

**An-Najah National University**

**Faculty of Graduate Studies**

**Computational Methods for Solving Nonlinear  
Volterra Integro - Differential Equation**

**By**

**Farah Khaled Shehadah Abu Thabit**

**Supervisor**

**Prof. Naji Qatanani**

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

**2019**

# **Computational Methods for Solving Nonlinear Volterra Integro - Differential Equation**

**By**

**Farah Khaled Shehadah Abu Thabit**

**This Thesis was defended successfully on 1/12/2019 and approved by:**

**Defense Committee Members**

**Signature**

<b>- Prof. Naji Qatanani</b>	<b>/ Supervisor</b>	<b>.....</b>
<b>- Dr. Mahmoud Almanassra</b>	<b>/ External examiner</b>	<b>.....</b>
<b>- Dr. Adnan Daraghmeh</b>	<b>/ Internal examiner</b>	<b>.....</b>

## Dedication

“لمن أحبطني،، واستمات لكي لا أكون،، ووضع كافة العراقيين في دربي حجراً حجراً،، لأتعرش بها !!،، فزادنتي إصراراً للصعود عليها“ ...

## **Acknowledgments**

All thanks to God who gave me the strength and determination to implement this achievement.

I would sincerely Thanks to Professor Naji Qatanani for guidance, encouragement and supervision during this study and the preparation of the thesis.

also,I would like to record my special thanks to my parents " Khaled & Nisreen " and to my sisters "Mahaba, Nour, Menat-Allah and Layan and for my only brother Mutaz for their support, encouragement and great efforts in all stages of my life .

In the end, thanks to all the people who helped me in this work.

## الإقرار

أنا الموقع أدناه مقدم الرسالة التي تحمل العنوان:

### Computational Methods for Solving Nonlinear Volterra Integro - Differential Equation

أقر بأن ما اشتملت عليه هذه الرسالة إنما هي نتاج جهدي الخاص، باستثناء ما تمت الإشارة إليه  
حيثما ورد، وأن هذا الرسالة ككل أو أي جزء منها لم يقدم من قبل لنيل أي درجة علمية أو بحث  
علمي لدى أي 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  
degree or qualification.

**Student's Name:**

اسم الطالب:

**Signature:**

التوقيع:

**Date:**

التاريخ:

## Table of Contents

No.	Content	Page
	Dedication	ii
	Acknowledgments	iv
	Declaration	v
	List of Tables	ix
	List of Figures	xi
	Abstract	xii
	Introduction	1
	<b>Chapter One: Mathematical Preliminaries</b>	4
1.1	Classification of Nonlinear Integro -Differential Equations	5
1.1.1	Types of nonlinear integro - differential equation	5
1.1.2	Singularity of nonlinear integro - differential equation	7
1.2	Systems of nonlinear Volterra Integro - Differential Equations	9
1.3	Systems of nonlinear Fredholm Integro - Differential Equations	10
1.4	Kinds of kernel	10
1.5	Existence of the solution of nonlinear Volterra integro-differential equation	12
1.6	Uniqueness of the solution of nonlinear Volterra integro-differential equation	21
1.7	Laplace Transforms	25
1.7.1	Properties of the Laplace transforms	26
1.7.2	Convolution theorem	27
	<b>Chapter Two: Computational Methods for Solving Nonlinear Volterra Integro-Differential Equation</b>	28
2.1	Differential transform method with Adomian polynomials (DTM).	28
2.2	Modified Laplace Adomian Decomposition Method	34
2.3	The Variational iteration method (VIM).	37
	<b>Chapter Three: Numerical Examples and results</b>	37
3.1	The numerical realization of equations (3.1) – (3.2) Using Differential Transform Method with Adomian Polynomials (DTM)	41
3.2	The numerical realization of equation (3.1) – (3.2) Using the Modified Laplace Adomian Decomposition Method (LADM)	45

3.3	The numerical realization of equation (3.1) – (3.2) Using The Variational Iteration Method (VIM)	50
3.4	The numerical realization of equation (3.3) – (3.4) Using Differential Transform Method with Adomian Polynomials (DTM) .	55
3.5	The numerical realization of equation (3.3) – (3.4) Using Variational Iteration Method (VIM).	57
3.6	The numerical realization of equation (3.5) – (3.6) Using the Modified Laplace Adomian Decompostion Method (LADM)	60
3.7	The numerical realization of equation (3.5) – (3.6) Using the Variational Iteration Method (VIM).	62
3.8	The numerical realization of equation (3.7) – (3.8) Using the Variational Iteration Method (VIM).	65
3.9	The numerical realization of equation (3.7) – (3.8) Using the Modified Laplace Adomian Decompostion Method (LADM).	67
3.10	The numerical realization of equation (3.7) – (3.8) Using the Differential Transform Method (DTM).	69
	Conclusions	72
	References	73
	Appendix	79
	Matlab Code for Differential Transform Method for Example 3.1	79
	Matlab Code for Differential Transform Method for Example 3.2	80
	Matlab Code for Differential Transform Method for Example 3.4	82
	Matlab Code for Modified Laplace Adomian Decompostion Method for Example 3.1	85
	Matlab Code for Modified Laplace Adomian Decompostion Method for Example 3.3	86
	Matlab Code for Modified Laplace Adomian Decompostion Method for Example 3.4	89
	Matlab Code for Variational Iteration Method for Example 3.1	92
	Matlab Code for Variational Iteration Method for Example 3.2	95
	Matlab Code for Variational Iteration Method for Example 3.3	97

	Matlab Code for Variational Iteration Method for Example 3.4	100
	المخلص	ب



## List of Tables

<b>No.</b>	<b>Title</b>	<b>Page</b>
3.1	The exact and numerical solutions of applying Algorithm 3.1 on equation (3.1)	40
3.2	The exact and numerical solutions of applying Algorithm 3.2 on equation (3.1)	45
3.3	The exact and numerical solutions of applying Algorithm 3.3 on equation (3.1)	49
3.4	The exact and numerical solutions of applying Algorithm 3.1 on equation (3.2)	52
3.5	The exact and numerical solutions of applying Algorithm 3.3 on equation (3.2)	54
3.6	The exact and numerical solutions of applying Algorithm 3.2 on equation (3.3)	57
3.7	The exact and numerical solutions of applying Algorithm 3.3 on equation (3.3)	59
3.8	The exact and numerical solutions of applying Algorithm 3.3 on equation (3.4)	62
3.9	The exact and numerical solutions of applying Algorithm 3.2 on equation (3.4)	64
3.10	The exact and numerical solutions of applying Algorithm 3.1 on equation (3.4)	64

## List of Figures

No.	Title	Page
(3.1)a	The exact and numerical solutions of applying Algorithm 3.1 on equation (3.1)	44
(3.1)b	The error resulting of applying Algorithm 3.1on equation (3.1)	44
(3.2)a	The exact and numerical solutions of applying Algorithm 3.2 on equation (3.1)	49
(3.2)b	The error resulting of applying Algorithm 3.2on equation (3.1)	49
(3.3)a	The exact and numerical solutions of applying Algorithm 3.3 on equation (3.1)	53
(3.3)b	The error resulting of applying Algorithm 3.3on equation (3.1)	53
(3.4)	The exact and numerical solutions of applying all Algorithms on equation (3.1)	54
(3.5)a	The exact and numerical solutions of applying Algorithm 3.1 on equation (3.2)	56
(3.5)b	The error resulting of applying Algorithm 3.1on equation (3.2)	56
(3.6)a	The exact and numerical solutions of applying Algorithm 3.3 on equation (3.2)	58
(3.6)b	The error resulting of applying Algorithm 3.3on equation (3.2)	58
(3.7)	The exact and numerical solutions of applying all Algorithms on equation (3.3)	59
(3.8)a	The exact and numerical solutions of applying Algorithm 3.2 on equation (3.3)	61
(3.8)b	The error resulting of applying Algorithm 3.2on equation (3.3)	61
(3.9)a	The exact and numerical solutions of applying Algorithm 3.3 on equation (3.3)	63
(3.9)b	The error resulting of applying Algorithm 3.3on equation (3.3)	63
(3.10)	The exact and numerical solutions of applying all Algorithms on equation (3.2)	64
(3.11)a	The exact and numerical solutions of applying Algorithm 3.3 on equation (3.4)	66

(3.11)b	The error resulting of applying Algorithm 3.3 on equation (3.4)	66
(3.12)a	The exact and numerical solutions of applying Algorithm 3.2 on equation (3.4)	68
(3.12)b	The error resulting of applying Algorithm 3.2 on equation (3.4)	68
(3.13)a	The exact and numerical solutions of applying Algorithm 3.1 on equation (3.4)	70
(3.13)b	The error resulting of applying Algorithm 3.1 on equation (3.4)	70
(3.14)	The exact and numerical solutions of applying all Algorithms on equation (3.4)	71

**Computational Methods for Solving Nonlinear Volterra Integro -  
Differential Equation.**

**By**

**Farah Khaled Shehada Abu Thabit**

**Supervisor**

**Prof. Naji Qatanani**

**Abstract**

In this thesis we focus on the numerical treatment of the nonlinear Volterra integro - differential equation. This equation has wide range of applications in mathematical physics, engineering, mechanics, chemistry, astronomy, biology, economics and potential theory.

After introducing some definitions and important concepts of this equation, we will focus our attention mainly on the numerical methods for solving

the nonlinear Volterra-integro differential equation. These methods are: Differential Transform method with Adomian polynomials (DTM), Modified Laplace Adomian Decomposition Method (LADM) and the Variational iteration method (VIM).

The mathematical framework of these numerical methods together with their convergence properties will be presented. To demonstrate the efficiency of these numerical methods, we construct some numerical examples. Numerical results show clearly that the Variational iteration method (VIM) is the most efficient numerical technique for solving the

nonlinear Volterra integro - differential equation in a comparison with the other numerical techniques.

## Introduction

The Volterra integro-differential equation was initiated by Volterra in 1884. It appears in a variety of applications in many fields including continuum mechanics, potential theory, geophysics, characterizing many social and many physical applications such as glass forming Process, nano hydrodynamics, heat transfer and all the diffusion process in general. In addition, this equation has an important role in neutron diffusion and biological species coexisting together with increasing and decreasing rates of generating, and wind ripple in the desert. More details about the applications of these equations can be found in [21, 32].

Nonlinear phenomena has a fundamental role in various fields of science and engineering. The nonlinear models of the real life problems are still difficult to solve either analytically or numerically.

There has been much attention devoted to the search for better and more efficient solution methods for determining solution of nonlinear models [3].

There are many several analytical and numerical methods for solving integro - differential equations such as, the Adomian decomposition method, the direct computation method, the series solution method, the successive approximation method and the conversion to equivalent differential equations. However, these analytical methods are not easy to use and require huge calculation [36-37]. Alternatively, integro-differential

equations can be solved using many numerical methods such as the Legendre wavelet method [33], the Haar wavelet method [25], the linearization method [34], the finite difference method [39], block-pulse functions [8], the Taylor polynomial method [9, 24] and the differential transform method [1, 2, 4, 5].

In recent years, much work has been concentrated on the solutions of Volterra integro-differential equations. For example in [32] the authors used the decomposition method for solving some nonlinear integro-differential equations that arise as model equations for describing turbulent diffusion. In addition, they gave a comparison between the implicit Runge Kutta method and the decomposition technique. In [40] the authors introduced a multi-grid method for solving the nonlinear Urysohn integral equation. Also, in [37] it was shown that the modified decomposition method for mixed nonlinear Volterra -Fredholm integral equations combined with the noise terms phenomena may provide the exact solution by using just two iterations. Moreover, Wazwaz in [37] implemented the modified decomposition method, where he obtained numerical solutions in a rapidly convergent series with components that are elegantly computed. In addition Sweilam [34] used The variational iteration method (VIM) and nonlinear boundary value problems for 4th order integro - differential equations, where Variational Iteration Method is simple and yet a powerful method for solving integro-differential equations [35].

