An-Najah National University

Faculty of Graduate Studies

Approximate Solutions of Systems of Integro-Differential Equations

By

Ahmad Jalal Issa

Supervisor

Prof. 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

Approximate Solutions of

Systems of Integro-Differential Equations

By

Ahmad Jalal Issa

This Thesis was Defended Successfully on 27/2/2019 and approved by:

Defense Committee Members	**Signature**
• **Prof. Naji Qatanani / Supervisor**	**...……**
• **Dr. Mahmoud Manasrah / External Examiner**	**....……**
• **Dr. Hadi Hamad / Internal Examiner**	**……**

Dedication

I dedicate my work to all my family members, to my parents, my brothers Osama and Ibrahim, my sisters Sabreen and Mejdalin, my friend Ayham shaer who encouraged me to learn, develop and who have been a source of encouragement and inspiration to me.

Acknowledgement

At the beginning, thanks God, I am grateful to have completed this thesis. I am heartily thankful to my Supervisor, Prof. Naji Qatanani, whose encouragement, guidance and support from the beginning to end level enabled me to develop and understand the topic of my research.

My thanks and appreciation goes to my thesis committee members Dr. Hadi Hamad and Dr. Mahmoud Manasrah for their support and valuable remarks.

الإقرار

أنا الموقع أدناه مقدم الرسالة التي تحمل العنوان:

Approximate Solutions of

Systems of Integro-Differential Equations

أقر بأن ما اشتملت عليه هذه الرسالة إنما هو من نتاج جهدي الخاص بإستثناء ما تمت الإشارة اليه حيثما ورد، وإن هذه الرسالة ككل، أو أي جزء منها لم يقدم لنيل أي درجة أو لقب علمي أو بحثي لدى أية مؤسسة تعليمية أو بحثية أخرى .

Declaration

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

Student's Name: **إسم الطالب:**

Signature: **التوقيع:**

Date: **التاريخ:**

Table of Contents

No.	Content	Page
	Dedication	III
	Acknowledgement	IV
	Declaration	V
	List of Tables	VII
	List of Figures	VIII
	Abstract	X
	Introduction	1
	Chapter One: Mathematical Preliminaries	5
1.1	Classification of Systems of Integro-Differential Equations	7
1.2	Linearity of Systems of Integro-Differential Equations	8
1.3	Homogeneity of Systems of Integro-Differential Equations	8
	Chapter Two: Numerical Techniques for Solving Systems of Integro-Differential Equations	9
2.1	Reconstruction of Variational Iteration Method	10
2.2	Sinc Collocation Method Based on Sinc Functions	14
2.3	Chebyshev Wavelets Method	19
	Chapter Three: Numerical examples for Systems of Integro-Differential Equations	28
3.1	Reconstruction of Variational Iteration Method	30
3.2	Sinc Collocation Method Based on Sinc Functions	34
3.3	Chebyshev Wavelets Method	39
3.4	Conclusions	70
	References	71
	Appendix	78
	الملخص	ب

List of Tables

No.	Title	Page
3.1	Exact and Numerical solutions using Algorithm 3.1 for system (3.1)	32
3.2	Exact and Numerical solutions using Algorithm 3.2 for system (3.1)	36
3.3	Exact and Numerical solutions using Algorithm 3.3 for system (3.1)	41
3.4	Exact and Numerical solutions using Algorithm 3.1 for system (3.2)	45
3.5	Resulting error of applying Algorithm 3.1 for system (3.2)	46
3.6	Exact and Numerical solutions using Algorithm 3.2 for system (3.2)	50
3.7	Resulting error of applying Algorithm 3.2 for system (3.2)	51
3.8	Exact and Numerical solutions using Algorithm 3.3 for system (3.2)	55
3.9	Resulting error of applying Algorithm 3.3 for system (3.2)	56
3.10	Exact and Numerical solutions using Algorithm 3.1 for system (3.3)	61
3.11	Exact and Numerical solutions using Algorithm 3.2 for system (3.3)	64
3.12	Exact and Numerical solutions using Algorithm 3.3 for system (3.3)	67

List of Figures

No.	Title	Page
2.1	Sinc function	14
2.2	The Chebyshev polynomials of the first kind	21
3.1.a	The exact and numerical solutions of u_1 using Algorithm 3.1 for system (3.1)	33
3.1.b	The exact and numerical solutions of u_2 using Algorithm 3.1 for system (3.1)	33
3.2.a	The exact and numerical solutions of u_1 using Algorithm 3.2 for system (3.1)	37
3.2.b	The exact and numerical solutions of u_2 using Algorithm 3.2 for system (3.1)	38
3.3	The resulting error of u_1& u_2 after applying algorithm 3.2 for system (3.1)	38
3.4.a	The exact and numerical solutions of u_1 using Algorithm 3.3 for system (3.1)	42
3.4.b	The exact and numerical solutions of u_2 using Algorithm 3.3 for system (3.1)	43
3.5	The resulting error of u_1& u_2 after applying algorithm 3.3 for system (3.1)	43
3.6.a	The exact and numerical solutions of u_1 using Algorithm 3.1 for system (3.2)	47
3.6.b	The exact and numerical solutions of u_2 using Algorithm 3.1 for system (3.2)	48
3.6.c	The exact and numerical solutions of u_3 using Algorithm 3.1 for system (3.2)	48
3.7	The resulting error of u_1 & u_2 & u_3 after applying algorithm 3.1 for system (3.2)	49
3.8.a	The exact and numerical solutions of u_1 using Algorithm 3.2 for system (3.2)	52
3.8.b	The exact and numerical solutions of u_2 using Algorithm 3.2 for system (3.2)	53
3.8.c	The exact and numerical solutions of u_3 using Algorithm 3.2 for system (3.2)	53
3.9	The resulting error of applying u_1 & u_2 & u_3 after algorithm 3.2 for system (3.2)	54
3.10.a	The exact and numerical solutions of u_1 using Algorithm 3.3 for system (3.2)	57
3.10.b	The exact and numerical solutions of u_2 using Algorithm 3.3 for system (3.2)	58

3.10.c	The exact and numerical solutions of u_3 using Algorithm 3.3 for system (3.2)	58
3.11	The resulting error of u_1 & u_2 & u_3 after applying algorithm 3.3 for system (3.2)	59
3.12.a	The exact and numerical solutions of u_1 using Algorithm 3.1 for system (3.3)	62
3.12.b	The exact and numerical solutions of u_2 using Algorithm 3.1 for system (3.3)	63
3.13	The resulting error of u_1 & u_2 after applying algorithm 3.1 for system (3.3)	63
3.14.a	The exact and numerical solutions of u_1 using Algorithm 3.2 for system (3.3)	65
3.14.b	The exact and numerical solutions of u_2 using Algorithm 3.2 for system (3.3)	66
3.15	The resulting error of u_1 & u_2 after applying algorithm 3.2 for system (3.3)	66
3.16.a	The exact and numerical solutions of u_1 using Algorithm 3.3 for system (3.3)	68
3.16.b	The exact and numerical solutions of u_2 using Algorithm 3.3 for system (3.3)	69
3.17	The resulting error of u_1 & u_2 after applying algorithm 3.3 for system (3.3)	69

Approximate Solutions of

Systems of Integro-Differential Equations

By

Ahmad Jalal Issa

Supervisor

Prof. Naji Qatanani

Abstract

In this thesis we will focus on the numerical handling of system of linear Volterra integro-differential equations.

In this thesis we will investigate some numerical techniques for solving system of linear Volterra integro-differential equations. These numerical techniques are: reconstruction of variational iteration method, sinc collocation method based on sinc function's and the Chebyshev wavelet method. Some illustrative examples to demonstrate the validity and applicability of these techniques are solved. A comparison between these methods is carried out. Numerical results have shown that Chebyshev wavelet method is one of the most efficient and powerful numerical techniques for solving system of linear Volterra integro-differential equations in comparison with the other numerical techniques.

Introduction

The system of integro-diferential equations is one of the most important mathematical tools in both pure and applied mathematics. These systems have motivated a huge amount of research work in recent years. They arise in many physical phenomena such as wind ripple in the desert, nano-hydrodynamics, population growth model, glass-forming process and oceanography. [5, 15]

Many numerical methods for solving system of linear integro-differential equations have been developed by many researchers. Rahimi [35] used the reconstruction of variational iteration method for solving systems of Volterra integro-differential equations and this method was compared with the homotopy perturbation method and proved that the RVIM method was more accurate and faster. In [24] Hesameddini et al., the Sinc-collocation method was used to approximate the solution of systems of linear Volterra integro-differential equations with initial conditions. They noted that the results could be improved by increasing number of dimensions. Aminikhah et al. [5] applied wavelet method for the numerical solution of linear system of integro-differential equations. They use the Chebyshev operational matrix of integration to solve these systems. Al-Faour et al. [3] used spectral method for solving system of linear Volterra integro-differential equations. They also used power functions and Chebyshev polynomials to solve some of the examples and concluded that power functions produce better approximation than the Chebyshev polynomials.

[17] Chandra et al. applied the single term Walsh series technique for solving system of linear second order Volterra integro-differential equations. They concluded that this method has more accurate results than the expansion method. Akyuz et al. [2] implemented Chebyshev collocation method to solve the systems of higher-order linear integro-differential equations. [29] Maleknejad et al. used rationalized Haar functions method for solving linear integro-differential equations system. They observed that the results could be improved by increasing number of dimensions. Arikoglu et al. [7] used differential transform method for solutions of integral and integro-differential equation systems. Biazar et al. [12] used homotopy perturbation method for solving systems of integro-differential equations. This method was characterized by rapid convergence. Biazar et al. [11] used Chebyshev wavelets basis on the interval [0, 1] for solving systems of integro-differential equations. Gachpazan [21] used the Power series method in which the Taylor expansion of the exact solution of linear or nonlinear integro-difierential equations system is obtained by recursive procedure. Jangveladze et al. [26] used the Finite difference approximation of the nonlinear integro-differential system associated with the penetration of a magnetic field into a substance is studied. In [25], Holmaker proved that the stationary solution of the formation of liver zones by a system of integro-differential equations is in fact globally asymptotically stable. That is, it is the limit (as time tends to infinity) of the solution of the integro-differential equations for arbitrary initial values. Bloom [14] recently demonstrated a system of integro-

differential equations governs the evolution of the components of the electric displacement field in a simple class of rigid holohedral isotropic dielectrics.

In this work, we implement some numerical techniques for solving a system of linear Volterra integro-differential equations. These are Reconstruction of variational method, Sinc collocation method based on sinc functions and Chebyshev wavelets method.

Reconstruction of Variational Iteration Method (RVIM) has been induced with Laplace transform from the Variational Iteration Method (VIM) that was developed from the Inokutti technique. The main feature of RVIM is that it provides rapidly convergent successive approximations to the exact solution and is very well-known for its simplicity in computation without any restrictive assumptions [35].

The Sinc collocation method based on sinc functions is widely used for obtaining the approximate solution of ordinary and partial differential equations and integral equations [24]. It is well-known that the sinc approximate solution converges exponentially to the exact solution.

The Chebyshev wavelets have been used by many authors for solving various functional. The main idea of using Chebyshev basis is that the problem under study reduces to a system of linear or nonlinear algebraic equations. This may be done by truncated series of orthogonal basis functions for the solution of problem and using the operational matrices [5]. This thesis is organized as follows: In chapter one, we introduce some

basic concepts of systems of integro-differential equations and their solvability. In chapter two, we implement some numerical methods for solving system of linear Volterra integro-differential equations, these are: reconstruction of variational iteration method, sinc collocation method based on sinc functions and the Chebyshev wavelet method. Numerical examples and results are presented in chapter three and conclusions are drawn**.**

Chapter One

Mathematical Preliminaries

Chapter One
Mathematical Preliminaries

In this chapter, we introduce some basic concepts of systems of integro-differential equations and their solvability.

Definition 1.1[17]

A system of integro-differential equations is the set of equations in which the unknown functions appear inside an integral sign and contains ordinary derivatives.

A system of integro-differential equations can be considerd in general as follows:

$$u_i^{(n)}(x) = f_i(x) + \int_{g(x)}^{h(x)} \left(\sum_{j=1}^{N} k_{ij}(x,t) u_j(t) \right) dt, \quad 1 \le i \le N \tag{1.1}$$

subject to the initial conditions:

$$u_i^{(s)}(0) = a_{is}, i = 1, 2, .., N, s = 0, 1, 2, .., (n-1), \tag{1.2}$$

where $u_i^{(n)}$ is the derivative of u_i of order n, the kernels $k_{ij}(x,t)$ and the functions $f_i(x)$ are given real-valued functions and $g(x)$ and $h(x)$ are limits of integration that may be both variables, constant, or mixed. $u_i(x)$ are the unknown functions known as the solutions of the system.

1.1 Classification of Systems of Integro-Differential Equations

1. System of Volterra integro-differential equations

A system of Volterra integro-differential equations can be written as:

$$u_1^{(n)}(x) = f_1(x) + \int_a^x \Big(k_{11}(x,t)u_1(t) + k_{12}(x,t)u_2(t) + .. + k_{1j}(x,t)u_j(t)\Big)\,dt$$
$$u_2^{(n)}(x) = f_2(x) + \int_a^x \Big(k_{21}(x,t)u_1(t) + k_{22}(x,t)u_2(t) + .. + k_{2j}(x,t)u_j(t)\Big)\,dt$$
$$u_3^{(n)}(x) = f_3(x) + \int_a^x \Big(k_{31}(x,t)u_1(t) + k_{32}(x,t)u_2(t) + .. + k_{3j}(x,t)u_j(t)\Big)\,dt$$
$$\vdots \qquad (1.3)$$
$$u_i^{(n)}(x) = f_i(x) + \int_a^x \Big(k_{i1}(x,t)u_1(t) + k_{i2}(x,t)u_2(t) + .. + k_{ij}(x,t)u_j(t)\Big)\,dt$$

The functions $u_1(x), u_2(x), u_3(x), .., u_i(x)$ are to be determined, the kernels $k_{i1}(x,t), k_{i2}(x,t), \dots, k_{ij}(x,t)$ and the functions $f_i(x)$ are given real-valued functions [42].

2. System of Fredholm integro-differential equations

A system of Fredholm integro-differential equations has the form:

$$u_1^{(n)}(x) = f_1(x) + \int_a^b \Big(k_{11}(x,t)u_1(t) + k_{12}(x,t)u_2(t) + .. + k_{1j}(x,t)u_j(t)\Big)\,dt$$
$$u_2^{(n)}(x) = f_2(x) + \int_a^b \Big(k_{21}(x,t)u_1(t) + k_{22}(x,t)u_2(t) + .. + k_{2j}(x,t)u_j(t)\Big)\,dt$$
$$u_3^{(n)}(x) = f_3(x) + \int_a^b \Big(k_{31}(x,t)u_1(t) + k_{32}(x,t)u_2(t) + .. + k_{3j}(x,t)u_j(t)\Big)\,dt$$
$$\vdots \qquad (1.4)$$
$$u_i^{(n)}(x) = f_i(x) + \int_a^b \Big(k_{i1}(x,t)u_1(t) + k_{i2}(x,t)u_2(t) + .. + k_{ij}(x,t)u_j(t)\Big)\,dt$$

1.2 (Linearity) [42]

The system of integro-differential equations

$$u_i^{(n)}(x) = f_i(x) + \int_{g(x)}^{h(x)} \left(\sum_{j=1}^{N} k_{ij}(x,t) u_j(t) \right) dt, \quad 1 \le i \le N \tag{1.5}$$

is said to be linear if the exponent of the unknown functions $u_j(x)$ under the integral sign for all j is one and all equations do not contain nonlinear functions of $u_j(x)$. Otherwise, the system is called nonlinear. Examples of nonlinear functions ($\sin u$, $\cos u$, $\sinh u$, $\cosh u$, e^u, ...).

1.3 (Homogeneity) [42]

The system of integro-differential equations

$$u_i^{(n)}(x) = f_i(x) + \int_{g(x)}^{h(x)} \left(\sum_{j=1}^{N} k_{ij}(x,t) u_j(t) \right) dt, \quad 1 \le i \le N \tag{1.6}$$

is said to be homogeneous if $f_i(x)$ is identically zero for all i. Otherwise, it is called nonhomogeneous.

Chapter Two

Numerical Techniques for Solving

Systems of Integro-Differential Equations

Chapter Two
Numerical Techniques for Solving Systems of Integro-Differential Equations

There are many numerical methods available for solving systems of integro-differential equations. In this chapter, we will discuss the following methods: Reconstruction of variational iteration method, sinc collocation method based on sinc functions and Chebyshev wavelets method.

2.1 Reconstruction of Variational Iteration Method

Reconstruction of Variational Iteration Method (RVIM) has been induced with Laplace transform from the variational iteration method (VIM) that was developed from the Inokutti technique. The main feature of RVIM is that it provides rapidly convergent successive approximations to the exact solution and is very well-known for its simplicity in computation without any restrictive assumptions [35].

Before introducing the main approach of the RVIM, it is necessary to present the following definitions and theorems related to the Laplace transform method.

Definition 2.1 [37]: Let $f(x)$ be a real-valued function defined for $x \geq 0$, then the Laplace transform of $f(x)$ denoted by $L[f(x)]$

is given as:

$$L[f(x)] = F(s) = \int_0^{\infty} e^{-sx} f(x)dx = \lim_{a \to \infty} \int_0^{a} e^{-sx} f(x)dx \qquad (2.1)$$

Definition 2.2 [37]**:** If $L[f(x)] = F(s)$, then $f(x) = L^{-1}[F(s)]$ is called the inverse Laplace transform of $F(s)$.

Definition 2.3 [35]**:** The convolution of two piecewise continuous functions $f, g: R \to R$ is the function $f * g$ given by

$$(f * g)(x) = \int_0^{x} f(\tau)g(x - \tau)d\tau$$

Theorem 2.1 [35]: Convolution Theorem

Let $f(x)$ and $g(x)$ are piecewise continuous functions on $[0, \infty)$, then

$$L[f * g] = L\left\{\int_0^{x} f(\tau)g(x - \tau)d\tau\right\} = L[f(x)].L[g(x)]$$

$$= F(s).G(s)$$

Also we can conclude that:

$$L^{-1}[F(s).G(s)] = f * g$$

$$= \int_0^{x} f(\tau)g(x - \tau)d\tau$$

Definition 2.4 [37]**:** The Laplace transform for the derivatives of $f(x)$ is given by:

$$L\left[f^{(n)}(x)\right] = s^n F(s) - s^{n-1} f(0) - s^{n-2} f'(0) - \cdots - f^{(n-1)}(0) \qquad (2.2)$$

We consider the system of integro-differential equation of the form:

$$u_i^{(n)}(x) = f_i(x) + \int_0^x \left(k_{i1}(x,t)u_1(t) + k_{i2}(x,t)u_2(t) + .. + k_{ii}(x,t)u_i(t)\right) dt \quad (2.3)$$

where $u_k^{(n)}$ is the derivative of u_k of order n, subject to the initial conditions:

$$u_k^{(j)} = c_j^k,\ 1 \le k \le i,\ 0 \le j \le n-1 \qquad (2.4)$$

We can write system (2.3) as

$$u_k^{(n)}(x) = M_k\left(x, u_1(x), u_2(x), \dots, u_i(x)\right), \qquad k = 1, \dots, i, \qquad (2.5)$$

with the zero artificial initial conditions.

Applying the Laplace transform to (2.5) and using the artificial initial conditions, we obtain:

$$s^n L\{u_k(x)\} = L\left\{M_k\left(x, u_1(x), u_2(x), \dots, u_i(x)\right)\right\}$$

Dividing both sides by s^n we get:

$$L\{u_k(x)\} = \frac{1}{s^n} L\left\{M_k\left(x, u_1(x), u_2(x), \dots, u_i(x)\right)\right\}$$

If we set $\frac{1}{s^n} = G(s)$, then by using the convolution theorem, we get :

$$L\{u_k(x)\} = G(s)L\left\{M_k\left(x, u_1(x), u_2(x), \dots, u_i(x)\right)\right\} = L\{(g * M_k)(x)\}, \qquad (2.6)$$

where $k = 1, \dots, i$, $L^{-1}\{G(s)\} = g(x)$.

Taking the inverse Laplace transform to both sides of (2.6), we obtain:

$$u_k(x) = \int_0^x g(x-t)\, M_k\left(t, u_1(t), u_2(t), \dots, u_i(t)\right) dt, \qquad k = 1, \dots, i$$

To impose the particular initial conditions to get the answer of (2.3), we have the following iteration form:

$$u_k^{\langle m+1\rangle}(x) = u_k^{(0)}(x) + \int_0^x g(x-t)\, M_k\left(t, u_1^{\langle m\rangle}(t), u_2^{\langle m\rangle}(t), .., u_i^{\langle m\rangle}(t)\right) dt,$$

for $k = 1, .., i$. (2.7)

The values $u_1^{(0)}(x), u_2^{(0)}(x), \dots, u_i^{(0)}(x)$ are given by:

$$u_k^{(0)}(x) = u_k(0) + x\acute{u}_k(0) + \frac{x^2 \acute{u}_k(0)}{2!} + \cdots + \frac{x^r u_k^{(r)}(0)}{r!} \qquad (2.8)$$

Therefore, the reconstruction of variation iteration method $u_k(x)$ is obtained as follows:

$$u_k(x) = \lim_{m \to \infty} u_k^{\langle m\rangle}(x), \quad k = 1, .., i, \qquad (2.9)$$

where $u_k^{\langle m\rangle}(x)$ indicates $m - th$ approximation of $u_k(x)$.

