \mathrm{b}]$ and that $R_1(x,t)$ and $R_2(x,t)$ in $[\mathrm{a}, \mathrm{b}] \times [\mathrm{a}, \mathrm{b}]$. Assume that $g(x), R_1(x,t)$, and $R_2(x,t)$ are sufficiently smooth continuous and arbitrary differentiable functions. Let $u(x)$ be the exact solution of (2.1) and $\bar{u}_N(x)$ is the N-order Taylor collocation solution of (2.7), then:

$$\|u(x) - \bar{u}_N(x)\|_\infty \leq \frac{M}{(N+1)!} |u^{N+1}(\xi)| + D \max_{0 \leq n \leq N} |e_n(d)|, \tag{2.20}$$

where

$$M = \max_{a \leq x \leq b} |(x-d)^{N+1}|,$$

$$D = \|h\|_\infty = \max_{a \leq x \leq b} \{|h_0(d)|, |h_1(d)|, \dots, |h_n(d)|\},$$

$$e_n(d) = u^{(n)}(d) - \bar{u}^{(n)}(d)\ ,\ a \leq \xi \leq b.$$

Proof:

One can easily find that

$$\|u(x) - \bar{u}_N(x)\|_\infty \le \|u(x) - T_n(x)\|_\infty + \|T_n(x) - \bar{u}_N(x)\|_\infty,$$

where

$$T_n(x) = \sum_{n=0}^{N} \frac{u^n(d)(x-d)^n}{n!},$$

$$\bar{u}_N(x) = \sum_{n=0}^{N} \frac{\bar{u}^n(d)(x-d)^n}{n!}.$$

Since $R_n(x) = u(x) - T_n(x) = \frac{u^{(N+1)}(\xi)}{(N+1)!}(x-c)^{N+1}$ is the remainder of the Taylor polynomial of degree N at $x = d$, we obtain

$$|R_n(x)| \le \frac{u^{(N+1)}(\xi)}{(N+1)!} . \max_{a \le x \le b} |(x-d)^{N+1}| = \frac{M}{(N+1)!} u^{(N+1)}(\xi). \quad (2.21)$$

set

$$\delta_n = \big(e_0(d), e_1(d), \dots, e_n(d), \dots, e_N(d)\big),$$

$$h = (h_0(d), h(d), \dots, h_n(d), \dots, h_N(d))^T,$$

where

$$e_n(d) = u^{(n)}(d) - \bar{u}^{(n)}, \quad h_n(d) = \frac{(x-d)^n}{n!}.$$

then we have

$$\begin{aligned} |T_n(x) - \bar{u}_N(x)| &= \sum_{n=0}^{N} (u^n(d) - \bar{u}_N(c)) \frac{(x-d)^n}{n!} = |\delta_n . h| \le \|\delta_n\|_\infty . \|h\|_\infty \\ &\le D\|\delta_n\|_\infty. \end{aligned} \quad (2.22)$$

It follows from (2.21) and (2.22) that

$$\|u(x) - \bar{u}_N(x)\|_\infty \le \frac{u^{N+1}(\xi)}{(N+1)!}. \max_{0\le n\le N} |(x-d)^{N+1}| + \|h\|_\infty . \|\delta_n\|_\infty$$
$$= \frac{M}{(N+1)!}|u^{N+1}(\xi)| + D \max_{0\le n\le N} |e_n(d)|.$$

2.3 Least Squares Approximation Method [34]

In this section, we present the least squares approximation method to obtain numerical solution for the Volterra-Fredholm integral equation of the second kind (2.1).

The operator $P\big(u(x)\big)$ corresponding to equation (2.1) can be written as:

$$P\big(u(x)\big) = u(x) - g(x) - \gamma_1 \int_a^x R_1(x,t)u(t)dt - \gamma_2 \int_a^b R_2(x,t)u(t)dt \qquad (2.23)$$

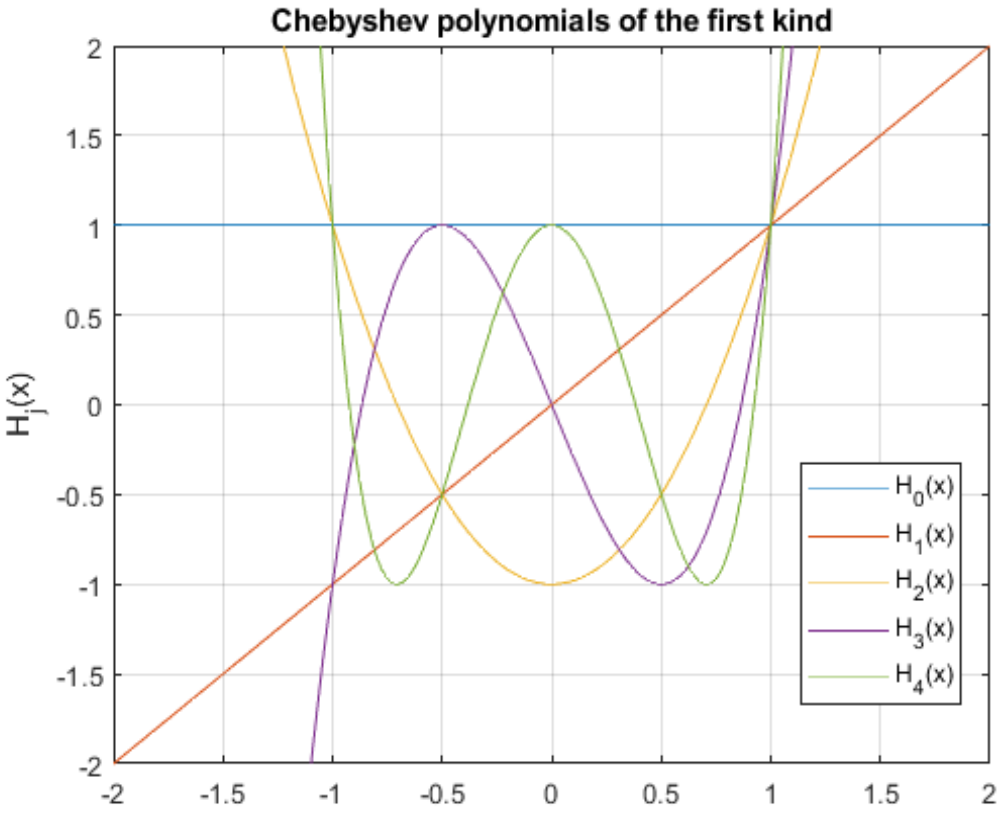
Let $\psi_0(x), \psi_1(x), \dots, \psi_n(x)$ be linearly independent functions on the interval $[a, b]$, where n is positive integer $n > 0$, and

$\Psi_n = \text{span}\{\psi_0(x), \psi_1(x), \dots, \psi_n(x)\}$ is generated by the basis.

We can use Chebyshev polynomials as a basis for the estimation of the solution of equation (2.1). Chebyshev polynomials satisfy the following recurrence relations,

$$H_0(x) = 1, \quad H_1(x) = x$$

$$H_j(x) = 2xH_{j-1}(x) - H_{j-2}(x), \quad j \ge 2, \quad -1 \le x \le 1 \qquad (2.24)$$



To use Chebyshev polynomials on our interval [0,1], we change the domain by substituting

$$y = 2x - 1, \qquad 0 \le x \le 1 \tag{2.25}$$

So, the shifted Chebyshev polynomials has the form

$$S_n(x) = T_n(2x - 1), \qquad 0 \le x \le 1 \tag{2.26}$$

Now, let $u_n(x) \in \Psi_n$, then there exists numbers sequence $e_0, e_1, \dots, e_n$ such that

$$u_n(x) = \sum_{i=0}^{n} e_i \psi_i(x) \tag{2.27}$$

Substituting (2. 27) into equation (2.1), we get

$$P\big(x,u_n(x)\big) = u_n(x) - g(x) - \gamma_1 \int_a^x R_1(x,t)u_n(t)dt - \lambda_2 \int_a^b R_2(x,t)u_n(t)dt \quad (2.28)$$

$$= \sum_{i=0}^{n} e_i.\,[\psi_i(x) - \gamma_1 \int_a^x R_1(x,t)\psi_i(t)dt - \gamma_2 \int_a^b R_2(x,t)\psi_i(t)dt] - g(x)$$

$$= \sum_{i=0}^{n} e_i.\,\alpha_i(x) - g(x) \quad (2.29)$$

where

$$\alpha_i(x) = \psi_i(x) - \gamma_1 \int_a^x R_1(x,t)\psi_i(t)dt - \gamma_2 \int_a^b R_2(x,t)\psi_i(t)dt\,, \quad (2.30)$$

$i = 0,1,\dots,n, \qquad x \in [a,b]$

$$I_n(x) = u_n(x) - u(x) - \gamma_1 \int_a^x R_1(x,t)(u_n(t) - u(t))dt - \gamma_2 \int_a^b R_2(x,t)\big(u_n(t) - u(t)\big)dt \quad (2.31)$$

Let $I_n(x) = P(x,u_n(x)) - P(x,u(x))$ is called the n-order remaining items of equation (2.1)

Remarks [47]

a. If $I_n(x) = 0$, then $u(x) = u_n(x)$; moreover if $\lim_{n\to\infty} I_n(x) = 0$, then $\lim_{n\to\infty} u_n(x) = u(x)$.

b. For any $x \in [a,b]$ and if $I_n(x) = 0$, then $u_n(x)$ is an exact solution of equation (2.1). If $\lim_{n\to\infty} I_n(x) = 0$, then $u_n(x)$ converges to the exact solution of equation (2.1).

$$\text{Now, let } M = M(e_0, e_1, \dots, e_n) = \int_a^b P^2(x, u_n(x))dx \tag{2.32}$$

Then the problem now is to find the real coefficients $e_0, e_1, \dots, e_n$ that minimize M.

A necessary condition for the minimization of M is

$$\frac{\partial M}{\partial e_i} = 0, \text{for each } i = 0,1, \dots, n. \tag{2.33}$$

By the relation (2.32) we can easily get

$$\frac{\partial M}{\partial e_i} = 2\int_a^b P(x, u_n(x))\frac{\partial P(x, u_n(x))}{\partial e_i}dx \tag{2.34}$$

$$= 2\int_a^b \{\sum_{j=0}^{n}[\psi_j(x) - \gamma_1\int_a^x R_1(x,t)\psi_j(t)dt - \gamma_2\int_a^b R_2(x,t)\psi_j(t)dt].e_j - g(x)\}$$

$$.\left[\psi_i(x) - \gamma_1\int_a^x R_1(x,t)\psi_i(t)dt - \gamma_2\int_a^b R_2(x,t)\psi_i(t)dt\right]dx \tag{2.35}$$

$$= 2\left[\int_a^b \sum_{j=0}^{n} e_j.\alpha_j(x).\alpha_i(x)dx - \int_a^b g(x)\alpha_i(x)dx\right] \tag{2.36}$$

$$= 0.$$

Thus we have:

$$\sum_{i=0}^{n} e_j . \int_a^b \alpha_j(x).\alpha_i(x)dx = \int_a^b g(x)\alpha_i(x)dx, \qquad i = 0,1,\dots,n. \quad (2.37)$$

To find $u_n(x)$, we will solve the $(n+1)$ linear equations for the (n+1) unknowns e_j .We write the system (2.32) as :

$$G_n \,.E = F \quad (2.38)$$

where

$$G_n = \begin{bmatrix} (\alpha_0,\alpha_0) & (\alpha_0,\alpha_1) & \dots & (\alpha_0,\alpha_n) \\ (\alpha_1,\alpha_0) & (\alpha_1,\alpha_1) & \dots & (\alpha_1,\alpha_n) \\ \vdots & \vdots & \vdots & \vdots \\ (\alpha_n,\alpha_0) & (\alpha_0,\alpha_1) & \dots & (\alpha_n,\alpha_n) \end{bmatrix},$$

$E = [e_0, e_1, \dots, e_n]^T$,

and

$F = [(\alpha_0, g(x_1)), (\alpha_1, g(x_2)), \dots, (\alpha_n, g(x_n))]^T$.

2.4 Convergence Analysis

We can show that the optimal squared approximation solution $u_n(x)$ converges to the exact solution $u(x)$ of equation (2.1).

Definition 2.4.1 [34]

For every $\varepsilon > 0$ and if $\int_a^b P^2\left(x, u_n(x)\right)dx \le \varepsilon$, then $u_n(x)$ is called an ε −approximate solution of equation (2.1).

Now let us assume that the functions $\alpha_i(x), i = 0,1,\dots,n,$ are linearly independent functions on the interval $[a,b]$, if the matrix G_n is a nonsingular matrix, then $E = G_n^{-1}F = E^0$ is a unique solution of equation (2. 32)

Definition 2.4.2 [34]

If equation (2.32) has a unique solution $E = E^0 = (e_0^0, e_1^0, \dots, e_n^0)^T$, then

$u_n(x) = \sum_{i=0}^{n} e_i^0 \psi_i(x)$ is called an optimal squared approximation solution of equation (2.1) defined on a set $\Psi_n = \text{span}\{\psi_1(x), \psi_2(x), \dots, \psi_n(x)\}, x \in [a, b]$.

Remark (2.1) [34]

If $\lim_{n\to\infty} \int_a^b P^2\big(x, u_n(x)\big)dx = 0$, then the optimal squared approximation solution $u_n(x) = \sum_{i=0}^{n} e_i^0 \psi_i(x)$ converges to the exact solution $u(x)$ of equation (2.1).