In this work, numerical simulations with different types of nonlinearities will be treated using some numerical techniques namely, Differential Transform of nonlinear integro-differential equation method with Adomian polynomials, Modified Laplace Adomian Decomposition Method (LADM)

And the Variational Iteration Method (VIM). In addition, a comparative study to examine the performance of these methods for solving integro - differential equations will be carried out. This can be realized by solving some numerical examples using Matlab software.

This thesis is organized as follows:

Chapter one introduces some basic concepts and definitions for the nonlinear integro - differential equations. In chapter two, some numerical methods namely, Differential transform of nonlinear integro-differential equation method with Adomian polynomials, Modified Laplace Adomian Decomposition Method (MLADM) and the Variational Iteration Method (VIM), will be addressed. Some numerical examples and results are presented in chapter three and conclusions have been drawn.



# Chapter One

## Mathematical Preliminaries

### Definition (1.1) [38]

An integro - differential equation is the equation in which unknown function  $q(x)$  appears under an integral sign and contains ordinary derivative  $q^{(n)}(x) = \frac{\partial^n q}{\partial x^n}$ ,  $n = 1, 2, 3, \dots$

The most standard form of the nonlinear integro-differential equation is given as:

$$q^{(n)}(x) = h(x) + \lambda \int_{m(x)}^{g(x)} \mathcal{B}(x, t) S(q(t)) dt, \quad (1.1)$$

Where  $q^{(n)}(x)$  is the  $n^{\text{th}}$  derivative of the unknown function  $q(x)$  that will be determined,  $h(x)$  is known analytic function,  $\mathcal{B}(x, t)$  is a known function of two variables  $x$  and  $t$  called the kernel of the integro-differential equation,  $\lambda$  is parameter “complex or real”  $g(x)$  and  $m(x)$  are limits of integration that may be both constants, variables or mixed, and  $S(q(t))$  is a nonlinear function.

Since equation (1.1) combines differential operator and integral operator, it is necessary to define some initial conditions  $q^{(k)}(0)$ ,  $0 \leq k \leq n - 1$

for the determination of the particular solution  $q(x)$ .

**Definition (1.2) [38]: Linearity:**

The integro-differential equation is called linear, if the unknown function  $q(x)$  inside the integral sign has exponent equal one. Otherwise if the unknown function  $q(x)$  has exponent other than one or contains nonlinear functions of  $q(x)$  then the integro-differential equation is called nonlinear, for example, the integro-differential equations

$$q^{(3)}(x) = 3 - x^2 + e^x + \int_0^x (x - t)q^2(t)dt$$

and

$$q^{(4)}(x) = x - x^2 e^x + \int_0^x e^{xt} \sinh q(t) dt$$

are a nonlinear integro-differential equations of the second kind.

**1.1 Classification of Nonlinear Integro-Differential Equations****1.1.1 Types of Nonlinear Integro-Differential Equations**

There are different kinds of nonlinear integro-differential equations:

**1. Volterra Nonlinear Integro-Differential Equation**

The Volterra nonlinear integro-differential equation of the second kind

appears in the form:

$$q^{(n)}(x) = h(x) + \lambda \int_a^x \mathcal{B}(x, t)S(q(t))dt, \quad (1.2)$$

where the lower limit of integration is constant and the upper limit is variable.

## 2. Fredholm Nonlinear Integro-Differential Equation

The Fredholm nonlinear integro-differential equation of the second kind has the form:

$$q^{(n)}(x) = h(x) + \lambda \int_a^b \mathcal{B}(x, t) S(q(t)) dt, \quad (1.3)$$

where the lower and the upper limit of integration are constant.

## 3. Volterra-Fredholm Nonlinear Integro-Differential Equation

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

$$\begin{aligned} q^{(n)}(x) = & h(x) \\ & + \lambda_1 \int_a^x \mathcal{B}_1(x, t) S(q(t)) dt + \lambda_2 \int_a^b \mathcal{B}_2(x, t) S(q(t)) dt \end{aligned} \quad (1.4)$$

and

$$q^{(n)}(x, t) = h(x, t) + \lambda \int_0^x \int_{\Omega} \mathcal{B}(x, t, r, s) S(q(r, s)) dr ds,$$

$$(x, t) \in \Omega \times [0, T] \quad (1.5)$$

where  $h(x, t)$  and  $\mathcal{B}(x, t, r, s)$  are analytic functions,  $D = \Omega \times [0, t]$  and  $\Omega$  is closed subset of  $R^n$ ,  $n = 1, 2, 3$ . It is interesting to note that (1.4) contains disjoint Volterra and Fredholm integrals, however (1.5) contains mixed integrals such that the Fredholm integral is the interior one, and Volterra is the exterior integral. Other derivatives of less order may appear as well.

**Definition (1.3) [37] : Analytic function:**

A function is said to be analytic if and only if its Taylor series about  $x_0$  converges to the function in some neighborhood for every  $x_0$  in its domain.

**Definition (1.4) [37] : Homogeneity:**

If  $h(x)$  is identically zero in the nonlinear integro-differential equation of the form:

$$q^{(n)}(x) = h(x) + \lambda \int_{m(x)}^{g(x)} \mathcal{B}(x, t) S(q(t)) dt,$$

then it is called homogeneous, otherwise it is called nonhomogeneous.

**1.1.2 Singularity of Nonlinear Integro - Differential Equation**

When one of the limits  $m(x)$  or  $g(x)$  or both are infinite or when the kernel becomes infinite at one or more points within the integration range then the equation is singular, for example, the integro - differential equations

$$q^{(n)}(x) = h(x) + \lambda \int_{-\infty}^{\infty} \mathcal{B}(x, t) S(q(t)) dt$$

and

$$q^{(n)}(x) = h(x) + \lambda \int_0^x \frac{1}{(x-t)^\alpha} S(q(t)) dt, \quad 0 < \alpha < 1$$

are a singular nonlinear integro-differential equation of the second kind.

### (i) Singular Nonlinear Integro-Differential Equation

If the kernel in (1.1) is of the form

$$\mathcal{B}(x, t) = \frac{v(x, t)}{x - t}, \quad (1.6)$$

where  $v(x, t)$  does not equal zero and is a differentiable function at  $a \leq x \leq b$ ,  $a \leq t \leq b$ , then the nonlinear integro-differential equation is said to be a singular equation with Cauchy kernel where  $\mathcal{B}(x, t) = \int_a^b \frac{v(x, t)}{x-t} h(t) dt$ , is understood in the sense of Cauchy Principle Value (CPV) and the notation P.V.  $\int_a^b \frac{v(x, t)}{x-t} dt$  is usually used to denote this.

Thus P.V.

$$\int_a^b \frac{v(x, t)}{x - t} dt = \lim_{x \rightarrow \varepsilon} \left\{ \int_a^{x-\varepsilon} \frac{v(x, t)}{x - t} dt + \int_{x+\varepsilon}^b \frac{v(x, t)}{x - t} dt \right\},$$

For example

$$u^{(2)}(x) = x^2 + \frac{2}{3}x^3 + x^5 \ln\left(\frac{1-x}{1+x}\right) + \operatorname{sech} x + \int_{-1}^1 \frac{q^2(t)}{x-t} dt, \quad |x| < 1$$

## (ii) Weakly Singular Nonlinear Integro-Differential Equation

The kernel is of the form

$$\mathcal{B}(x, t) = \frac{v(x, t)}{|x - t|^\gamma}, \quad (1.7)$$

Where  $v(x, t)$  is bounded, “i.e. several times continuously differentiable

$a \leq x \leq b$  ,  $a \leq t \leq b$  with  $v(x, t) \neq 0$  and  $\gamma$  is a constant such that  $0 < \gamma < 1$ . For example, the equation:

$$q^{(n)}(x) = \lambda \int_0^x \frac{1}{|x - t|^\gamma} S(u(t)) dt, \quad 0 < \gamma < 1$$

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

## 1.2 System of Nonlinear Volterra Integro-Differential Equations

A system of nonlinear Volterra integro - differential equations has the form [38]:

$$\left. \begin{aligned} q^{(i)}(x) &= h_1(x) + \int_0^x \left( \mathcal{B}_1(x, t) S_1(q(t)) + \tilde{\mathcal{B}}_1(x, t) \tilde{S}_1(p(t)) \right) dt \\ p^{(i)}(x) &= h_2(x) + \int_0^x \left( \mathcal{B}_2(x, t) S_2(q(t)) + \tilde{\mathcal{B}}_2(x, t) \tilde{S}_2(p(t)) \right) dt \end{aligned} \right\}, \quad (1.8)$$

The nonlinear functions  $S_i$  ,  $\tilde{S}_i = 1, 2$  are specified. The kernels  $\mathcal{B}_i(x, t)$  and  $\tilde{\mathcal{B}}_i(x, t)$  and the functions  $h_i(x)$ ,  $i = 1, 2$  are given as

real - valued function .

### 1.3 System of Nonlinear Fredholm Integro-Differential Equations

A system of nonlinear Fredholm integro - differential equations has the form[38]:

$$\left. \begin{aligned} q^{(i)}(x) &= h_1(x) + \int_a^b \left( \mathcal{B}_1(x, t) S_1(q(t)) + \tilde{\mathcal{B}}_1(x, t) \tilde{S}_1(p(t)) \right) dt \\ p^{(i)}(x) &= h_2(x) + \int_a^b \left( \mathcal{B}_2(x, t) S_2(q(t)) + \tilde{\mathcal{B}}_2(x, t) \tilde{S}_2(p(t)) \right) dt \end{aligned} \right\}, \quad (1.9)$$

The nonlinear functions  $S_i, \tilde{S}_i = 1, 2$  are specified. The kernels  $\mathcal{B}_i(x, t)$  and  $\tilde{\mathcal{B}}_i(x, t)$  and the functions  $h_i(x), i = 1, 2$  are given as

real - valued function .

### 1.4 Kinds of Kernels

#### 1. Separable Kernel

A kernel  $\mathcal{B}(x, t)$  is said to be separable or (degenerate) if it can be expressed as the sum of a finite number of terms, that is,

$$\mathcal{B}(x, t) = \sum_{i=1}^n r_i(x) c_i(t), \quad (1.10)$$

where the functions  $r_i(x)$  and  $c_i(t)$  are linearly independent.

#### 2. Difference Kernel

A kernel  $\mathcal{B}(x, t)$  is called difference kernel , when

$$\mathcal{B}(x, t) = \mathcal{B}(x - t), \quad (1.11)$$

### 3. Symmetric (or Hermitian) Kernel

A complex-valued function  $\mathcal{B}(x, t)$  is called symmetric (or Hermitian) if

$$\mathcal{B}(x, t) = \mathcal{B}^*(t, x), \quad (1.12)$$

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

$$\mathcal{B}(x, t) = \mathcal{B}(t, x), \quad (1.13)$$

### 4. Skew-Symmetric Kernel

The kernel is of the form

$$\mathcal{B}(x, t) = -\mathcal{B}(t, x) \quad (1.14)$$

### 5. Abel's Kernels

The kernel  $\mathcal{B}(x, t)$  is of the form

$$\mathcal{B}(x, t) = \frac{v(x, t)}{|x - t|^\gamma}, \quad (1.15)$$

where  $\gamma \in (0, 1)$  and the function  $v(x, t)$  is assumed to be differentiable.

### 6. Polar Kernel

$$\mathcal{B}(x, t) = \frac{v(x, t)}{(x - t)^\gamma} + k(x, t), \quad 0 < \gamma < 1, \quad (1.16)$$

where  $v$  and  $k$  are bounded and  $v(x, t) \neq 0$ .



## 7. Logarithmic Kernel

The kernel  $\mathcal{B}(x, t)$  called a logarithmic kernel if it has the form

$$\mathcal{B}(x, t) = v(x, t) \ln(x - t) + k(x, t), \quad (1.17)$$

where  $v$  and  $k$  satisfy the same conditions as in equation (1.16).