2.2 Sinc Collocation Method Based on Sinc Functions

The Sinc collocation method based on sinc functions is widely used for obtaining the approximate solution of ordinary and partial differential equations and integral equations [24]. It is well-known that the sinc approximate solution converges exponentially to the exact solution.

Definition 2.5 [31]**:** The sinc function is defined on the whole real line $-\infty < x < \infty$ by

$$sinc(x) = \begin{cases} \dfrac{\sin(\pi x)}{\pi x} & x \neq 0 \\ 1 & x = 0 \end{cases} \tag{2.10}$$

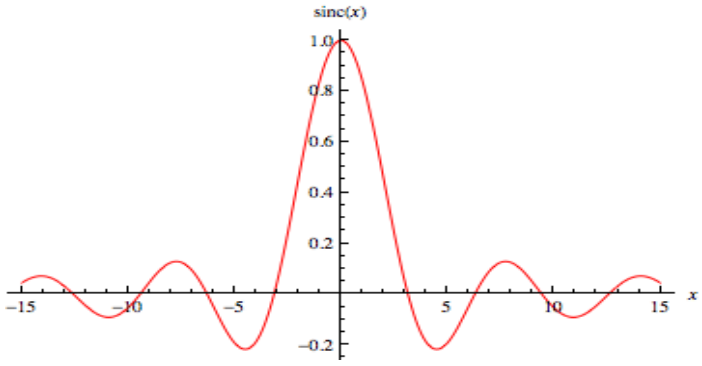

Figure 2.1: Sinc function

Definition 2.6 [31]**:** Let $k = 0, \mp 1, \mp 2, \mp 3, \ldots$, then the translated sinc basis functions are defined as

$$s(k,r)(x) = sinc\left(\frac{x-kr}{r}\right) = \begin{cases} \dfrac{\sin\left[\frac{\pi}{r}(x-kr)\right]}{\frac{\pi}{r}(x-kr)} & x \neq kr \\ 1 & x = kr \end{cases} \tag{2.11}$$

which are called the $k-th$ sinc functions.

Remark 2.1 [24]**:** The Sinc function for the interpolating points $x_d = dr$, is given by

$$s(k,r)(dr) = \mu_{kd} = \begin{cases} 0 & k \neq d \\ 1 & k = d \end{cases} \tag{2.12}$$

Remark 2.2 [24]**:** If $p(x)$ is defined on the real axis and r is a positive integer, then the series

$$a(p,r)(x) = \sum_{k=-\infty}^{\infty} p(kr)s(k,r)(x) \tag{2.13}$$

is called the Whittaker Cardinal expansion of $p(x)$.

The properties of the Whittaker Cardinal expansion have been extensively studied by [27]. These properties are derived in the infinite strip D_S of the complex w-planes where for any $g > 0$, $G_S = \left\{w = t + is: |s| < g \leq \frac{\pi}{2}\right\}$.

To construct an approximation on the interval (a, b) the conformal map.

$$w = \varphi(z) = \ln\left(\frac{z - a}{b - z}\right)$$

The map carries the eye-shaped region

$$G_E = \left\{z \in \mathbb{C}: \left|arg\left(\frac{z-a}{b-z}\right)\right| < g \leq \frac{\pi}{2}\right\}.$$

For the sinc method, the basis functions on (a, b) for $z \in G_E$ are derived from the composite translated sinc functions,

$$s(k,r) \circ \varphi(x) = sinc\left(\frac{\varphi(x) - kr}{r}\right), \tag{2.14}$$

where $s(k,r) \circ \varphi(x) = s(k,r)(\varphi(x))$.

The inverse map of $w = \varphi(z)$ is

$$z = \varphi^{-1}(w) = \frac{a + be^{w}}{1 + e^{w}} \tag{2.15}$$

Also we define the range of φ^{-1} on the real line as

$$\tau = \{\varphi^{-1}(u) \in G_E : -\infty < u < \infty\}$$

and the interpolation points $\{x_d\}$ are then given by:

$$x_d = \varphi^{-1}(dr) = \frac{a + be^{dr}}{1 + e^{dr}}, d = 0, \mp 1, \mp 2, ... \tag{2.16}$$

Definition 2.7 [31]**:** Let $L_\alpha(G_E)$ be the set of all analytic functions. Then, there exists a constant c, such that:

$$|u(z)| \leq c\frac{|\beta(z)|^{\alpha}}{[1 + |\beta(z)|]^{2\alpha}}, z \in G_E, 0 < \alpha \leq 1, \tag{2.17}$$

where $\beta(z) = e^{\varphi(z)}$.

Theorem 2.2 [31]: Let $u \in L_\alpha(G_E)$ and N is a natural number, and r be selected by the formula

$$r = \left(\frac{\pi g}{\alpha N}\right)^{1/2},$$

Where

$$0 < \alpha \leq 1,\ g \leq {}^{\pi}/_{2}.$$

Then, there exists a positive constant c_1, independent of N, such that

$$\sup_{z \in \tau} \left| u(z) - \sum_{d=-N}^{N} u(z_d) s(d,r) \circ \varphi(z) \right| \leq c_1 e^{-(\pi g \alpha N)^{1/2}}. \tag{2.18}$$

Theorem 2.3 [24]: Let $\frac{u}{\acute{\varphi}} \in L_\alpha(G_E)$ and N is a natural number, and r be given as

$r = \left(\frac{\pi g}{\alpha N}\right)^{1/2}$, where $0 < \alpha \leq 1,\ g \leq {}^{\pi}/_{2}$.

Moreover, let $\mu_{kd}^{(-1)}$ be defined as

$$\mu_{kd}^{(-1)} = \frac{1}{2} + \int_{0}^{k-d} \frac{\sin(\pi t)}{\pi t}\, dt\,.$$

Then, there exists a positive constant c_2, independent of N, such that

$$\left| \int_{a}^{z_k} u(t)dt - r \sum_{d=-N}^{N} \mu_{kd}^{(-1)} \frac{u(z_d)}{\acute{\varphi}(z_d)} \right| \leq c_3 e^{-(\pi g \alpha N)^{1/2}}. \tag{2.19}$$

Theorem 2.4 [24]: Let φ be a conformal injective map of the simply connected domain G_E onto G_S. Then

$$\mu_{kd}^{(0)} = [S(k,d) \circ \varphi(x)]|x = x_d = \begin{cases} 1, & k = d, \\ 0, & k \neq d. \end{cases}$$

$$\mu_{kd}^{(1)} = r \frac{d}{d\varphi} [S(k,d) \circ \varphi(x)]|x = x_d = \begin{cases} 0, & k = d, \\ \dfrac{(-1)^{d-k}}{d-k}, & k \neq d. \end{cases}$$

$$\mu_{kd}^{(2)} = r^2 \frac{d^2}{d\varphi^2}[S(k,d) \circ \varphi(x)]|x = x_d = \begin{cases} \frac{-\pi^2}{3}, & k = d, \\ \frac{-2(-1)^{d-k}}{(d-k)^2}, & k \neq d. \end{cases} \quad (2.20)$$

We consider the system of linear Volterra integro-differential equations of the form:

$$u_i^{(n)}(x) = f_i(x) + \int_a^x \left(\sum_{j=1}^{N} k_{ij}(x,t) u_j(t) \right) dt, 1 \leq i \leq N, \quad (2.21)$$

Subject to the initial conditions:

$$u_i^{(s)}(0) = a_{is}, i = 1, 2, \ldots, N, \quad s = 0, 1, 2, \ldots, (n-1), \quad (2.22)$$

in the domain $[0, 1]$ and let $u_i(x) \in L_\alpha(G_E)$. By using theorem (2.2), $u_i(x)$ is approximated as follows:

$$u_i(x) = R_i(x) + A_i(x), \quad (2.23)$$

where

$$R_i(x) = \sum_{k=-N}^{N} c_k^i w(x) sinc\left(\frac{\varphi(x) - kr}{r}\right), A_i(x) = \sum_{j=0}^{n} a_j^i x^j \quad (2.24)$$

where c_k are unknown coefficients and $w(x) = x^n(x-1)^n$.

Integrating both sides of (2.23) from 0 to x we get

$$\int_0^x u_i(t)dt = \int_0^x R_i(t)dt + \int_0^x A_i(t)dt, \tag{2.25}$$

and by differentiating both sides of (2.23) with respect to x we get

$$u_i^{(n)}(x) = R_i^{(n)}(x) + A_i^{(n)}(x),$$

where

$$R_i^{(n)}(x) = \sum_{k=-N}^{N} c_k \frac{d^n\left(w(x)sinc\left(\frac{\varphi(x)-kr}{r}\right)\right)}{dx^n} \tag{2.26}$$

Substituting (2.25) and (2.26) into system (2.21), evaluating the result at the Sinc points $x_j = \frac{e^{jr}}{1+e^{jr}}$, where $j = -N-1, \dots, N$, and by using theorems (2.3) and (2.4), a system of algebraic equations is obtained.

Then we can solve it to obtain unknown coefficients $\left\{c_k^i\right\}_{k=-N}^{N}$ and $\left\{a_j^i\right\}_{j=0}^{n}$.

2.3 Chebyshev Wavelets Method (CWM)

The main idea of using Chebyshev basis is that the problem under study reduces to a system of linear or nonlinear algebraic equations. This may be done by truncated series of orthogonal basis functions for the solution of problem and using the operational matrices [5].

Wavelets constitute a family of functions constructed from dilation and translation of a single function called the mother wavelet [23, 18, 19]. When the dilation p and the translation q vary continuously we have the following family of continuous wavelets as

$$\psi_{p,q}(x) = \frac{1}{\sqrt{|p|}}\,\psi\left(\frac{x-q}{p}\right), p, q \in \mathbb{R}, p \neq 0.$$

If we choose the dilation and translation p^{-a}, and bqp^{-a}, respectively where $p > 1$, $q > 0$. Then we have the following family of continuous wavelets as $\psi_{a,b}(x) = \sqrt{|p|^a}\ \psi(p^a x - bq), a, b \in \mathbb{Z}^+$, where $\psi_{a,b}$ forms a wavelet basis for $L^2(R) = \left\{g\colon\ \mathbb{R} \to \mathbb{C} \mid \int_{-\infty}^{\infty} |g(x)|^2 dx < \infty\right\}$,

where $L^2(\mathbb{R})$: set of all square integrable functions equipped with norm

$$\|f\|_{L^2[a,b]} = \left(\int_a^b |f(x)|^2 dx\right)^{1/2}.$$

For the particular case, when $p = 2$ and $q = 1$, then $\psi_{a,b}(x)$ forms an orthogonal basis.

Chebyshev wavelets $\psi_{b,c}(x) = \psi(a, b, c, x)$ have four parameters, $b = 1, 2, 3, \ldots, 2^a - 1$, $a \in \mathbb{Z}^+$and c is the degree of Chebyshev polynomials of the first kind. They are defined on the interval $0 \leq x \leq 1$ by:

$$\psi_{b,c}(x) = \psi(a, b, c, x)$$

$$= \begin{cases} 2^{a/2}\tilde{T}_c(2^a x - 2b + 1), & for \quad \frac{b-1}{2^{a-1}} \leq x \leq \frac{b}{2^{a-1}} \\ 0 \quad , & otherwise \end{cases}$$

where

$$\tilde{T}_C(x) = \begin{cases} \frac{1}{\sqrt{\pi}}, & c = 0 \\ \sqrt{\frac{2}{\pi}}\, T_C(x), & c > 0 \end{cases}$$

and $c = 0, 1, 2, \ldots, C-1$, and $b = 1, 2, 3, \ldots, 2^a - 1$.

$T_C(x)$ are the famous Chebyshev polynomials of the first kind of degree c which are orthogonal with respect to the weight function $w(x) = 1/\sqrt{1-x^2}$, on the interval $-1 \leq x \leq 1$ and satisfy the following recurrence relation

$T_0(x) = 1$, $T_1(x) = x$, $T_{c+1}(x) = 2xT_c(x) - T_{c-1}(x)$, $c = 1, 2, 3, \ldots$

See Figur

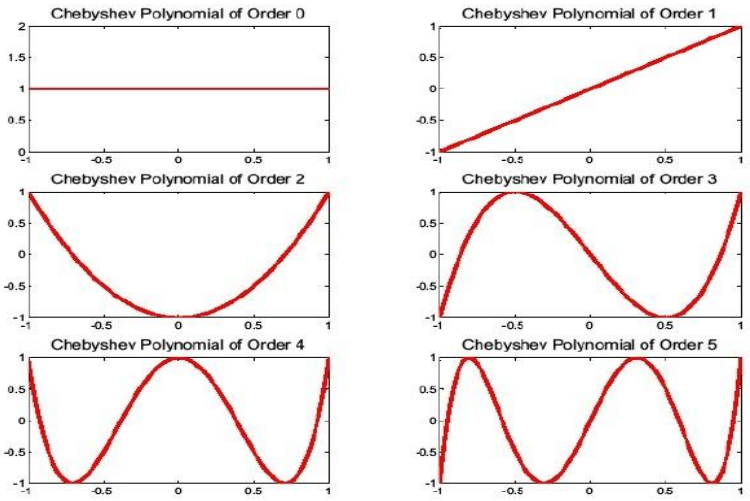

Figure 2.2: The Chebyshev polynomials of the first kind.

Remark 2.4 [5]**:** The set of Chebyshev wavelets is an orthogonal set with respect to the weight function $w_b(x) = w(2^a x - 2b + 1)$.

Definition 2.8 [5]**:** A function $h(x)$ defined on the interval $0 \le x \le 1$ is called the wavelet series if this function is written in the following form

$$h(x) = \sum_{b=1}^{\infty} \sum_{c=0}^{\infty} d_{bc} \psi_{bc}(x),$$

where $d_{bc} = \left(h(x), \psi_{bc}(x)\right) w_b(x)$ is the inner product in $L^2_{w_b}[0,1]$.

Remark 2.5 [5]**:** The wavelet series in $L^2[0,1]$ is convergent if

$$\lim_{u_1, u_2 \to \infty} \left\| h(x) - \sum_{b=1}^{u_1} \sum_{c=0}^{u_2} d_{bc} \psi_{bc}(x) \right\| = 0.$$

If the wavelet series is truncated, then it can be written as

$$h(x) \cong \sum_{b=1}^{2^{a-1}} \sum_{c=0}^{C-1} d_{bc} \psi_{bc}(x) = D^T \psi(x),$$

where D and $\psi(x)$ are $2^{a-1} C \times 1$ matrices given by

$$D = \left[d_{1,0}, d_{1,1}, .., d_{1,C-1}, d_{2,0}, d_{2,1}, .., d_{2,C-1}, .., d_{2^{a-1},0}, .., d_{2^{a-1},C-1}\right]^T$$

$$= \left[d_1, d_2, \dots, d_C, d_{C+1}, \dots, d_{2^{a-1}C}\right]^T,$$

$$\psi(x) = [\psi_{1,0}(x), \psi_{1,1}(x), \dots, \psi_{1,C-1}(x), \psi_{2,0}(x), \psi_{2,1}(x),$$

$$\dots, \psi_{2,C-1}(x), \dots, \psi_{2a-1,0}(x), \dots, \psi_{2a-1,C-1}(x)]^T$$

$$= \left[\psi_1(x), \psi_2(x), \dots, \psi_C(x), \psi_{C+1}(x), \dots, \psi_{2^{a-1}C}(x)\ \right]^T. \qquad (2.27)$$

Remark 2.6 [11]**:** The integral of the multiple of two Chebyshev wavelets vector functions with respect to $w_b(x)$ from 0 to 1 is an identity matrix. Moreover, A function $h(x,y)$ defined on $[0,1]\times[0,1]$ can be approximated as :

$$h(x,y) \cong \sum_{i=1}^{2^{a-1}C} \sum_{j=1}^{2^{a-1}C} a_{ij}\psi_i(x)\,\psi_j(y) = \psi^T(x) A\,\psi(y)$$

where $A=[a_{ij}]$ is a matrix of the entries $2^{a-1}C\times 2^{a-1}C$, that can be determined by:

$$a_{ij} = \Big(\psi_i(x), \Big(h(x,y), \psi_j(y)\Big) w_b(y)\Big) w_b(x),\ i = 1,2,3,\dots,2^{a-1}C,$$

$$j = 1,2,3,\dots,2^{a-1}C.$$

The integral of the vector $\psi(x)$ defined in (2.27), can be achieved as:

$$\int_0^x \psi(t)dt = B\,\psi(x)$$

where B is the $2^{a-1}C\times 2^{a-1}C$ operational matrix of integration [8].

This matrix has form :

$$B = 2^{-a}\begin{pmatrix} M & E & E & \cdots & E \\ O & M & E & \ddots & \vdots \\ O & O & M & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & E \\ O & \cdots & O & O & M \end{pmatrix},$$

where M, E and O are $C \times C$ matrices given by

$$M = \begin{pmatrix} 1 & 1/\sqrt{2} & 0 & 0 & 0 & \cdots & 0 \\ -\sqrt{2}/4 & 0 & 1/4 & 0 & 0 & \cdots & 0 \\ -\sqrt{2}/3 & -1/2 & 0 & 1/6 & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \sqrt{2}(-1)^{\lambda}(\frac{1}{\lambda-2}-\frac{1}{\lambda})/2 & \cdots & -1/2(\lambda-2) & 0 & 1/2\lambda & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \sqrt{2}(-1)^{C}(\frac{1}{C-2}-\frac{1}{C})/2 & 0 & 0 & 0 & \cdots & -1/2(C-2) & 0 \end{pmatrix}$$

$$E = \begin{pmatrix} 2 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ -2\sqrt{2}/3 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sqrt{2}(\frac{1-(-1)^{\lambda}}{\lambda}-\frac{1-(-1)^{\lambda-2}}{\lambda-2})/2 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sqrt{2}(\frac{1-(-1)^{C}}{C}-\frac{1-(-1)^{C-2}}{C-2})/2 & 0 & 0 & \cdots & 0 \end{pmatrix}$$

$$O = \begin{pmatrix} 0 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 0 \end{pmatrix}$$

The product characteristic of two Chebyshev wavelets vector functions are given as

$$\psi(x)\psi^{T}(x)\,Z \approx \tilde{Z}\,\psi(x),$$

where Z is a given vector and $\tilde{Z} = \left[\tilde{z}_{ij}\right]_{2^{a-1}C \times 2^{a-1}C}$ is an operational matrix of product.

We consider the system of linear Volterra integro-differential equations of the form:

$$u_i^{(n)}(x) = f_i(x) + \sum_{j=1}^{c} \int_0^x k_{ij}(x,t) u_j(t) dt, \tag{2.28}$$

with the following conditions

$$u_i^{(r)}(0) = a_{ir},\ i = 1, 2, 3, \dots, n, r = 0, 1, 2, \dots, (n-1), c = 1, 2, 3, \dots. \tag{2.29}$$

Now we approximate $u_i^{(n)}(x)$ by using Chebyshev wavelet space as follows

$$u_i^{(n)}(x) = D_i^T \psi(x), i = 1, 2, 3, \dots, n, \tag{2.30}$$

Therefore we have

$$u_i^{(r)}(x) = D_i^T B^{n-r} \psi(x) + \sum_{j=0}^{n-r-1} a_{ir} \frac{x^j}{j!},$$

where $i = 1, 2, 3, \dots, n$ and D_i are $2^{a-1}C \times 1$ matrices given by

$$D = \left[d^i{}_{1,0}, d^i{}_{1,1}, \dots, d^i{}_{1,C-1}, d^i{}_{2,0}, d^i{}_{2,1}, \dots, d^i{}_{2^{a-1},0}, \dots, d^i{}_{2^{a-1},C-1}\right]^T$$

$$= \left[d_{i,1}, d_{i,2}, \dots, d_{i,C}, d_{i,C+1}, \dots, d_{i,2^{a-1}C}\right]^T,$$

$$\psi(x) = [\psi_{1,0}(x), \psi_{1,1}(x), \dots, \psi_{1,C-1}(x), \psi_{2,0}(x), \psi_{2,1}(x),$$

$$\dots, \psi_{2,C-1}(x), \dots, \psi_{2a-1,0}(x), \dots, \psi_{2a-1,C-1}(x)]^T$$

$$= \left[\psi_1(x), \psi_2(x), \dots, \psi_C(x), \psi_{C+1}(x), \dots, \psi_{2^{a-1}C}(x)\ \right]^T. \tag{2.31}$$

The use of (2.29) and (2.30) other conditions will be considered the following approximation:

$$f_i(x) \cong F_i^T\ \psi(x),$$

$$u_i(x) \cong D_i^T B\psi(x) + E_i^T \psi(x), \tag{2.32}$$

$$k_{ij}(x,t) \cong \psi^T(x) K_{ij} \psi(t),$$

where K_{ij} and F_i are known matrices for $i = 1, 2, 3, \dots, n, j = 1, 2, 3, \dots, c$.