Theorem (2.2) [34]:

Suppose that $u(x)$ is an exact solution of equation (2.1) defined on $[a, b]$ and $u_n(x)$ is an optimal squared approximation solution of equation (2.1). Moreover,

If there exists $P_n(x) = \sum_{i=0}^{n} a_i x^i,\ a_i \in \mathbb{R}$, such that $\forall x \in [a, b]$,
$\lim_{n\to\infty} P_n(x) = u(x)$, then:

$$\lim_{n\to\infty} \int_a^b P^2(x, u_n(x)dx = 0$$

2.5 Legendre Collocation Method [9]

Legendre Polynomials are defined on $[-1,1]$ as follows

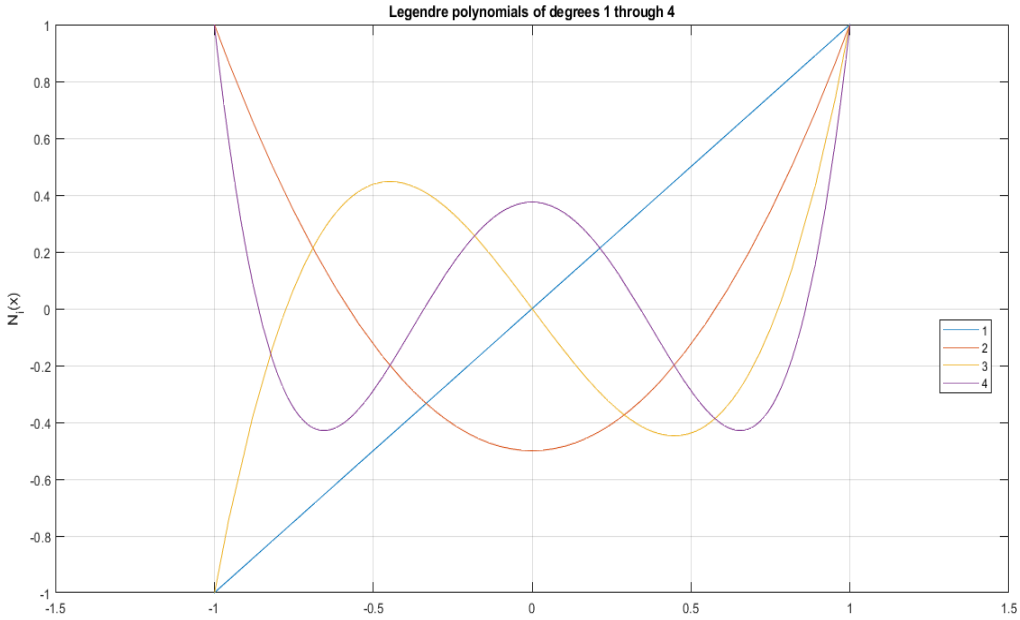
$N_0(x) = 1$

$$N_n(x) = \frac{(-1)^n}{2^n n!}\frac{d^n}{dx^n}((1-x^2)^n), n \geq 1$$

As a result

$$N_1(x) = x$$

$$N_{i+1}(x) = \frac{2i+1}{i+1}xN_i(x) - \frac{i}{i+1}N_{i-1}(x), i = 1,2,3,\ldots \quad (2.39)$$



In order to use these polynomials on [0,1], we define the shifted Legendre polynomials:

$$\phi_i(x) = N_i(2x-1), \quad i = 0,1,2,\ldots \quad (2.40)$$

The nodes r_i $(0 \leq i \leq n)$ of standard Legendre polynomials are the roots of these polynomials on $[-1,1]$, and the nodes of the shifted Legendre polynomials on $[0,1]$ are defined by [7,20,41]

$$s_j = \frac{1}{2}(r_j + 1), \qquad 0 \leq j \leq n \quad (2.41)$$

The Legendre collocation method [27] is to seek polynomial $u(x)$ that we can expressed in terms of shifted Legendre polynomials as:

$$u(x) \cong \sum_{i=0}^{n} d_i \phi_i(x) \tag{2.42}$$

where $d_i, i = 0,1, \dots, n$ are the coefficients to be determined, substituting (2.42) in (2.1) gives:

$$\sum_{i=0}^{n} d_i \phi_i(x) = g(x) + \gamma_1 \int_0^x R_1(x,t) \sum_{i=0}^{n} d_i \phi_i(t)\, dt + \gamma_2 \int_0^1 R_2(x,t) \sum_{i=0}^{n} d_i \phi_i(t)\, dt \tag{2.43}$$

Suppose that

$$h_i(x) = \phi_i(x) - \gamma_1 \int_0^x R_1(x,t)\phi_i(t)dt - \gamma_2 \int_0^1 R_2(x,t)\phi_i(t)dt \tag{2.44}$$

Then equation (2.43) can be written as

$$\sum_{i=0}^{n} d_i h_i(x) = g(x) \tag{2.45}$$

Collocating equation (2.44) in (n +1) roots of the shifted Legendre polynomial $\phi_{n+1}(x)$ and the shifted Gauss-Legendre nodes, we obtain:

$$\sum_{i=0}^{n} d_i h_i(x_j) = f(x_j), \quad for\ j = 0,1, \dots, n \tag{2.46}$$

Equation (2.46) in matrix form:

$$H^T D = F \tag{2.47}$$

where

$F = [g(x_0), g(x_1), \dots, g(x_n)]^T$

$D_{ij} = d_i(x_j), \qquad i, j = 0,1, \dots, n.$

the unknown vector D is computed by:

$D = (H^T)^{-1}F$

So, the approximate solution is given by $u(x) = H^T\phi(x)$.

2.6 Lagrange Interpolating Polynomial [26]

The Lagrange interpolating polynomial of $u(x)$ is the polynomial $P_n(x)$ of degree n that passes through the $(n+1)$ distinct points:

$(x_0, u_0), (x_1, u_1), \dots, (x_n, u_n)$,

and given by:

$$P_n(x) = \sum_{j=0}^{n} u_j \, E_{n,j}(x) \tag{2.48}$$

where the polynomials $E_{n,j}(x), 0 \le j \le n$ have the property:

$$E_{n,j}(x_i) = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$$

The polynomials $E_{n,j}(x), 0 \le j \le n$ are called Lagrange polynomials defined by:

$$E_{n,j}(x) = \prod_{\substack{r=0 \\ r \neq j}}^{n} \frac{x - x_r}{x_j - x_r} \tag{2.49}$$

To solve Volterra-Fredholm integral equation (2.1) using Lagrange polynomials, we first, define Lagrange formula for a set of (n+1) data points $\{(x_0, t_0), (x_1, t_1), \dots, (x_n, t_n)\}$ as we define in equation (2.48) and substitute in (2.1), to get:

$$\sum_{j=0}^{n} u_j \, E_{n,j}(x) - \lambda_1 \int_a^x R_1(x,t) \sum_{j=0}^{n} u_j \, E_{n,j}(t) \, dt$$

$$-\lambda_2 \int_a^b R_2(x,t) \sum_{j=0}^{n} u_j \, E_{n,j}(t) \, dt = g(x)$$

This yields:

$$u_0 \prod_{\substack{r=0 \\ r\neq j}}^{n} \frac{x-x_r}{x_0-x_r} + u_1 \prod_{\substack{r=0 \\ r\neq j}}^{n} \frac{x-x_r}{x_1-x_r} + \cdots + u_n \prod_{\substack{r=0 \\ r\neq j}}^{n} \frac{x-x_r}{x_n-x_r}$$
$$- \lambda_1 \int_a^x K_1(x,t)[u_0 \prod_{\substack{r=0 \\ r\neq j}}^{n} \frac{t-x_r}{x_0-x_r} + u_1 \prod_{\substack{r=0 \\ r\neq j}}^{n} \frac{t-x_r}{x_1-x_r} + \cdots$$
$$+ u_n \prod_{\substack{r=0 \\ r\neq j}}^{n} \frac{t-x_r}{x_n-x_r}]dt$$
$$- \lambda_2 \int_a^b K_2(x,t) \left[u_0 \prod_{\substack{r=0 \\ r\neq j}}^{n} \frac{t-x_r}{x_0-x_r} + u_1 \prod_{\substack{r=0 \\ r\neq j}}^{n} \frac{t-x_r}{x_1-x_r} + \cdots \right.$$
$$\left. + u_n \prod_{\substack{r=0 \\ r\neq j}}^{n} \frac{t-x_r}{x_n-x_r} \right] dt = g(x) \qquad (2.50)$$

Now, we simplify this equation to get

$$\frac{u_0}{\prod_{r=1}^{n}(x_0-x_r)} \left\{ \prod_{r=1}^{n}(x-x_r) - \gamma_1 \int_a^x R_1(x,t)[\prod_{r=1}^{n}(t-x_r)]dt - \right.$$
$$\gamma_2 \int_a^b R_2(x,t)[\prod_{r=1}^{n}(t-x_r)]dt\} + \frac{u_1}{\prod_{\substack{r=0 \\ r\neq 1}}^{n}(x_1-x_r)} \{\prod_{\substack{r=0 \\ r\neq 1}}^{n}(x-x_r) -$$
$$\gamma_1 \int_a^x R_1(x,t) \left[\prod_{\substack{r=0 \\ r\neq 1}}^{n}(t-x_r)\right] dt - \gamma_2 \int_a^b R_2(x,t) \left[\prod_{\substack{r=0 \\ r\neq 1}}^{n}(t-x_r)\right] dt\} +$$
$$\cdots + \frac{u_n}{\prod_{\substack{r=0 \\ r\neq n}}^{n}(x_n-x_r)} \{\prod_{\substack{r=0 \\ r\neq n}}^{n}(x-x_r) - \gamma_1 \int_a^x R_1(x,t) \prod_{\substack{r=0 \\ r\neq n}}^{n}(t-x_r)\, dt -$$
$$\left. \gamma_2 \int_a^b R_2(x,t) \prod_{\substack{r=0 \\ r\neq n}}^{n}(t-x_r)\, dt \right\} = g(x) \qquad (2.51)$$

We put $x = x_i$, $for\ i = 0,1,\dots,n$, so we get system of $n+1$ equations, that is:

$$AU = F \qquad (2.52)$$

where

$$A = a_{ij} = \begin{cases} 1 - \frac{1}{q}\left[\int_a^{x_i} R_1(x_i,t)q_1 dt + \int_a^b R_2(x_i,t)q_1 dt\right], & if\ i = j \\ -\frac{1}{q}\left[\int_a^{x_i} R_1(x_i,t)q_1 dt + \int_a^b R_2(x_i,t)q_1 dt\right], & if\ i \neq j \end{cases}$$

where

$$q = \prod_{\substack{r=0 \\ r\neq j}}^{n}(x_j - x_r) \text{ and } q_1 = \prod_{r=0}^{n}(t - x_r) for\ all\ i,j = 0,1,\dots,n$$

2.7 Convergence Analysis

Theorem (2.3) [32]

Suppose that $n \geq 0$, and that u is a real-valued function, defined and continuous on the closed real interval $[a,b]$, such that the derivative of u of order $(n+1)$ exists and is continuous on $[a,b]$. Then, for every $x \in [a,b]$, there exists $\delta = \delta(x)$ in (a,b) such that:

$$u(x) - p_n(x) = \frac{u^{(n+1)}(\delta(x))}{(n+1)!}\prod_{r=0}^{n}(x - x_r) \tag{2.53}$$

Moreover

$$|u(x) - p_n(x)| \leq \frac{M_{n+1}}{(n+1)!}|\omega_{n+1}| \tag{2.54}$$

where

$$M_{n+1} = \max_{\mu\in[a,b]}|u^{n+1}(\mu)|$$

Convergence Theorem (2.4) [32]

If $\lim_{n\to\infty}\frac{M_{n+1}}{(n+1)!}\max_{x\in[a,b]}|\omega_{n+1}(x)| = 0$

then, by (2.54),

$$\lim_{n\to\infty}\max_{x\in[a.b]}|u(x) - p_n(x)| = 0 \tag{2.55}$$

and we say that the sequence of interpolation polynomials p_n with equally spaced points on [a, b], converges uniformly to u as $n \to \infty$, on the interval [a, b].

Chapter Three

Numerical Examples and Results

In this Chapter, four numerical methods, namely: Taylor collocation method, Least squares approximation method, Legendre collocation method and Lagrange interpolating method will be implemented to solve some numerical examples with known exact solution. The algorithm for each method will be implemented using the Matlab Software.

Example (3.1):

Consider the Volterra-Fredholm integral equation of the second kind:

$$u(x) = x - 2e^{x} + e^{-x} + 1 + \int_{0}^{x} t.e^{x}u(t)dt + \int_{0}^{1} e^{(t+x)}u(t)dt \qquad (3.1)$$

The exact solution of equation (3.1) [26] is:

$$u(x) = e^{-x}. \qquad (3.2)$$

Algorithm (3.1): Taylor Collocation Algorithm:

This algorithm is coded using MATLAB software.

Input $a, b, c, N, L1, L2, A, B$

where $L1 = \lambda_1, L2 = \lambda_2., A = 1, B = 0$

Input the functions: $h, h_t, f, k1, k2, Y_{ex}$.

where $h = h(x), h_t = h(x)|_{x=t}, Y_{ex} =$ exact solution.

calculate U, V, F and the right side vector, and that will be in two “For” loops that start from 0 to N.

```
for n = 0:N
x1(n+1) = vpa(a+((b-a)/(N))n);
for m=0:N
    U =A(x − c)^m  +  B(h − c)^m;
    V = int(k1(t − c)^m,t,a,h);
    F = int(k2(ht − c)^m,t,a,b);
D(n+1,m+1)= subs((1/factorial(m))(U - L1V - L2F),x,x1(n+1));
end
  F_s(n+1) = subs(f,x,x1(n+1));
end
```

where “D” is the main matrix, and “F_s” is the right side term.