### 1.5 Existence of the Solution of Nonlinear Volterra Integro-Differential Equation

For convenience we consider the following nonlinear Volterra integro-differential equation :

$$\frac{dq(t)}{dt} = h\left(t, q(t), \int_0^t \mathcal{B}(t, u) S(u, q(u)) du\right), \quad (1.18)$$

with the initial condition :

$$q(0) = q_0 \quad (1.19)$$

where  $h : I \times R^n \times R^n \rightarrow R^n$ ,  $S : I \times R^n \rightarrow R^n$  and  $\mathcal{B} : I \times I \rightarrow R^n$  are continuous functions and  $q_0$  is a given constant; in which  $I = [0, T]$ ,  $R = (-\infty, \infty)$  and  $R^n$  denotes the Euclidean  $n$  - space with norm  $\| \cdot \|$ . and let  $B = C(I, R^n)$  be the Banach space of all continuous functions from  $I$  into  $R^n$  equipped with supremum norm  $\|q\|_B = \sup \{ \|q(t)\| : t \in I \}$ .

**Definition (1.5): Normed vector space  $(\mathcal{V}, \|\cdot\|)$ :**

Let the set  $\mathcal{V}$  be a linear vector space, a mapping,  $\|\cdot\|: \mathcal{V} \rightarrow R$ , is called normed vector space if the following properties satisfy:

1.  $\|q\| \geq 0$
2.  $\|q\| = 0$  if and only if  $q = 0$
3.  $\|q + y\| \leq \|q\| + \|y\|$
4.  $\|\delta q\| \geq \delta \|q\|$

**Definition (1.6): Complete space:**

A normed vector space  $(\mathcal{V}, \|\cdot\|)$  is called complete if every Cauchy sequence in  $\mathcal{V}$  converges to an element  $v \in \mathcal{V}$ .

**Definition (1.7): Banach space:**

A Banach space is a complete normed vector space  $(\mathcal{V}, \|\cdot\|)$ .

The existence theorem is based on the topological transversality theorem given by Tikhonov [36] and is known as Leray – Schauder alternative.

For a normed linear space  $D$  and a number  $\rho$  where  $\rho \in (0, \infty)$ , we let

$$k_\rho = \{ q \in D : \|q\| \leq \rho \} \quad \text{and} \quad Q_\rho = \{ q \in D : \|q\| = \rho \}$$

**Definition (1.8) [26]:**

We should say that  $\mathcal{H}$  is of the Leray - Schauder type provided for any  $k_\rho$  ball in  $D$ , either

- (a) There is an  $q \in k_\rho$  such that  $q \in \mathcal{H}q$ , or
- (b) There exist  $q \in Q_\rho$ , and  $\lambda \in (0,1)$  such that  $y \in \lambda \mathcal{H}y$ .

**Theorem (1.1) [36] (Leray – Schauder alternative)**

Let  $D$  be a convex subset of a normed linear space  $D$  and let  $\mathcal{H} : D \rightarrow D$  be a completely continuous operator and let

$$F(\mathcal{H}) = \{ q \in D : q \in \lambda \mathcal{H}q \text{ for some } 0 < \lambda < 1 \},$$

Then either

1.  $F(\mathcal{H})$  is unbounded, or
2. The operator  $\mathcal{H}$  has a fixed point.

**Proof:** Assume  $D$  is bounded and let  $k_\rho$  be a ball containing  $F(\mathcal{H})$  in its interior. Since no  $q \in Q_\rho$  can satisfy the second property in Definition (1.8), then the operator  $\mathcal{H}$  has a fixed point and the proof is complete.

**Theorem (1.2) (Banach fixed point space)** A fixed point is a point that does not change upon application of a map, system of differential equations, etc. In particular, a fixed point of a function  $q(x)$  is a point  $x_0$  such that  $q(x_0) = x_0$ .

**Theorem (1.3) [31] (The Arzela – Ascoli Theorem)**

Let  $T \subseteq (\varphi(I), \rho)$ , then the following statements are equivalent:

(1)  $T$  is compact.

(2)  $T$  is closed subset of  $(\varphi(I), \rho)$  and is both uniformly bounded and equicontinuous over  $I$ .

For convenience we list the following hypotheses :

**(H<sub>1</sub>)** There exists a constant  $\sigma \geq 0$  such that  $\|\mathcal{B}(t, u)\| \leq \sigma$  for

$t \geq u \geq 0$ .

**(H<sub>2</sub>)** There exists a continuous function  $\beta: I \rightarrow R_+ = [0, \infty)$

such that

$$\|\mathcal{B}(t, u)\| \leq \beta(t) H(\|u\|),$$

for every  $t \in I$  and  $u \in R^n$ , where  $H: R_+ \rightarrow (0, \infty)$  is a continuous non - decreasing function.

**(H<sub>3</sub>)** For each  $t^* \in I$ ,

$$\lim_{t \rightarrow t^*} \int_0^T |\mathcal{B}(t, u) - \mathcal{B}(t^*, u)| du = 0,$$

is satisfied for  $t \in I$ .

**(H<sub>4</sub>)** There exists a continuous function  $\omega: I \rightarrow R_+$  such that

$$\|h(t, q, x)\| \leq \omega(t)(\|q\| + \|x\|),$$

for every  $t \in I$  and  $q, x \in R^n$ .

Our main results are given through the following theorem :

**Theorem (1.4) [33]:** Suppose that the hypotheses  $(H_1)$ ,  $(H_2)$  and  $(H_4)$  are satisfied. Then equations (1.18) and (1.19) have a solution  $q$  defined on  $I$  provided  $T$  satisfies

$$\int_0^T M(u)du < \int_c^\infty \frac{du}{u+H(u)}, \quad (1.20)$$

Where  $c = \|q_0\|$  and  $M(t) = \max \{ \omega(t), \sigma\beta(t) \}$  for  $t \in I$ .

**Proof .** To prove the existence of a solution of equations (1.18) - (1.19), we start by applying Theorem (1.1), first we establish the priori bounds on the solution of the problem

$$\frac{dq(t)}{dt} = \lambda h \left( t, q(t), \int_0^t \mathcal{B}(t, u) S(u, q(u)) du \right), \quad (1.21)$$

under the initial condition equation(1.19) for  $\lambda \in (0,1)$ . Let  $q(t)$  be a solution of equation (1.21) with initial condition in equation (1.19), then  $q(t)$  satisfies the equivalent integral equation

$$q(t) = q_0 + \lambda \int_0^t h \left( u, q(u), \int_0^u \mathcal{B}(u, \tau) S(\tau, q(\tau)) d\tau \right) du \quad (1.22)$$

upon using the hypotheses  $(H_1)$ ,  $(H_2)$ ,  $(H_4)$  and equation (1.22) we obtain

$$\|q(t)\| \leq \|q_0\| + \int_0^t \omega(u) \left( \|q(u)\| + \int_0^u \sigma\beta(\tau)H(\|q(\tau)\|)d\tau \right) du \quad (1.23)$$

if we set the right side of equation (1.23) as  $L(t)$ , then

$$\|q(t)\| \leq L(t), \quad L(0) = \|q_0\| \quad \text{and}$$

$$\frac{dL(t)}{dt} \leq \omega(t) \left( L(t) + \int_0^t \sigma\beta(\tau)H(L(\tau))d\tau \right), \quad (1.24)$$

let

$$z(t) = L(t) + \int_0^t \sigma\beta(\tau)H(L(\tau))d\tau \quad (1.25)$$

then we obtain  $L(t) \leq z(t)$ ,  $z(0) = L(0) = \|q_0\|$  and

$$\dot{z}(t) \leq \omega(t)z(t) + \sigma\beta(t)H(z(t)) \leq M(t)(z(t) + H(z(t)))$$

i.e.

$$\frac{\dot{z}(t)}{z(t) + H(z(t))} \leq M(t), \quad (1.26)$$

Integrating both sides of equation (1.26) from 0 to  $t$  and use the change of variables and the condition in equation (1.20) yields

$$\int_c^{z(t)} \frac{du}{u + H(u)} \leq \int_0^t M(u)du \leq \int_0^T M(u)du < \int_c^\infty \frac{du}{u + H(u)} \quad (1.27)$$

Consequently, we conclude that there is a constant  $k$  independent of

$\lambda \in (0,1)$  such that  $z(t) \leq k$  and hence  $L(t) \leq k$  for  $t \in I$ .

Thus we have  $\|q(t)\| \leq k$  for  $t \in I$  and then

$$\|q(t)\| = \sup \{ \|q(t)\| : t \in I \} \leq k .$$

We define  $B = C(I, R^n)$  and rewrite the initial value problem equations (1.18) - (1.19) as follows: if  $y \in B$  and  $q(t) = y(t) + q_0$ ,  $t \in I$ , then it is easy to see that  $y$  satisfies  $y(0) = y_0 = 0$ ,

$$y(t) = \int_0^t h \left( u, y(u) + q_0, \int_0^u \mathcal{B}(u, \tau) S(\tau, y(\tau) + q_0) d\tau \right) du ,$$

Then we define  $\mathcal{H} : D_0 \rightarrow D_0$ ,  $D_0 = \{ y \in D : y_0 = 0 \}$  by

$$\mathcal{H}y(t) = \int_0^t h \left( u, y(u) + q_0, \int_0^u \mathcal{B}(u, \tau) S(\tau, y(\tau) + q_0) d\tau \right) du \quad (1.28)$$

for  $t \in I$ , If and only if  $q$  Satisfies (1.18) - (1.19). Then  $\mathcal{H}$  is clearly continuous. Now we need to prove that  $\mathcal{H}$  is completely continuous. So we let a bound sequence  $\{v_i\}$  in  $D_0$ , i.e.  $\|v_m\| \leq d$  for all  $i$ , where  $d$  is positive constant. From (1.28) and using the hypotheses  $(H_1)$ ,  $(H_2)$ ,  $(H_4)$  and letting  $M^* = \sup \{M(t) : t \in I\}$  we have

$$\|\mathcal{H}v_m\| = TM^* \left( b + c + \frac{TM^*}{2} H(b + c) \right)$$

consequently  $\{ \mathcal{H}v_m \}$  is uniformly bounded.

Now we shall show that the sequence  $\{ \mathcal{H}v_m \}$  is equicontinuous. Let

$0 \leq t_1 \leq t_2 \leq T$ , from (1.28) and using the hypotheses  $(H_1)$ ,  $(H_2)$ ,  $(H_4)$

and letting  $M^* = \sup \{ M(t) : t \in I \}$  we have

$$\begin{aligned}
 & | \mathcal{H} v_i(t_2) - \mathcal{H} v_i(t_1) | \\
 & \leq \int_{t_1}^{t_2} | h(u, v_i(u) + q_0, \int_0^u \mathcal{B}(u, \tau) v_i(\tau) + q_0) d\tau | du \\
 & \leq \int_{t_1}^{t_2} p(u) (v_i(u) + \|q_0\| + \int_0^u l o(\tau) H(\|v_i(\tau)\| + \|q_0\|) d\tau) d\tau du \\
 & \leq \int_{t_1}^{t_2} M^* (b + c + T M^* H(b + c)) du \tag{1.29}
 \end{aligned}$$

By equation (1.29) we conclude that  $\{ \mathcal{H} v_i \}$  is equicontinuous and hence by theorem (1.2) the operator  $\mathcal{H}$  is completely countinous.

Moreover, the set  $F(\mathcal{H}) = \{ y \in D_0 : q \in \lambda \mathcal{H} y \text{ for some } 0 < \lambda < 1 \}$ , is bounded, since for every  $y$  in  $F(\mathcal{H})$  the function  $q = y + h$  is a solution of (1.21), for which we have proved  $\|q\| \leq k$  and hence  $\|y\| \leq k$ . Now an application of Theorem (1.1), the operator  $\mathcal{H}$  has a fixed point in  $D_0$ . This means that equations (1.18) – (1.19) has a solution. This completes the proof of the theorem.



## 1.6 Uniqueness of the Solution of Nonlinear Volterra Integro - Differential Equation

Now to show the uniqueness of the solution of nonlinear Volterra integro -differential equations we need to employ the analysis based on the applications of the Banach fixed point theorem coupled with Bielecki type norm and the integral inequalities with explicit estimates (for more details see [6, 19, 20, 28, 29]).