By substituting the approximations (2.30) and (2.32) into the system (2.28), we obtain:

$$D_i^T \psi(x) \cong F_i^T \psi(x) + \sum_{j=1}^{c} \int_0^x \left(\psi^T(x) K_{ij} \psi(t)\right)(D_i^T B\psi(t) + E_i^T \psi(t))dt$$

$$= F_i^T \psi(x) + \sum_{j=1}^{c} \left(\psi^T(x) K_{ij}\right)\left(\int_0^x \psi(t)(D_i^T B\psi(t) + E_i^T \psi(t))dt\right)$$

$$= F_i^T \psi(x) + \sum_{j=1}^{c} \left(\psi^T(x) K_{ij}\right)\left(\int_0^x \psi(t)(D_i^T B + E_i^T)\psi(t)dt\right)$$

Therefore,

$$D_i^T \psi(x) \cong F_i^T \psi(x) + \sum_{j=1}^{c} \psi^T(x) K_{ij} \tilde{Z}_i B\, \psi(x), i = 1, 2, 3, \dots, n, \tag{2.33}$$

$c = 1, 2, 3, \dots.$, where B is the $2^{a-1}C \times 2^{a-1}C$ operational matrix of integration and $\tilde{Z}_i$ are $2^{a-1}C \times 1$ matrices.

We multiply both sides of (2.33) by $w_n(x)\,\psi^T(x)$ and integrating with respect to x from 0 to 1, we obtain a linear system in terms of input D_i, $i = 1, 2, 3, \dots, n$, the vector functions D_i elements are calculated by solving this system.

Chapter Three

Numerical Examples for

Systems of Integro-Differential Equations

Chapter Three
Numerical Examples for
Systems of Integro-Differential Equations

In this chapter, three numerical techniques are considered to solve systems of integro-differential equations. These techniques are reconstruction of variational iteration method, sinc collocation method based on sinc functions and Chebyshev wavelets method. The algorithm of each technique will be implemented through Mathematica software to solve three numerical examples.

Example 3.1

Consider the system of Volterra integro-differential equations:

$$\begin{aligned} u_1'(x) &= 1 - 7x^2 - \frac{x^3}{6} - 4x^4 - \frac{x^5}{20} + \int_0^x (x-t)u_1 + 6xu_2\,dt, \\ u_2'(x) &= 4x + \frac{10}{3}x^2 - \frac{x^3}{3} + x^4 - \frac{x^5}{5} + \int_0^x tu_1 + (2t-3x)u_2\,dt, \end{aligned} \tag{3.1}$$

together with the initial conditions

$u_1(0) = 0, u_2(0) = \frac{5}{3}$,

The exact solution of system (3.1) [41] is $u_1(x) = x^3 + x, u_2(x) = 2x^2 + \frac{5}{3}$.

We seek to find an approximate solution to the system (3.1) using the following numerical techniques:

3.1 Reconstruction of Variational Iteration Method

The following algorithm implements the reconstruction of variational iteration method using Mathematica software.

Algorithm 3.1

Input N, n

Input $f_i(x)$ for i =1, 2, ….., N

Input $k_{ij}(x)$ for i =1, 2, ….., N ; j =1 , 2 , ….., N

Input initial condition $U_i(0)$; $U_i^{(d)}(0)$ for i =1 , 2 , ….., N ; d=1,…n-1

Define $u_i(x) = f_i(x) + \int_0^x \sum_{j=1}^N k_{ij}(x,t) * u_j(t)\ dt$

Applying Laplace transformation for both sides

Substitute artificial initial condition $u_i(0) = 0$ for i =1, 2, ….., N ;

$u_i^{(d)}(0) = 0$ for i =1 , 2 , ….., N ; d=1,…n-1

Use convolution theory

Set $u_{i0}(t) = U_i(0) + \sum_{j=1}^{n-1} \frac{U_i^{(j)}(0) \times t^{(j)}}{j!}$ for i =1 , 2 , ….., N ;

Applying recurrence iteration to calculate $u_{im}(x)$ for i =1 , 2 , ….., N ; for m =1 , 2 , ….., M ;

Set $u_{aprrox\ i}(x) = u_{iM}(x)$ for i =1, 2, ….., N

Input $u_{exact\ i}(x)$ for i =1, 2, ….., N

Plot $u_{aprrox\,i}(x)$; $u_{exact\ i}(x)$ for i =1 , 2 , ….., N

Define error= $|u_{exact\ i}(x) - u_{aprrox\,i}(x)|$; Plot error.

Table 3.1 contains the exact and numerical solutions together and the absolute error using algorithm 3.1 for system (3.1)

Table 3.1: The exact and numerical solutions of applying Algorithm 3.1 for system (3.1).

x	$Exact\ solution$ $u_1(x) = x^3 + x$	$Numerical\ solution$ u_{1app}	$Absolute\ error$ $\lvert u_1 - u_{1app} \rvert$	$Exact\ solution$ $u_2(x) = 2x^2 + \frac{5}{3}$	$Numerical\ solution$ $u_{2\ app}$	$Absolute\ error$ $\lvert u_2 - u_{2\ app} \rvert$
0	0	0	0.	1.6666666666666667	1.6666666666666667	0.
0.1	0.1010	0.1010	0.	1.6866666666666668	1.6866666666666668	0.
0.2	0.2080	0.2080	0.	1.7466666666666668	1.7466666666666668	0.
0.3	0.3270	0.3270	0.	1.8466666666666667	1.8466666666666667	0.
0.4	0.4640	0.4640	0.	1.9866666666666668	1.9866666666666668	0.
0.5	0.6250	0.6250	0.	2.166666666666667	2.166666666666667	0.
0.6	0.8160	0.8160	0.	2.3866666666666667	2.3866666666666667	0.
0.7	1.0430	1.0430	0.	2.646666666666667	2.646666666666667	0.
0.8	1.3120	1.3120	0.	2.946666666666667	2.946666666666667	0.
0.9	1.6290	1.6290	0.	3.286666666666667	3.286666666666667	0.

Figure 3.1.a compares the exact solution $u_1(x) = x^3 + x$ and the approximate solution with $M = 10$

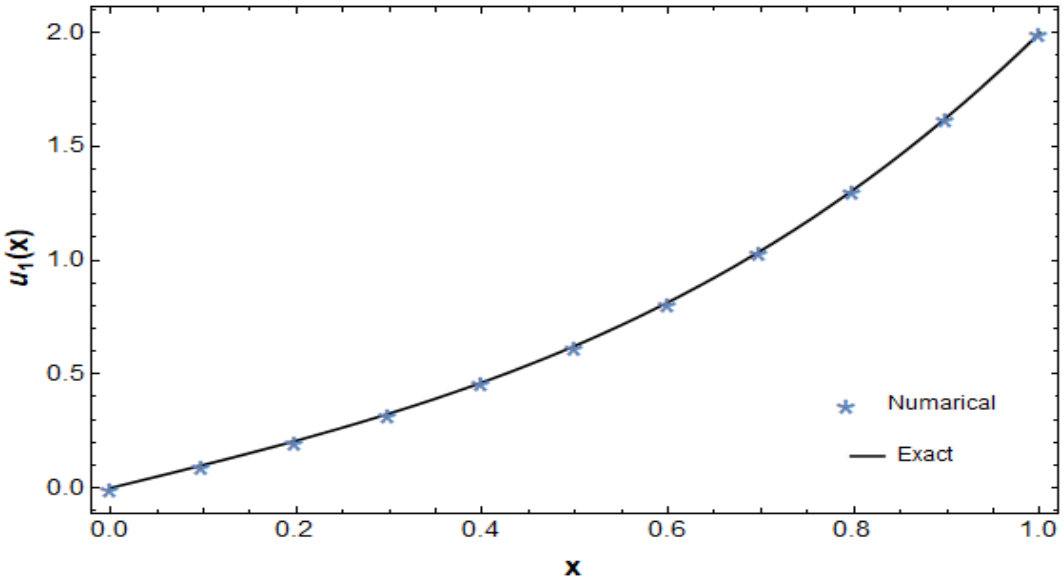

Figure 3.1.a: The exact and numerical solutions of u_1 using Algorithm 3.1 for system (3.1).

Figure 3.1.b compares the exact solution $u_2(x) = 2x^2 + \frac{5}{3}$ and the approximate solution with $M = 10$

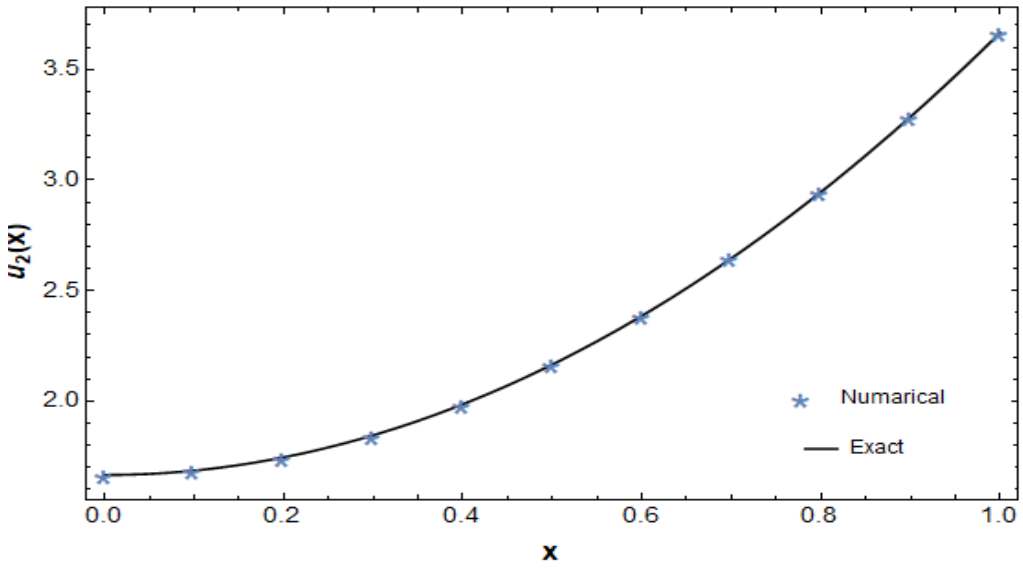

Figure 3.1.b: The exact and numerical solutions of u_2 using Algorithm 3.1 for system (3.1).

3.2 Sinc Collocation Method Based on Sinc Functions

The following algorithm implements the Sinc collocation method based on sinc functions using Mathematica software.

Algorithm 3.2

Input N, M, α, d, n

Input $f_i(x)$ for i =1, 2, ….., N

Input $k_{ij}(x)$ for i=1, 2,….., N ; j =1 , 2 , ….., N

Input initial condition $U_i(0)$ for i =1, 2, ….., N ;

Input initial condition $U_i^{(d)}(0)$ for i =1, 2, ….., N ; d=1,…n-1

Define sinc Function $S(k,h,x)$

Define $P_i(x) = \sum_{j=0}^{n} a_{ij}\, x^j$ for i =1, 2, ….., n ;

Define $Y_i(x) = \sum_{j=-M}^{M} C_{ij} S(k,h,x)$ for i =1, 2, ….., N ;

Replace $u_i(x) = Y_i(x) + P_i(x)$ for i =1, 2, ….., N ; in each equation

Calculate Sinc point x_p for p =-M-1 , -M , -M+1,……. , 0 , 1, .., M;

Evaluate $u_i(x_p) = f_i(x_p) + \int_0^{x_p} \sum_{j=1}^{N} k_{ij}(x_p,t) * u_j(t)\, dt$ for p =-M-1 , -M , -M + 1 ,……. , 0 , 1, .., M ; for i =1 , 2 , ….., N ;

From this step, we get (2 (M +1) * N) equation

Evaluate $u_i(0) = U_i(0)$ for i =1, 2, ….., N ;

From this step, we get (N) equation

Evaluate $u_i^{(d)}(0) = U_i^d(0)$ for i =1, 2, ….., N ; d=1,…n-1

From this step, we get (N*(n-1)) equation

Solving the algebraic system to get $c_{ij}\ and\ a_{ik}$ for i =1, 2, ….., N ;

j =1, 2, ….., N; k =1 , 2 , ….., n

Set $u_{aprrox\ i}(x) = sub\ c_{ij}\ and\ a_{ik}\ in\ u_i(x)$

Input $u_{exact\ i}(x)$

Plot $u_{aprrox\ i}(x)$; $u_{exact\ i}(x)$

Define error= $|u_{exact\ i}(x) - u_{aprrox\ i}(x)|$

Plot error.

Table 3.2 contains the exact and numerical solutions together and the absolute error using algorithm 3.2 for system (3.1)

Table 3.2: The exact and numerical solutions of applying Algorithm 3.2 for system (3.1**).**

x	$Exact\ solution$ $u_1(x) = x^3 + x$	$Numerical\ solution$ u_{1app}	$Absolute\ error$ $\lvert u_1 - u_{1app}\rvert$	$Exact\ solution$ $u_2(x) = 2x^2 + \frac{5}{3}$	$Numerical\ solution$ $u_{2\ app}$	$Absolute\ error$ $\lvert u_2 - u_{2\ app}\rvert$
0	0	0	**0**	1.6666666666666667	1.6666666666666667	0.
0.1	0.1010	0.10100000039140195	$3.914019386375145 \times 10^{-10}$	1.6866666666666668	1.6866666667257286	$5.9061866508614 \times 10^{-11}$
0.2	0.2080	0.2079999987555035	$1.244496505847792 \times 10^{-9}$	1.7466666666666668	1.7466666664854993	$1.811675254259626 \times 10^{-10}$
0.3	0.3270	0.32699997849740003	$2.15026000338625 \times 10^{-8}$	1.8466666666666667	1.8466666633866398	$3.280026916741008 \times 10^{-9}$
0.4	0.4640	0.46399993875638507	$6.124361495718489 \times 10^{-8}$	1.9866666666666668	1.9866666571094465	$9.557220348455076 \times 10^{-9}$
0.5	0.6250	0.6249999148772128	$8.512278715233634 \times 10^{-8}$	2.166666666666667	2.166666653108086	$1.355858092466633 \times 10^{-8}$
0.6	0.8160	0.8159999445594568	$5.544054337836002 \times 10^{-8}$	2.3866666666666667	2.3866666576718694	$8.994797351391526 \times 10^{-9}$
0.7	1.0430	1.0430000177818546	$1.778185443335189 \times 10^{-8}$	2.646666666666667	2.646666669626049	$2.959382072731387 \times 10^{-9}$
0.8	1.3120	1.3120000754536585	$7.545365821037819 \times 10^{-8}$	2.946666666666667	2.946666679401857	$1.273518979161281 \times 10^{-8}$
0.9	1.6290	1.6290000803363964	$8.033639642412993 \times 10^{-8}$	3.286666666666667	3.2866666806256957	$1.3959028599686 \times 10^{-8}$

The accuaracy of the results in the Sinc collocation method based on sinc functions to solve system (3.1) has a max error of $u_1 \approx 8.5 \times 10^{-8}$ and max error of $u_2 \approx 1.42 \times 10^{-8}$

Figure 3.2.a compares the exact solution $u_1(x) = x^3 + x$ and the approximate solution with $M = 8$

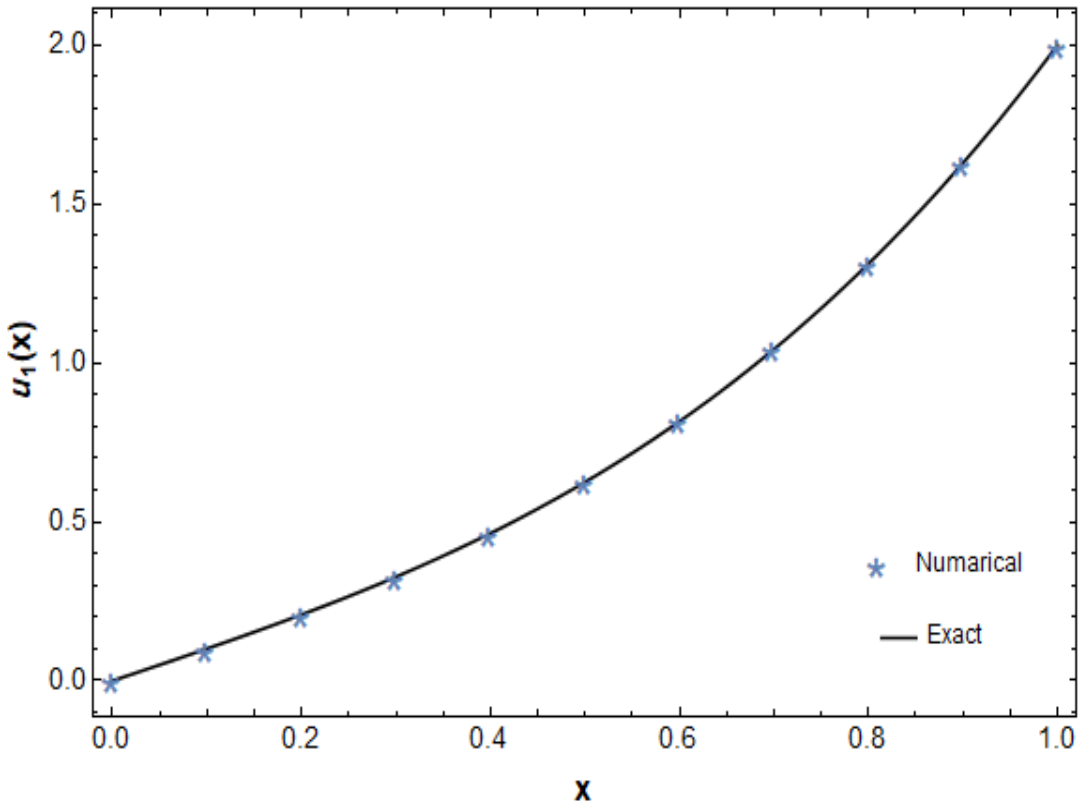

Figure 3.2.a: The exact and numerical solutions of u_1 using Algorithm 3.2 for system (3.1).

Figure 3.2.b compares the exact solution $u_2(x) = 2x^2 + \frac{5}{3}$ and the approximate solution with $M = 8$

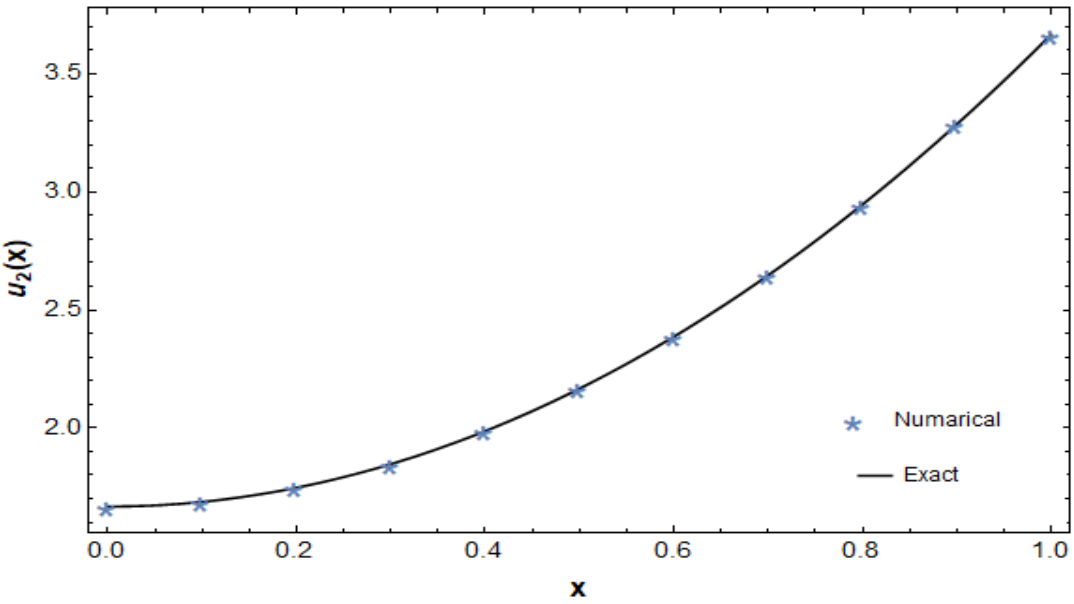

Figure 3.2.b: The exact and numerical solutions of u_2 using Algorithm 3.2 for system (3.1).

Figure 3.3 shows the absolute error resulting of applying algorithm 3.2 for system (3.1)

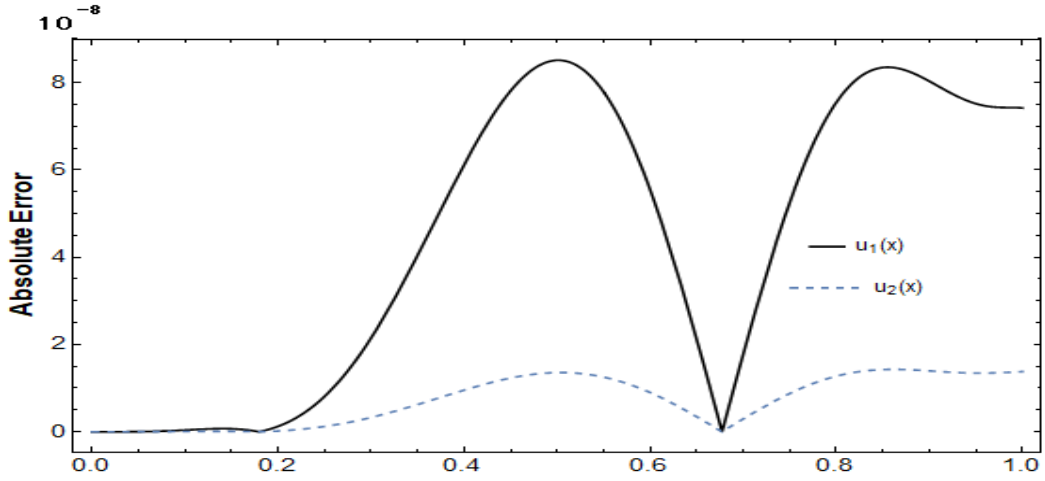

Figure 3.3: The resulting error of u_1& u_2 after applying Algorithm 3.2 for system (3.1).