4. Calculating the unknowns “U_s” of the supposed series from the Equation:

```
U_s = inv(D)*F_s'
```

5. Charting the results in two steps:

5.1.Reforming the supposed series “u_N” with no unknowns and just functioned to variable x, and here we form the Error Equation “delta”.

```
for q = 0 : N
  m = m + 1;
u_N = u_N+(1/factorial(q))U_s(m)(x − c)^q;
delta(m) =  double(Yex-u_N);
```

```
end
```

5.2. Substituting the variable x with values from 0 to 1 by step 0.01. So we can have points of the above Equations and we can chart them with horizontal axis from 0 to 1.

```
for z=0:0.01:1
i=i+1; xx(i)= z;
Ynn(i) = double(subs(u_N,x,z));
Yexx(i)= double(subs(u_e,x,z));
end
```

5.3. Start charting with “plot” command:

```
plot(xx,Ynn,'k')
gridon; hold all; plot(xx, u_e,'m')
%% Error Comparsion
delta = sqrt(abs(int(((u_e-u_N))^2,a,b)))
```

Table (3.1) shows the exact and numerical solutions together with the absolute error using Taylor collocation method for example (3.1).

Table (3.1): The exact and numerical solutions using Taylor collocation algorithm with *N=8*.

x	Exact solution $u_e = e^{-x}$	Numerical solution u_i	Absolute error $Error = \|u_e - u_i\|$
0	1.0000000000	1.000000001	1.1493e-12
0.05	0.9512294245	0.951229441	6.3456e-11
0.1	0.9048374180	0.904837423	1.9112e-11
0.15	0.8607079764	0.860707971	7.6528e-12
0.2	0.8187307531	0.818730745	7.2494e-12
0.25	0.7788007831	0.778800778	1.644e-12
0.3	0.7408182207	0.740818221	5.9044e-12
0.35	0.7046880897	0.704688093	5.7032e-13
0.4	0.6703200460	0.67032005	3.8858e-14
0.45	0.6376281516	0.637628154	2.4587e-12
0.5	0.6065306597	0.60653066	5.0651e-12
0.55	0.5769498104	0.576949809	4.7458e-12
0.6	0.5488116361	0.548811635	1.6814e-12
0.65	0.5220457768	0.522045777	8.3655e-14
0.7	0.4965853038	0.496585304	1.6814e-12
0.75	0.4723665527	0.472366552	4.0521e-12
0.8	0.4493289641	0.44932896	1.2962e-11
0.85	0.4274149319	0.427414926	1.4884e-11
0.9	0.4065696597	0.406569655	7.3973e-12
0.95	0.3867410235	0.386741024	4.7214e-11
1	0.3678794412	0.367879444	2.9537e-13

It can be observed that the maximum error is 1.64919e-08. Exact and numerical solutions are presented in figure 3.1(a). The absolute error is shown in figure 3.1(b).

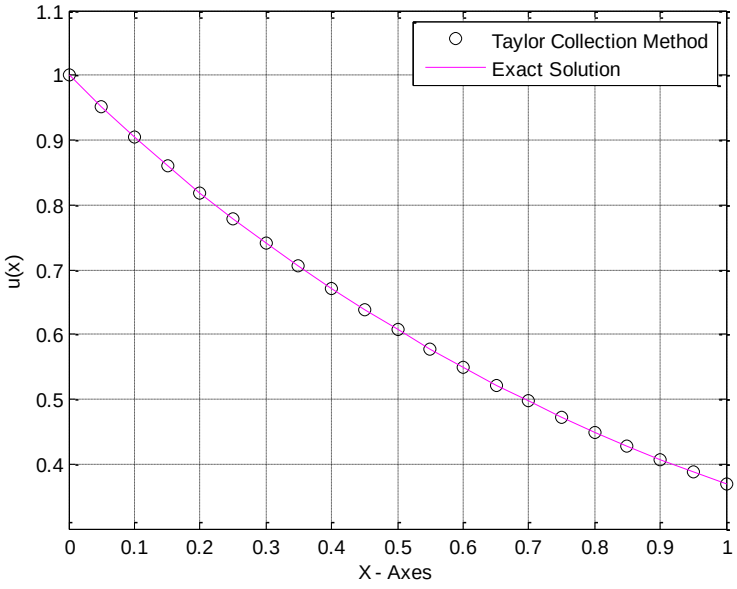


Figure 3.1(a) A comparison between the exact and *N* umerical solutions in example (3.1).

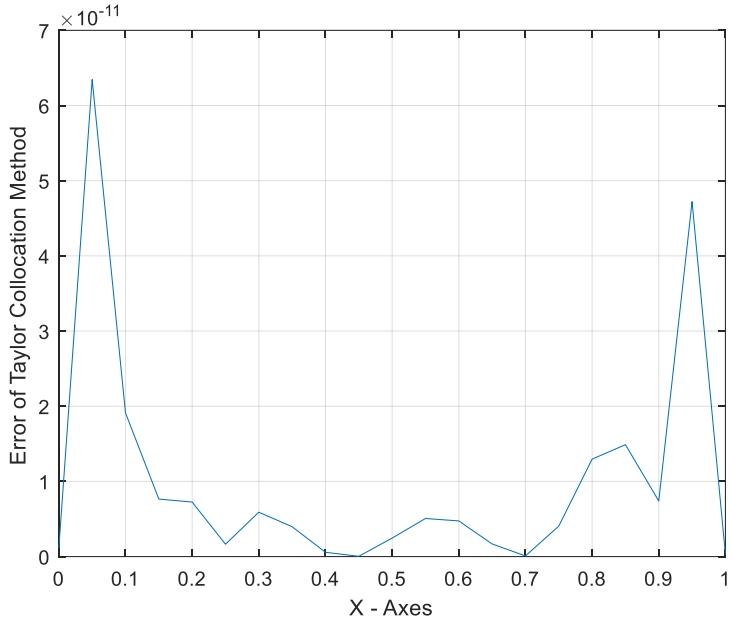


Figure 3.1(b) The absolute error for example (3.1).

Algorithm (3.2): Least Square Algorithm:

1. Input $a, b, c, N, L1, L2, A, B$

$where\ L1 = \lambda_1, L2 = \lambda_2., A = 1, B = 0$

2. Input the functions: $h, h_t, f, k1, k2, Y_{ex}$.

$$where\ h = h(x), h_t = h(x)|_{x=t}, Y_{ex} = exact\ solution.$$

3. Calculate $\varphi(x)$ and $\alpha(x)$, for number of points from 0 to N, using "For" loop:

```
phi(1:2) = [1 x];
if N >= 3
fori = 3:1:N+1
phi(i) = 2*x*phi(i-1)-phi(i-2);
end
end
for m = 0 : N
i = i + 1;
alpha(i) = A*phi(i) + B*subs(phi(i),x,h)...
    - L1*int(k1*subs(phi(i),x,t),t,a,h)...
    - L2*int(k2*subs(phi(i),x,ht),t,a,b);
end
```

4. Calculate the main matrix "G" and the right hand term "F":

```
fori = 1 : N+1
for j = 1 : N+1
```

```
    G(i,j)=double(int((alpha(j)*alpha(i)),x,a,b));
end
F(i) = double(int((f*alpha(i)),x,a,b));
end
```

5. Calculate the unknowns "C" of the supposed series $\varphi(x)$.

```
C = inv(G) * F'
```

6. Reforming the supposed series "u_x" so it haves no unknowns and functioned just to x. Also forming the error function:

```
fori = 1 : N+1
u_x = u_x + C(i) * phi(i);
delta(i) =  double(Yex-u_x);
end
```

7. Drawing after substituting x from 0 to 1 by step 0.01:

```
for j = 0.01:0.01:1
i=i+1; xx(i) = j; Y(i) = double(subs(u_N,x,j));
Yexx(i)= double(subs(Yex,x,j));
end
plot(xx,Y,'LineWidth',2); grid on; hold all
gridon; hold all; plot(xx,Yexx,'m') % True Chart
%% Error Comparsion
delta = sqrt(abs(int(((Yex-u_N))^2,a,b)))
```

Table (3.2) shows the exact and numerical solutions with the absolute error using least square method for example (3.1).

Table (3.2) : The exact and numerical solutions using Least Square algorithm with *N=8*.

x	Exact solution $u_e = e^{-x}$	Numerical solution u_i	Absolute error $Error = \|u_e - u_i\|$
0	1.0000000000	0.9999986150	1.385e-06
0.05	0.9512294245	0.9512297217	2.971e-07
0.1	0.9048374180	0.9048379569	5.389e-07
0.15	0.8607079764	0.8607081902	2.138e-07
0.2	0.8187307531	0.8187305773	1.758e-07
0.25	0.7788007831	0.7788003878	3.952e-07
0.3	0.7408182207	0.7408178303	3.903e-07
0.35	0.7046880897	0.7046878767	2.130e-07
0.4	0.6703200460	0.6703200818	3.580e-08
0.45	0.6376281516	0.6376284026	2.510e-07
0.5	0.6065306597	0.6065310141	3.544e-07
0.55	0.5769498104	0.5769501241	3.137e-07
0.6	0.5488116361	0.5488117836	1.475e-07
0.65	0.5220457768	0.5220456975	7.930e-08
0.7	0.4965853038	0.4965850309	2.728e-07
0.75	0.4723665527	0.4723662149	3.377e-07
0.8	0.4493289641	0.4493287489	2.151e-07
0.85	0.4274149319	0.4274150015	6.959e-08
0.9	0.4065696597	0.4065700090	3.490e-07
0.95	0.3867410235	0.3867412726	2.491e-07
1	0.3678794412	0.3678785520	8.891e-07

It can be observed that the maximum error is 1.385e-06. Exact and numerical solutions are illustrated in figure 3.2 (a). The absolute error is presented in figure 3.2(b).

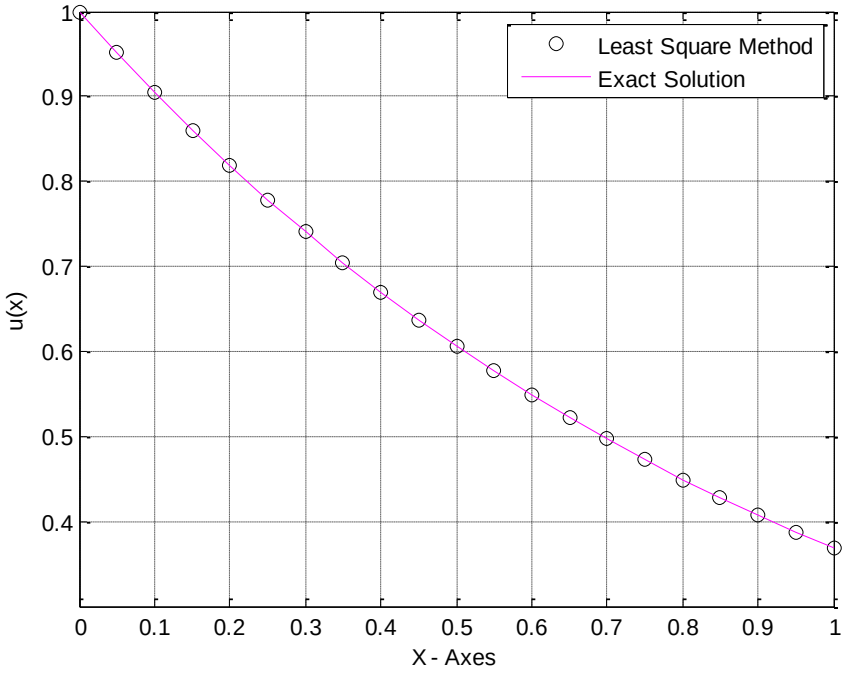


Figure 3.2(a) comparison between the exact and numerical solutions in example (3.1).

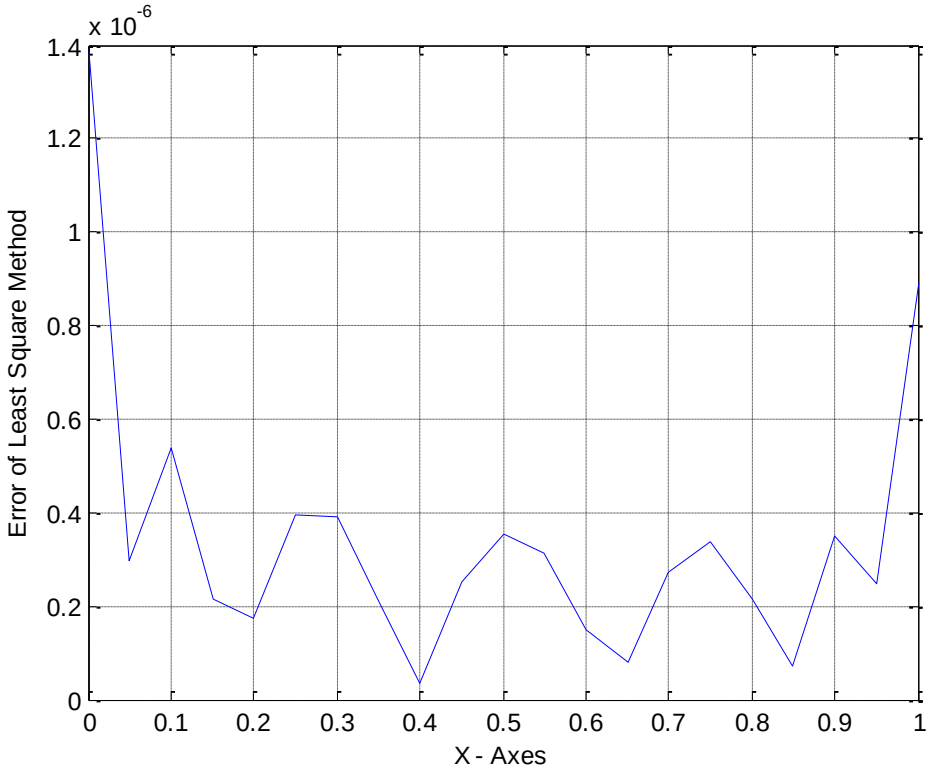


Figure 3.2(b) The absolute error for example (3.1).