We first build the appropriate metric space for our analysis and we let  $\alpha > 0$ , be a constant and consider the space of continuous functions  $C(I, R^n)$  such that  $\sup_{t \in I} \frac{|q(t)|}{e^{\alpha t}} < \infty$ , and denote this special space by  $C_\alpha(I, R^n)$ . We couple the linear space  $C_\alpha(I, R^n)$ , with Bielecki's metric :

$$d_\alpha^\infty(q, y) = \sup_{t \in I} \frac{\|q(t) - y(t)\|}{e^{\alpha t}},$$

and with Bielecki's norm:

$$\|q\|_\alpha^\infty = \sup_{t \in I} \frac{\|q(t)\|}{e^{\alpha t}}$$

We are now ready to present the main results for the uniqueness of solution of equations (1.18) - (1.19).

### **Theorem (1.5) [27]:**

Let  $\rho > 0, \alpha > 0, M \geq 0, \gamma > 1$  be constants with

$\alpha = \rho \gamma$ . Suppose that the functions  $h, \mathcal{B}$  in equations (1.18) - (1.19) satisfy

the conditions:

$$i. \quad \|h(t, x, v) - h(t, \bar{x}, \bar{v})\| \leq M[\|x - \bar{x}\| + \|v - \bar{v}\|],$$

$$ii. \quad \|\mathcal{B}(t, u, x) - \mathcal{B}(t, u, \bar{x})\| \leq \rho[\|x - \bar{x}\|],$$

$$iii. \quad d = \sup_{t \in I} \left\| \frac{1}{e^{\alpha t}} \left( q_0 + \int_0^t h \left( \sigma, 0, \int_0^\sigma \mathcal{B}(\sigma, u) S(u, 0) du \right) d\sigma \right) \right\| < \infty$$

If  $M/\alpha(1 + 1/\theta) < 1$ , then the equations (1.18) - (1.19) has a unique solution  $x \in C_\alpha(I, R^n)$ .

**Proof:** To prove Theorem (1.4), we need to consider the following equivalent formulation for the equations (1.18)-(1.19), with let  $q \in C_\alpha(I, R^n)$  and define the operator  $T$  by

$$\begin{aligned} (Tq)(t) = & q_0 + \int_0^t h \left( \sigma, q(\sigma), \int_0^\sigma \mathcal{B}(\sigma, u) S(u, q(\sigma)) du \right) d\sigma \\ & - \int_0^t h \left( \sigma, 0, \int_0^\sigma \mathcal{B}(\sigma, u) S(u, 0) du \right) d\sigma \\ & + \int_0^t h \left( \sigma, 0, \int_0^\sigma \mathcal{B}(\sigma, u) S(u, 0) du \right) d\sigma, \end{aligned} \quad (1.30)$$

Now we shall show that  $T$  into itself, From (1.30) and using the hypotheses we have

$$\|Tq\|_\alpha^\infty = \sup_{t \in I} \frac{\|(Tq)(t)\|}{e^{\alpha t}}$$

$$\begin{aligned}
& \leq \sup_{t \in I} \frac{1}{e^{\alpha t}} \left\| h \left( \sigma, q(\sigma), \int_0^\sigma \mathcal{B}(\sigma, u) S(u, q(\sigma)) du \right) d\sigma \right. \\
& \quad \left. - h \left( \sigma, 0, \int_0^\sigma \mathcal{B}(\sigma, u) S(u, 0) du \right) d\sigma \right\| \\
& \quad + \sup_{t \in I} \frac{1}{e^{\alpha t}} \left\| h \left( \sigma, 0, \int_0^\sigma \mathcal{B}(\sigma, u) S(u, 0) du \right) d\sigma \right\| \\
& \leq d_1 + \sup_{t \in I} \frac{1}{e^{\alpha t}} M \left[ \|q(t)\| + \int_0^t \rho \|q(\sigma)\| d\sigma \right] \\
& = d_1 + \sup_{t \in I} \frac{1}{e^{\alpha t}} M \left[ \sup_{t \in I} \frac{\|q(t)\|}{e^{\alpha t}} + \rho \sup_{t \in I} \frac{1}{e^{\alpha t}} \int_0^t e^{\alpha \sigma} \frac{\|q(\sigma)\|}{e^{\alpha \sigma}} d\sigma \right] \\
& \leq d_1 + M \left[ \|q\|_\alpha^\infty + \rho \|q\|_\alpha^\infty \sup_{t \in I} \frac{1}{e^{\alpha t}} \int_0^t e^{\alpha \sigma} d\sigma \right] \\
& = d_1 + M \|q\|_\alpha^\infty \left[ 1 + \rho \sup_{t \in I} \frac{1}{e^{\alpha t}} \left( \frac{e^{\alpha t} - 1}{\alpha} \right) \right] \\
& = d_1 + M \|q\|_\alpha^\infty \left[ 1 + \frac{\rho}{\alpha} \right] = d_1 + \|q\|_\alpha^\infty M \left( 1 + \frac{1}{\gamma} \right) < \infty
\end{aligned}$$

This proves that the operator  $T$  maps  $C_\alpha(I, R^n)$  into itself.

Now we verify that the operator  $T$  is a contraction map. Let

$x, v \in C_\alpha(I, R^n)$ . From equation (1.30) and using the hypotheses we have

$$d_\alpha^\infty(Tx, Tv) = \sup_{t \in I} \frac{\| (Tx)(t) - (Tv)(t) \|}{e^{\alpha t}}$$

$$\begin{aligned}
&= \sup_{t \in I} \frac{1}{e^{\alpha t}} \left\| h \left( \sigma, x(\sigma), \int_0^\sigma \mathcal{B}(\sigma, u) S(u, x(\sigma)) du \right) d\sigma \right. \\
&\quad \left. - h \left( \sigma, v(\sigma), \int_0^\sigma \mathcal{B}(\sigma, u) S(u, v(\sigma)) du \right) d\sigma \right\| \\
&\leq \sup_{t \in I} \frac{1}{e^{\alpha t}} M \left[ \|x(t) - v(t)\| + \int_0^t \rho \|u(\sigma) - v(\sigma)\| d\sigma \right] \\
&= M \left[ \sup_{t \in I} \frac{\|x(t) - v(t)\|}{e^{\alpha t}} + \sup_{t \in I} \frac{1}{e^{\alpha t}} \int_0^t e^{\alpha \sigma} \frac{\|x(t) - v(t)\|}{e^{\alpha t}} d\sigma \right] \\
&= M \left[ d_\alpha^\infty(x, v) + \rho d_\alpha^\infty(x, v) \sup_{t \in I} \frac{1}{e^{\alpha t}} \int_0^t e^{\alpha \sigma} d\sigma \right] \\
&= M d_\alpha^\infty(x, v) \left[ 1 + \rho \sup_{t \in I} \frac{1}{e^{\alpha t}} \left( \frac{e^{\alpha t} - 1}{\alpha} \right) \right] \\
&= M d_\alpha^\infty(x, v) \left[ 1 + \frac{\rho}{\alpha} \right] = M \left( 1 + \frac{1}{\gamma} \right) d_\alpha^\infty(x, v)
\end{aligned}$$

Since  $M(1 + 1/\gamma) < 1$ , it follows from the Banach fixed point theorem (see that T has a unique fixed point in  $C_\alpha(I, R^n)$ ). The fixed point of T is however solution of equation (1.1). The proof is complete.

## 1.7 Laplace Transforms

**Definition (1.5) [36]** : Let  $f(x)$  be a real valued function defined for  $x \geq 0$ . Suppose that  $f(x)$  is multiplied by  $e^{-sx}$  and the result is integrated with respect to  $x$  from 0 to  $\infty$ , if the integral converges then it is a function of  $s$ , that is

$$\mathcal{L}\{f(x)\} = \mathcal{F}(s) = \int_0^{\infty} e^{-sx} f(x) dx \quad (1.30)$$

or

$$\mathcal{L}\{f(x)\} = \mathcal{F}(s) = \lim_{A \rightarrow \infty} \int_0^A e^{-sx} f(x) dx \quad (1.31)$$

$\mathcal{F}(s)$  is called the Laplace transform of  $f(x)$ . Moreover we have

$$f(x) = \mathcal{L}^{-1}\{\mathcal{F}(s)\} \quad (1.32)$$

called the inverse Laplace transform.

### 1.7.1 Properties of the Laplace Transforms

Laplace transforms has the following properties:

#### 1. Constant Multiple:

$$\mathcal{L}\{af(x)\} = a\mathcal{L}\{f(x)\} = a\mathcal{F}(s), \quad a \text{ is a constant.} \quad (1.33)$$

For example:

$$\mathcal{L}\{4e^x\} = 4\mathcal{L}\{e^x\} = \frac{4}{s-1}$$

## 2. Linearity Property:

$$\mathcal{L}\{\alpha f(x) + \beta r(x)\} = \alpha \mathcal{L}\{f(x)\} + \beta \mathcal{L}\{r(x)\} = \alpha \mathcal{F}(s) + \beta \mathcal{R}(s) ,$$

$$\alpha, \beta \text{ are constants} \quad (1.34)$$

For example:

$$\mathcal{L}\{4x + 3x^2\} = 4\mathcal{L}\{x\} + 3\mathcal{L}\{x^2\} = \frac{4}{s^2} + \frac{6}{s^3}$$

## 3. Multiplication by $x$ :

$$\mathcal{L}\{x^n f(x)\} = (-1)^n \frac{d^n}{ds^n} \mathcal{L}\{f(x)\} = (-1)^n \frac{d^n}{ds^n} \mathcal{F}(s) \quad (1.35)$$

For example:

$$\mathcal{L}\{x \sin x\} = -\frac{d}{ds} \mathcal{L}\{\sin x\} = -\frac{d}{ds} \left( \frac{1}{s^2 + 1} \right) = \frac{2s}{(s^2 + 1)^2}$$

### 1.7.2 Convolution

**Definition (1.6) [38] :** If  $M(x)$  and  $r(x)$  are piecewise continuous functions on  $[0, x)$  and both satisfy the conditions needed for the existence of Laplace transform, then the convolution of  $M$  and  $r$  denoted by  $(M * r)(x)$  is defined by the integral :

$$(M * r)(x) = \int_0^x M(x-t)r(t)dt$$

or

$$(r * M)(x) = \int_0^x r(x-t)M(t)dt$$

note that

$$(M * r)(x) = (r * M)(x)$$

we can easily show that the Laplace transform of the convolution product

$(M * r)(x)$  is

$$\mathcal{L}\{(M * r)(x)\} = \mathcal{L}\left\{\int_0^x M(x-t) \cdot r(t) dt\right\} = \mathcal{M}(s) \cdot \mathcal{R}(s)$$

Where  $\mathcal{M}(s)$  and  $\mathcal{R}(s)$  are the Laplace transform for the functions  $M(x)$  and  $r(x)$  respectively.

## **Chapter Two**

### **Computational Methods**

There are many computational techniques available for solving nonlinear Volterra integro-differential equations. In this chapter we will discuss the following methods: Differential Transform Method with Adomian Polynomials (DTM), Modified Laplace Adomian Decomposition Method and the Variational Iteration Method (VIM).

#### **2.1 Differential Transform Method with Adomian Polynomials (DTM)**

In this section, the (DTM) was discussed in order to solve nonlinear integro - differential equations in a more comprehensive and effective way; the idea is based on the methodology of [10] where the nonlinear term is replaced by its Adomian polynomials and the dependent variable components is replaced by its corresponding differential transform component of the same index.

#### **Differential Transform Method**

This method is based on the (DTM) presented by Zhou [39] in his study of electric circuits, where basic definitions and fundamental theorems of the differential transformation and its applicability to different types of differential and integral equation are given in [3, 4, 39].