3.3 Chebyshev Wavelets Method

The following algorithm implements the Chebyshev wavelets method using Mathematica software.

Algorithm 3.3

Input N, M, k , b, n

Input $f_i(x)$ for i =1, 2, ….., N

Input $k_{ij}(x)$ for i =1, 2, ….., N ; j =1 , 2 , ….., N

Input initial condition $U_i(0)$ for i =1, 2, ….., N ;

Input initial condition $U_i^{(r)}(0)$ for i =1, 2, ….., N ; r=1,…n-1

Define Chebyshev Function $T(k, b, m, x)$

Define weight function $W(x)$

Define $D_i = [d_{i1}, \ldots\ldots, d_{iM}]$ for i =1 , 2 , ….., N ;

Define $\psi(x) = [T(k, b, 0, x), \ldots\ldots, T(k, b, M-1, x)]$

Define $DM[h(x)]$ as definition 2.8

Define operator $DM[h(x, y)]$ as remark 2.6

Calculate $F_i = DM[f_i(x)]$ for i =1, 2, ….., N ;

Calculate $Q_i = DM[U_i(0)]$ for i =1, 2, ….., N ;

Calculate $K_{ij} = DM[k_{ij}(x)]$ for i =1, 2, ….., N ; j =1 , 2 , ….., N

Define operation matrix B

Define $u_i^{(n)}(x) = {D_i}^T . \psi(x)$ for i =1, 2, ….., N

Define $u_i^{(r)}(x) = D_i . B^{n-r} . \psi(x) + \sum_{j=0}^{n-r-1} a_{ir} \frac{x^j}{j!}$ for i =1, 2, ….., N

for r =1 , 2 , ….., n-1

Substituting $u_i^{(r)}(x)$ for i =1, 2, ….., N ; for r =1 , 2 , ….., n in the system

Multiplying each equation by W(x).$\psi^T(x)$

Applying $\int_0^1 . dx$ for all equations

From this step, we get (M*N) equation

Solving the algebraic system to get d_{ij} for i =1, 2, ….., N ;

j =1 , 2 , ….., N;

Set $u_{aprrox\,i}(x) = sub\ d_{ij}\ in\ u_i(x)$

Input $u_{exact\ i}(x)$

Plot $u_{aprrox\,i}(x)$; $u_{exact\ i}(x)$

Define error= $|u_{exact\ i}(x) - u_{aprrox\,i}(x)|$

Plot error.

Table 3.3 contains the exact and numerical solutions together and the absolute error using algorithm 3.3 for system (3.1).

Table 3.3: The exact and numerical solutions of applying Algorithm 3.3 for system (3.1**).**

x	$Exact\ solution$ $u_1(x) = x^3 + x$	$Numerical\ solution$ u_{1app}	$Absolute\ error$ $\lvert u_1 - u_{1app}\rvert$	$Exact\ solution$ $u_2(x) = 2x^2 + \frac{5}{3}$	$Numerical\ solution$ $u_{2\,app}$	$Absolute\ error$ $\lvert u_2 - u_{2\,app}\rvert$
0	0	$1.123904917174857 \times 10^{-16}$	$1.123904917174857 \times 10^{-16}$	1.6666666666666667	1.666666666666667	$2.220446049250313 \times 10^{-16}$
0.1	0.1010	0.10100000000000009	$8.326672684688674 \times 10^{-17}$	1.6866666666666668	1.686666666666667	$2.220446049250313 \times 10^{-16}$
0.2	0.2080	0.20800000000000007	$5.551115123125783 \times 10^{-17}$	1.7466666666666668	1.7466666666666668	0.
0.3	0.3270	0.3270000000000003	$2.220446049250313 \times 10^{-16}$	1.8466666666666667	1.846666666666667	$2.220446049250313 \times 10^{-16}$
0.4	0.4640	0.4640000000000004	$3.885780586188048 \times 10^{-16}$	1.9866666666666668	1.9866666666666672	$4.440892098500626 \times 10^{-16}$
0.5	0.6250	0.6250000000000004	$4.440892098500626 \times 10^{-16}$	2.166666666666667	2.166666666666667	0.
0.6	0.8160	0.8160000000000006	$4.440892098500626 \times 10^{-16}$	2.3866666666666667	2.386666666666667	$4.440892098500626 \times 10^{-16}$
0.7	1.0430	1.0430000000000008	$6.661338147750939 \times 10^{-16}$	2.646666666666667	2.646666666666667	0.
0.8	1.3120	1.3120000000000007	$4.440892098500626 \times 10^{-16}$	2.946666666666667	2.946666666666667	0.
0.9	1.6290	1.6290000000000007	$6.661338147750939 \times 10^{-16}$	3.286666666666667	3.286666666666667	0.

The accuaracy of the results in the Chebyshev wavelets method to solve system (3.1) has a max error of $u_1 \approx 5.7 \times 10^{-16}$ and the max error of $u_2 \approx 4.4 \times 10^{-16}$

Figure 3.4.a compares the exact solution $u_1(x) = x^3 + x$ and the approximate solution with $M = 8$

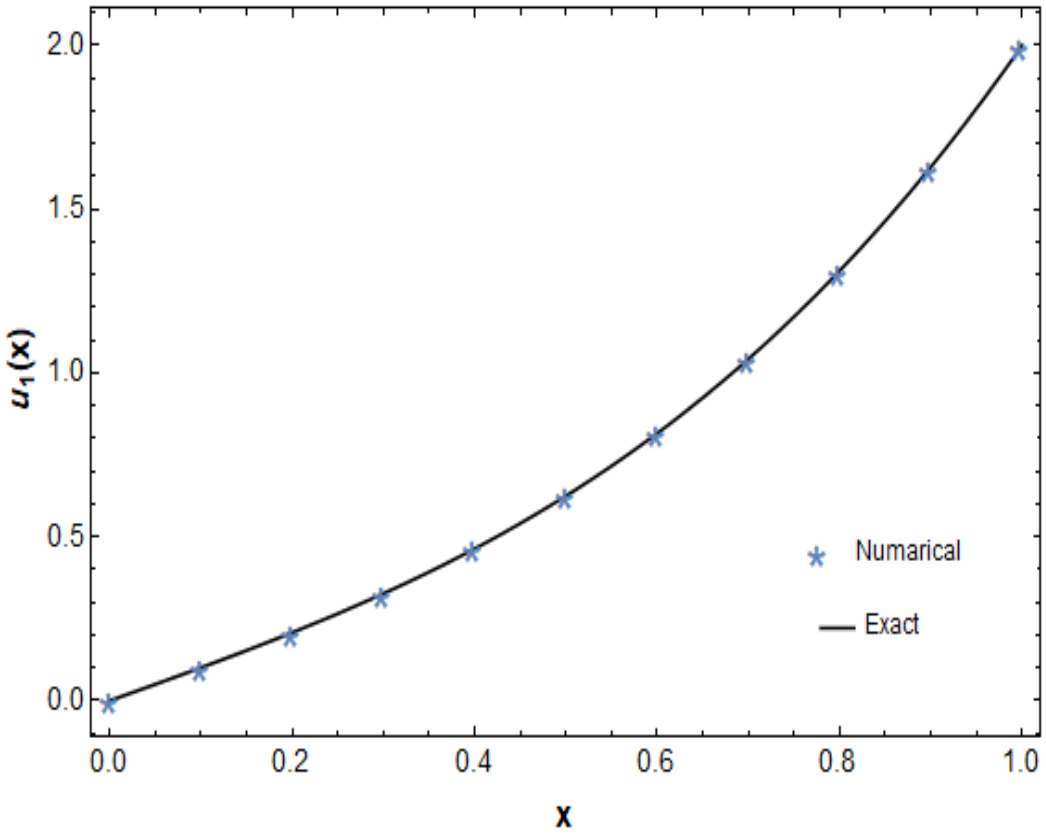

Figure 3.4.a: The exact and numerical solutions of u_1 using Algorithm 3.3 for system (3.1)

Figure 3.4.b compares the exact solution $u_2(x) = 2x^2 + \frac{5}{3}$ and the approximate solution with $M = 8$

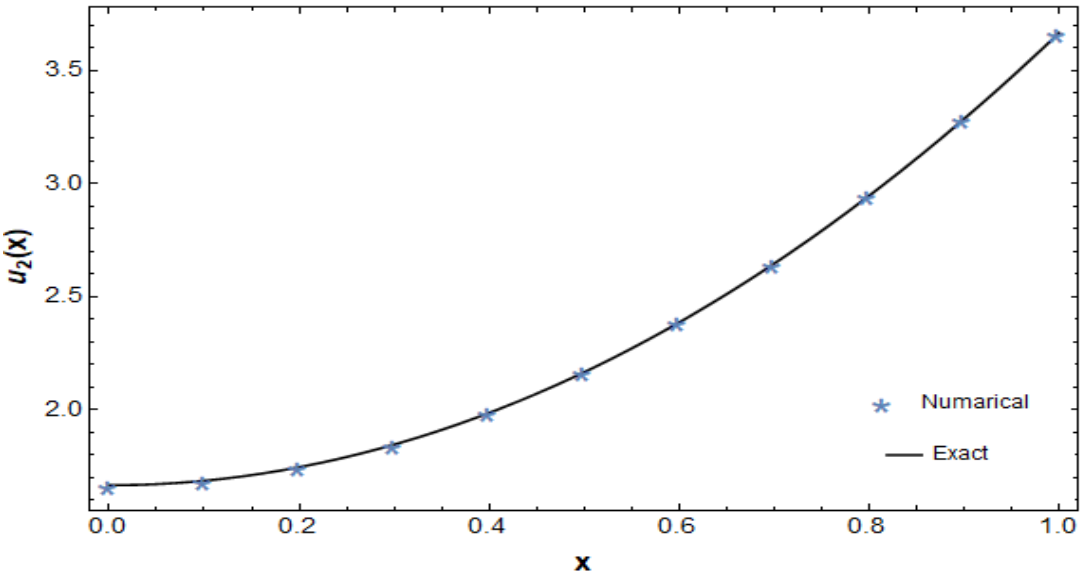

Figure 3.4.b: The exact and numerical solutions of u_2 using Algorithm 3.3 for system (3.1)

Figure 3.5 shows the absolute error resulting of applying algorithm 3.3 for system (3.1)

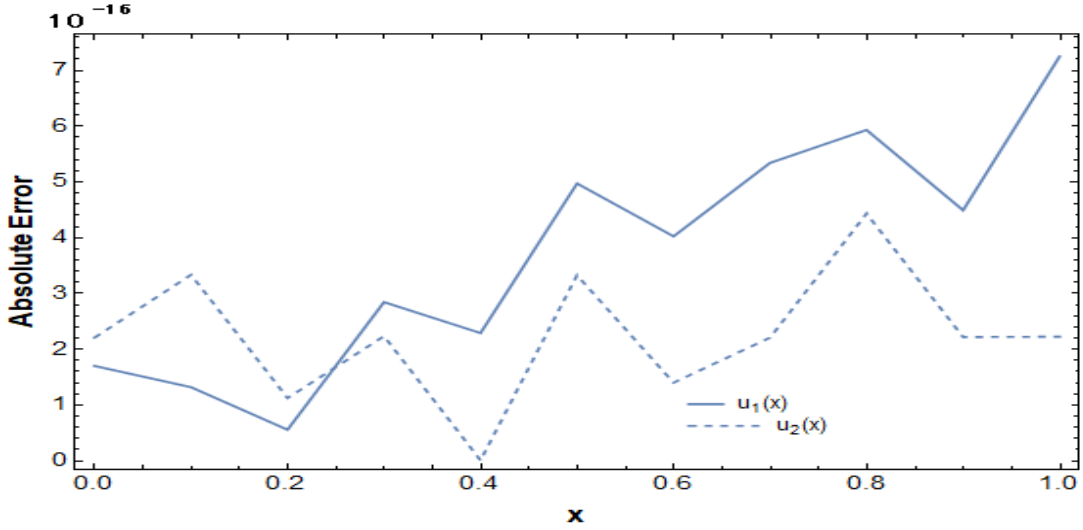

Figure 3.5: The resulting error ofu_1& u_2 after applying Algorithm 3.3 for system (3.1)

Example 3.2

Consider the system of Volterra integro-differential equations:

$$\begin{aligned} u_1'(x) &= -2 + x^2 - x^4 + \frac{3}{20}x^5 + 2x^6 + \frac{1}{5}x^7 - \frac{1}{8}x^8 \\ &\quad + \int_0^x (t^3 - x^2)u_1 + (12t^2 - x)u_2 dt, \\ u_2'(x) &= 4 - 8x - \frac{1}{3}x^3 + 2x^4 - \frac{8}{5}x^5 + \frac{1}{30}x^6 - 4e^x \\ &\quad + \int_0^x (t - x)u_1 + 8(1 - t)u_2 + 2u_3 dt, \\ u_3'(x) &= 3 - \frac{7}{2}x^2 + \frac{4}{3}x^3 + \frac{6}{5}x^5 - \frac{7}{30}x^6 + \int_0^x (2x - t)u_1 + 6tu_2 + u_3 dt, \end{aligned} \tag{3.2}$$

together with the initial conditions

$u_1(0) = 0,\ u_2(0) = 1, u_3(0) = 2$

The exact solution of system (3.2) [41] is

$u_1(x) = x^4 - 2x,\ u_2(x) = 1 - x^3,\ u_3(x) = x + 2e^x.$

We find an approximate solution to a system (3.2) using the aforementioned methods.

Tables 3.4 and 3.5 contain the exact and numerical solutions using algorithm 3.1 for system (3.2) together with the resulting error.

Table 3.4: The exact and numerical solutions of applying Algorithm 3.1 for system (3.2**).**

x	$Exact\ solution$ $u_1(x) = x^4 - 2x$	$Numerical\ solution$ u_{1app}	$Exact\ solution$ $u_2(x) = 1 - x^3$	$Numerical\ solution$ $u_{2\ app}$	$Exact\ solution$ $u_3(x) = x + 2e^x$	$Numerical\ solution$ $u_{3\ app}$
0	0.	0.	1.	1.	2	2.
0.1	-0.1999	-0.19989013671875	0.999	0.9989995956420898	2.3103418361512955	2.310341827571392
0.2	-0.3984	-0.3984375	0.992	0.991999626159668	2.64280551632034	2.6428054869174957
0.3	-0.5919	-0.59185791015625	0.973	0.9730005264282227	2.999717615152006	2.999717593193054
0.4	-0.7744	-0.77435302734375	0.936	0.9360003471374512	3.3836493952825406	3.3836494088172913
0.5	-0.9375	-0.9375	0.875	0.875	3.7974425414002564	3.7974424362182617
0.6	-1.0704	-1.07037353515625	0.784	0.783998966217041	4.244237600781018	4.244237184524536
0.7	-1.1599	-1.15985107421875	0.657	0.6569967269897461	4.727505414940953	4.727502703666687
0.8	-1.1904	-1.1903076171875	0.488	0.4879894256591797	5.251081856984936	5.25106954574585
0.9	-1.1439	-1.14398193359375	0.271	0.27097034454345703	5.8192062223139	5.819161415100098

Table 3.5 shows the resulting error of using the numerical solution.

x	$Absolute\ error$ $\lvert u_1 - u_{1app}\rvert$	$Absolute\ error$ $\lvert u_2 - u_{2app}\rvert$	$Absolute\ error$ $\lvert u_3 - u_{3app}\rvert$
0	0.	0.	0.
0.1	0.000009863281250022116	$4.043579101553618 \times 10^{-7}$	$8.579903454375426 \times 10^{-9}$
0.2	0.000037499999999968114	$3.738403320241446 \times 10^{-7}$	$2.940284415942074 \times 10^{-8}$
0.3	0.000042089843750092726	$5.264282226802308 \times 10^{-7}$	$2.195895199008646 \times 10^{-8}$
0.4	0.00004697265624997726	$3.471374512287184 \times 10^{-7}$	$1.353475065357656 \times 10^{-8}$
0.5	0.	0.	$1.051819946695786 \times 10^{-7}$
0.6	0.00002646484375001812	0.00000103378295890355	$4.16256481727828 \times 10^{-7}$
0.7	0.00004892578124993108	0.000003273010253823649	0.000002711274266431473
0.8	0.00009238281249990266	0.00001057434082019082	0.000012311239085960324
0.9	0.00008193359375008313	0.00002965545654287638	0.00004480721380240027

The accuaracy of the results in the reconstruction of variational iteration method to solve system (3.2) has a max error of $u_1 \approx 9.44 \times 10^{-5}$ and max error of $u_2 \approx 2.98 \times 10^{-5}$ and max error of $u_3 \approx 5.11 \times 10^{-5}$

Figure 3.6.a compares the exact solution $u_1(x) = x^4 - 2x$ and the approximate solution with $M = 3$

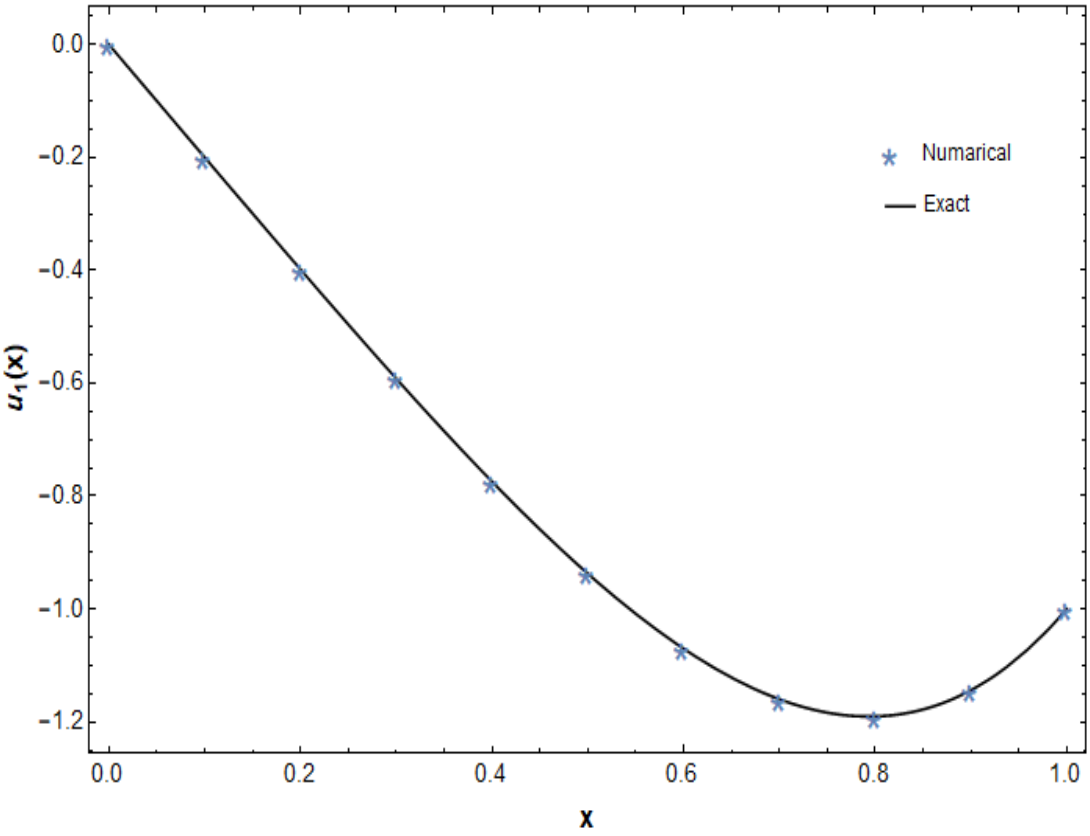

Figure 3.6.a: The exact and numerical solutions of u_1 using Algorithm 3.1 for system (3.2)

Figure 3.6.b compares the exact solution $u_2(x) = 1 - x^3$ and the approximate solution with $M = 3$

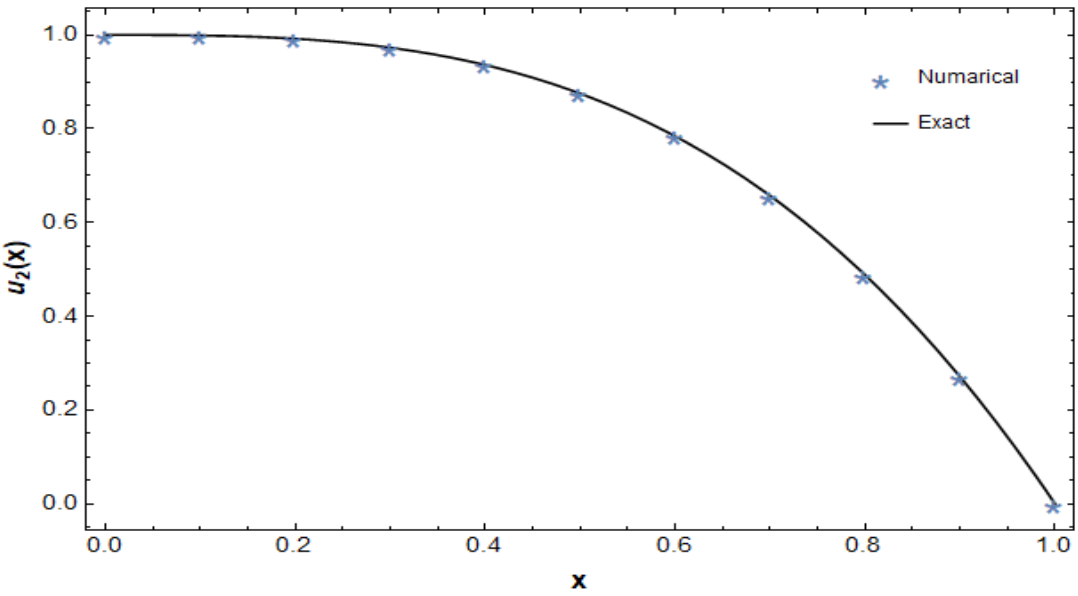

Figure 3.6.b: The exact and numerical solutions of u_2 using Algorithm 3.1 for system (3.2)