Algorithm (3.3): Legendre Collocation Method:

1. Input $a, b, c, N, L1, L2, A, B$

where $L1 = \lambda_1, L2 = \lambda_2., A = 1, B = 0$

2. Input the functions: $h, h_t, f, k1, k2, Y_{ex}$.

where $h = h(x), h_t = h(x)|_{x=t}, Y_{ex} =$ exact solution.

3. Calculating X values, which the Legendre roots after substituting in the Equation $y = (x + 1)/2$. Then sorting the results from smaller to bigger. The programing has been done using "switch" loop.

4. Forming Legendre polynomial and shifted Legendre polynomial, for points from 1 to N:

```
L(1:2) = [1 x]; i=0;
for m = 1 : N0
i=i+1;
L(i+2) = simplify(((2*m+1)/(m+1))*x*L(i+1)-(m/(m+1))*L(i));
end

 % Shifted Legendre Polynomial
fori = 1 : N0+1
```

```
psi(i) = subs(L(i),x,(2*x-1));
end
```

5. Calculating the main matrix “g” and left hand term “F”, using two “For” loop:

```
fori = 1 : N0+1
for j = 1 : N0+1
    MAT = simplify(A*psi(i) + B*subs(psi(i),x,h)...
        - L1*int(k1*subs(psi(i),x,t),t,0,h)...
        - L2*int((k2*subs(psi(i),x,ht)),t,0,b));
g(i,j) = double(subs( MAT ,x,x1(j)));
end
F(i) = subs(f,x,x1(i));
end
```

6. Calculate the unknowns “C”:

```
c = inv(g')*F'
```

7. Reforming the shifted Legendre polynomial as function of x without unknowns:

```
fori = 1: N0+1
   y = y + c(i) * psi(i)
```

```
end
```

8. Drawing and showing the results:

```
for j = 0:0.01:1
i=i+1; xx(i) = j;  Y(i) = double(subs(y,x,j));
Yexx(i)= double(subs(Yex,x,j));
end
plot(xx,Y,'LineWidth',2) % ,'m'
gridon; hold all; plot(xx,Yexx,'m') % True Chart

%% Error Comparsion
delta = sqrt(abs(int(((Yex-y))^2,a,b)))
```

Table (3.3) shows the exact and numerical solutions with the absolute error using Legendre collocation method for example (3.1).

Table (3.3) The Exact and numerical solutions using Legendre collocation algorithm with *N=8*.

x	Exact solution $u_e = e^{-x}$	Numerical solution u_i	Absolute error $Error = \|u_e - u_i\|$
0	1.0000000000	0.999999999	1.28e-09
0.05	0.9512294245	0.951229425	5.20e-10
0.1	0.9048374180	0.904837418	2.28e-11
0.15	0.8607079764	0.860707976	3.87e-10
0.2	0.8187307531	0.818730753	2.71e-10
0.25	0.7788007831	0.778800783	2.86e-11
0.3	0.7408182207	0.740818221	3.31e-10
0.35	0.7046880897	0.70468809	3.02e-10
0.4	0.6703200460	0.670320046	6.05e-11
0.45	0.6376281516	0.637628151	2.11e-10
0.5	0.6065306597	0.606530659	3.33e-10
0.55	0.5769498104	0.57694981	2.32e-10
0.6	0.5488116361	0.548811636	2.56e-11
0.65	0.5220457768	0.522045777	2.68e-10
0.7	0.4965853038	0.496585304	3.17e-10
0.75	0.4723665527	0.472366553	1.10e-10
0.8	0.4493289641	0.449328964	2.16e-10
0.85	0.4274149319	0.427414932	3.43e-10
0.9	0.4065696597	0.40656966	1.08e-11
0.95	0.3867410235	0.386741024	4.27e-10
1	0.3678794412	0.36787944	1.01e-09

It can be observed that the maximum error is 1.28e-09. Exact and numerical solutions are illustrated in figure 3.3(a). Figure 3.3(b) shows the absolute error.

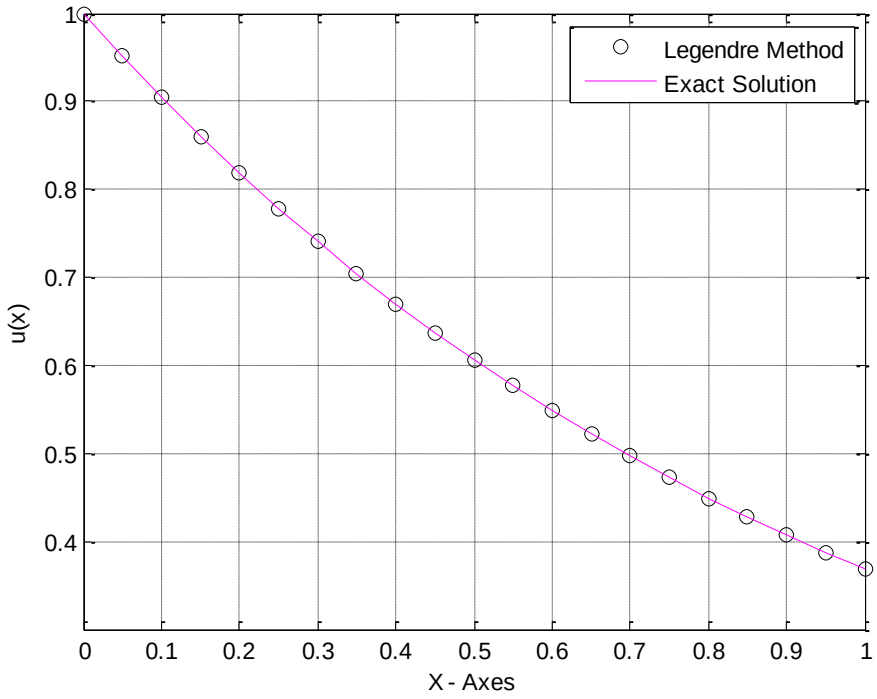


Figure 3.3(a) A comparison between the exact and numerical solutions in example (3.1).

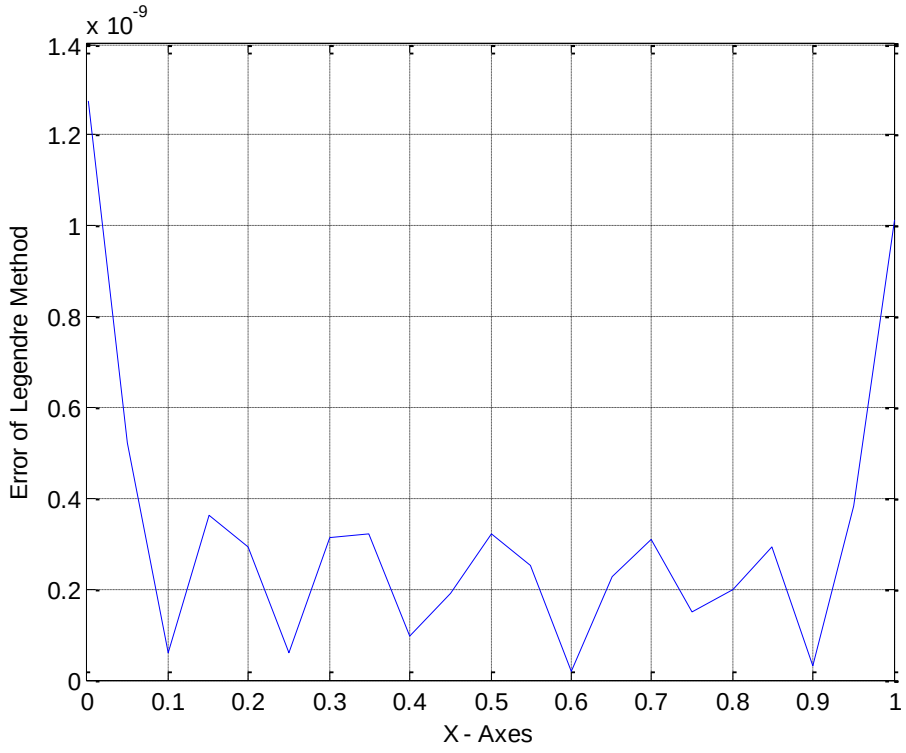


Figure 3.3 (b) The absolute error for example (3.1).

Algorithm (3.4): Lagrange Interpolating Algorithm:

The following algorithm explain how Lagrange method works, the code has been built in MATLAB software.

1.Defining the constants and the functions as in step 1-2 in Taylor collection method.

2. Calculating $x1$values based on the given N number from the following Equation:

```
j=0;j=0;
fori=0:N
 j=j+1;
 x1(j) = vpa(a+((b-a)/(N+1))*j);
end
```

3. Calculating the parameters P1 and P:

```
%%%%%% P & P1 Calculations %%%%%%%%%%
        p=1;p1=1;
for k = 1:N+1
                D = (x1(j)-x1(k));
if D ==0;
                D = 1 ;D1 = 1; p = p*D;
else
                x1(j); x1(k); D = (x1(j)-x1(k));
                D1=(t-x1(k));  p = p*D;
end
```

```
                p1 = p1*D1;
end
```

4.Computing the main matrix aijand the right hand side from the following code:

```
%%%%%% A Calculations %%%%%%%%%%%%%%
ifi==j
            A(i,j)=1-
(1/p)*(vpa(subs((int(k1*p1,t,a,x1(i))+int(k2*p1,t,
a,b)),x,x1(i))));
else
            A(i,j)= -
(1/p)*(vpa(subs((int(k1*p1,t,a,x1(i))+int(k2*p1,t,
a,b)),x,x1(i))));
end
%%%%%% Right Hand Side Calculations %
   B(i) = subs(f,x,x1(i));
```

5.Computing the set of Equations from the code:

```
C = vpa(inv(A) * B')
```

6. Computing the solution u as function of x:

```
u = 0;
for j=1:N+1
     p=1;p1=1;
```

```
for k = 1:N+1
        D = (x1(j)-x1(k));
if D ==0;
             D = 1 ;D1 = 1; p = p*D;
else
              x1(j); x1(k); D = (x1(j)-x1(k));
             D1=(x-x1(k));  p = p*D;
end
             p1 = p1*D1;
end
        u = vpa(u + C(j)*(p1/p));
end
```

6. Drawing and presenting tables after substituting x from 0 to 1 by step 0.05:

```
i=0;
for z=0.000001:0.05:1.000001
i=i+1; xx(i)= z;
Ynn(i) = double(subs(u,x,z));
Yexx(i)= double(subs(Yex,x,z));
En(i)  = abs(Yexx(i) - Ynn(i));
end
plot(xx,Ynn,'ok')                          % Numerical
Chart
grid on; hold all; plot(xx,Yexx,'m') % True Chart
```

```
xlabel('X - Axes'); ylabel('u(x)')
figure; plot([0.00001:0.05:1.00001],En)    % Error
Chart
grid on; xlabel('X - Axes'); ylabel('Error of
Lagrrange Method')
axis
%% Error Comparsion
delta = vpa(sqrt(abs(int(((Yex-u))^2,a,b))))
%%% Tables
disp(['No    ','Numerical Sol.    ','Analytical
Sol.    ','Abs. Error'])
fori = 1:length(Ynn)
disp([num2str(i),'     ',num2str(Ynn(i)),'
',num2str(Yexx(i)),'           ',num2str(En(i))])
end
```

7. The results will be on the main window of MATLAB software, and they are the main matrix and the right hand term, two plots show the answer and the error.

Table (3.4) shows the exact and numerical solutions with the absolute error using Lagrange interpolating method for example (3.1).

Table (3.4) The exact and numerical solutions using Lagrange algorithm with *N*=8

x	Exact solution $u_e = e^{-x}$	Numerical solution u_i	Absolute error $Error = \|u_e - u_i\|$
0	1.0000000000	0.999998998415113	1.58538659977836e-09
0.05	0.9512294245	0.951228472923512	3.48252648940672e-10
0.1	0.9048374180	0.904836513190301	8.69282423821005e-12
0.15	0.8607079764	0.860707115754011	3.64992480683668e-11
0.2	0.8187307531	0.818729934369757	2.21189733196070e-11
0.25	0.7788007831	0.778800004285080	1.40688571903524e-11
0.3	0.7408182207	0.740817479880306	1.64384061918099e-11
0.35	0.7046880897	0.704687385051773	2.07968087195809e-11
0.4	0.6703200460	0.670319375738746	2.28179697359110e-11
0.45	0.6376281516	0.637627514017329	2.33880692590560e-11
0.5	0.6065306597	0.606530053207141	2.48638887256902e-11
0.55	0.5769498104	0.576949233459045	2.80799827834244e-11
0.6	0.5488116361	0.548811087314638	3.19729798192725e-11
0.65	0.5220457768	0.522045254750744	3.52434748052133e-11
0.7	0.4965853038	0.496584807244596	3.82423537281795e-11
0.75	0.4723665527	0.472366080417891	4.31927826838319e-11
0.8	0.4493289641	0.449328514840360	5.18782794500794e-11
0.85	0.4274149319	0.427414504596010	6.20013484997628e-11
0.9	0.4065696597	0.406569253237621	6.64783228465637e-11
0.95	0.3867410235	0.386740636777622	6.39505670640972e-11
1	0.3678794412	0.367879073385887	9.37022681668509e-11

It can be observed that the maximum error is 1.58538659977836e-09. Exact and numerical solutions are illustrated in figure 3.4(a). Figure 3.4(b) shows the absolute error.