The  $k^{th}$  derivative transformation of a function  $q(x)$  is defined as follows:

$$Q(k) = \frac{1}{k!} \left[ \frac{d^k}{dx^k} q(x) \right]_{x=x_0}, \quad (2.1)$$

where  $q(x)$  is the premier analytical function and  $Q(x)$  is the transformed function. Differential inverse transformation of  $Q(x)$  is defined as follows:

$$q(x) = \sum_{k=0}^{\infty} Q(k)(x - x_0)^k, \quad (2.2)$$

Since  $q(x)$  is the analytical function, it is clear that  $q(x) = q_0(x)$ . By combination of equations (2.1) and (2.2), with  $x_0 = 0$ , the function

$q(x)$  can be written as:

$$q(x) = \sum_{k=0}^{\infty} \frac{1}{k!} \left[ \frac{d^k}{dx^k} q(x) \right]_{x=0} x^k, \quad (2.3)$$

Basic mathematical properties of the differential transform can easily be obtained and summarized in the following theory:

**Theorem (2.1) [7]:** If  $G(k)$ ,  $T(k)$  and  $\mathcal{W}(k)$  are differential transforms of the functions  $g(x)$ ,  $t(x)$  and  $w(x)$  respectively then :

1. If  $g(x) = t(x) \pm w(x)$  then  $G(k) = T(k) \pm \mathcal{W}(k)$ .
2. If  $g(x) = t(x)w(x)$  then  $G(k) = \sum_{l=0}^k T(l) \mathcal{W}(k-l)$ .
3. If  $g(x) = \alpha w(x)$  then  $G(k) = \alpha \mathcal{W}(k)$ .
4. If  $g(x) = \frac{d^n}{dx^n} t(x)$  then  $G(k) = (k+1)(k+2) \cdots (k+n)T(k+n)$ .

5. If  $u(x) = x^n$  then  $G(k) = \delta(k - n) = \begin{cases} 1, & k = n \\ 0, & o.w \end{cases}$
6. If  $g(x) = \cos(\omega x + \alpha)$  then  $G(k) = \frac{\omega^k}{k!} \cos\left(\frac{\pi k}{2} + \alpha\right)$ .
7. If  $g(x) = \sin(\omega x + \alpha)$  then  $G(k) = \frac{\omega^k}{k!} \sin\left(\frac{\pi k}{2} + \alpha\right)$ .
8. If  $g(x) = e^{\lambda x}$  then  $G(k) = \frac{\lambda^k}{k!}$ .

**Theorem (2.2) [ 7 ]:**

If  $G(k)$ ,  $T_1(k)$  and  $T_2(k)$  are differential transforms of the functions  $g(x)$ ,  $t_1(x)$  and  $t_2(x)$  respectively, then for  $k = 1, 2, \dots, N$ ,

we have :

1. If  $g(x) = \int_{x_0}^x t_1(s)t_2(s)ds$  then  $G(k) = \frac{1}{k} \sum_{l=0}^{k-1} T_1(l) T_2(k - l - 1)$ ,
2. If  $g(x) = w(x) \int_{x_0}^x t_1(s)t_2(s)ds$  then

$$G(k) = \sum_{l=0}^k \sum_{s=0}^{k-l-1} \frac{1}{k-l} \mathcal{W}(l) t_1(s) t_2(k - l - s - 1), \quad k \geq 1.$$

**The Modified Differential Transform Method**

The Adomian decomposition method is utilized to the following general nonlinear equation:

$$\mathcal{L}_q + \mathcal{R}_q + \mathcal{M}_q = U(x), \quad (2.4)$$

Where  $q$  is the unknown function,  $\mathcal{L}_q$  is the highest - order derivative which assumed to be easily reversible,  $\mathcal{R}_q$  is the remaining linear operator ,  $\mathcal{M}_u$  represents a general nonlinear differential operator , and  $U(x)$  is the

source term. Applying the inverse operator  $\mathcal{L}^{-1}$  to both sides of equation (2.4) and using the specific conditions we obtain

$$q = f(x) - \mathcal{L}^{-1}(\mathcal{R}_q) - \mathcal{L}^{-1}(\mathcal{M}_q), \quad (2.5)$$

where the function  $f(x)$  represents the terms arising from integrating the source term  $U(x)$ . The nonlinear operator  $\mathcal{M}_q = S(q)$  is decomposed as

$$S(q) = \sum_{n=0}^{\infty} \tilde{A}_n(q_0, q_1, q_2, \dots, q_n) \quad (2.6)$$

where  $\tilde{A}_n$ ,  $n \geq 0$  are the Adomian polynomials determined formally as follows [3,2,10,39]

$$\tilde{A}_n = \frac{1}{n!} \left[ \frac{d^n}{d\lambda^n} \left[ S \left( \sum_{i=0}^n \lambda^i q_i \right) \right] \right]_{\lambda=0} \quad (2.7)$$

the Adomian polynomials of  $S(q)$  are introduced as

$$\tilde{A}_0 = S(q_0),$$

$$\tilde{A}_1 = q_1 \dot{S}(q_0),$$

$$\tilde{A}_2 = q_2 \dot{S}(q_0) + \frac{1}{2!} q_1^2 S^{(2)}(q_0),$$

$$\tilde{A}_3 = q_3 \dot{S}(q_0)(q_0) + q_1 q_2 S^{(2)}(q_0) + \frac{1}{3!} q_1^3 S^{(3)}(q_0),$$

$$\tilde{A}_4 = q_4 \dot{S}(q_0) + \left( q_1 q_3 + \frac{1}{2!} q_2^2 \right) S^{(2)}(q_0) + \frac{1}{2!} q_1^2 q_2 S^{(3)}(q_0) + \frac{1}{4!} q_1^4 S^{(4)}(q_0),$$

and so on.

**Lemma (2.1) [4]:** If  $f(x) = S(q(x))$ , then  $F(k) = A_k$  where  $A_k$ 's are the Adomian polynomials  $\tilde{A}_k$  with replacing  $q_k$  by  $Q(k)$ ,  $k = 0, 1, 2, \dots$

**Proof:** ( for more details see [4]).

The differential transform components of  $q(x)$  are computed by utilizing their properties. Then we can write it in the following formula ( for  $x = 0$  )

$$S(0) = S(Q(0)),$$

$$S(1) = Q(1)\dot{S}(Q(0)),$$

$$S(2) = Q(2)\dot{S}(Q(0)) + \frac{1}{2!}Q^2(1)S^{(2)}(Q(0)),$$

$$S(3) = Q(3)\dot{S}(Q(0)) + Q(1)Q(2)S^{(2)}(Q(0)) + \frac{1}{3!}Q^3(1)S^{(3)}(Q(0)),$$

$$S(4) = Q(4)\dot{S}(Q(0)) + \left( Q(1)Q(3) + \frac{1}{2!}Q^2(2) \right) S^{(2)}(Q(0)) + \\ \frac{1}{2!}Q^2(1)Q(2)S^{(3)}(Q(0)) + \frac{1}{4!}Q^4(1)S^{(4)}(Q(0)) ,$$

*and so on.*

To use the differential transform method we need to replace the nonlinear term by its Adomian polynomials of the index  $k$ , and hence the dependent variable components are replaced by their corresponding differential transforms in the recurrence relation of the same index.

## 2.2 Modified Laplace Adomian Decomposition Method (LADM)

In this section, we show how the modified Laplace decomposition method can be used to solve the nonlinear Volterra integro - differential equation (1.1).

To solve the nonlinear Volterra integro - differential equation (1.1) using the Laplace transform method, it is essential to use the Laplace transforms of the derivatives of  $q(x)$ .

Applying the Laplace transform to both sides of equation (1.1) yields:

$$\begin{aligned} s^n \mathcal{L}\{q(x)\} - s^{n-1}q(0) - \dots - sq^{(n-2)}(0) - q^{(n-1)}(0) \\ = \mathcal{L}\{h(x)\} + \mathcal{L}\left\{\int_0^x B(x,t)S(u(t))dt\right\}. \end{aligned} \quad (2.8)$$

or equivalently

$$\begin{aligned} \mathcal{L}\{q(x)\} = \frac{1}{s}q(0) - \dots - \frac{1}{s^{n-1}}q^{(n-2)}(0) - \frac{1}{s^n}q^{(n-1)}(0) + \frac{1}{s^n}\mathcal{L}\{h(x)\} \\ + \frac{1}{s^n}\mathcal{L}\left\{\int_0^x B(x,t)S(u(t))dt\right\} \end{aligned} \quad (2.9)$$

The next step in Laplace decomposition method is representing the solution in an infinite series given as

$$q(x) = \sum_{n=0}^{\infty} q_n(x). \quad (2.10)$$

To overcome the nonlinearity of  $S(q(t))$ , we use the Adomian decomposition method given in the form of an infinite series of the Adomian polynomials  $\tilde{A}_n$ , that is

$$S(q(t)) = \sum_{n=0}^{\infty} \tilde{A}_n(x) \quad (2.11)$$

where for every  $n \in N$ ,  $\tilde{A}_n$  is the Adomian polynomial given as

$$\tilde{A}_n = \frac{1}{n!} \left[ \frac{d^n}{d\lambda^n} \left[ S \left( \sum_{i=0}^n \lambda^i u_i \right) \right] \right]_{\lambda=0}. \quad (2.12)$$

Substituting equation (2.10) and equation (2.11) into equation (2.9) gives

$$\begin{aligned} \mathcal{L} \left\{ \sum_{i=0}^{\infty} q_n(x) \right\} &= \frac{1}{s} q(0) - \dots - \frac{1}{s^{n-1}} q^{(n-2)}(0) - \frac{1}{s^n} q^{(n-1)}(0) \\ &+ \frac{1}{s^n} \mathcal{L}\{h(x)\} + \frac{1}{s^n} \mathcal{L} \left\{ \int_0^x \mathcal{B}(x, t) \sum_{n=0}^{\infty} \tilde{A}_n(x) dt \right\}, \end{aligned} \quad (2.13)$$

The Adomian decomposition method admits the use of the following recursive relation

$$\mathcal{L}\{q_0(x)\} = f_1(s), \quad (2.14)$$

$$\mathcal{L}\{q_1(x)\} = f_2(s) + \frac{1}{s^n} \mathcal{L} \left\{ \int_0^x \mathcal{B}(x, t) \sum_{i=0}^{\infty} \tilde{A}_n(x) dt \right\}, \quad (2.15)$$

$$\mathcal{L}\{q_{n+1}(x)\} = \frac{1}{s^n} \mathcal{L} \left\{ \int_0^x \mathcal{B}(x, t) \sum_{n=0}^{\infty} \tilde{A}_n(x) dt \right\}, \quad n \geq 1, \quad (2.16)$$

where  $f_1(s)$  and  $f_2(s)$  are Laplace transforms of  $f_1(x)$  and  $f_2(x)$  respectively. Applying the inverse Laplace transform to equations (2.14)-(2.16) gives the required recursive relation as follows

$$u_0(x) = f_1(x), \quad (2.17)$$

$$u_1(x) = f_2(s) + \mathcal{L}^{-1} \left\{ \frac{1}{s^n} \mathcal{L} \left\{ \int_0^x \mathcal{B}(x, t) \sum_{n=0}^{\infty} \tilde{A}_n(x) dt \right\} \right\}, \quad (2.18)$$

$$u_{n+1}(x) = \mathcal{L}^{-1} \left\{ \frac{1}{s^n} \mathcal{L} \left\{ \int_0^x \mathcal{B}(x, t) \sum_{n=0}^{\infty} \tilde{A}_n(x) dt \right\} \right\}, \quad n \geq 1, \quad (2.19)$$

### 2.3 The Variational Iteration Method (VIM)

In this section, we will study the newly developed variational iteration method (VIM) was established by Ji-Huan He [11-12], the method provides rapidly convergent successive approximations of the exact solution only if such a closed form solution exists, and not components as in Adomian decomposition method. In this method the problem considered has the form:

$$\mathcal{A}_q + \mathcal{N}_q = U(x), \quad (2.20)$$

where  $\mathcal{A}$  is a linear operator,  $\mathcal{N}$  is a nonlinear operator, and  $U(x)$  is a given function. According to (VIM) [13 - 16], we can build the following correction functional :

$$q_{i+1}(x) = q_i(x)$$

$$+ \int_0^x \lambda(\xi) \{ \mathcal{A}q_i(\xi) + \mathcal{N}\tilde{q}_i(\xi) - U(\xi) \} d\xi, \quad i \geq 0, \quad (2.21)$$

where  $\lambda$  is Lagrange multiplier [18], which can be determined optimally by the variational theory and may be here constant or function, the subscript  $i$  denotes the  $i^{th}$  approximation,  $\tilde{q}_i$  is considered as a restricted value that means it behaves as a constant, hence  $\delta\tilde{q}_i = 0$ , where  $\delta$  is the variational derivative [18].