Figure 3.6.c compares the exact solution $u_3(x) = x + 2e^x$ and the approximate solution with $M = 3$

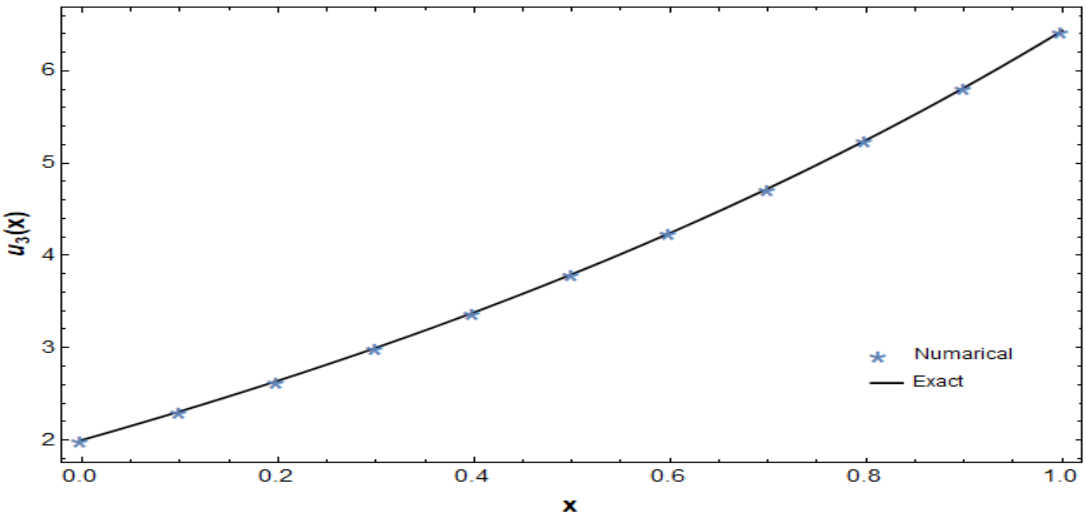

Figure 3.6.c: The exact and numerical solutions of u_3 using Algorithm 3.1 for system (3.2)

Figure 3.7 shows the absolute error resulting of applying algorithm 3.1 for system (3.2)

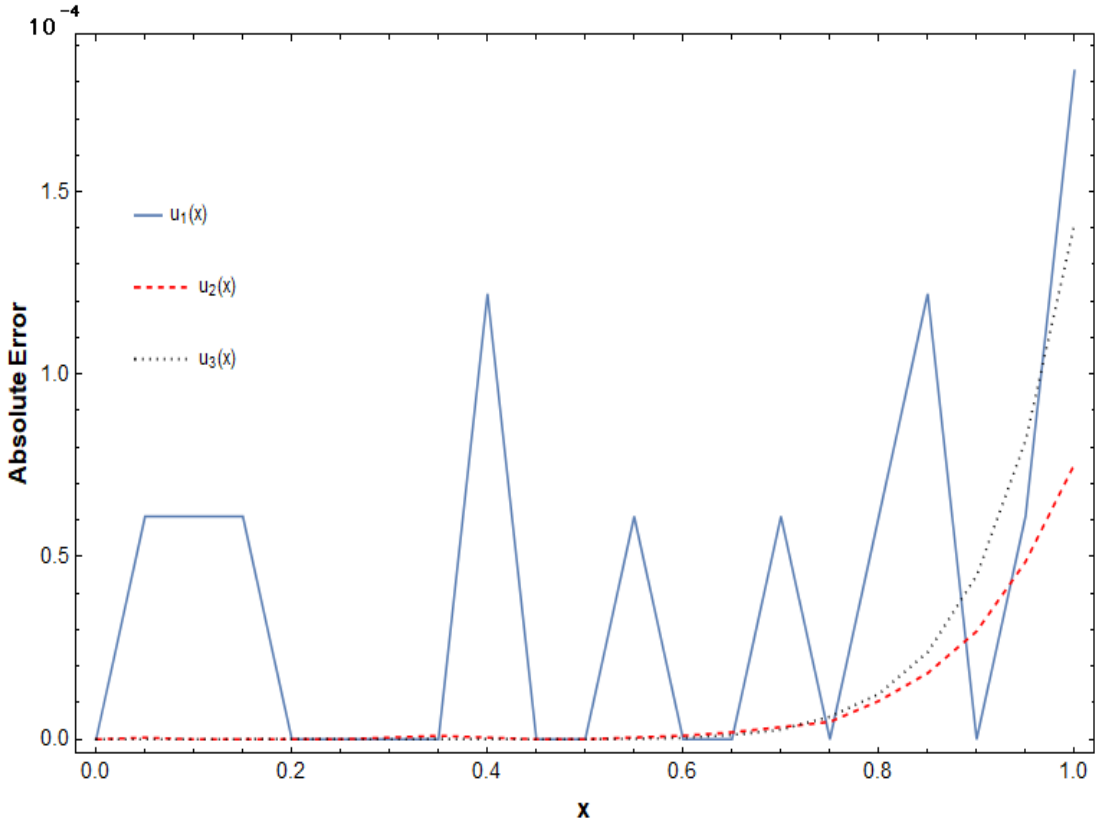

Figure 3.7: The resulting error of u_1& u_2 & $u_3$after applying Algorithm 3.1 for system (3.2)

Tables 3.6 and 3.7 contain the exact and numerical solutions using algorithm 3.2 for system (3.2) together with the resulting error.

Table 3.6: The exact and numerical solutions of applying Algorithm 3.2 for system (3.2**).**

x	$Exact\ solution$ $u_1(x) = x^4 - 2x$	$Numerical\ solution$ u_{1app}	$Exact\ solution$ $u_2(x) = 1 - x^3$	$Numerical\ solution$ $u_{2\ app}$	$Exact\ solution$ $u_3(x) = x + 2e^x$	$Numerical\ solution$ $u_{3\ app}$
0	0.	0.	1.	1.	2	2.
0.1	−0.1999	−0.1998999986410037	0.999	0.9989999996164405	2.3103418361512955	2.310341836911406
0.2	−0.3984	−0.39840000469382025	0.992	0.992000001268595	2.64280551632034	2.642805513660997
0.3	−0.5919	−0.5919000795392859	0.973	0.9730000216099085	2.999717615152006	2.9997175706314225
0.4	−0.7744	−0.774400228030929	0.936	0.9360000615945837	3.3836493952825406	3.3836492678242998
0.5	−0.9375	−0.9375003156846542	0.875	0.8750000858865024	3.7974425414002564	3.7974423632300143
0.6	−1.0704	−1.0704001918522101	0.784	0.784000056734032	4.244237600781018	4.244237484652817
0.7	−1.1599	−1.1598998894031196	0.657	0.6569999842789827	4.727505414940953	4.72750545433764
0.8	−1.1904	−1.190399643757478	0.488	0.4879999282244343	5.251081856984936	5.251082019406921
0.9	−1.1439	−1.1438996142131692	0.271	0.2709999270259329	5.8192062223139	5.819206393371196

Table 3.7 shows the resulting error of using the numerical solution.

x	$Absolute\ error$ $\lvert u_1 - u_{1app}\rvert$	$Absolute\ error$ $\lvert u_2 - u_{2app}\rvert$	$Absolute\ error$ $\lvert u_3 - u_{3app}\rvert$
0.0	0.	0.	0.
0.1	$1.358996248868038 \times 10^{-9}$	$3.835595174805917 \times 10^{-10}$	$7.601106410959346 \times 10^{-10}$
0.2	$4.693820221390865 \times 10^{-9}$	$1.268595006820305 \times 10^{-9}$	$2.659342968058808 \times 10^{-9}$
0.3	$7.953928582438152 \times 10^{-8}$	$2.160990852928535 \times 10^{-8}$	$4.452058366410938 \times 10^{-8}$
0.4	$2.280309290281224 \times 10^{-7}$	$6.159458376675531 \times 10^{-8}$	$1.274582408505864 \times 10^{-7}$
0.5	$3.156846541951807 \times 10^{-7}$	$8.588650235452633 \times 10^{-8}$	$1.781702421155273 \times 10^{-7}$
0.6	$1.91852210118526 \times 10^{-7}$	$5.673403202788307 \times 10^{-8}$	$1.161282012773767 \times 10^{-7}$
0.7	$1.10596880320557 \times 10^{-7}$	$1.572101726576846 \times 10^{-8}$	$3.939668680175146 \times 10^{-8}$
0.8	$3.562425212599862 \times 10^{-7}$	$7.177556560211684 \times 10^{-8}$	$1.624219851947828 \times 10^{-7}$
0.9	$3.857868307033385 \times 10^{-7}$	$7.297406701134435 \times 10^{-8}$	$1.710572963276035 \times 10^{-7}$

The accuaracy of the results in the Sinc collocation method based on sinc functions to solve system (3.2) has a max error of $u_1 \approx 3.9 \times 10^{-7}$ and max error of $u_2 \approx 8.5 \times 10^{-8}$ and max error of $u_3 \approx 1.79 \times 10^{-7}$

Figure 3.8.a compares the exact solution $u_1(x) = x^4 - 2x$ and the approximate solution with $M = 8$

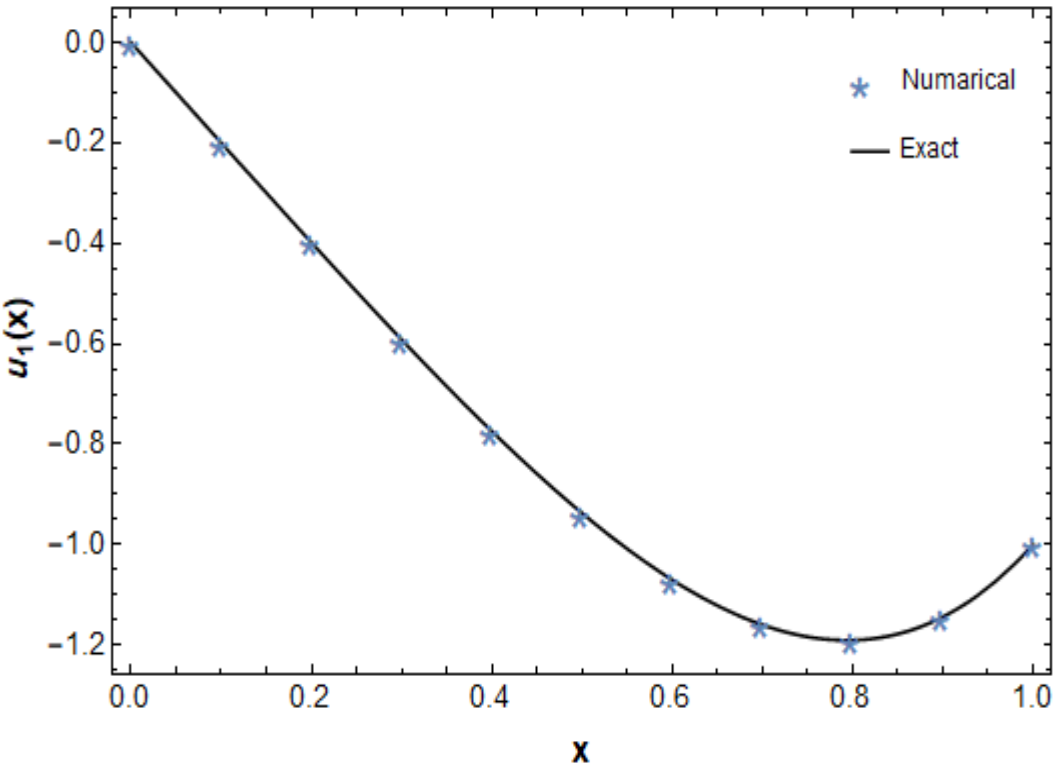

Figure 3.8.a: The exact and numerical solutions of u_1 using Algorithm 3.2 for system (3.2)

Figure 3.8.b compares the exact solution $u_2(x) = 1 - x^3$ and the approximate solution with $M = 8$

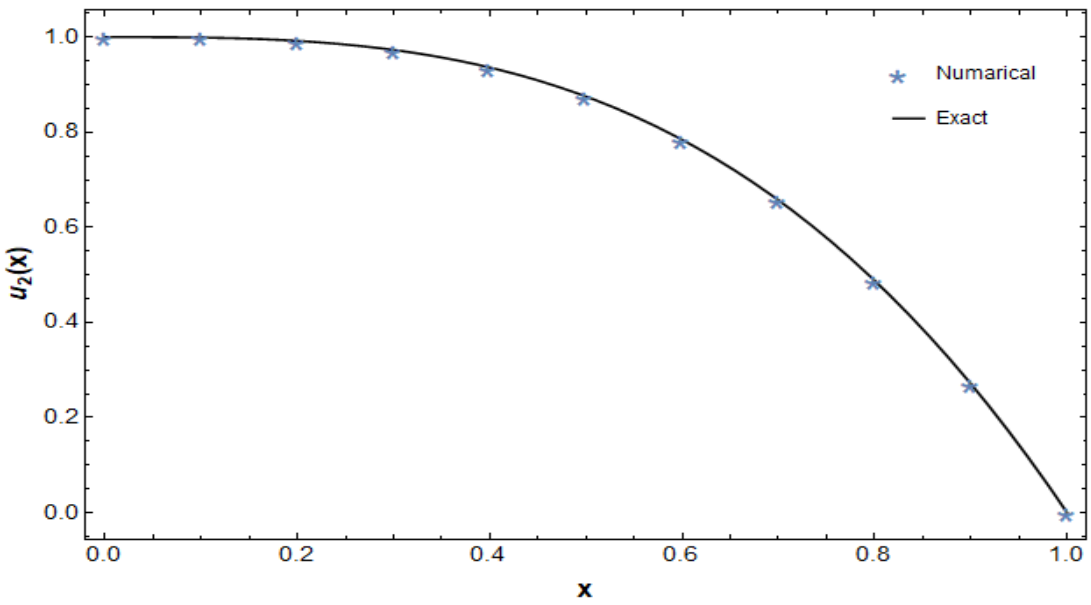

Figure 3.8.b: The exact and numerical solutions of u_2 using Algorithm 3.2 for system (3.2)

Figure 3.8.c compares the exact solution $u_3(x) = x + 2e^x$ and the approximate solution with $M = 8$

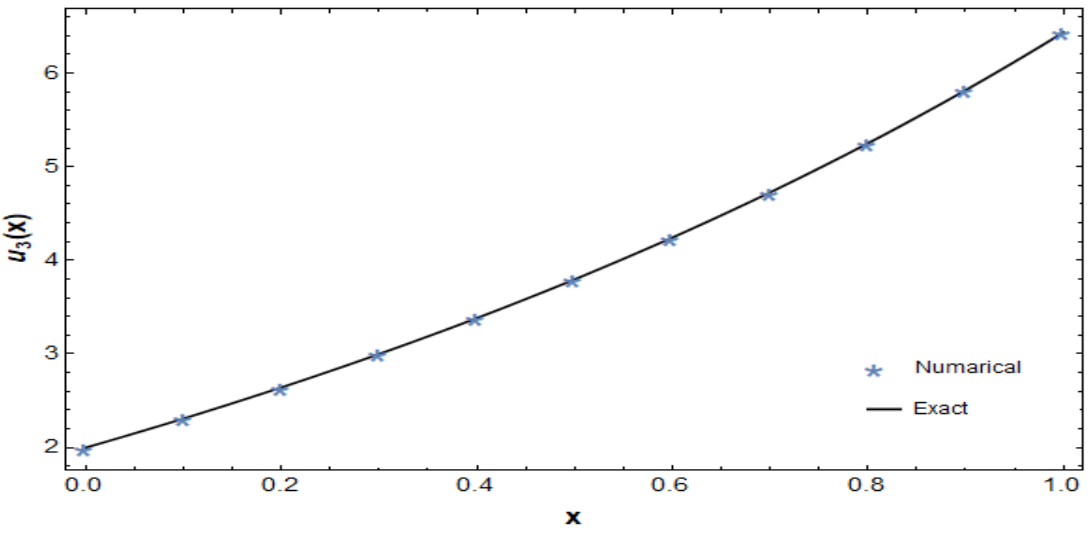

Figure 3.8.c: The exact and numerical solutions of u_3 using Algorithm 3.2 for system (3.2)

Figure 3.9 shows the absolute error resulting of applying algorithm 3.2 for system (3.2)

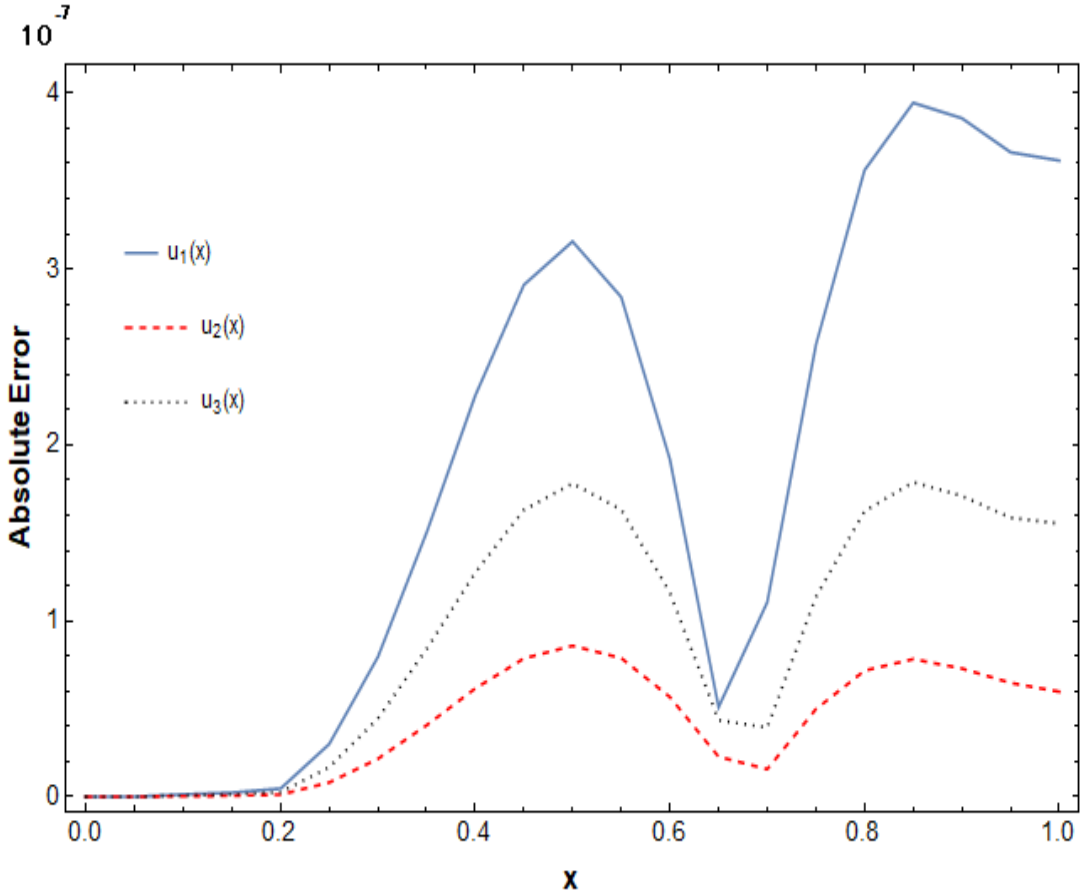

Figure 3.9: The resulting error of u_1& u_2 & u_3 after applying Algorithm 3.2 for system (3.2)

Tables 3.8 and 3.9 contain the exact and numerical solutions using algorithm 3.3 for system (3.2) together with the resulting error.