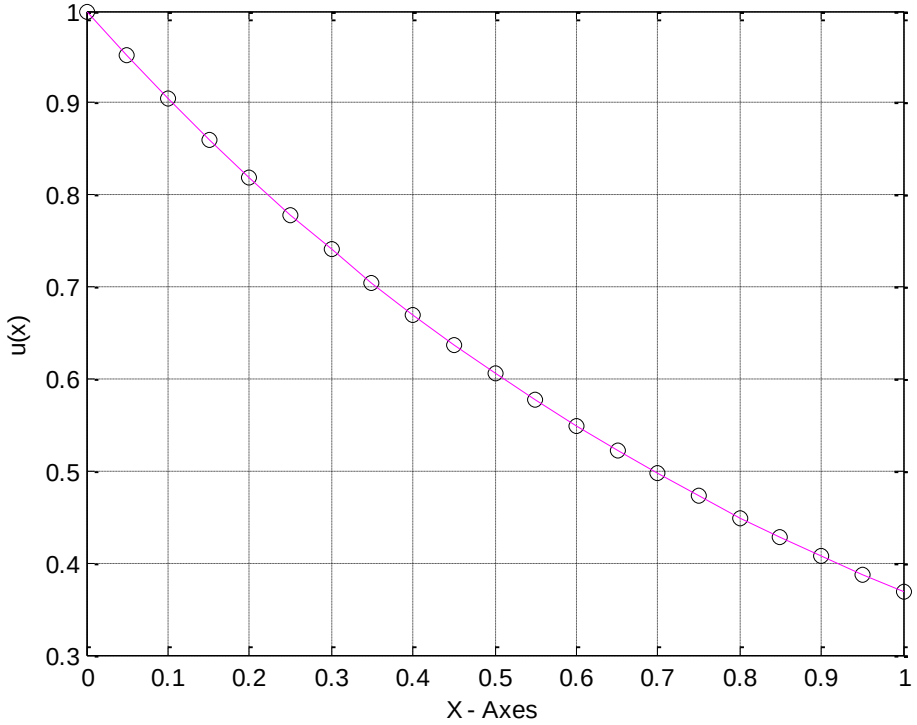


Figure 3.4(a) A comparison between the exact and numerical solutions in example (3.1)

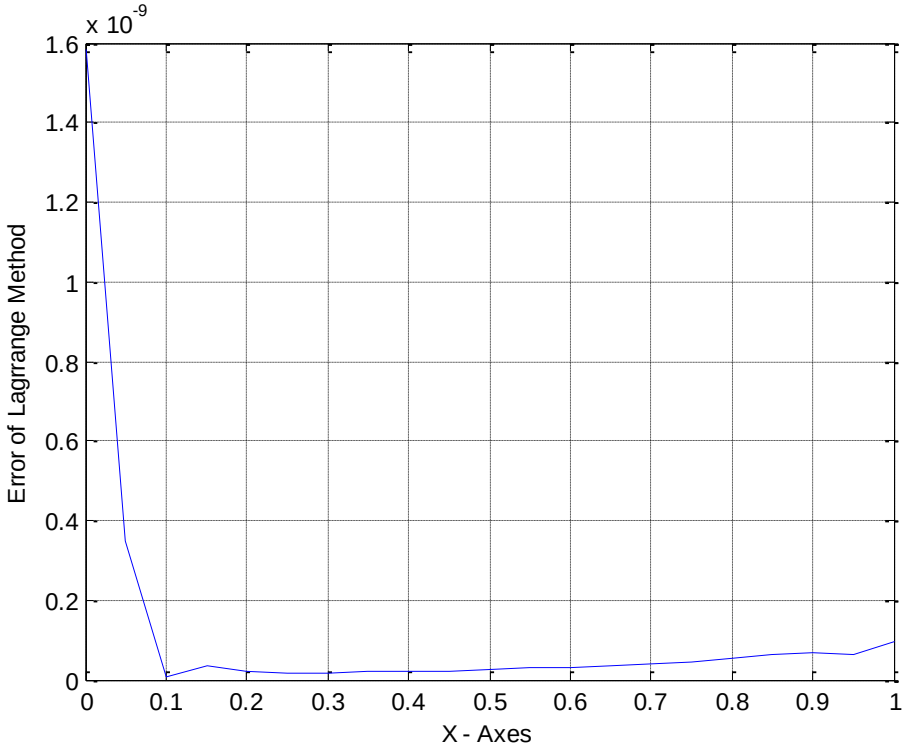


Figure 3.4(b) The absolute error for example (3.1)

Example (3.2):

Consider the Volterra-Fredholm integral equation of the second kind:

$$u(x) = \cos(x)\,.\,(0.5x - 0.25) + 0.2\cos(2 - x) + \int_0^x \sin(x - t)\,.\,u(t)dt$$

$$+ \int_0^1 \cos(x - t)\,.\,u(t)dt \qquad (3.3)$$

The exact solution of equation (3.3) [26] is:

$$u(x) = \sin(x). \qquad (3.4)$$

We solve equation (3.3) using similar techniques as for the aforementioned example (3.1), we have:

Taylor Collocation Method

Table (3.5) contains the exact and numerical solutions using Taylor collocation method together with the absolute error. Moreover, exact and numerical results are illustrated in figure 3.5(a). The absolute error is shown in figure 3.5(b).

Table (3.5) The exact and numerical solutions using Taylor collocation algorithm with *N*=8.

x	Exact solution $u_e = \sin(x)$	Numerical solution u_i	Absolute error $Error = \lvert u_e - u_i \rvert$
0	0	-2.97e-10	2.97159e-11
0.05	0.049979169	0.049979168	1.64477e-09
0.1	0.099833417	0.099833416	9.51358e-10
0.15	0.149438132	0.149438132	3.11233e-10
0.2	0.198669331	0.198669331	1.99256e-10
0.25	0.247403959	0.247403959	4.24601e-10
0.3	0.295520207	0.295520206	6.66514e-10
0.35	0.342897807	0.342897807	7.20205e-10
0.4	0.389418342	0.389418342	5.57071e-10
0.45	0.434965534	0.434965534	2.81471e-10
0.5	0.479425539	0.479425539	4.6162e-11
0.55	0.522687229	0.522687229	2.98348e-11
0.6	0.564642473	0.564642473	8.64604e-11
0.65	0.605186406	0.605186405	3.32128e-10
0.7	0.644217687	0.644217687	5.80692e-10
0.75	0.68163876	0.681638759	7.05117e-10
0.8	0.717356091	0.71735609	6.5027e-10
0.85	0.751280405	0.751280405	4.85661e-10
0.9	0.78332691	0.783326909	4.02727e-10
0.95	0.813415505	0.813415504	6.15982e-10
1	0.841470985	0.841470984	1.12334e-09

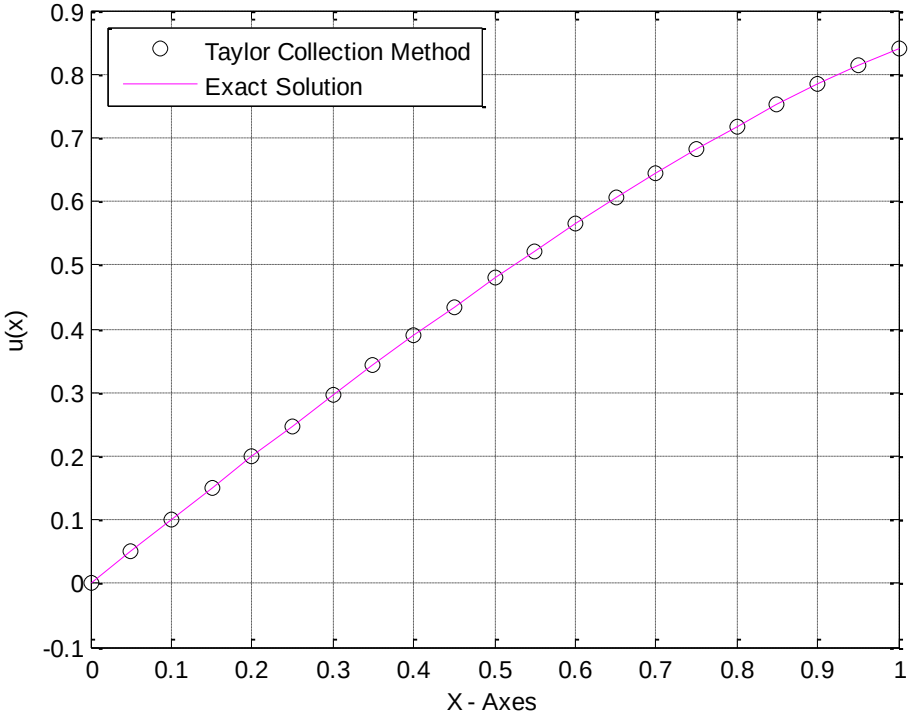


Figure 3.5(a) A comparison between the exact and numerical solutions in example (3.2).

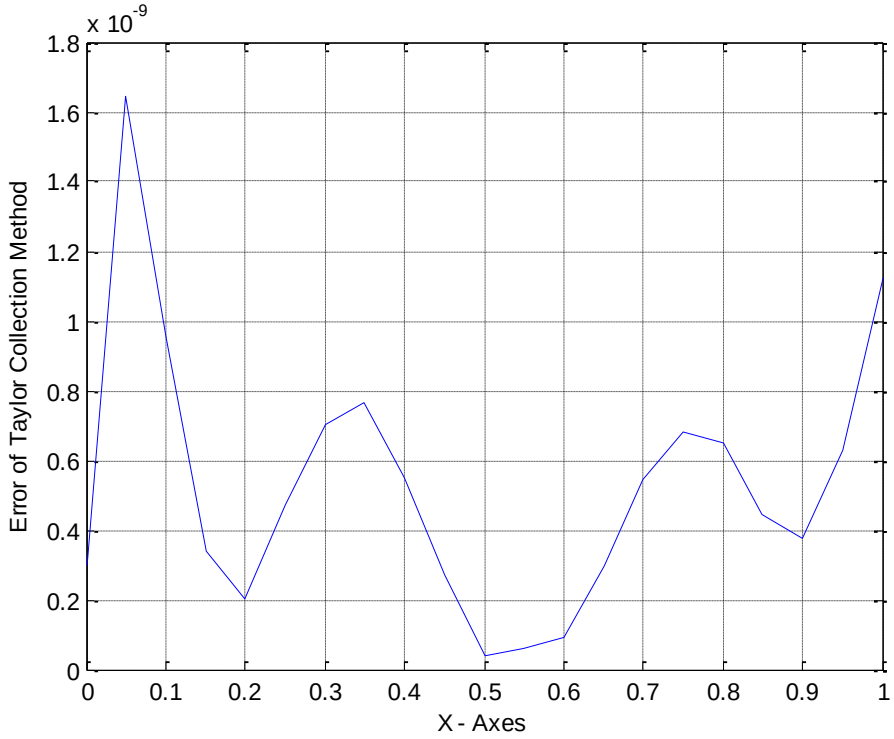


Figure 3.5(b) The Absolute error for example (3.2).

Least Square Method

Table (3.6) shows the exact and the numerical solutions with the absolute error using Least square method for example (3.2). Exact and numerical results are illustrated in figure 3.6(a), and the absolute error is presented in figure 3.6(b). It can be observed that the maximum error is 1.218e-04.

Table (3.6) The Exact and numerical solutions using least square algorithm with *N*=8

x	Exact solution $u_e = \sin(x)$	Numerical solution u_i	Absolute error $Error = \lvert u_e - u_i \rvert$
0	0	0.000121805	5.136e-8
0.05	0.049979169	0.0499344155	4.475e-10
0.1	0.099833417	0.0998042381	2.917e-10
0.15	0.149438132	0.1494563645	1.823e-10
0.2	0.198669331	0.1987085223	3.919e-09
0.25	0.247403959	0.2474310472	2.708e-09
0.3	0.295520207	0.2955183997	1.807e-09
0.35	0.342897807	0.3428705699	2.723e-10
0.4	0.389418342	0.3893827158	3.562e-10
0.45	0.434965534	0.4349413815	2.415e-10
0.5	0.479425539	0.4794256449	1.063e-10
0.55	0.522687229	0.5227115437	2.431e-10
0.6	0.564642473	0.5646781331	3.565e-10
0.65	0.605186406	0.6052135278	2.712e-09
0.7	0.644217687	0.6442192831	1.595e-09
0.75	0.68163876	0.6816114742	2.728e-09
0.8	0.717356091	0.7173168295	3.926e-09
0.85	0.751280405	0.7512622807	1.812e-10
0.9	0.78332691	0.7833562907	2.938e-10
0.95	0.813415505	0.8134603194	4.481e-10
1	0.841470985	0.8413488034	1.221e-08

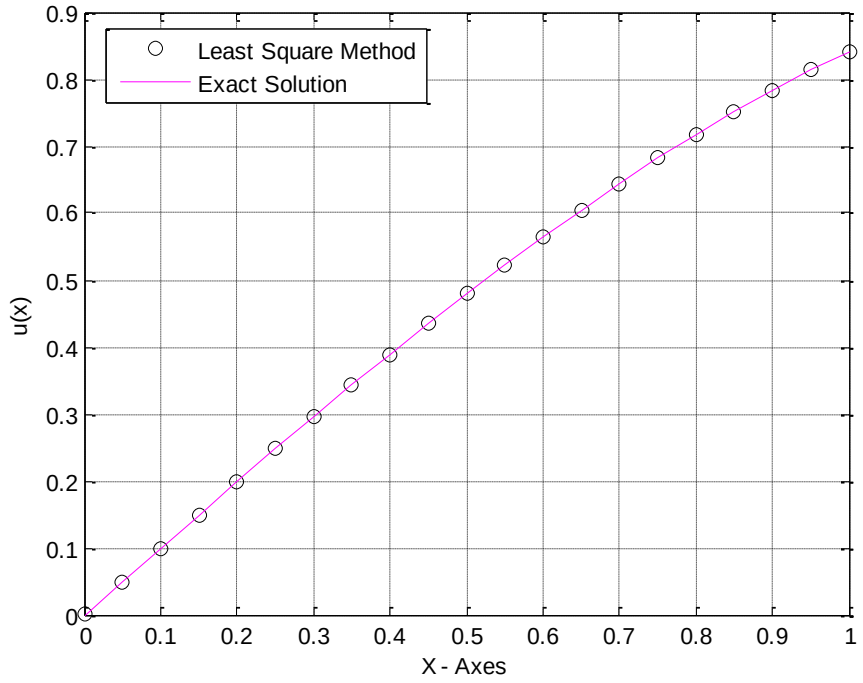


Figure 3.6(a) A comparison between exact and numerical solutions in example (3.2).