In particular, the correction functional for the nonlinear integro - differential equation (1.1) is

$$q_{i+1}(x) = q_i(x) + \int_0^x \lambda(\tau) \left[ q_i^{(n)}(\tau) - k(\tau) - \int_0^s L(\tau, r) S(\tilde{q}_n(r)) dr \right] d\tau, \quad (2.22)$$

The variational iteration method needs applying two basic steps, namely:

1. The determination of the Lagrange multiplier  $\lambda$  that can be identified optimally via integration by parts and by using a restricted variation. after selecting  $\lambda$ , an iteration formula, without restricted variation, should be used to determine of the successive approximations of  $q_{i+1}(x), i \geq 0$  of the solution  $q(x)$ .
2. The zeroth approximation  $q_0(x)$  can be any selective function. However, the initial values  $q(0), \dot{q}(0), \dots$  are preferably used for the selective zeroth approximation  $q_0$ .



The following is a summary of the Lagrange multipliers as derived in [38], and the selective zeroth approximations:

$$\dot{q} + k(q(\tau), \dot{q}(\tau)) = 0, \quad \lambda = -1, \quad q_0(x) = q(0),$$

$$q^{(2)} + k(q(\tau), \dot{q}(\tau), q^{(2)}(\tau)) = 0, \quad \lambda = \tau - x, \quad q_0(x) = q(0) + \dot{q}(0)x$$

$$q^{(3)} + k(q(\tau), \dot{q}(\tau), q^{(2)}(\tau), q^{(3)}(\tau)) = 0, \quad \lambda = -\frac{1}{2!} (\tau - x)^2,$$

$$q_0(x) = q(0) + \dot{q}(0)x + \frac{1}{2!} q^{(2)}(0)x^2,$$

and so on.

Consequently, the solution is given by

$$q(x) = \lim_{i \rightarrow \infty} q_i(x).$$

## Chapter Three

### Numerical Examples and Results

In this chapter we implement the aforementioned numerical methods in chapter two to solve some numerical examples in order to test the efficiency and accuracy of these methods. This will be carried out using proper algorithms and Matlab software. Comparison between the exact and the approximate solutions will be tabulated and graphically illustrated.

#### Example 3.1

Consider the Volterra nonlinear integro-differential equation :

$$q^{(2)}(x) - 6q(x) = -4 + 8 \int_0^x tq(t) \ln q(t) dt \quad (3.1)$$

with the initial conditions :

$$q(0) = 1, \dot{q}(0) = 0 \quad (3.2)$$

The exact solution of equation (3.1) is :

$$q(x) = e^{x^2}.$$

Now, we apply the aforementioned numerical methods in chapter two to solve equations (3.1) – (3.2).

### 3.1 The Numerical Realization Of Equations (3.1) – (3.2) Using The Differential Transform Method With Adomian Polynomials (DTM)

The following Algorithm implements the Differential Transform Method with Adomian Polynomials (DTM) using the Matlab software

#### Algorithm (3.1)

This algorithm can be illustrated as follows:

1. Input :

a)  $g(x)$  ,  $m(x)$ : Limits of integration.

b)  $\lambda$  : Is a constant parameter.

c)  $h(x)$ : The function of the integral equation.

d)  $\mathcal{B}(x, t)$ : The kernel function.

e)  $S(q(t))$  : Nonlinear term.

f) Set  $m(x) = 0$  and  $g(x) = x$ .

g) The first known iterations  $q(x)$ ,  $i = 0, 1, \dots, n$

2. Calculate: the Adomian polynomials for the nonlinear term .

3. Calculate: The iterations  $q_i(x)$ , from  $n$  to  $\infty$

for  $i = n:m$

( if  $i \leq m$  )

$$q(k+1) = -\frac{(k-2)}{k}u(k-1) + \frac{1}{6(k+1)} \left[ \frac{(-1)^k * (6 - 11k + 6k^2 - k^3)}{k!} \right];$$

( if  $i > m$  )

$$q(k+1) = -\frac{(k-2)}{k}u(k-1) + \frac{1}{6(k+1)} \left[ \frac{(-1)^k * (6 - 11k + 6k^2 - k^3)}{k!} + \frac{G(k-5)}{k-2} \right];$$

*end*

4. Evaluation sum of all  $u$ 's in this form :  $Q(x) = \sum_{k=0}^{\infty} q(k) x^k$

*syms x*

*yapp = 0;*

*for i = 1 : k*

*yapp = yapp + (x<sup>i-1</sup>q(i))*

*end*

*yapp*

5. For any subinterval of discrete points of  $x$  :

i. Input exact solution and compute exact at all points of  $x$ .

ii. Find absolute error =  $|q - q_n|$ .

- iii. Plot the error and all values  $x$ 's.
  - iv. Output  $(x_i, \text{approximate}_i, \text{Exact}_i, \text{error}_i)$ .
  - v. Print the table.
  - vi. Plot and Stop (The process is complete).
6. Output: Approximations  $q(x)$  about interval.

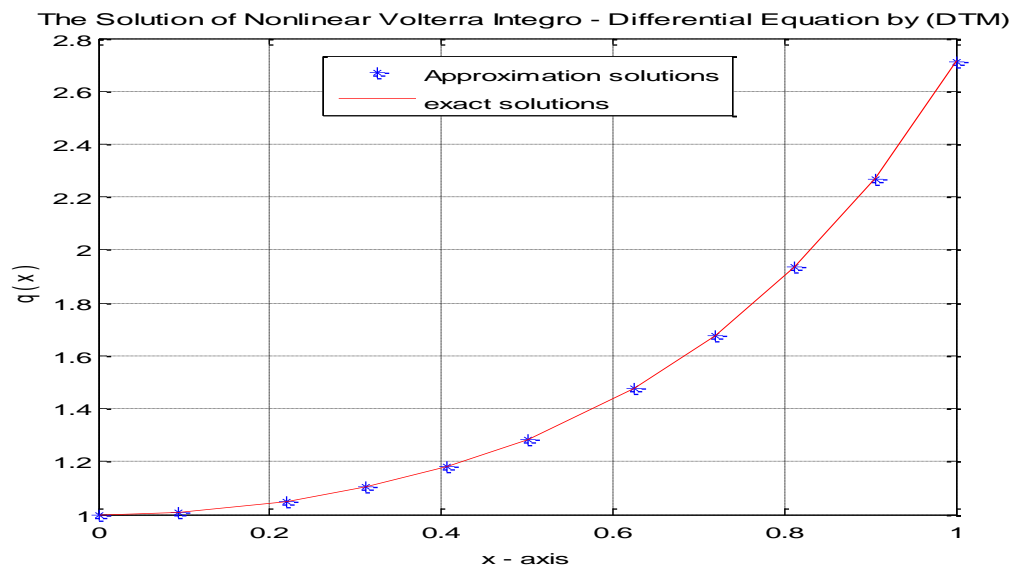
We use Algorithm (3.1) to solve equations (3.1) – (3.2). Table (3.1) contains both the exact and numerical solutions using the Differential Transform Method (DTM) for equations (3.1) – (3.2).

**Table (3.1): The exact and numerical solutions using the Differential Transform Method with Adomian Polynomials (DTM) with  $n = 10$ .**

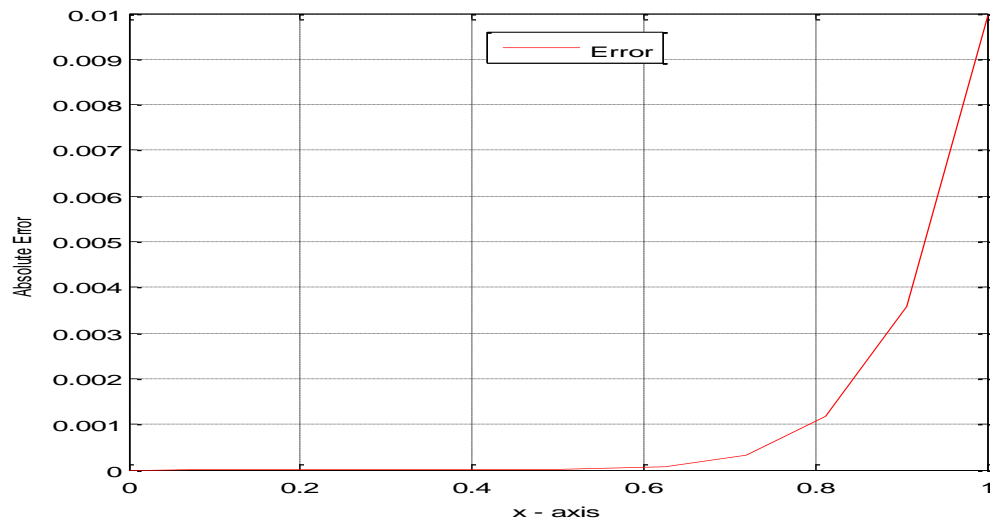
$x$	Exact solution $q(x) = e^{x^2}$	Numerical solution $q_n(x)$	Absolute error $=  q - q_n $
0	1.0000000000000000	1.0000000000000000	0
0.0938	1.00883726004162	1.00883726004118	0.000000000000044
0.2188	1.04903788068848	1.04903787857613	0.00000000211235
0.3125	1.10258370680894	1.10258363157239	0.00000007523655
0.4062	1.17939127885956	1.17939023105735	0.00000104780220
0.5	1.28402541668774	1.28401692708333	0.00000848960440
0.6250	1.47790419541173	1.47782318045695	0.00008101495478
0.7188	1.67644158014955	1.67610624329884	0.00033533685071
0.8125	1.93509466932358	1.93392305604841	0.00117161327516
0.9062	2.27322252652152	2.26962882885807	0.00359369766344
1	2.71828182845904	2.70833333333333	0.009948495125712

It can be observed that the maximum error is 0.009948495125712.

The exact and approximate solutions are shown in Figure 3.1 (a) and the resulted error is shown in Figure 3.1 (b).