Table 3.8: The exact and numerical solutions of applying Algorithm 3.3 for system (3.2**).**

	$Exact\ solution\ u_1(x) = x^4 - 2x$	$Numerical\ solution\ u_{1app}$	$Exact\ solution\ u_2(x) = 1 - x^3$	$Numerical\ solution\ u_{2\ app}$	$Exact\ solution\ u_3(x) = x + 2e^x$	$Numerical\ solution\ u_{3\ app}$
0	0.	$2.690833666996184 \times 10^{-13}$	1.	0.9999999999543171	2	1.9999999974820484
0.1	-0.1999	-0.19990000000022942	0.999	0.9990000000727636	2.3103418361512955	2.3103418351947798
0.2	-0.3984	-0.39839999999984066	0.992	0.991999999990548	2.64280551632034	2.6428055151638277
0.3	-0.5919	-0.5919000000004384	0.973	0.9729999999906025	2.999717615152006	2.999717617646919
0.4	-0.7744	-0.7744000000009819	0.936	0.9360000000694738	3.3836493952825406	3.3836493955712172
0.5	-0.9375	-0.9374999999997702	0.875	0.875000000131186	3.7974425414002564	3.7974425389296056
0.6	-1.0704	-1.0703999999956941	0.784	0.7840000001191981	4.244237600781018	4.244237600772698
0.7	-1.1599	-1.1598999999883794	0.657	0.6570000000729469	4.727505414940953	4.727505417496077
0.8	-1.1904	-1.190399999977122	0.488	0.4880000000954549	5.251081856984936	5.2510818561222665
0.9	-1.1439	-1.1438999999585011	0.271	0.27100000021849296	5.8192062223139	5.819206221276965

Table 3.9 shows the resulting error of using the numerical solution.

x	$Absolute\ error$ $\lvert u_1 - u_{1app} \rvert$	$Absolute\ error$ $\lvert u_2 - u_{2app} \rvert$	$Absolute\ error$ $\lvert u_3 - u_{3app} \rvert$
0	$2.690833666996184 \times 10^{-13}$	$4.568290190576363 \times 10^{-11}$	$2.517951624980696 \times 10^{-9}$
0.1	$2.293998324631729 \times 10^{-13}$	$7.276357294472291 \times 10^{-11}$	$9.565157554902726 \times 10^{-10}$
0.2	$1.593725151849412 \times 10^{-13}$	$9.451994742448733 \times 10^{-12}$	$1.156512219324668 \times 10^{-9}$
0.3	$4.383160501220118 \times 10^{-13}$	$9.397482791939638 \times 10^{-12}$	$2.494912720862885 \times 10^{-9}$
0.4	$9.818812429784884 \times 10^{-13}$	$6.947387110045611 \times 10^{-11}$	$2.886766381493544 \times 10^{-10}$
0.5	$2.298161660974074 \times 10^{-13}$	$1.311859509911528 \times 10^{-10}$	$2.470650795061146 \times 10^{-9}$
0.6	$4.305888978706207 \times 10^{-12}$	$1.191982068604602 \times 10^{-10}$	$8.319567257331073 \times 10^{-12}$
0.7	$1.162048235414658 \times 10^{-11}$	$7.294698178839099 \times 10^{-11}$	$2.555123224112776 \times 10^{-9}$
0.8	$2.287792177924075 \times 10^{-11}$	$9.545503276697787 \times 10^{-11}$	$8.62669047307918 \times 10^{-10}$
0.9	$4.14988043928588 \times 10^{-11}$	$2.184930569804066 \times 10^{-10}$	$1.0369349823236 \times 10^{-9}$

The accuaracy of the results in the Chebyshev wavelets method to solve system (3.2) has a max error of $u_1 \approx 7.7 \times 10^{-11}$ and max error of $u_2 \approx 2.5 \times 10^{-10}$ and max error of $u_3 \approx 2.6 \times 10^{-9}$

Figure 3.10.a compares the exact solution $u_1(x) = x^4 - 2x$ and the approximate solution with $M = 8$

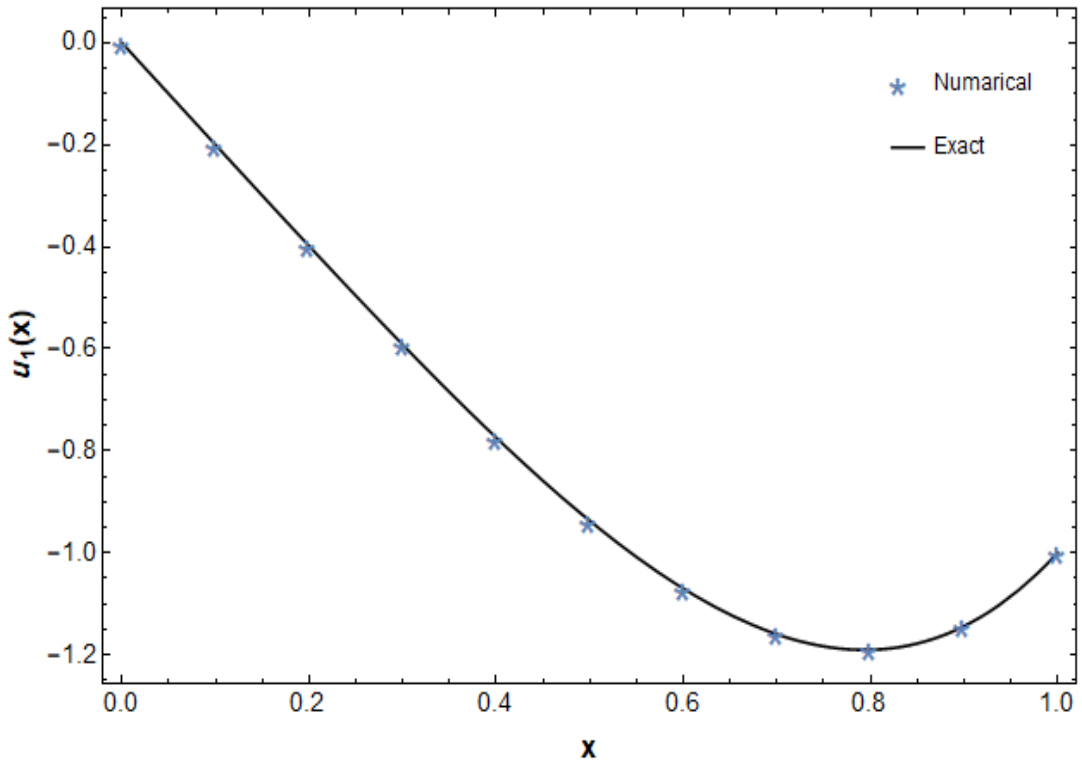

Figure 3.10.a: The exact and numerical solutions of u_1 using Algorithm 3.3 for system (3.2)

Figure 3.10.b compares the exact solution $u_2(x) = 1 - x^3$ and the approximate solution with $M = 8$

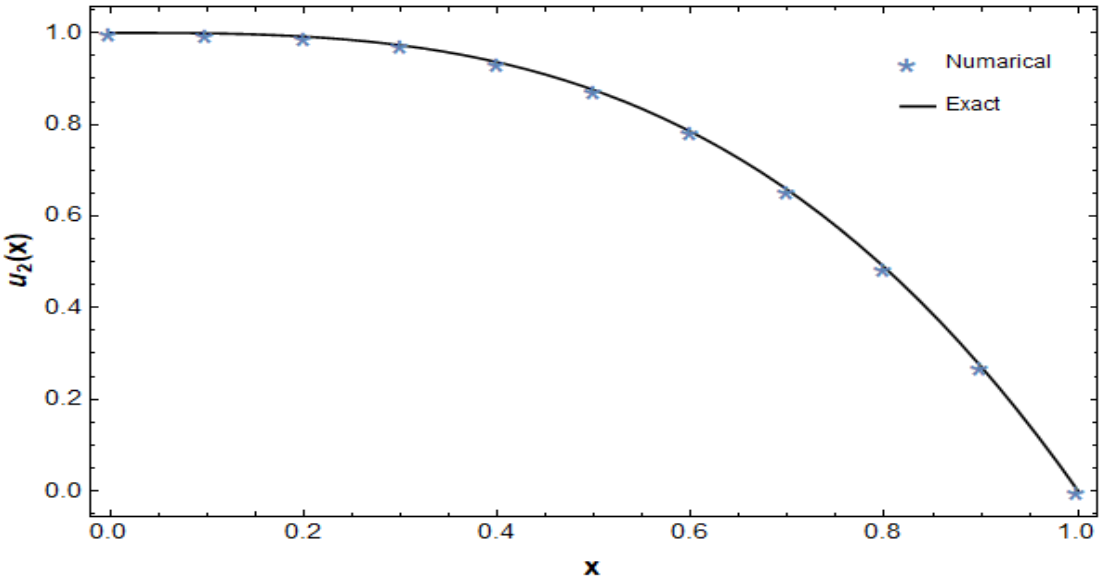

Figure 3.10.b: The exact and numerical solutions of u_2 using Algorithm 3.3 for system (3.2)

Figure 3.10.c compares the exact solution $u_3(x) = x + 2e^x$ and the approximate solution with $M = 8$

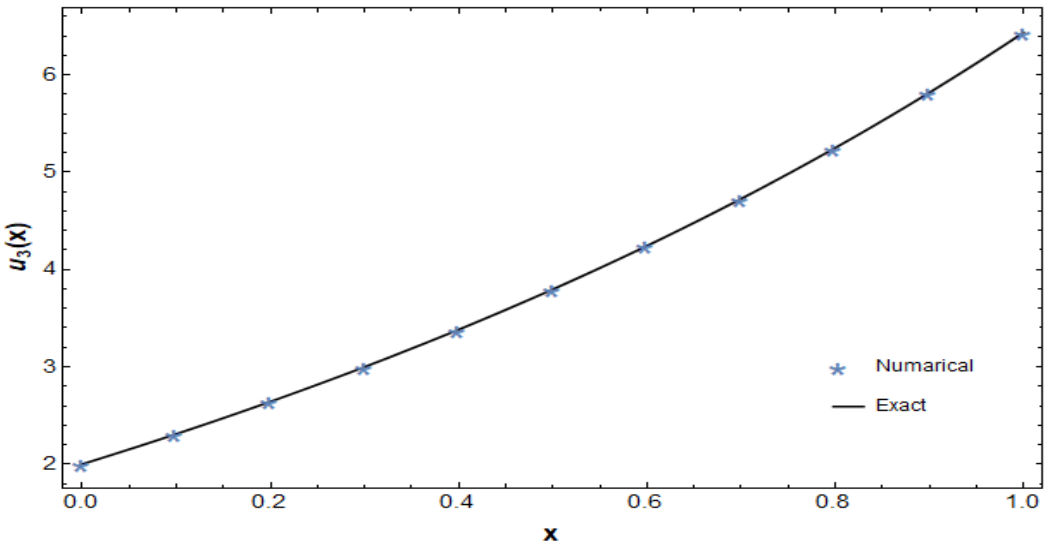

Figure 3.10.c: The exact and numerical solutions of u_3 using Algorithm 3.3 for system (3.2)

Figure 3.11 shows the absolute error resulting of applying algorithm 3.3 for system (3.2)

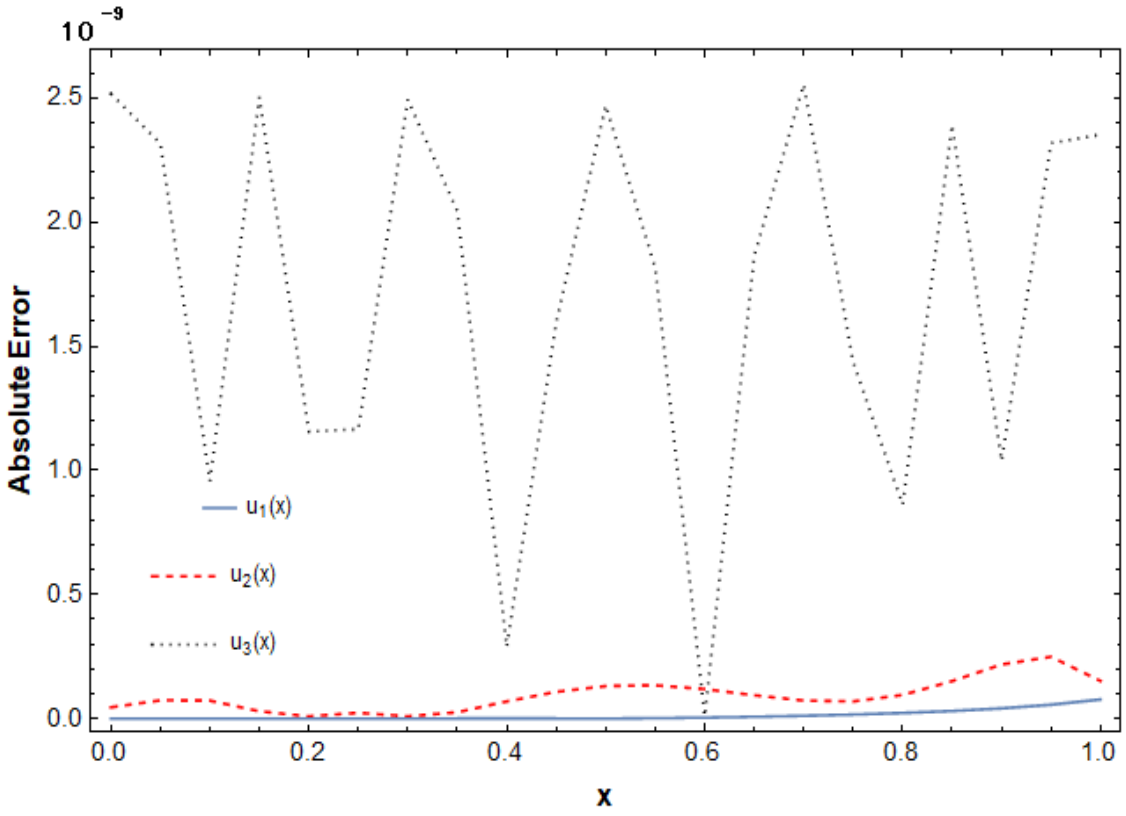

Figure 3.11: The resulting error of u_1& u_2 & u_3 after applying Algorithm 3.3 for system (3.2)

Example 3.3

Consider the system of Volterra integro-differential equations:

$$u_1''(x) = -1 - x + \cosh x - \frac{\sin^3 x}{3} - \sinh x + e^x + \int_0^x (e^{-t})\, u_1 + (\sin^2 t)\, u_2 dt, \tag{3.3}$$

$$u_2''(x) = -3 + x^2 - \frac{2}{3}x^3 - 2e^x(x-1) + \int_0^x (x^2 - t^2)\, u_1 + (x-t)\, u_2 dt,$$

together with the initial conditions

$u_1(0) = 2, u_1'(0) = 1, u_2(0) = 1, u_2'(0) = 0,$

The exact solution of system (3.3) [41] is

$u_1(x) = e^x + 1,\ u_2(x) = \cos x.$

We seek to find an approximate solution to a system (3.3) by the following numerical methods:

Table 3.10 contains the exact and numerical solutions together and the absolute error using algorithm 3.1 for system (3.3).

Table 3.10: The exact and numerical solutions of applying Algorithm 3.1 for system (3.3**).**

x	$Exact\ solution$ $u_1(x) = e^x + 1$	$Numerical\ solution$ u_{1app}	$Absolute\ error$ $\lvert u_1 - u_{1app} \rvert$	$Exact\ solution$ $u_2(x) = \cos x$	$Numerical\ solution$ $u_{2\ app}$	$Absolute\ error$ $\lvert u_2 - u_{2\ app} \rvert$
0	2.	2	0.	1.	1.	0.
0.1	2.1051709180756477	2.105170918075422	$2.255973186038318 \times 10^{-13}$	0.9950041652780258	0.9950041652780328	$6.994405055138486 \times 10^{-15}$
0.2	2.2214027581601696	2.2214027581072777	$5.289191307156216 \times 10^{-11}$	0.9800665778412416	0.980066577840887	$3.54605234065275 \times 10^{-13}$
0.3	2.349858807576003	2.349858806343985	$1.232018043140215 \times 10^{-9}$	0.955336489125606	0.9553364891056901	$1.991584674954083 \times 10^{-11}$
0.4	2.4918246976412703	2.491824686493494	$1.114777647970299 \times 10^{-8}$	0.9210609940028851	0.9210609936577114	$3.451736674264793 \times 10^{-10}$
0.5	2.648721270700128	2.6487212107128038	$5.99873244411242 \times 10^{-8}$	0.8775825618903728	0.8775825587614037	$3.128969083832089 \times 10^{-9}$
0.6	2.822118800390509	2.822118568272824	$2.321176850728079 \times 10^{-7}$	0.8253356149096782	0.8253355960995421	$1.881013611537696 \times 10^{-8}$
0.7	3.0137527074704766	3.013751992402314	$7.150681624601418 \times 10^{-7}$	0.7648421872844884	0.7648421021871812	$8.509730720085429 \times 10^{-8}$
0.8	3.225540928492468	3.225539063307953	0.000001865184514837636	0.6967067093471654	0.6967063969228278	$3.124243376229074 \times 10^{-7}$
0.9	3.45960311115695	3.4595988198549152	0.000042913020346091 46	0.6216099682706644	0.6216089910088343	$9.772618301262526 \times 10^{-7}$

The accuaracy of the results in the reconstruction of variational iteration method to solve system (3.3) has a max error of $u_1 \approx 4.68 \times 10^{-6}$ and max error of $u_2 \approx 1.01 \times 10^{-6}$

Figure 3.12.a compares the exact solution $u_1(x) = e^x + 1$ and the approximate solution with $M = 3$

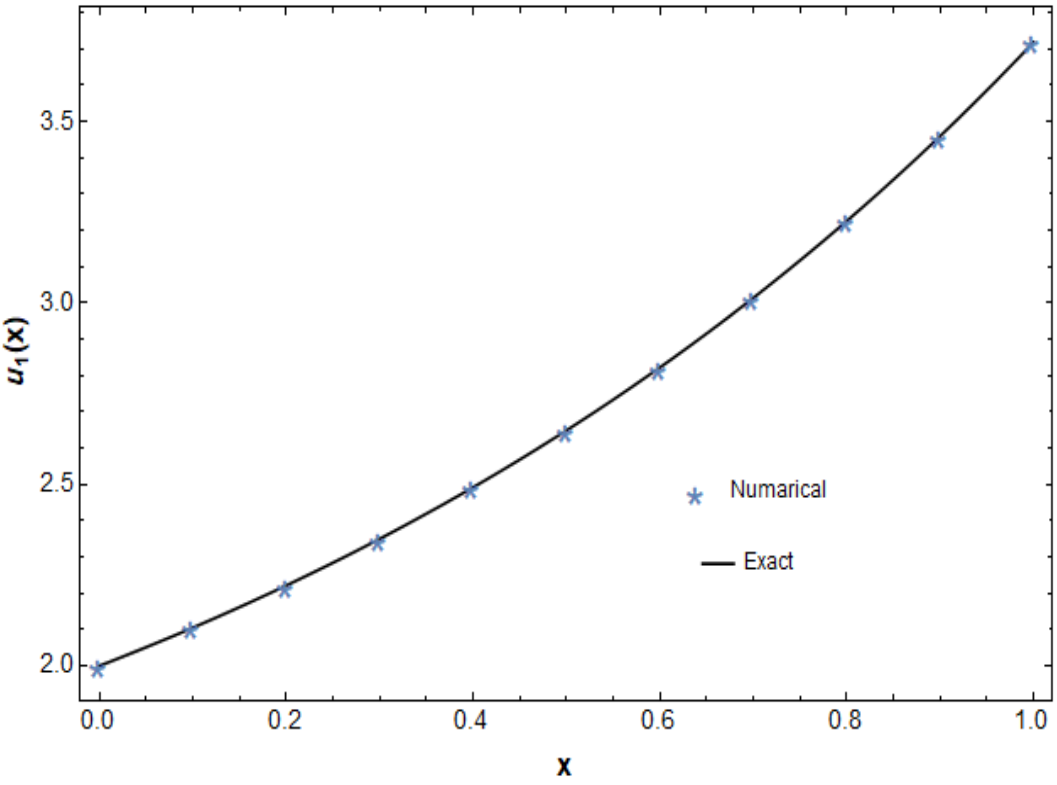

Figure 3.12.a: The exact and numerical solutions of u_1 using Algorithm 3.1 for system (3.3)

Figure 3.12.b compares the exact solution $u_2(x) = \cos x$ and the approximate solution with $M = 3$

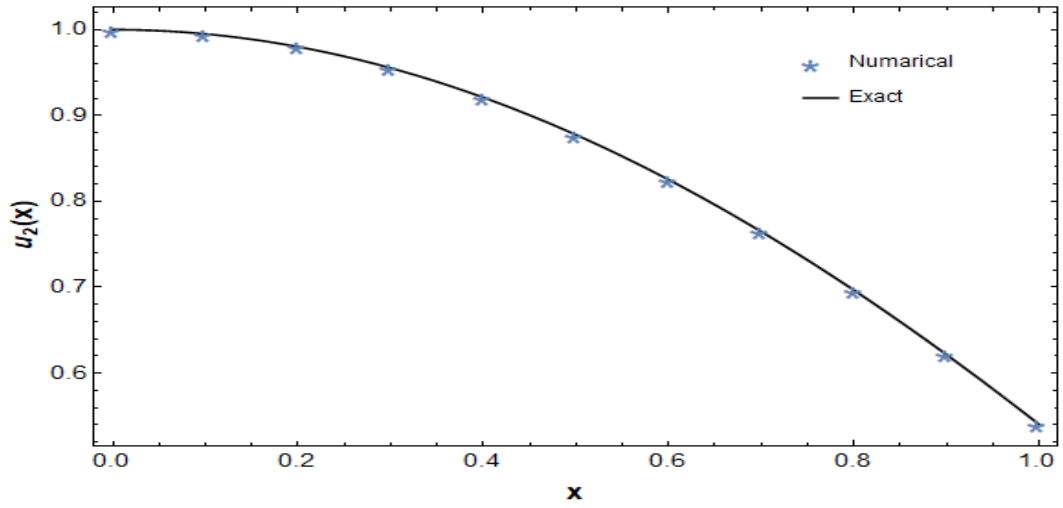

Figure 3.12.b: The exact and numerical solutions of u_2 using Algorithm 3.1 for system (3.3)

Figure 3.13 shows the absolute error resulting of applying algorithm 3.1 for system (3.3)

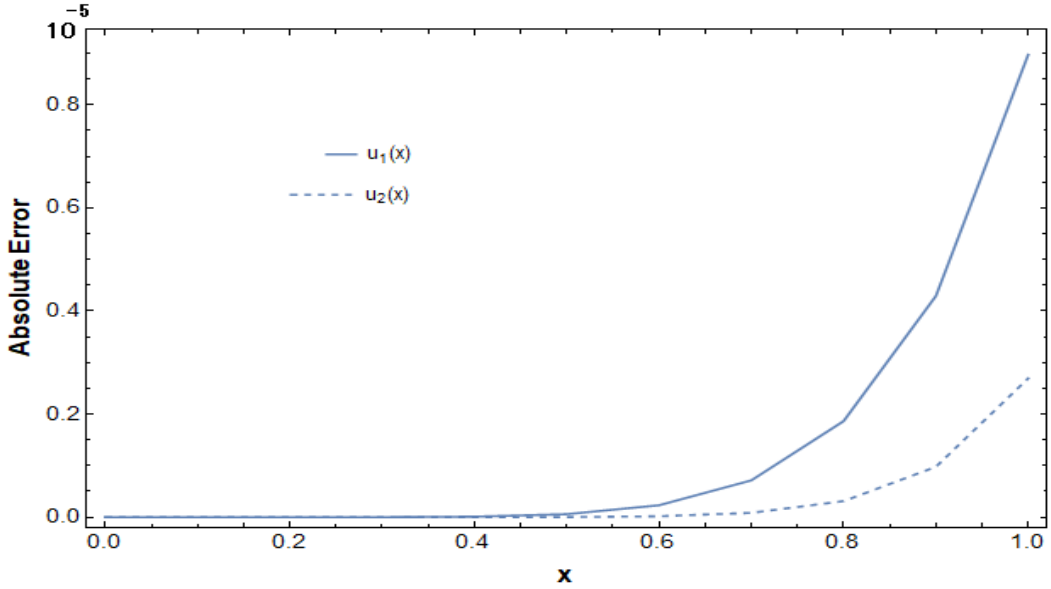

Figure 3.13: The resulting error of u_1& u_2 after applying Algorithm 3.1 for system (3.3)

Table 3.11 contains the exact and numerical solutions together and the absolute error using algorithm 3.1 for system (3.3).