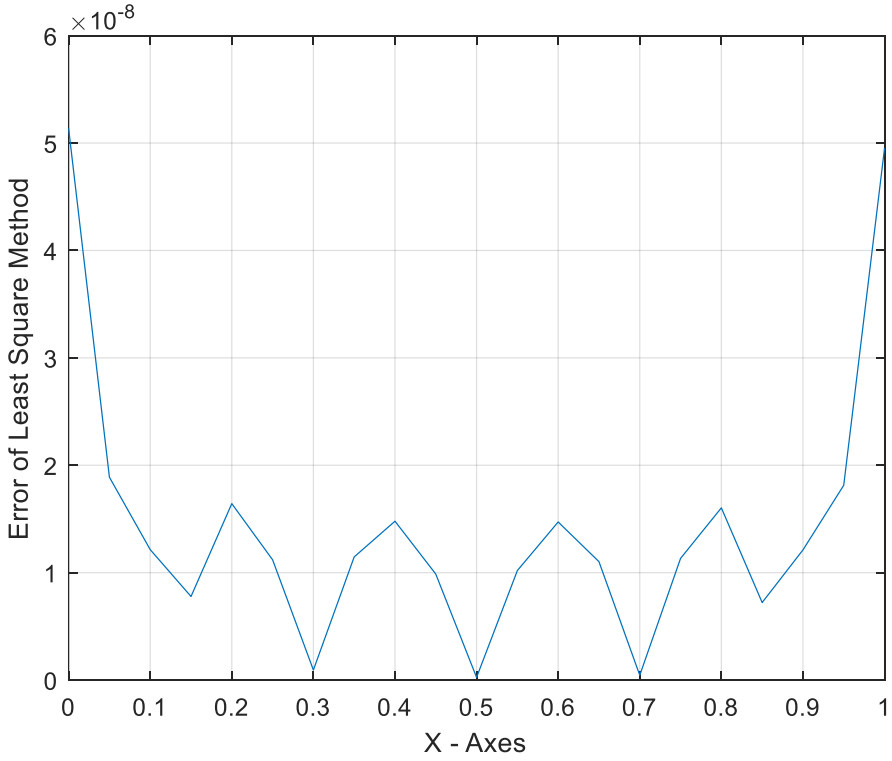


Figure 3.6(b) The absolute error for example (3.2).

Legendre Collocation Method

Table (3.7) shows the exact and the numerical solutions with the absolute error using Legendre collocation method for example (3.2). Exact and numerical solutions are illustrated in figure 3.7(a), and absolute error is presented in figure 3.7(b). It can be observed that the maximum error is 1.01148E-09.

Table (3.7) The exact and numerical solutions using Legendre algorithm with *N*=8.

x	Exact solution $u_e = \sin(x)$	Numerical solution u_i	Absolute error $Error = \lvert u_e - u_i \rvert$
0	0	-8.25E-10	8.24803e-10
0.05	0.049979169	0.04997917	3.41972e-10
0.1	0.099833417	0.099833417	1.53619e-11
0.15	0.149438132	0.149438132	2.60229e-10
0.2	0.198669331	0.198669331	1.82116e-10
0.25	0.247403959	0.247403959	6.49621e-11
0.3	0.295520207	0.295520207	2.36087e-10
0.35	0.342897807	0.342897808	2.13921e-10
0.4	0.389418342	0.389418342	3.67945e-11
0.45	0.434965534	0.434965534	1.6268e-10
0.5	0.479425539	0.479425538	2.49878e-10
0.55	0.522687229	0.522687229	1.66028e-10
0.6	0.564642473	0.564642473	3.83317e-11
0.65	0.605186406	0.605186406	2.27439e-10
0.7	0.644217687	0.644217687	2.56228e-10
0.75	0.68163876	0.68163876	7.20959e-11
0.8	0.717356091	0.717356091	2.056e-10
0.85	0.751280405	0.751280405	2.99939e-10
0.9	0.78332691	0.78332691	1.83036e-11
0.95	0.813415505	0.813415505	4.11195e-10
1	0.841470985	0.841470984	1.01148e-09

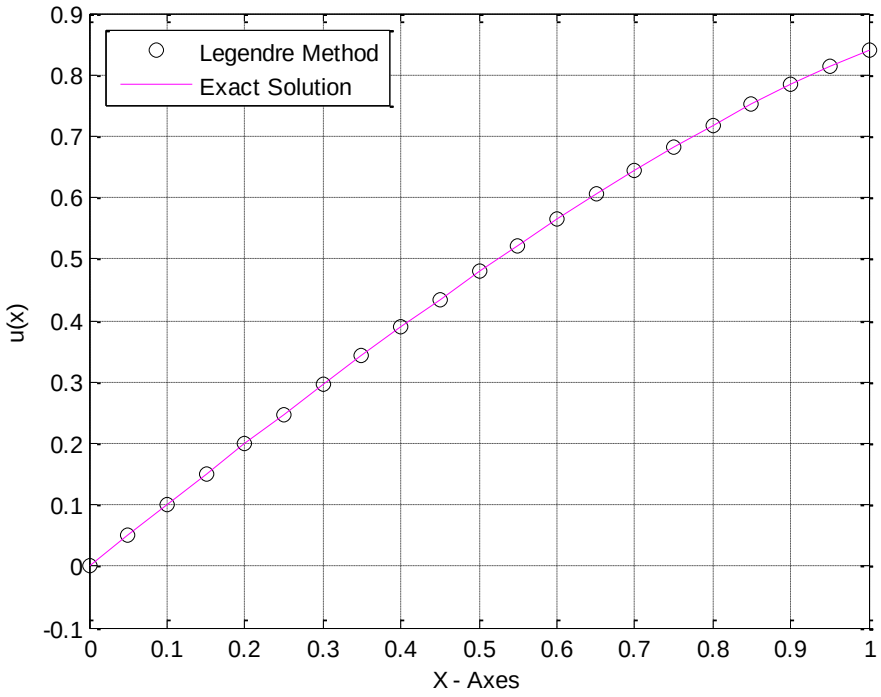


Figure 3.7(a) A comparison between exact and numerical solutions in example (3.2).

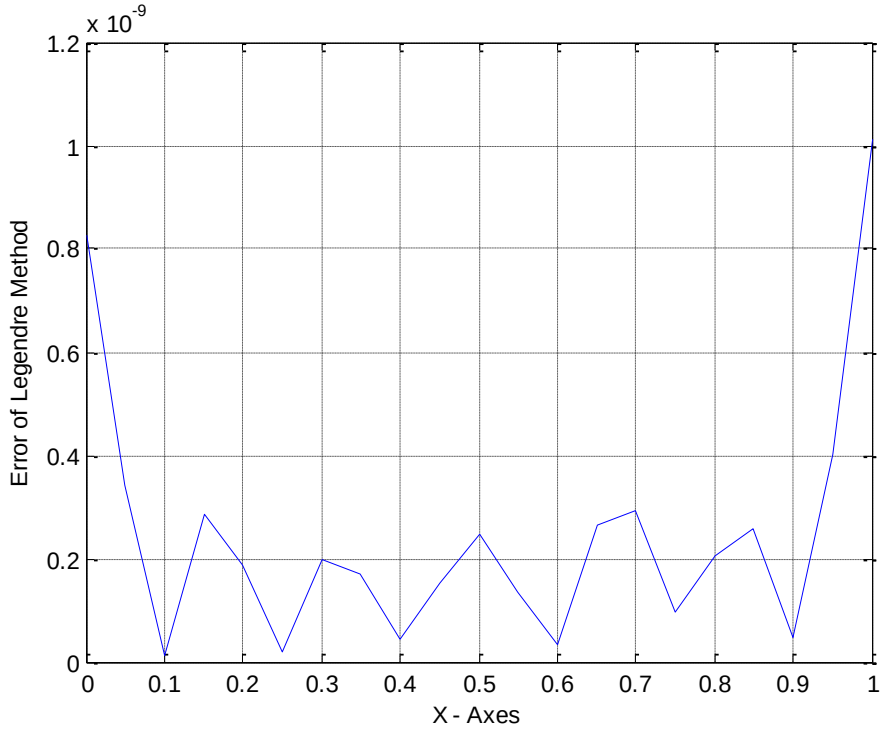


Figure 3.7(b) The absolute error for example (3.2).

Lagrange Interpolating Method

Table (3.8) shows the exact and the numerical solutions with the absolute error using Legendre collocation method for example (3.2). Exact and numerical solutions are illustrated in figure 3.8(a), and absolute error is presented in figure 3.8(b). It can be observed that the maximum error is 1.01148E-09.

Table (3.8) The exact and numerical solutions using Lagrange algorithm with *N*=8

x	Exact solution $u_e = \sin(x)$	Numerical solution u_i	Absolute error Error $= \lvert u_e - u_i \rvert$
0	0	1.00107436604573e-06	1.841471523259336e-09
0.05	0.049979169	0.0499801673184054	7.02508315408856e-10
0.1	0.099833417	0.0998344104369503	1.21399323926852e-09
0.15	0.149438132	0.149439119934744	1.30985897128788e-09
0.2	0.198669331	0.198670309539945	1.32159444499180e-09
0.25	0.247403959	0.247404926824540	1.34228067549813e-09
0.3	0.295520207	0.295521160620143	1.37753819462461e-09
0.35	0.342897807	0.342898745412693	1.41530026587233e-09
0.4	0.389418342	0.389419261920050	1.44940026647333e-09
0.45	0.434965534	0.434966433077015	1.48110002040625e-09
0.5	0.479425539	0.479426414672822	1.51370321832545e-09
0.55	0.522687229	0.522688079906655	1.54826451659318e-09
0.6	0.564642473	0.564643297147283	1.58308499642601e-09
0.65	0.605186406	0.605187200203389	1.61614621685402e-09
0.7	0.644217687	0.644218450431729	1.64782698597321e-09
0.75	0.68163876	0.681639490030728	1.68113456489039e-09
0.8	0.717356091	0.717356785887677	1.71819669603224e-09
0.85	0.751280405	0.751281063368194	1.75486913889245e-09
0.9	0.78332691	0.783327529457122	1.77993808581078e-09
0.95	0.813415505	0.813416084680574	1.79148273993235e-09
1	0.841470985	0.841471523259336	1.85044579659177e-09

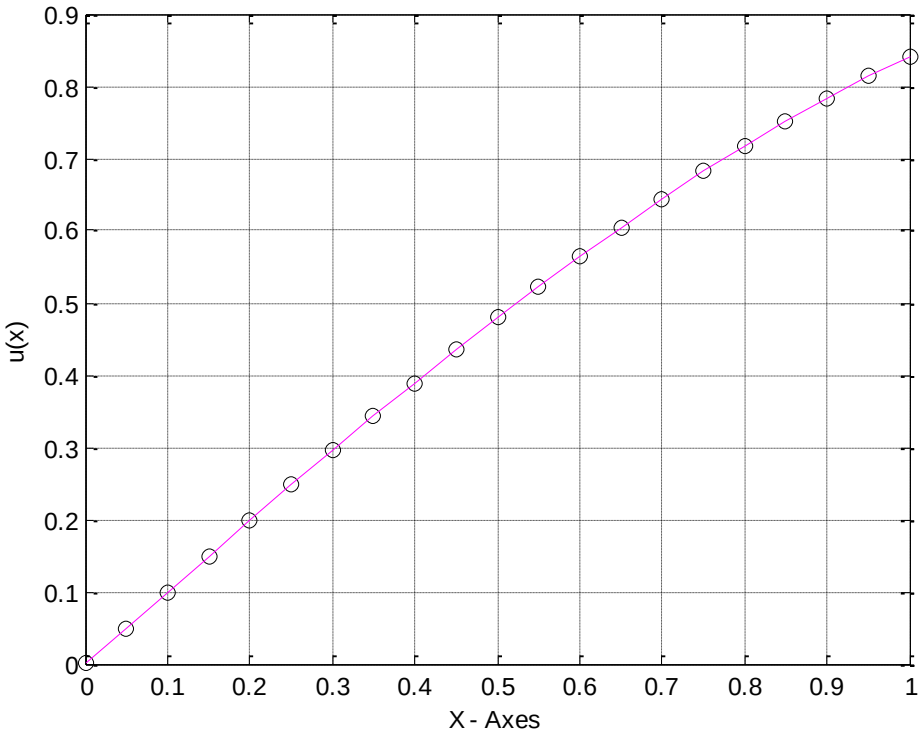


Figure 3.8(a) A comparison between the exact and numerical solutions in example (3.2)