**Fig. 3.1 (a):** A comparison between the exact and approximate solutions in examples (3.1).



**Fig. 3.1 (b):** Absolute error between exact and numerical solutions in example (3.1).

### 3.2 The Numerical Realization Of Equations (3.1) – (3.2) Using The Modified Laplace Adomian Decopostion Method (LADM)

#### Algorithm 3.2

This algorithm can be illustrated as follows:

1. Input :

- a)  $g(x)$  ,  $m(x)$ : Limits of integration.
  - b)  $\lambda$  : Is a constant parameter.
  - c)  $h(x)$ : The function of the integral equation.
  - d)  $\mathcal{B}(x, t)$ : The kernel function.
  - e)  $S(q(t))$  : Nonlinear term.
  - f) Set  $m(x) = 0$  and  $g(x) = x$ .
2. Calculate: The Adomian polynomials for the nonlinear term .
3. Calculate: The iterations  $q_i(x)$
4. Depending on the following recursive relation

$$\mathcal{L}\{u_0(x)\} = h_1(s),$$

$$\mathcal{L}\{u_1(x)\} = h_2(s) + \frac{1}{s^n} \mathcal{L} \left\{ \int_0^x \mathcal{B}(x, t) \sum_{i=0}^{\infty} A_i(x) dt \right\},$$

$$\mathcal{L}\{u_{i+1}(x)\} = \frac{1}{s^n} \mathcal{L} \left\{ \int_0^x \mathcal{B}(x, t) \sum_{i=0}^{\infty} A_i(x) dt \right\}, \quad i \geq 1,$$

we will need to find all iteration  $q_i(x)$  using Matlab as follow

```
%%for i = 1:1
```

```
U1 = k1(x, t);
```

```
%%for i = 2:2
```

```
A1 =  $\mathcal{B}(x, t) * G_0$ ;
```

```
F = int(A1, t, 0, x);
```

```
D = laplace(F);
```

```
B = (O * D);
```

```
C = ilaplace(B);
```

```
U2 = k2(x, t) + subs(C, t, x);
```

```
%%for i = 3:m
```

```
Ai - 1 =  $\mathcal{B}(x, t) * G_{i-2}$ 
```

```
Fi - 2 = int(Ai - 1, t, 0, x);
```

```
Di - 2 = laplace(Fi - 2);
```

```
Bi - 2 = (O * Di - 2);
```

```
C1 = ilaplace(Bi - 2);
```



$Ui = (subs(C1, t, x));$

*End*

5. For any subinterval of discrete points of  $x$  :

i. Input exact solution and compute exact at all points of  $x$ .

ii. Find absolute error =  $|q - q_n|$ .

iii. Plot the error and all values  $x$ 's.

iv. Output  $(x_i, approximate_i, Exact_i, error_i)$ .

v. Print the table.

vi. Plot and Stop (The process is complete).

2. Output: Approximations  $q(x)$  about interval.

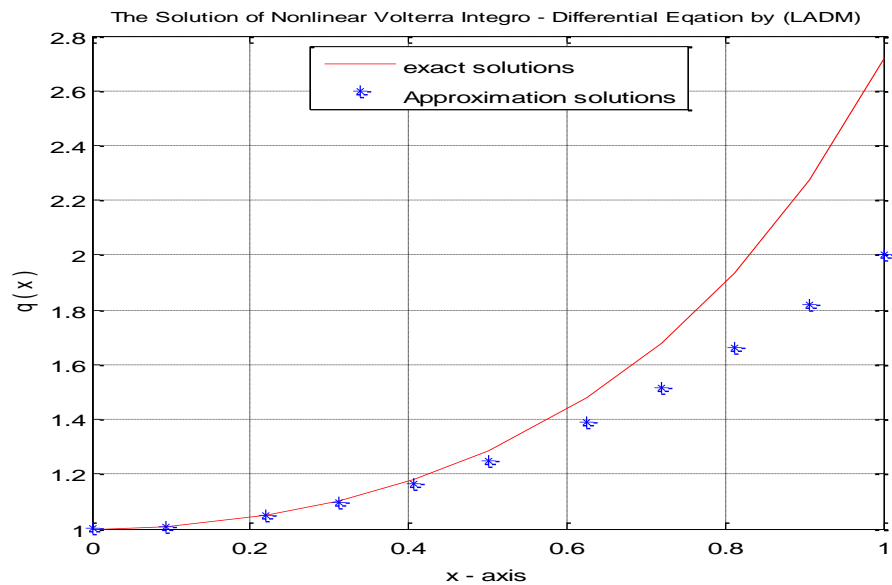
We use Algorithm (3.2) to solve equations (3.1) – (3.2). Table (3.2) contains both the exact and numerical solutions using the Laplace Adomian decomposition method for equations (3.1) – (3.2).

**Table (3.2): The exact and numerical solutions using the Laplace Adomian decomposition method with  $n = 2$ .**

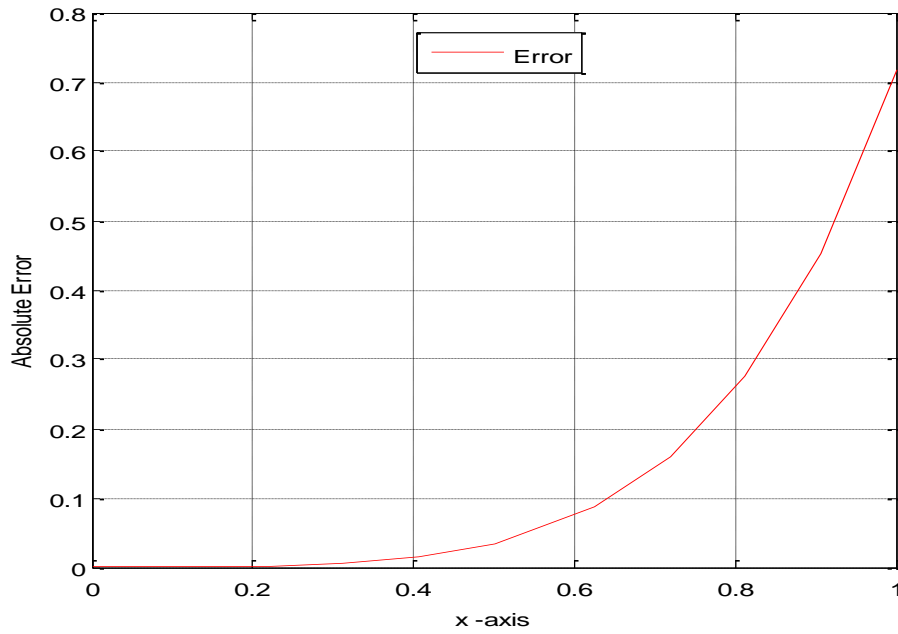
$x$	Exact solution $q(x) = e^{x^2}$	Numerical solution $q_n(x)$	Absolute error $=  q - q_n $
0	1.00000000000000	1.000011200126562	0.000011200126562
0.0938	0.66666684327790	1.008809640126562	0.000027619915065
0.2188	1.04903788068848	1.047884640126562	0.001153240561927
0.3125	1.10258370680894	1.097667450126562	0.004916256682381
0.4062	1.17939127885956	1.165009640126562	0.014381638732999
0.5	1.28402541668774	1.250012200126562	0.034014216561180
0.6250	1.47790419541173	1.390636200126562	0.087267995285177
0.7188	1.67644158014955	1.516684640126562	0.159756940022989
0.8125	1.93509466932358	1.660167450126562	0.274927219197021
0.9062	2.27322252652152	1.821209640126562	0.452012886394959
1	2.71828182845904	2.000011200126561	0.718270628332484

It can be observed that the maximum error is  $= 0.718270628332484$ .

The exact and approximate solutions are shown in Figure 3.2 (a) and the resulted error is shown in Figure 3.2 (b).



**Fig. 3.2 (a):** A comparison between the exact and approximate solutions in examples (3.1).



**Fig. 3.2 (b):** Absolute error between exact and numerical solutions in example (3.1).

### 3.3 The Numerical Realization Of Equation (3.1) – (3.2) Using The Variational Iteration Method (VIM)

#### Algorithm 3.3

This Algorithm can be illustrated as follows:

1. Input :

- a)  $g(x)$  ,  $m(x)$ : Limits of integration.
- b)  $\lambda$  : Is a constant parameter.
- c)  $h(x)$ : The function of the integral equation.
- d)  $\mathcal{B}(x, t)$ : The kernel function.
- e)  $S(q(t))$  : Nonlinear term.

f) Set  $m(x) = 0$  and  $g(x) = x$ .

2. Calculate: The iterations  $q_i(x)$

When  $i = 1$

$$A = \int_0^t e^{t-r} y(1)^2 dr$$

$$B = \int_0^x (0 - 1 - e^t + 2te^t + e^{2t} - A) dt$$

$$Q1 = y(1) - (B);$$

$$A = \text{expand}(U1);$$

3. Set and solve more iterations.

4. For any subinterval of discrete points of  $x$  :

i. Input exact solution and compute exact at all points of  $x$ .

ii. Find absolute error  $= |q - q_n|$ .

iii. Plot the error and all values  $x$ 's.

iv. Output  $(x_i, \text{approximate}_i, \text{Exact}_i, \text{error}_i)$ .

v. Print the table.

vi. Plot and Stop (The process is complete).

5. Output: Approximations  $q(x)$  about interval.

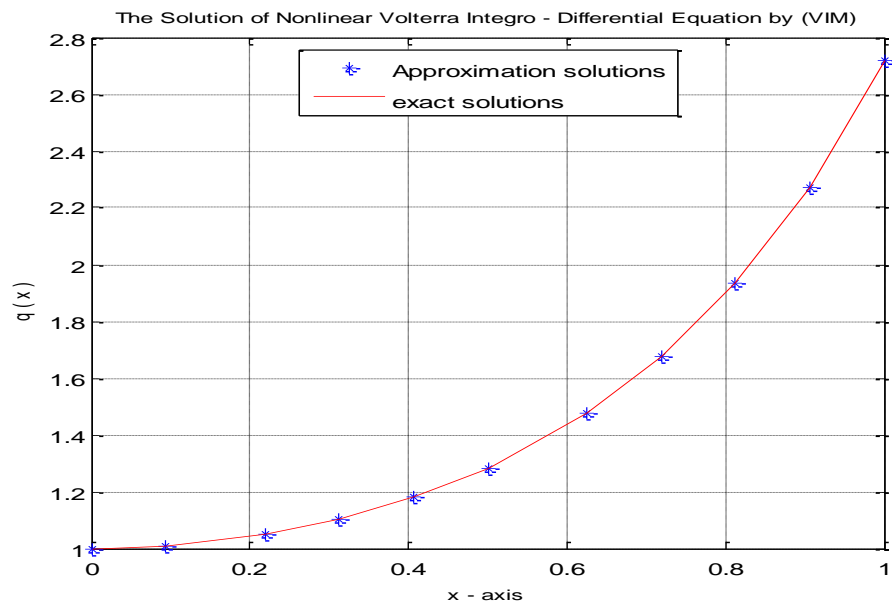
We use Algorithm (3.3) to solve equations (3.1) – (3.2). Table (3.3) contains both the exact and numerical solutions using the Variational Iteration Method (3.1) – (3.2).

**Table (3.3): The exact and numerical solutions using the Variational Iteration Method with Adomian Polynomials (DTM) with  $n = 3$ .**

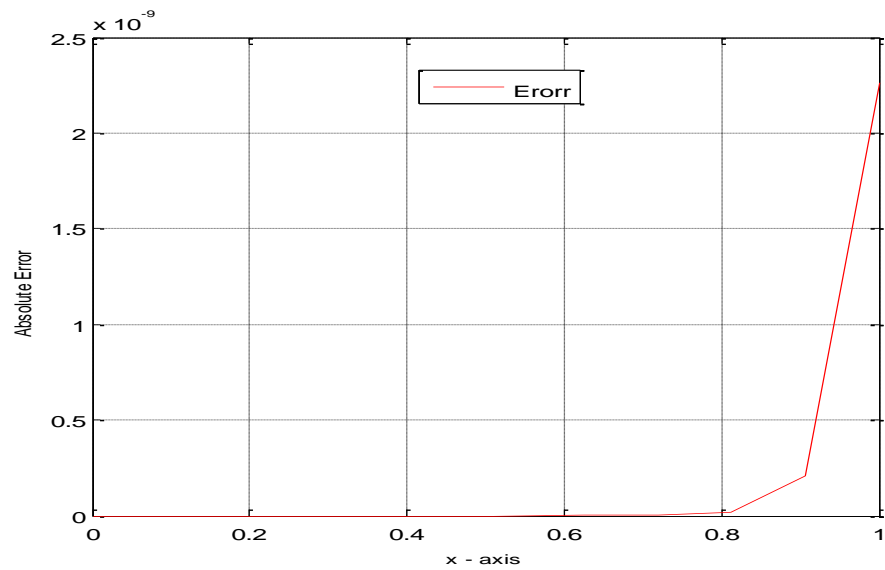
$x$	Exact solution $q(x) = e^{x^2}$	Numerical solution $q_n(x)$	Absolute error $=  q - q_n $
0	1.000000000000000	1.000000000000000	0
0.0938	1.008837260041627	1.008837260041627	0
0.2188	1.049037880688489	1.049037880688489	0
0.3125	1.102583706808942	1.02583706808942	0
0.4062	1.179391278859560	1.179391278859560	0
0.5	1.284025416687741	1.284025416687741	0
0.6250	1.477904195411739	1.477904195411739	0.0000000000000027
0.7188	1.676441580149551	1.676441580149551	0.00000000000000787
0.8125	1.935094669323582	1.935094669323582	0.000000000015065
0.9062	2.273222526521520	2.273222526521520	0.000000000209527
1	2.718281828459046	2.718281828459046	0.000000002260553

It can be observed that the maximum error is 0.000000002260553

The exact and approximate solutions are shown in Figure 3.3 (a) and the resulted error is shown in Figure 3.3 (b).