Table 3.11: The exact and numerical solutions of applying Algorithm 3.2 for system (3.3**).**

x	$Exact\ solution$ $u_1(x) = e^x + 1$	$Numerical\ solution$ u_{1app}	$Absolute\ error$ $\lvert u_1 - u_{1app}\rvert$	$Exact\ solution$ $u_2(x) = \cos x$	$Numerical\ solution$ $u_{2\ app}$	$Absolute\ error$ $\lvert u_2 - u_{2\ app}\rvert$
0	2.	2.	0.	1.	1.	0.
0.1	2.1051709180756477	2.1051397636262448	0.00003115444940293699	0.9950041652780258	0.9949953813521188	0.00000878392590697441
0.2	2.2214027581601696	2.2212621771873557	0.00014058097281388626	0.9800665778412416	0.9800269511507578	0.00003962669048385159
0.3	2.349858807576003	2.353150577162296	0.0032917695862928475	0.955336489125606	0.9562638272781463	0.0009273381525403135
0.4	2.4918246976412703	2.5101191909529916	0.018294493311721283	0.9210609940028851	0.9262143139859589	0.005153319983073779
0.5	2.648721270700128	2.694898115680843	0.046176844980714726	0.8775825618903728	0.890587846895059	0.013005285004686229
0.6	2.822118800390509	2.8955876162197285	0.07346881582921938	0.8253356149096782	0.8460218395509642	0.020686224641286
0.7	3.0137527074704766	3.094820636647796	0.08106792917731953	0.7648421872844884	0.7876576255740133	0.022815438289524925
0.8	3.225540928492468	3.289093529586907	0.06355260109443917	0.6967067093471654	0.7145821371224699	0.017875427775304487
0.9	3.45960311115695	3.4945194299151083	0.03491631875815848	0.6216099682706644	0.631444245918761	0.009834277648096634

The accuaracy of the results in the Sinc collocation method based on sinc functions to solve system (3.3) has a max error of $u_1 \approx 0.081$ and max error of $u_2 \approx 0.229$

Figure 3.14.a compares the exact solution $u_1(x) = e^x + 1$ and the approximate solution with $M = 8$

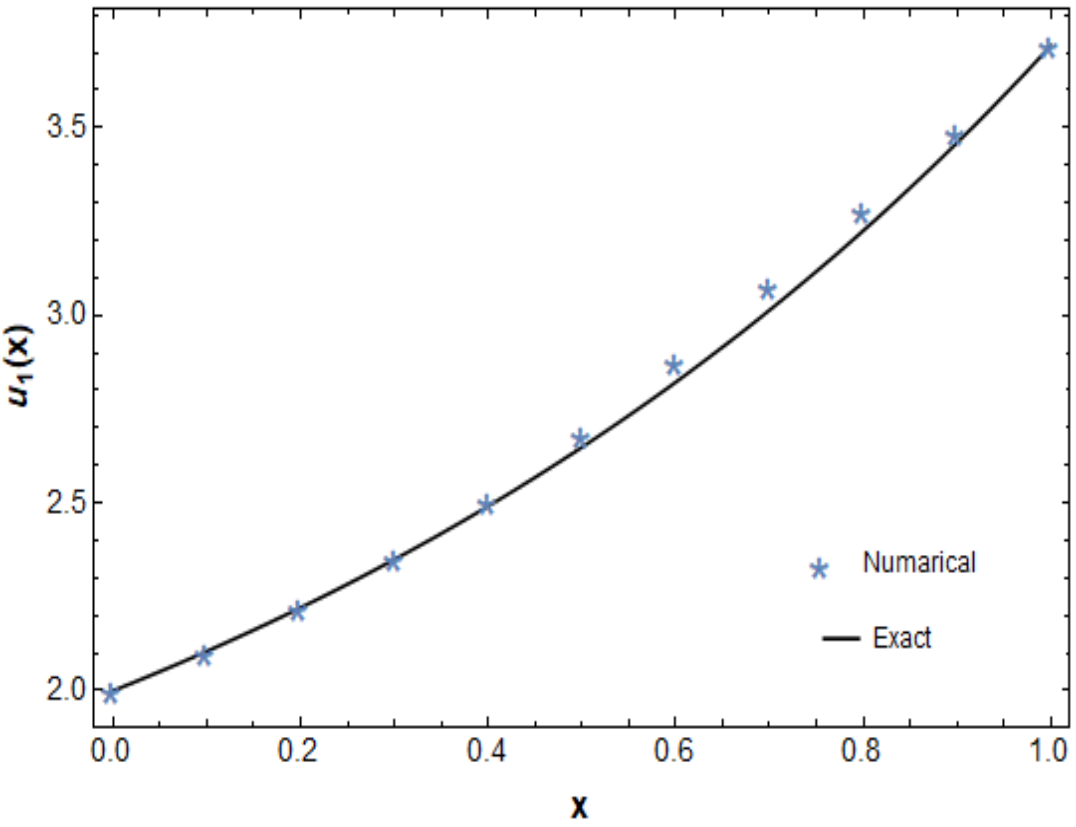

Figure 3.14.a: The exact and numerical solutions of u_1 using Algorithm 3.2 for system (3.3)

Figure 3.14.b compares the exact solution $u_2(x) = \cos x$ and the approximate solution with $M = 8$

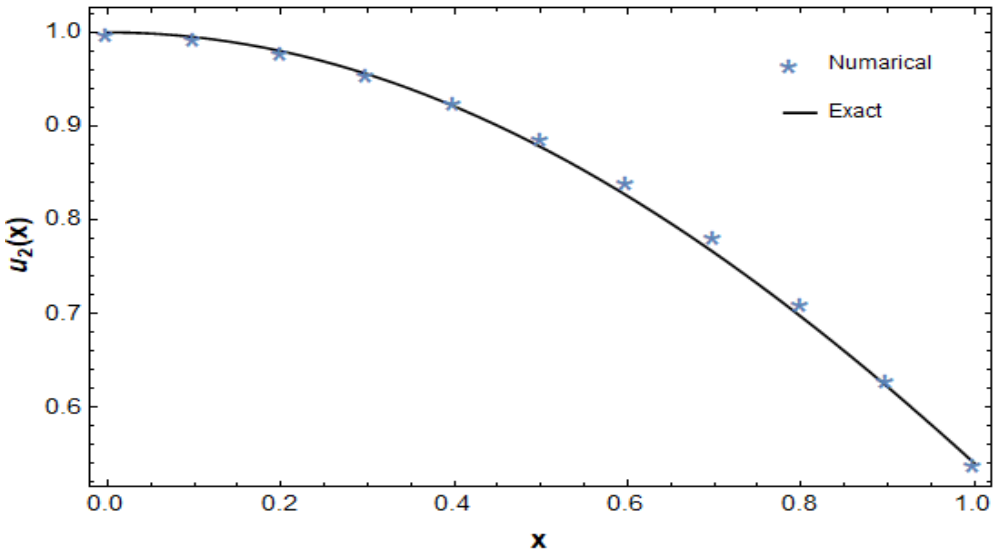

Figure 3.14.b: The exact and numerical solutions of u_2 using Algorithm 3.2 for system (3.3)

Figure 3.15 shows the absolute error resulting of applying algorithm 3.2 for system (3.3)

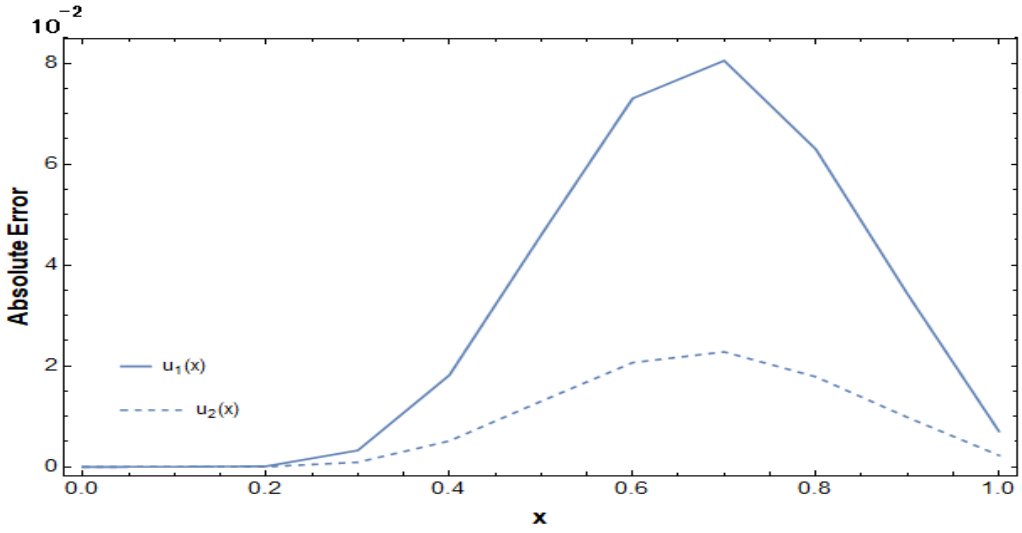

Figure 3.15: The resulting error of u_1& u_2 after applying Algorithm 3.2 for system (3.3)

Table 3.12 contains the exact and numerical solutions together and the absolute error using algorithm 3.3 for system (3.3).

Table 3.12: The exact and numerical solutions of applying Algorithm 3.3 for system (3.3**).**

x	$Exact\ solution$ $u_1(x) = e^x + 1$	$Numerical\ solution$ u_{1app}	$Absolute\ error$ $\lvert u_1 - u_{1app}\rvert$	$Exact\ solution$ $u_2(x) = \cos x$	$Numerical\ solution$ $u_{2\,app}$	$Absolute\ error$ $\lvert u_2 - u_{2\,app}\rvert$
0	2.	1.999999998740645	$1.25935506467556 \times 10^{-9}$	1.	0.9999999993413167	$6.586833301014394 \times 10^{-10}$
0.1	2.1051709180756477	2.105170917580036	$4.956115517984472 \times 10^{-10}$	0.9950041652780258	0.9950041649575281	$3.204977394588582 \times 10^{-10}$
0.2	2.2214027581601696	2.22140275754089	$6.192797386006532 \times 10^{-10}$	0.9800665778412416	0.9800665775316376	$3.096040090966312 \times 10^{-10}$
0.3	2.349858807576003	2.349858808759274	$1.183270814664183 \times 10^{-9}$	0.955336489125606	0.9553364897343352	$6.087291781753379 \times 10^{-10}$
0.4	2.4918246976412703	2.491824697676863	$3.559286199106282 \times 10^{-11}$	0.9210609940028851	0.9210609939384683	$6.441680522328852 \times 10^{-11}$
0.5	2.648721270700128	2.6487212692830155	$1.417112649448881 \times 10^{-9}$	0.8775825618903728	0.8775825611033266	$7.870462059855754 \times 10^{-10}$
0.6	2.822118800390509	2.822118800114823	$2.756861405828203 \times 10^{-10}$	0.8253356149096782	0.8253356147660844	$1.435938035143635 \times 10^{-10}$
0.7	3.0137527074704766	3.01375270838393	$9.134533129895317 \times 10^{-10}$	0.7648421872844884	0.764842187697923	$4.134346198725325 \times 10^{-10}$
0.8	3.225540928492468	3.2255409275988933	$8.935745476890133 \times 10^{-10}$	0.6967067093471654	0.6967067087823426	$5.64822744131277 \times 10^{-10}$
0.9	3.45960311115695	3.459603110051884	$1.105065816631167 \times 10^{-9}$	0.6216099682706644	0.6216099676731845	$5.974798433783235 \times 10^{-10}$

The accuaracy of the results in the Chebyshev wavelets method to solve system (3.3) has a max error of $u_1 \approx 1.7 \times 10^{-9}$ and max error of $u_2 \approx 9.57 \times 10^{-10}$

Figure 3.16.a compares the exact solution $u_1(x) = e^x + 1$ and the approximate solution with $M = 8$

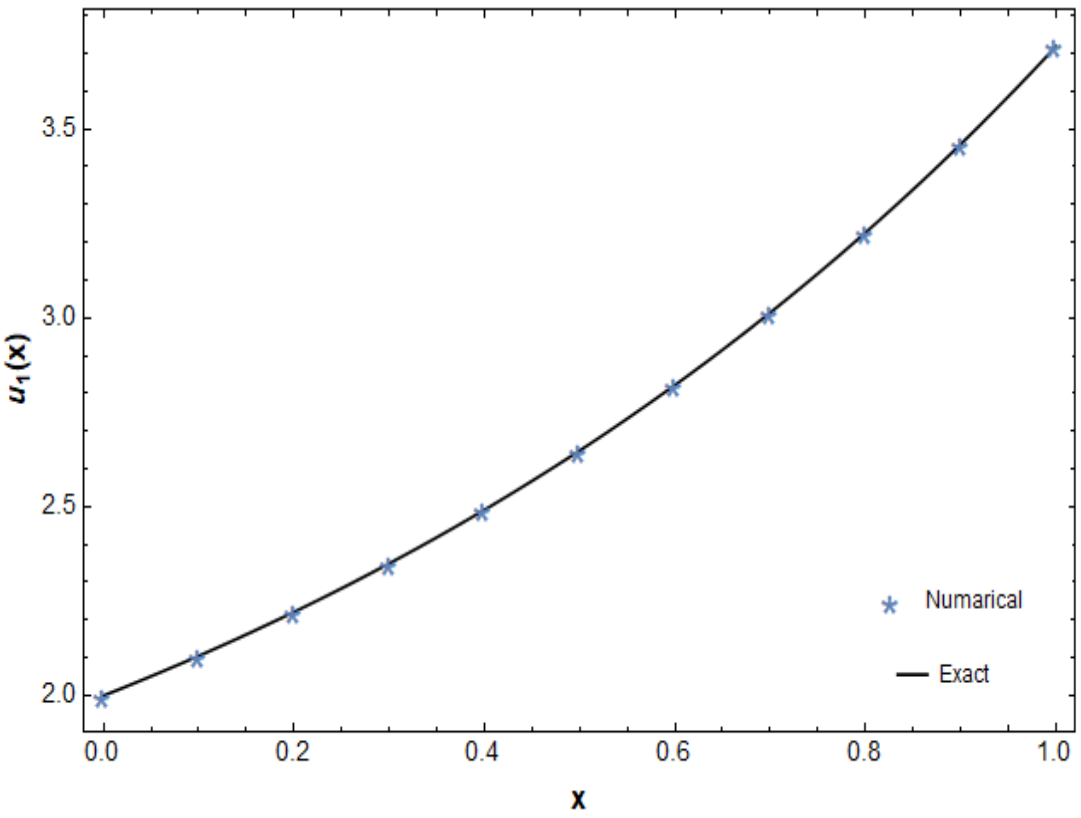

Figure 3.16.a: The exact and numerical solutions of u_1 using Algorithm 3.3 for system (3.3)

Figure 3.16.b compares the exact solution $u_2(x) = \cos x$ and the approximate solution with $M = 8$

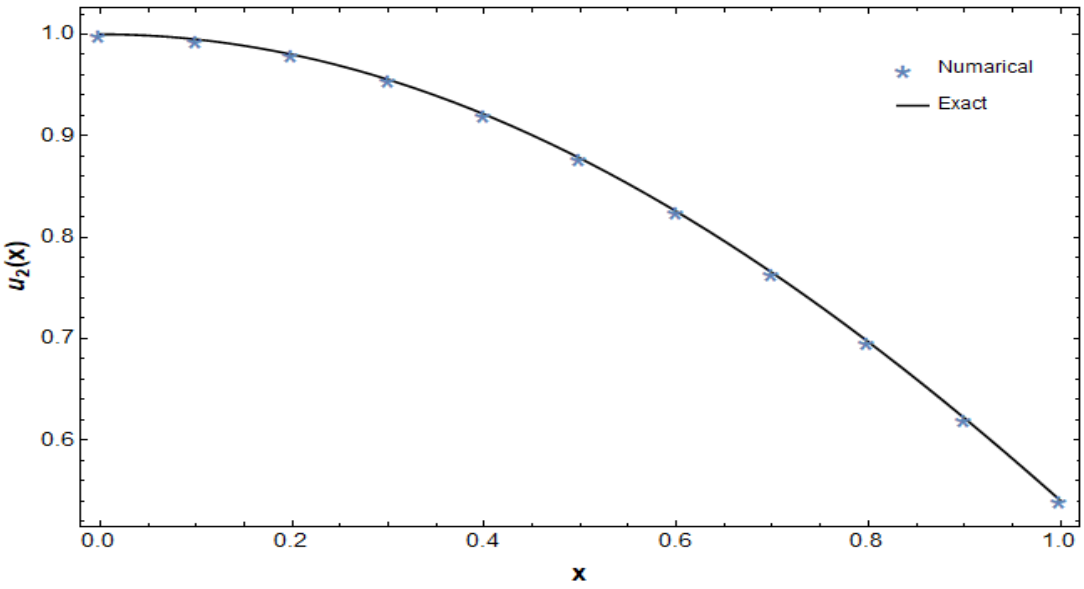

Figure 3.16.b: The exact and numerical solutions of u_2 using Algorithm 3.3 for system (3.3)

Figure 3.17 shows the absolute error resulting of applying algorithm 3.3 for system (3.3)

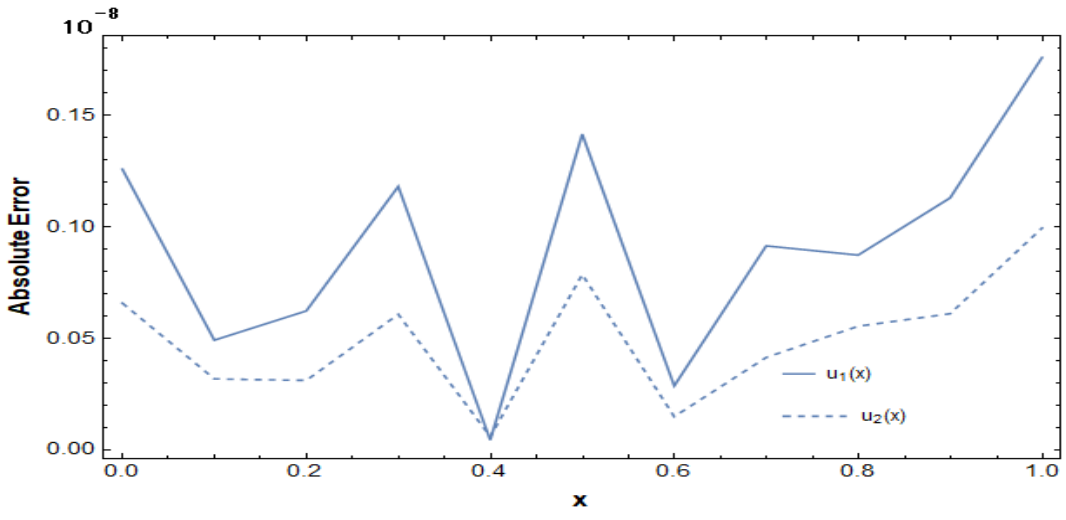

Figure 3.17: The resulting error of u_1& u_2 after applying Algorithm 3.3 for system (3.3)

3.4 Conclusion

In this thesis we have solved a system of linear Volterra integro-differential equations using several numerical methods. These include the Reconstruction of variational iteration method, Sinc collocation method based on sinc functions and Chebyshev wavelets method.