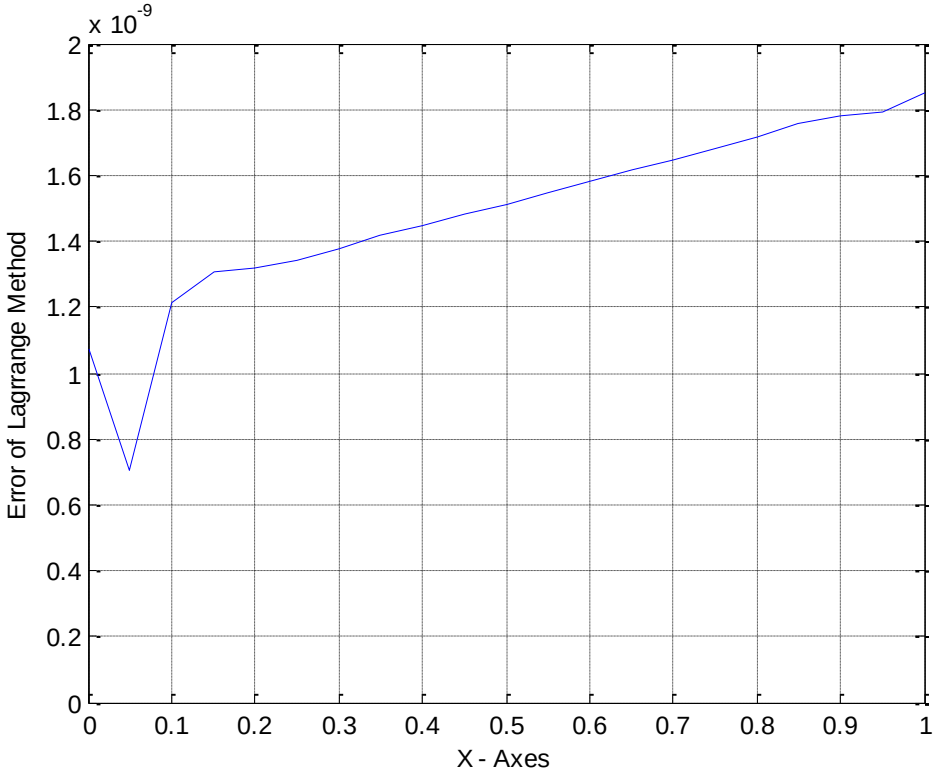


Figure 3.8(b) The absolute error for example (3.2).

Conclusion

In this thesis we have solved Volterra-Fredholm integral equations of the second kind using various numerical methods. These numerical methods were implemented in a form of algorithms to solve some numerical test cases using Matlab software by the processor AMD E2-9010 RADEON R2, 4 COMPUTE CORES 2C+2G. A comparison between these numerical techniques for example (3.1) is shown in the following table:

	Taylor Collocation method		Legendre collocation method		Least Square method		Lagrange interpolation method	
	Maximum error	Time (sec)	Maximum error	Time (sec)	Maximum error	Time (sec)	Maximum error	Time (sec)
N=6	4.61e-8	8.45	1e-6	20.22	3e-04	219.43	7.47e-7	45
N=7	1.681e-9	10.83	3.86e-8	20.97	1.7e-05	465.57	3.67e-8	105.72
N=8	6.34e-11	14.77	1.2e-9	27.25	1.3e-06	694.7	1.5e-9	359

And a comparison between these numerical techniques for example (3.2) is shown in the following table:

	Taylor Collocation method		Legendre collocation method		Least Square method		Lagrange interpolation method	
	Maximum error	Time (sec)	Maximum error	Time (sec)	Maximum error	Time (sec)	Maximum error	Time (sec)
N=6	1.051 e-7	19.17	8.061e-7	22.43	3.432e-6	414.4	7.621e-6	100.23
N=7	3.928e-9	22.63	5.204e-8	28.62	7.521e-7	470	5.416e-8	197.25
N=8	1.351e-10	30.31	1.0114e-9	38.13	5.136e-8	503	1.804e-9	600.78

These two table show that as *N* increases the error decreases and the Taylor collocation method is one of the most powerful and accurate numerical methods for solving Volterra-Fredholm integral equations of the second kind

in comparison with other numerical methods. Also, the Taylor collocation method has shown to be one of the fastest methods for solving Volterra-Fredholm integral equations of the second kind.

References

[1] M. Abdoue, **On Asymptotic Methods for Fredholm-Volterra Integral Equation of the Second Kind in Contact Problems**, J. Comp. Appl. Math. 154(2003), 431-446.

[2] M. Abdou and F. Salama , **Volterra-Fredhdm Integral Equation of the First Kind and Spectral Relationships**, Appl. Math. Comput. 153 (2004), 141-153 .

[3] H. Adibi and A. Rismani**, Numerical Solution to a Functional Integral Equations Using the Legendre-Spectral Method**, Australian Journal of Basic and Applied Sciences, 4(3) (2010), 481-486.

[4] B. Ambreen, K. Abid, U. Hayat and S. Mohyud-Ain, **New Iterative Method for Time-Fractional Schrondinger Equation**, World Journal of Modeling and Simulation, (2013), 9: 89-95.

[5] R. Bentez and V. Bols, **Existence and Uniqueness of Nontrivial Collocation Solutions of Implicitly Linear Homogeneous Volterra Integral Equations**, Journal of Computational and Applied Mathematics, 235 (2011) 3661–3672.

[6] S. Bhaleker and V. Dafterder-gejji, **Solving Fractional-Order Logistic Equation Using a New Iterative Method**. Hindawi Publishing Corporation, International Journal of Differential Equations, (2012), 20: 12-16.

[7] R. Burden and J. Faires, **Numerical Analysis**, Cengage Learning, Canada, (2010).

[8] F. Calio, E. Marchetti and V. Muresan, **On Some Volterra-Fredholm Integral Equations**, International Journal of Pure and Applied Mathematics, (2006), 174-184.

[9] C. Chokri, **On The Numerical Solution of the Volterra-Fredholm Integral Equations with Abel Kernel Using Legendre Polynomials**, International Journal of Advanced Scientific and Technical Research, Vol.1 No.3, (2013), 2249-9954.

[10] C. Consterda, **Integral Equation of the First Kind in Plane Elasticity**, J. Quort. Appl. Math., Vol. LIII, No.4, (1995), 783-791.

[11] V. Dafterder-Gejji and H. Jafari, **An Iterative Method for Solving Nonlinear Functional Equations**, Journal of Mathematical Analysis and Applications, 316 (2006) 753-763.

[12] V. Dafterder-Gejji and S. Bhaleker, **Solving Fractional Diffussion-Wave Equations Using New Iterative Method**, Fractional calculus and Applied Analysis: International Journal of Theory and Applications, (2008), 11: 11311-1454.

[13] M. Gülsu and M. Sezer, **A Taylor Collocation Method for the Approximate Solution of General Linear Fredholm–Volterra integro-Difference Equations with Mixed Argument**, International Journal of Computer Mathematics, 175 (2006), 675–690.

[14] A. Hamada, **New Iterative Method: An Application for Solving Fractional Physical Differential Equation**, Hindawi Publishing Corporation, (2012), 13: 23-27.

[15] H. Ibrahim, F. Attah and G. Gyegwe, **On the Solution of Volterra-Fredholm and Mixed Volterra-Fredholm Integral Equations Using the New Iterative Method**, Applied Mathematics, (2016), 6(1): 1-5.

[16] H. Ibrahim and P.V. Ayoo, **Approximation of Systems of Volterra Integro-Differential Equations Using the New Iterative Method**. International Journal of Science and Research, Vol.4, No.5, (2013), 2319-7064.

[17] R. Kanwal, **Linear Integral Equations Theory and Techniques**, Academic Press, (1971).

[18] PG. Kauthen, **Continuous Time Collocation Methods for Voltera-Fredholm Integral Equations**, Numer Math, Vol. 56, No.5, (1989), 409-424.

[19] H. Kocak and A. Yildrin, **An Efficent NIM for Finding Exact Solutions of Nonlinear Time-Fractional Partial Differential Equations**, Nonlinear Analysis, Modeling and Control, 16 (2011), 403-414.

[20] A. Lowan, N. Davids and A. Levenson, **Table of the Zeros of the Legendre Polynomials of Order 1-16 and the Weight Cooficients for Gauss Mechanical Quadrature Formula**, New York, National Bureau of standards, (1941).

[21] S. Majeed and H. Omran, **Numerical Methods for Solving Linear Volterra-Fredholm Integral Equations**, Journal of Al-Nahrain University, 11(3) (2008), 131-134.

[22] K. Maleknejad and F. Yami, **A Computational Method for System of Volterra-Fredholm Integral Equations**, Appl. Math. Comput., 188 (2006), 589-595.

[23] K. Maleknejad and S. Sohrabi, **Legendre Polynomial Solution of Nonlinear Volterra-Fredholm Integral Equations,** Iust International Journal of Engineering Science, Vol. 19, No. 1-5, (2008), 49-52.

[24] F. Mirzaee and E. Hadadiyan, **Approximate Solutions for Mixed Nonlinear Volterra-Fredholm Type Integral Equations Via Modified Block-Pulse Functions**, Journal of the Association of Arab Universities for Basic and Applied Sciences, 12 (2012), 65-73.

[25] M. Muskhelishvili, **Some Basic Problems of Mathematical Theory Elasticity**, Noordhoff, Holland, (1953).

[26] M. Mustafa and I. Ghanim**, Numerical Solution of Linear Volterra-Fredholm Integral Equations Using Lagrange Polynomials**, Mathematical Theory and Modeling, Vol.4, 5 (2014).

[27] S. Nemati**, Numerical Solution of Volterra-Fredholm Integral Equations Using Legendre Collocation Method**, Journal of Computational and Applied Mathematics, 278 (2015), 29-36.

[28] M. Ramadan and M. Al-luhaibi, **NIM for Solving the Fornberg-Withan Equation and Comparison With the Homotopy Perturbation Transform Method**, British Journal of Mathematics and Computer Science, Vol.4, No.9, (2014), 1231-1227.

[29] I. Rus, **Metrical Fixed Point Theorems**, Univ. of Cluj-Napoca, (1979).

[30] I. Rus, **Picard Mappings: Results and Problems, Seminar on Fixed Point Theory, Preprint,** Cluj-Napoca, Babe¸s-Bolyai Univ., 6 (1987), 55-64.

[31] V. Srivastava and K. Rai, **A Multi-Term Fractional Diffusion Equation for Oxygen Delivery Through a Capillary to Tissues**. Mathematical and Computer Modeling, 5 (2010), 616-624.

[32] E. Suli and D. Mayers, **An Introduction to Numerical Analysis**, Cambridge University Press, (2003).

[33] G. Targonski, **On Carleman Integral Operators**, Proc. Amer. Math. Soc., (1967), 450-456.

[34] Q. Wang, K. Wang and S. Chen**, Least Squares Approximation Method for the Solution of Volterra-Fredholm Integral Equations**, Journal of Computational and Applied Mathematics, 272 (2014), 141-147.

[35] K. Wang and Q. Wang**, Taylor Collocation Method and Convergence Analysis for the Volterra-Fredholm Integral Equations**, Journal of Computational and Applied Mathematics, 260 (2014), 294-300.

[36] A. Wazwaz, **A First Course in Integral Equations**. World Scientific Publishing Co. Pte. Ltd., (1997).

[37] A. Wazwaz**, Linear and Nonlinear Integral Equations: Methods and Applications**, Springer Heidelberg, Dordrecht London, (2011).

[38] S. Yalsinbas, **Taylor Polynomial Solution of Nonlinear Volterra-Fredholm Integral Equations**, Appl. Math. Comput., 127 (2002), 195-206.

[39] M. Yaseen, M. Samraig and S, Naheed, **The DI Method for Exact Solutions of Laplace Equation**, International Journal of Mathematical Physics, (2012), 12: 1208-3350

[40] S. Yousefi and S. M. Razzaghi, **Legendre Wavelets Method for Nonlinear Volterra-Fredholm Integral Equations**, Math. Comput. Simul, 70 (2005), 1-8.

[41] E. Yusufoglu and E. Erbas, **Numerical Expansion Methods for Solving Volterra-Fredholm Type Linear Integral Equations by Interpolation and Quadrature Rules**, Kybernetes 37 (6), (2008), 768-785.