Numerical methods were implemented in the form of algorithms to solve some numerical examples. The results show the following observations:

1) The results for example 3.1 show clearly that the Reconstruction of Variational Iteration Method (RVIM) is more efficient in comparison with the sinc and Chebyshev methods. This is because the RVIM is very well known for its fast convergence and consequently requires less CPU time in the case when the kernel and the exact solution are polynomials.

2) Examples 3.2 and 3.3 show that Chebyshev wavelet method is more effective than counterparts used in this study for solving system of linear Volterra integro-differential equations in particular when the kernel and the exact solution are not polynomials.

References

[1] S. Abbasbandy and A. Taati, ***Numerical solution of the system of nonlinear Volterra integro-differential equations with nonlinear differential part by the operational tau method and error estimation***, **Journal of Computation and Applied Mathematics**, Vol. 231, Issue 1, 2009, 106–113.

[2] A. Akyuz and M. Sezer, ***Chebyshev polynomial solutions of systems of higher-order linear Fredholm–Volterra integro-differential equations***, **Journal of the Franklin Institute** 342, 2005, 688–701.

[3] O. Al-Faour and R. Saeed, ***Solution of a system of linear Volterra integral and integro-differential equations by spectral method***, **Al-Nahrain Univ. J. Sci.**Vol. 6, 2006, 30–46.

[4] S. Alkan, **A numerical method for solution of integro-differential equations of fractional order**, Sakarya Üniversitesi Fen Bilimleri Enstitüsü Dergisi,Vol. 21, Issue 2, 2017, 82-89.

[5] H. Aminikhah and S. Hosseini, **Numerical solution of linear system of integro-differential equations by using wavelet method**, Math. Sci. Letter, No. 4, No. 1, 2015, 45-50.

[6] M. Araghia , S. Daliri and M. Bahmanpour, ***Numerical Solution of Integro-Differential Equation by using Chebyshev Wavelet Operational Matrix of Integration***, **International Journal of Mathematical Modelling and Computations** Vol. 02, No. 02, 2012, 127 – 136.

[7] A. Arikoglu and I. Ozkol, **Solutions of integral and integro-differential equation systems by using differential transform method**, Comput. Math. Appl. 56, 2008, 2411-2417.

[8] E. Babolian and F. Fattahzadeh, **Numerical computation method in solving integral equations by using Chebyshev wavelet operational matrix of integration**, Applied Mathematics and Computation, Vol. 188, Issue 1, 2007, 1016-1022.

[9] M. Bagheri and E. Miralikatouli, ***Comparison Differential Transform Method with Homotopy Perturbation Method for Nonlinear Integral Equations***, **Journal of Mathematics and Computer Science** Vol .5 No.4, 2012, 288-296.

[10] C. Baker and A. Tang, **Stability analysis of continuous implicit Runge Kutta methods for Volterra integro-differential systems with unbounded delays**, Applied Numerical Mathematics. 24, 1997, 153-173.

[11] J. Biazar and H. Ebrahimi, **A Strong Method for solving systems of integro-differential equations**, Applied Mathematics, Vol. 2 No. 9, 2011, 1105-1113.

[12] J. Biazar, H. Ghazvini and M. Eslami, **He's homotopy perturbation method for systems of integro-differential equations**, Vol. 39, Issue 3, 2009, 1253-1258.

[13] J. Biazar, **Solution of Systems of Integral-Differential Equations by Adomian Decomposition Method**, Applied Mathematics and Computation, Vol. 168, No. 2, 2005, 1232-1238.

[14] F. Bloom, ***Concavtty arguments and growth estimates for damped linear integro differential equations with applications to a class of holohedral isotropic dielectrics***, **Journal of Applied Mathematics and Physics (ZAMP)** Vol. 29, Issue 4, 644–663.

[15] T. Bo, L. Xie and X. Zheng, ***Numerical approach to wind ripple in desert***, **International Journal of Nonlinear Sciences and Simulation**, Vol. 8, Issue 2, 2007, 223-228.

[16] H. Brunner, **Theory and numerical solution of Volterra functional integral equations**, Hong Kong Baptist University, 2010.

[17] R. Chandra, G. Sekar and K. Murugesan, **System of linear second order Volterra integro-differential equations using Single Term Walsh Series technique**, Applied Mathematics and Computation, Vol. 273, 2016, 484–492.

[18] O. Christensen and K. Christensen, **Approximation Theory from Taylor Polynomial to Wavelets**, Springer Science, New York, 2004.

[19] I. Daubeches, **Ten lectures on wavelets**, CBMS-NSF, 1992.

[20] M. El-Gamel, **Sinc-Collocation Method for Solving Linear and Nonlinear System of Second-Order Boundary Value Problems**, Applied Mathematics, vol. 3, No. 11, 2012, 1627-1633.

[21] M. Gachpazan, ***Numerical Scheme to Solve Integro-Difierential Equations System***, **Journal of Advanced Research in Scientific Computing**, Vol. 1, No. 1, 2009, 11-21.

[22] M. Ganesh and I. Sloan, **Optimal order spline methods for nonlinear differential and integro-differential equations**, Applied Mathematics and Computation, Vol. 29, Issue 4, 1999, 445-478.

[23] D. Gottlieb and S. Orszag, **Numerical Analysis of Spectral Methods**, SIAM, Philadelphia, PA, 1997.

[24] E. Hesameddini and E. Asadolahifard, ***Solving systems of linear Volterra integro-differential equations by using Sinc-collocation method***, **International Journal of Mathematical Engineering and Science**, Vol. 2. Issue 7, 2013.

[25] K. Holmaker, **Global Asymptotic Stability For A stationary Solution Of A system of integro-differential equations describing the formation of liver zones**, Siam j. Math. Anal. Vol. 24, No. 1, 1993, 116-128.

[26] T. Jangveladze, Z. Kiguradze and B. Neta, **Finite difference approximation of a nonlinear integro-differential system**, Applied Mathematics and Computation, Vol. 215, Issue 2, 2009, 615-628.

[27] J. Lund and K. Bowers**, Sinc methods for quadrature and differential equations,** PA, Philadelphia, SIAM, 1992.

[28] K. Maleknejad and M. TavassoliKajani, **Solving Linear integro-differential equation system by Galerkin methods with hybrid functions**, Applied Mathematics and Computation, Vol. 159, Issue 3, 2004, 603-612.

[29] K. Maleknejad, F. Mirzae and S. Abbasbandy, **Solving Linear integro-differential equations system by using rationalized Haar functions method**, Applied Mathematics and Computation, Vol. 155, Issue 2, 2004, 317-328.

[30] K. Maleknejad, H. Safdari and M. Nouri, ***Numerical solution of an integral equations system of the first kind by using an operational matrix with block pulse functions***, **International Journal of Systems Science**, Vol. 42, issue 1, 2011, 195-199.

[31] O. Mohammed, Fadhel and M. AL-Safi, **Sinc-Jacobi collocation algorithm for solving the timefractional diffusion-wave eguations**, 2015.

[32] S. Mohyud-Din, H. Khan, M. Arif and M. Rafiq, **Chebyshev wavelet method to nonlinear fractional Volterra–Fredholm integro-differential equations with mixed boundary conditions**, Advances in Mechanical Engineering, Vol. 9, Issue 3, 2017, 1–8.

[33] X. Oton, **Integro-differential equations Regularity theory and Pohozaev identities**, 2014.

[34] J. Pour-Mahmoud, M. Y. Rahimi-Ardabili and S. Shahmorad, **Numerical solution of the system of Fredholm integro-differential equations by the Tau method**, Applied Mathematics and Computation, Vol. 168, Issue 1, 2005, 465-478.

[35] A. Rahimi and E. Hesameddini, **A new numerical scheme for solving systems of integro-differential equations**, Computational Methods for Differential Equations, Vol. 1, No. 2, 2013, 108-119.

[36] J. Saberi-N and M. Tamamgar. **The Variational iteration method : A highly promising method for solving the system of the integro-difierential equations,** Computers and Mathematics with Applications, Vol. 56, Issue 2, 2008, 346-351.

[37] J. Schiff, **The Laplace Transform: Theory and Applications**, 1999.

[38] M. Sezer, ***Chebyshev polynomial solutions of systems of higher-order linear Fredholm–Volterra integro-differential equations***, **Journal of the Franklin Institute**, Vol. 342, Issue 6, 2005, 688–701.

[39] F. Sun, M. Gao et al., ***The fractal dimension of the fractal model of drop-wise condensation and its experimental study***, **International Journal of Nonlinear Sciences and Numerical Simulation** Vol. 8, Issue 2, 2007, 211-222.

[40] V. Volterra, **Theory of Functionals of Integral and Integro-Differential Equations**, Dover, New York, 1959.

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

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

[43] E. Yusufoglu, **An Efficient Algorithm for Solving Integro-Differential Equations System**, Applied Mathematics and Computation, Vol. 192, No. 1, 2007, 51-55.

Appendix

Mathematica Code For RVIM

```
Clear[f1, f2, k1, k2, k11, k22, G, M1, M2, eq1, eq2, v, x, u, Cov, g, uu, vv]

f1[x_] := 1 - 7 x^2 - x^3/6 - 4 x^4 - x^5/20;

f2[x_] := 4 x + 10 x^2/3 - x^3/3 + x^4 - x^5/5;

k1[x_, t_] := x - t;
k11[x_, t_] := 6 x;
k2[x_, t_] := t;
k22[x_, t_] := -(3 x - 2 t);
G[s_] := 1 / s;
g[x_] := InverseLaplaceTransform[G[s], s, x];

Cov[g_, f_, x_] := Integrate[g[x - τ] * f[τ], {τ, 0, x}];


M1[x_] := (f1[x] + Integrate[k1[x, t] * u[t] + k11[x, t] * v[t], {t, 0, x}]);
M2[x_] := (f2[x] + Integrate[k2[x, t] * u[t] + k22[x, t] * v[t], {t, 0, x}]);

eq1 = Cov[g, M1, x] //. {u[t] → uu[t, n], v[t] → vv[t, n]};

eq2 = Cov[g, M2, x] //. {u[t] → uu[t, n], v[t] → vv[t, n]};

Table[uu[t_, 0] := 0;
  vv[t_, 0] := 5 / 3;
  u[x_, n + 1] := (uu[x, 0] + eq1);
  v[x_, n + 1] := (vv[x, 0] + eq2);
  vv[x_, n + 1] := Evaluate[v[x, n + 1]];
  uu[x_, n + 1] := Evaluate[u[x, n + 1]];
  n, {n, 0, 10}];
```

mathematica code for sinc method

```
ClearAll[NN, α, d, h, u, y, P, v, yy, Pp, sinc, s, eq1, eq2, xj,
 equation, variable, neq, sol, ppppp, uaprox, vaprox, w, Ppp, yyy]

NN = 3; α = 1 / 2; d = Pi / 2; h = Sqrt[(Pi * d) / (α * NN)];

u[x_] = y[x] + P[x];

v[x_] = yy[x] + Pp[x];

w[x_] = yyy[x] + Ppp[x];

y[x_] := Sum[c[k] * x^1 * (x - 1)^1 * s[k, h, x], {k, -NN, NN}]

yy[x_] := Sum[cc[k] * x^1 * (x - 1)^1 * s[k, h, x], {k, -NN, NN}]

yyy[x_] := Sum[ccc[k] * x^1 * (x - 1)^1 * s[k, h, x], {k, -NN, NN}]

P[x_] := Sum[a[n] * x^n, {n, 0, 1}]

Pp[x_] := Sum[aa[n] * x^n, {n, 0, 1}]

Ppp[x_] := Sum[aaa[n] * x^n, {n, 0, 1}]

sinc[x_] := If[x == 0, 1, (Sin[Pi * x] / (Pi * x))]

s[k_, h_, x_] := sinc[(x - k * h) / (h)]

k1[x_, t_] := t^3 - x^2;
k11[x_, t_] := 12 t^2 - x;
k111[x_, t_] := 0;
k2[x_, t_] := t - x;
k22[x_, t_] := 8 * (1 - t);
k222[x_, t_] := 2;
k3[x_, t_] := 2 x - t;
k33[x_, t_] := 6 t;
k333[x_, t_] := 1;
```

$$eq1 = u'[x] == -2 + x^2 - x^4 + \frac{3x^5}{20} + 2x^6 + \frac{x^7}{5} - \frac{x^8}{8} +$$
Integrate[k1[x, t] * u[t] + k11[x, t] * v[t] + k111[x, t] * w[t], {t, 0, x}];

$$eq2 = v'[x] == +4 - 4e^x - 8x - \frac{x^3}{3} + 2x^4 - \frac{8x^5}{5} + \frac{x^6}{30} +$$
Integrate[k2[x, t] * u[t] + k22[x, t] * v[t] + k222[x, t] * w[t], {t, 0, x}];

```
sol = NSolve[neq, variable]
```

```
{{c[-3] → 11.263, cc[-3] → 0.654125, ccc[-3] → 2.26741, c[-2] → 10.3128,
  cc[-2] → 1.71098, ccc[-2] → 1.66114, c[-1] → 2.53556, cc[-1] → 0.784575,
  ccc[-1] → 0.940851, c[0] → 1., cc[0] → -1., ccc[0] → 1.43656, c[1] → 6.0982,
  cc[1] → -2.81262, ccc[1] → 2.73024, c[2] → 15.8625, cc[2] → -4.27425,
  ccc[2] → 5.37158, c[3] → 14.3409, cc[3] → -2.97355, ccc[3] → 4.84042,
  a[0] → 0., aa[0] → 1., aaa[0] → 2., a[1] → -1., aa[1] → -1., aaa[1] → 4.43656}}
```

```
readyfunction1 = u[x] //. sol;
readyfunction2 = v[x] //. sol; readyfunction3 = w[x] //. sol;
```

```
Show[{Plot[{(x^4 - 2 x)}, {x, 0, 1}, PlotStyle → {{Black}, Dashed},
    Frame → True, FrameStyle → Directive[Black, 14],
    FrameLabel → {Style["x", {16, Bold}], Style["u1(x)", {16, Bold}]},
    PlotLegends → Placed[{Exact}, {Scaled[{.63, 0.2}], {0, 0.5}}]], ListPlot[
    Partition[Flatten[Table[{n, Re[readyfunction1] //. x → n}, {n, 0, 1, .1}]], 2],
    PlotMarkers → {"*", 30},
    PlotLegends → Placed[{Numarical}, {Scaled[{.6, 0.3}], {0, 0.5}}]]]]
```

```
Show[{Plot[{(1 - x^3)}, {x, 0, 1}, PlotStyle → {{Black}, Dashed},
    Frame → True, FrameStyle → Directive[Black, 14],
    FrameLabel → {Style["x", {16, Bold}], Style["u2(x)", {16, Bold}]},
    PlotLegends → Placed[{Exact}, {Scaled[{.13, 0.2}], {0, 0.5}}]], ListPlot[
    Partition[Flatten[Table[{n, Re[readyfunction2] //. x → n}, {n, 0, 1, .1}]], 2],
    PlotMarkers → {"*", 30},
    PlotLegends → Placed[{Numarical}, {Scaled[{.1, 0.3}], {0, 0.5}}]]]]
```

```
Show[{Plot[{(x + 2 * E^x)}, {x, 0, 1}, PlotStyle → {{Black}, Dashed},
    Frame → True, FrameStyle → Directive[Black, 14],
    FrameLabel → {Style["x", {16, Bold}], Style["u3(x)", {16, Bold}]},
    PlotLegends → Placed[{Exact}, {Scaled[{.63, 0.2}], {0, 0.5}}]], ListPlot[
    Partition[Flatten[Table[{n, Re[readyfunction3] //. x → n}, {n, 0, 1, .1}]], 2],
    PlotMarkers → {"*", 30},
    PlotLegends → Placed[{Numarical}, {Scaled[{.6, 0.3}], {0, 0.5}}]]]]
```

Mathematica code for Chebeysheve wavelet method

```
T[m_, x_] := If[m == 0, 1 / Sqrt[Pi], Sqrt[2 / Pi] ChebyshevT[m, x]]; ψ[k_, n_, m_, x_] :=
 If[(n - 1) / (2^(k - 1)) <= x < (n) / (2^(k - 1)), 2^(k / 2) T[m, 2^k * x - 2 n + 1], 0];
W[x_, n_] := 1 / Sqrt[1 - (2^k * x - 2 n + 1)^2]; k = 1; Dim = 8
```

```
8
```

```
8
```

```
8
```

```
D1M[f_, x_] := Table[Integrate[f * ψ[1, 1, m, x] * W[x, 1], {x, -1, 1}], {m, 0, Dim - 1}];
```

```
D2M[f_, x_, y_] := Table[Integrate[f * ψ[1, 1, m, x] * ψ[1, 1, nn, y] * W[x, 1] * W[y, 1],
   {x, -1, 1}, {y, -1, 1}], {m, 0, Dim - 1}, {nn, 0, Dim - 1}]
```

```
OpM = Table[
   D1M[Assuming[0 < x ≤ 1, Refine[Integrate[ψ[1, 1, m, x], x]]], x], {m, 0, Dim - 1}];
```

```
psi[x_] := Transpose[{Table[ψ[1, 1, m, x], {m, 0, Dim - 1}]}];
```

```
co1 = {Table[c1[m], {m, 0, Dim - 1}]};
```

```
co2 = {Table[c2[m], {m, 0, Dim - 1}]};
```

$$f1 = \left(1 - 7x^2 - \frac{x^3}{6} - 4x^4 - \frac{x^5}{20}\right);$$

```
w11 = 0;
w12 = 0;
```

```
k11 = x - t;
k12 = 6 x;
```

```
x10 = 0;
```

$$f2 = \left(4x + \frac{10x^2}{3} - \frac{x^3}{3} + x^4 - \frac{x^5}{5}\right);$$

```
w21 = 0;
w22 = 0;
```

```
k21 = (t);
k22 = -(3 x - 2 t);
```

```
x20 = 5 / 3;
```

```
d1M = D1M[x10, x]; d2M = D1M[x20, x]
```

$$\left\{\frac{5\sqrt{\frac{\pi}{2}}}{3}, 0, 0, 0, 0, 0, 0, 0\right\}$$

```
f1M = D1M[f1, x]; d1M = D1M[x10, x]; w11M = D1M[w11, x]; w12M = D1M[w12, x];

fin1 = Assuming[0 ≤ x < 1, Refine[Expand[first]]];
fin2 = Assuming[0 ≤ x < 1, Refine[Expand[second]]];

T[m_, x_] := If[m == 0, 1 / Sqrt[Pi], Sqrt[2 / Pi] ChebyshevT[m, x]];

ans1 = Integrate[fin1.Transpose[psi[x]] * W[x, 1], {x, 0, 1}];
ans2 = Integrate[fin2.Transpose[psi[x]] * W[x, 1], {x, 0, 1}];

fin1t = fin1 //. x → t; fin2t = fin2 //. x → t;

ans12 = Integrate[fin1t, {t, 0, x}]; ans22 = Integrate[fin2t, {t, 0, x}];

v11 = Integrate[Transpose[psi[x]].k11M.ans12.Transpose[psi[x]] * W[x, 1], {x, 0, 1}];
v12 = Integrate[Transpose[psi[x]].k12M.ans22.Transpose[psi[x]] * W[x, 1], {x, 0, 1}];

v21 = Integrate[Transpose[psi[x]].k21M.ans12.Transpose[psi[x]] * W[x, 1], {x, 0, 1}];
v22 = Integrate[Transpose[psi[x]].k22M.ans22.Transpose[psi[x]] * W[x, 1], {x, 0, 1}];
```

جامعة النجاح الوطنية

كلية الدراسات العليا

الحلول التقريبية لنظام المعادلات التكاملية التفاضلية

إعداد

أحمد جلال أحمد عيسى

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.

2019

الحلول التقريبية لنظام المعادلات التكاملية التفاضلية

إعداد

أحمد جلال أحمد عيسى

إشراف

أ.د. ناجي قطناني

الملخص

في هذه الأطروحة ركزنا على الحل التقريبي لنظام معادلات فولتيرا التكاملية التفاضلية الخطية (system of linear Volterra integro-differential equations)، وقمنا باستقصاء بعض الطرق العددية لحل نظام معادلات فولتيرا التكاملية التفاضلية الخطية. هذه الطرق العددية هي: إعادة بناء طريقة تكرار التباين (reconstruction of variational iteration method)، طريقة التجميع السينك على أساس إقترانات السينك (sinc collocation method based on sinc function's)، طريقة مويجات شيبيشيف (Chebyshev wavelet method). إن الأمثلة العددية التي تناولناها نفذت باستخدام هذه الطرق العددية لحل نظام معادلات فولتيرا التكاملية التفاضلية الخطية.

تم وضع مقارنة بين هذه الطرق العددية حيث أظهرت لنا النتائج العددية أن طريقة مويجات شيبيشيف أكثر كفاءة وفاعلية بالمقارنة مع الطرق العددية الأخرى التي تم دراستها وذلك بناء على الأمثلة التي استخدمناها في الرسالة.