Appendix

MATLAB code for Taylor Collocation Method

```
clc
clear all
% close all

%% Inputs

% %% Example(1)
% % Constant Inouts
% a = 0; b = 1; N = 8;
% L1= 1; L2= 1;
% c=0;
% % Functional Inputs
% syms t x
% A = 1; B = 0;
% h = x; ht = t;
% f = x - 2*exp(x) + exp(-x) + 1;
% k1=t*exp(x); k2=exp(x+t);
% Yex = exp(-x); % Exact Solution

%% Example(2)
% Constant Inouts
a = 0; b = 1; N = 8;
```

```
L1= 1; L2= 1;
c=0;
% Functional Inputs
syms t x
A = 1; B = 0;
h = x; ht = t;
f = cos(x)*(0.5*x - 0.25) + 0.25*cos(2-x);
k1=sin(x-t); k2=cos(x-t);
Yex = sin(x); % Exact Solution


tic
%% Programing
for n = 0:N
    x1(n+1) = vpa(a+((b-a)/(N))*n);
    for m=0:N
        U = A*(x-c)^m + B*(h-c)^m;
        V = int(k1*(t-c)^m,t,a,h);
        F = int(k2*(ht-c)^m,t,a,b);
        D(n+1,m+1)= subs((1/factorial(m))*(U -
L1*V - L2*F),x,x1(n+1));
    end
    F_s(n+1) = subs(f,x,x1(n+1));
end
```

```
U_s = inv(D)*F_s'
toc

%% Ploting the Results
u_N = 0; m=0; en=0;
for q = 0 : N
    m = m + 1;
    u_N = u_N +(1/factorial(q))*U_s(m)*(x-c)^q;
%     delta(m) =  double(Yex-u_N);
end
% delta =  double(Yex-u_N);
i=0;
for z=0:0.05:1
    i=i+1; xx(i)= z;
    Ynn(i) = double(subs(u_N,x,z));
    Yexx(i)= double(subs(Yex,x,z));
    En(i)  = abs(Yexx(i) - Ynn(i));
end
plot(xx,Ynn,'ok')                          % Numerical
Chart
grid on; hold all; plot(xx,Yexx,'m') % True Chart
xlabel('X - Axes'); ylabel('u(x)')
figure; plot([0:0.05:1],En)    % Error Chart
```

```
grid on; xlabel('X - Axes'); ylabel('Error of
Taylor Collocation Method')
%% Error Comparsion
delta = sqrt(abs(int(((Yex-u_N))^2,a,b)))
```

MATLAB code for Least Square Approximation method

```
%% Least Square Method
clc
clear all
% close all
% %% Example(5)
% % Constant Inouts
% a = 0; b = 1; N = 8;
% L1= 1; L2= 1;
% c=0;
% % Functional Inputs
% syms t x
% A = 1; B = 0;
% h = x; ht = t;
% f = x - 2*exp(x) + exp(-x) + 1;
% k1=t*exp(x); k2=exp(x+t);
% Yex = exp(-x); % Exact Solution



%% Example(6)
% Constant Inouts
a = 0; b = 1; N = 8;
L1= 1; L2= 1;
c=0;
```

```
% Functional Inputs
syms t x
A = 1; B = 0;
h = x; ht = t;
f = cos(x)*(0.5*x - 0.25) + 0.25*cos(2-x);
k1=sin(x-t); k2=cos(x-t);
Yex = sin(x); % Exact Solution

% i=0;
% for m = 0 : N
%     i = i + 1;
%     phi(i) = (1/factorial(m))*(x-c)^m;
% end

% Chepyshev
T(1:2) = [1 x];
if N >=3
for i = 3:1:N+1
    T(i) = 2*x*T(i-1)-T(i-2);
end
end
% shfted Chpyshev
Cheb = subs(T,x,2*x-1);
```

```
tic
%% Programing
i=0;
for m = 0 : N
    i = i + 1;
    alpha(i) = A*Cheb(i) + B*subs(Cheb(i),x,h) -
L1*int(k1*subs(Cheb(i),x,t),t,a,h)...
                                                -
L2*int(k2*subs(Cheb(i),x,ht),t,a,b);
end

for i = 1 : N+1
    for j = 1 : N+1
        G(i,j) = double(int((alpha(j) *
alpha(i)),x,a,b));
    end
        F(i) = double(int((f*alpha(i)),x,a,b));
end
G
F

% Full Matrix
C = inv(G) * F'
toc
```

```
% T1 = 0;
% for i = 1 : N+1
%     T1 = T1 + C(i)*alpha(i);
% end
% T = T1 - f;
% I = int(T^2,x1,a,b)

% forming Tylor Series
u_x = 0;
for i = 1 : N+1
    u_x = u_x + C(i) * Cheb(i);
    delta(i) =  double(sqrt(int(((Yex-
u_x))^2,a,b)));

end
u_N = simplify(u_x)
toc

% Drawing
i=0;
for j = 0:0.05:1
    i=i+1; xx(i) = j; Y(i) =
double(subs(u_N,x,j));
    Yexx(i)= double(subs(Yex,x,j));
```

```
    En(i)  = abs(Yexx(i) - Y(i));
end
plot(xx,Y,'ok') % ,'m'  'LineWidth',2
grid on; hold all
grid on; hold all; plot(xx,Yexx,'m') % True Chart
xlabel('X - Axes'); ylabel('u(x)')
figure; plot([0:0.05:1],En)    % Error Chart
grid on; xlabel('X - Axes'); ylabel('Error of
Least Square Method')

%% Error Comparsion
delta = sqrt(abs(int(((Yex-u_N))^2,a,b)))
```

MATLAB code for Legendre Collocation Method

```
clc
clear all
% close all



% % Example(5)
% % Constant Inouts
% a = 0; b = 1;N = 8; N0 = N-1;
% L1= 1; L2= 1;
% c=0;
% % Functional Inputs
% syms t x
% A = 1; B = 0;
% h = x; hc=c; ht=t;
% f = x - 2*exp(x) + exp(-x) + 1;
% k1=t*exp(x); k2=exp(x+t);
% Yex = exp(-x); % Exact Solution


%% Example(6)
% Constant Inouts
a = 0; b = 1; N = 8; N0 = N-1;
L1= 1; L2= 1;
c=0;
```

```
% Functional Inputs
syms t x
A = 1; B = 0;
h = x; hc = c; ht = t;
f = cos(x)*(0.5*x - 0.25) + 0.25*cos(2-x);
k1=sin(x-t); k2=cos(x-t);
Yex = sin(x); % Exact Solution

%% X values
switch N0
    case  1
        x1 = sort(subs((x+1)/2,[0.5773 -0.5773]));
    case  2
        x1 = sort(subs((x+1)/2,[0 0.7745 -
0.7745]));
    case  3
        x1 = sort(subs((x+1)/2,[0.3399 -0.3399
0.8611 -0.8611]));
    case  4
        x1 = sort(subs((x+1)/2,[0 0.5384 -0.5384
0.9061 -0.9061]));
    case  5
        x1 = sort(subs((x+1)/2,[0.2386 -0.2386
0.6612 -0.6612 0.9324 -0.9324]));
```

```
    case  6
        x1 = sort(subs((x+1)/2,[0 0.4058 -0.4058
0.7415 -0.7415 0.9491 -0.9491]));
    case  7
        x1 = sort(subs((x+1)/2,[0.1834 -0.1834
0.5255 -0.5255 0.7966 -0.7966 0.9602 -0.9602]));
end

tic
%% Legendre Polynomial:
L(1:2) = [1 x]; i=0;
for m = 1 : N0
    i=i+1;
    L(i+2) = simplify(((2*m+1)/(m+1))*x*L(i+1)-
(m/(m+1))*L(i));
end
% Shifted Legendre Polynomial
for i = 1 : N0+1
    psi(i) = subs(L(i),x,(2*x-1));
end


%% G & F:
for i = 1 : N0+1
```

```
    for j = 1 : N0+1
      MAT = simplify(A*psi(i) + B*subs(psi(i),x,h)
- L1*int(k1*subs(psi(i),x,t),t,0,h)...

- L2*int((k2*subs(psi(i),x,ht)),t,0,b));
      g(i,j) = double(subs( MAT ,x,x1(j)));
    end
      F(i) = subs(f,x,x1(i));
end

c = inv(g')*F'

y=0;
for i = 1: N0+1
    y = y + c(i) * psi(i);
end
y = simplify(y)
toc

%% Drawing
i=0;
for j = 0:0.05:1
    i=i+1; xx(i) = j;  Y(i) = double(subs(y,x,j));
    Yexx(i)= double(subs(Yex,x,j));
```

```
    En(i)  = abs(Yexx(i) - Y(i));
end
plot(xx,Y,'ok') % ,'m' 'LineWidth',2
grid on; hold all; plot(xx,Yexx,'m') % True Chart
xlabel('X - Axes'); ylabel('u(x)')
figure; plot([0:0.05:1],En)    % Error Chart
grid on; xlabel('X - Axes'); ylabel('Error of
Legendre Method')

%% Error Comparsion
delta = sqrt(abs(int(((Yex-y))^2,a,b)))
```

MATLAB code for Lagrange interpolation Method

```
clc
clear all
close all

% %% Inputs
% % Example(5)
% Constant Inouts
a = 0; b = 1; N = 8;
L1= 1; L2= 1;
c=0;
% Functional Inputs
syms t x
A = 1; B = 0;
h = x; ht = t;
f = x - 2*exp(x) + exp(-x) + 1;
k1=t*exp(x); k2=exp(x+t);
Yex = exp(-x); % Exact Solution

%% Example(6)
% % Constant Inouts
% a = 0; b = 1; N = 8;
% L1= 1; L2= 1;
```

```
% c=0;
% % Functional Inputs
% syms t x
% A = 1; B = 0;
% h = x; ht = t;
% f = cos(x)*(0.5*x - 0.25) + 0.25*cos(2-x);
% k1=sin(x-t); k2=cos(x-t);
% Yex = sin(x); % Exact Solution


tic
j=0;j=0;
for i=0:N
 j=j+1;
 x1(j) = vpa(a+((b-a)/(N+1))*j);
end

for i=1:N+1
    for j=1:N+1
%%%%%% P & P1 Calculations %%%%%%%%%%
        p=1;p1=1;
            for k = 1:N+1
               D = (x1(j)-x1(k));
              if D ==0;
```

```
                D = 1 ;D1 = 1; p = p*D;
               else
                x1(j); x1(k); D = (x1(j)-x1(k));
                D1=(t-x1(k));  p = p*D;
               end
                p1 = p1*D1;
             end
%%%%%% A Calculations %%%%%%%%%%%%%%
        if i==j
           A(i,j)=1-
(1/p)*(vpa(subs((int(k1*p1,t,a,x1(i))+int(k2*p1,t,
a,b)),x,x1(i))));
        else
           A(i,j)= -
(1/p)*(vpa(subs((int(k1*p1,t,a,x1(i))+int(k2*p1,t,
a,b)),x,x1(i))));
        end
    end
%%%%%% Right Hand Side Calculations %
   B(i) = subs(f,x,x1(i));
end

% Full Matrix
C = vpa(inv(A) * B')
```

```
u = 0;
for j=1:N+1
     p=1;p1=1;
       for k = 1:N+1
        D = (x1(j)-x1(k));
          if D ==0;
             D = 1 ;D1 = 1; p = p*D;
          else
              x1(j); x1(k); D = (x1(j)-x1(k));
             D1=(x-x1(k));  p = p*D;
          end
             p1 = p1*D1;
       end
        u = vpa(u + C(j)*(p1/p));
 end
 toc
 u
%%%%%%
i=0;
for z=0.000001:0.05:1.000001
    i=i+1; xx(i)= z;
    Ynn(i) = double(subs(u,x,z));
    Yexx(i)= double(subs(Yex,x,z));
    En(i)  = abs(Yexx(i) - Ynn(i));
```

```
end

plot(xx,Ynn,'ok')                          % Numerical
Chart
grid on; hold all; plot(xx,Yexx,'m') % True Chart
xlabel('X - Axes'); ylabel('u(x)')
% axis([0 1 0 0.9])
figure; plot([0.00001:0.05:1.00001],En)    % Error
Chart
grid on; xlabel('X - Axes'); ylabel('Error of
Lagrrange Method')
% axis([0 1 0 2e-9])

axis
%% Error Comparsion
delta = vpa(sqrt(abs(int(((Yex-u))^2,a,b))))

%%% Tables
disp(['No    ','Numerical Sol.    ','Analytical
Sol.    ','Abs. Error'])
for i = 1:length(Ynn)
disp([num2str(i),'     ',num2str(Ynn(i)),'
',num2str(Yexx(i)),'           ',num2str(En(i))])
end
```

جامعة النجاح الوطنية

كلية الدراسات العليا

الطرق العددية لحل معادلة فولتيرا–فريدهولم التكاملية من النوع الثاني

إعداد

ربا ناصر علي سلمان

إشراف

أ.د ناجي قطناني

قدمت هذه الأطروحة استكمالا لمتطلبات الحصول على درجة الماجستير في الرياضيات البحتة بكلية الدراسات العليا في جامعة النجاح الوطنية في نابلس–فلسطين.

2019

الطرق العددية لحل معادلة فولتيرا–فريدهولم التكاملية من النوع الثاني

إعداد

ربا ناصر علي سلمان

إشراف

أ.د ناجي قطناني

الملخص

في هذه الرسالة قمنا بالتركيز على الطرق العددية لحل معادلة فولتيرا–فريدهولم التكاملية من النوع الثاني بسبب استخدامها في نطاق واسع في التطبيقات الفيزيائية مثل نظرية المرونة ، نظرية الجنيح ، مشاكل ثابتة مرنة والتوصيل الجزيئي.

بعد تصنيف المعادلات التكاملية ، قمنا باستقصاء بعض الطرق العددية لحل معادلة فولتيرا–فريدهولم التكاملية من النوع الثاني. وتشمل : طريقة ارتصاف تايلور ، طريقة تقريب المربعات الصغرى ، طريقة التجميع ليجيندري وطريقة لاغرانج الاستيفاء.

تم تحليل الإطار الرياضي لهذه الطرق العددية مع خصائصها التقاربية. وقد تم توضيح بعض الأمثلة العددية لتنفيذ هذه الطرق العددية.

تظهر النتائج العددية بوضوح أن طريقة الاستكمال الداخلي في لاجرانج هي واحدة من أكثر الطرق الرقمية فعالية وكفاءة لحل معادلة فولتيرا–فريدهولم التكاملية من النوع الثاني مقارنة بالطرق العددية الأخرى. من ناحية أخرى ، طريقة ارتداد تايلور هي واحدة من أسرع الطرق من حيث وقت وحدة المعالجة المركزية لحل معادلة فولتيرا–فريدهولم التكاملية من النوع الثاني.