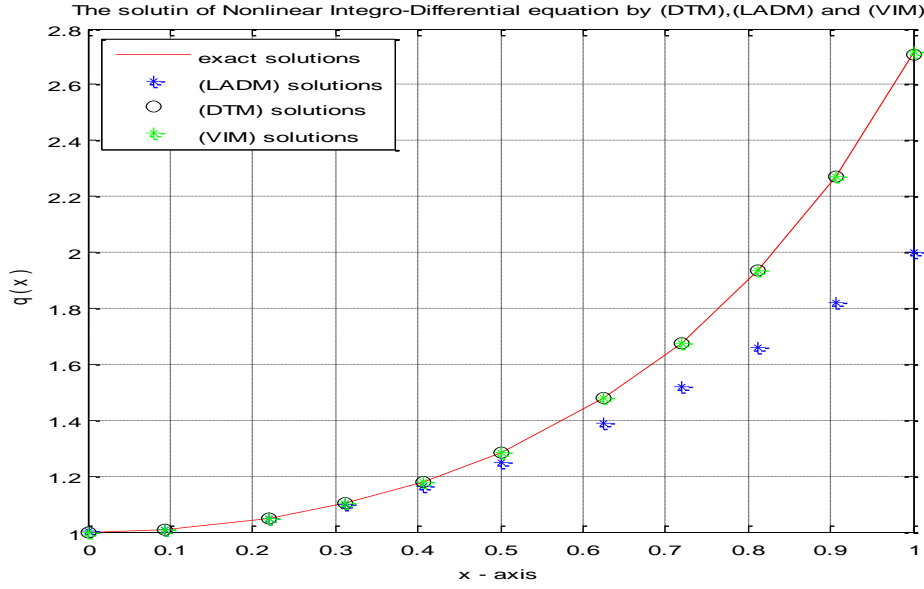


**Fig. 3.3 (a):** A comparison between the exact and approximate solutions in examples (3.1).



**Fig. 3.3 (b):** Absolute error between exact and numerical solutions in example (3.1) .

The exact and approximate solutions of all methods are shown in Figure 3.4.



**Fig. 3.4:** A comparison between the exact and approximate solutions in example (3.1).

### Example 3.2

Consider the Volterra nonlinear integro-differential equation :

$$6(x^2 + 1) \frac{d}{dx} q(x) = (x^3 + 3x^2 + 6x + 6)e^{-x} + \int_0^x t^3 e^{-\tan q(t)} dt, \quad (3.3)$$

with the initial condition:

$$q(0) = 0 \quad (3.4)$$

The exact solution of equation (3.3) is :

$$q(x) = \tan^{-1} x$$

### 3.4 The Numerical Realization Of Equations (3.3) – (3.4) Using The Differential Transform Method with Adomian Polynomials (DTM)

Using Algorithm (3.1) for equations (3.3) – (3.4) . Table (3.5) contains both the exact and numerical solutions using the Differential Transform Method with Adomian Polynomials (DTM) for equations (3.3) – (3.4).

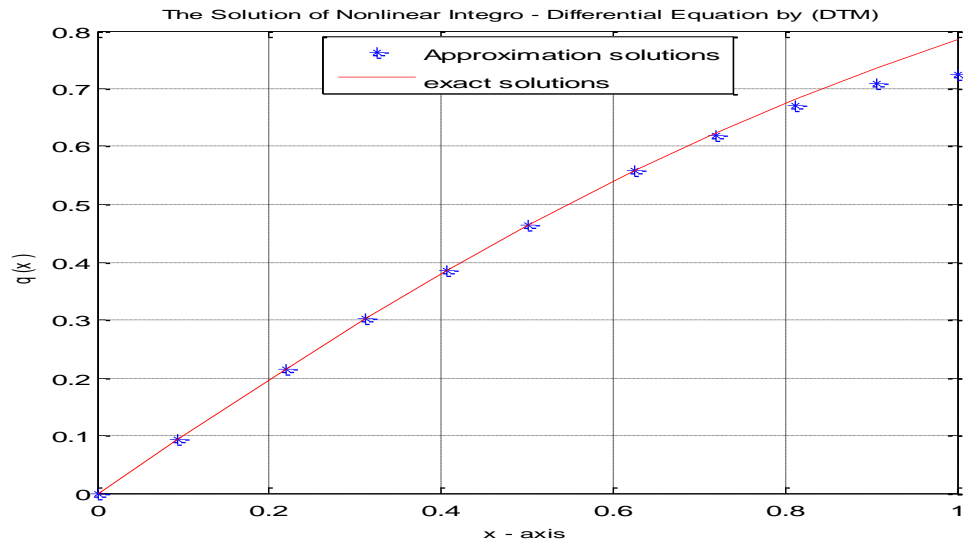
**Table (3.6): The exact and numerical solutions using the Differential Transform Method with Adomian Polynomials (DTM) with  $n = 9$**

$x$	Exact solution $q(x) = \tan^{-1}x$	Numerical solution $q_n(x)$	Absolute error $=  q - q_n $
0	0	0	0
0.0938	0.09352634530384	0.09352634524183	0.00000000006201
0.2188	0.21540541588183	0.21540529299132	0.00000012289051
0.3125	0.30288486837497	0.30288194350543	0.00000292486953
0.4062	0.38583975163390	0.38581026419894	0.00002948743495
0.5	0.46364760900080	0.46346726190476	0.00018034709604
0.6250	0.55859931534356	0.55737143471127	0.00122788063228
0.7188	0.62323229760101	0.61921887947618	0.00401341812483
0.8125	0.68231655487474	0.67113242450924	0.01118413036550
0.9062	0.73622997521529	0.70867363217887	0.02755634303641
1	0.78539816339744	0.72380952380952	0.06158863958792

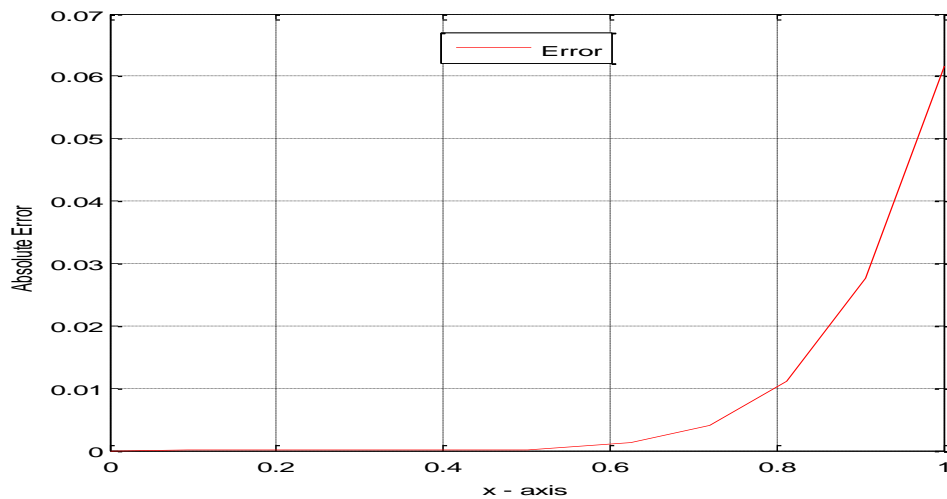
It can be observed that the maximum error is 0.06158863958792.

The exact and approximate results of  $q(x)$  are shown in Figure 3.4 (a) and the resulted error is shown in Figure 3.5 (b).





**Fig. 3.5 (a):** A comparison between the exact and approximate solutions in examples (3.2).



**Fig. 3.5 (b):** Absolute error between exact and numerical solutions in example (3.2) .

### 3.5 The Numerical Realization Of Equations (3.3) – (3.4) Using The Variational Iteration Method (VIM)

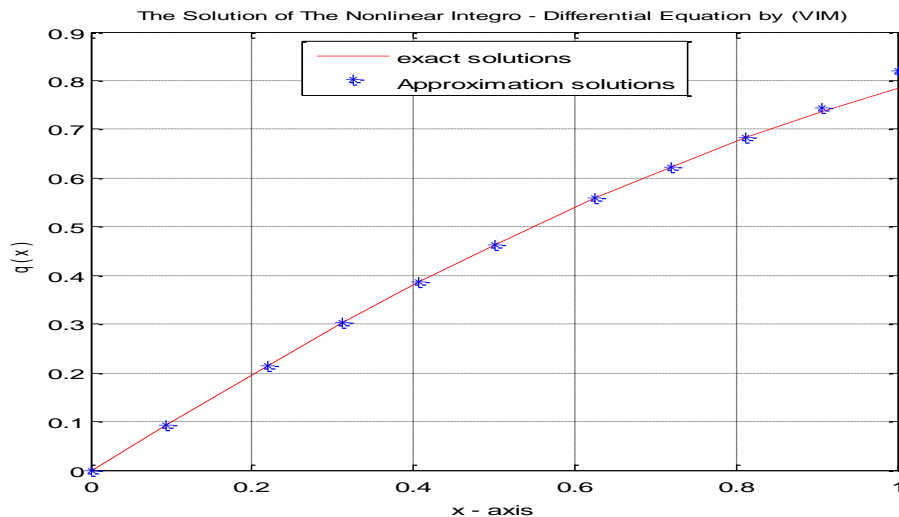
Using Algorithm (3.3) for equations (3.3) – (3.4) . Table (3.5) contains both the exact and numerical solutions using the Variational Iteration Method (VIM) for equations (3.3) – (3.4) .

**Table (3.5): The exact and numerical solutions using the Variational Iteration with  $n = 6$ .**

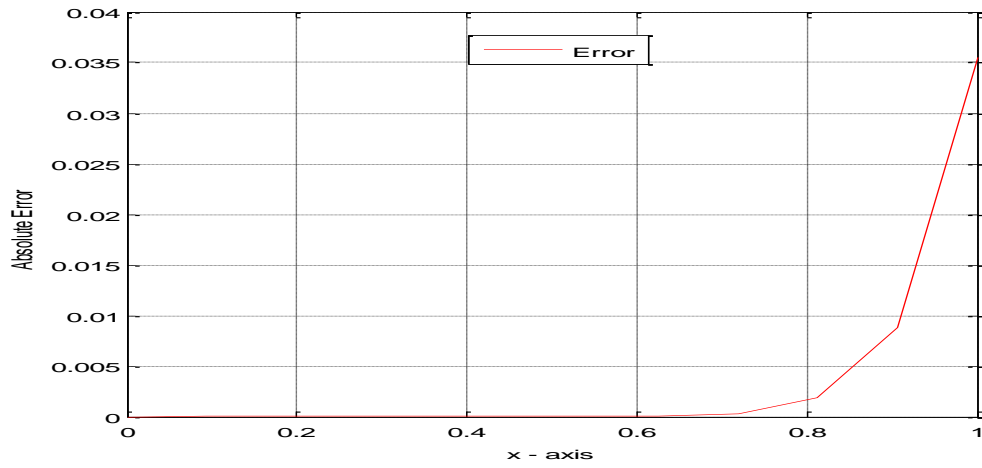
$x$	Exact solution $q(x) = \tan^{-1}x$	Numerical solution $q_n(x)$	Absolute error $=  q - q_n $
0	0	0	0
0.0938	0.09352634530384	0.093526345303846	0.00000000008066
0.2188	0.21540541588183	0.215405415889905	0.000000001624805
0.3125	0.30288486837497	0.302884869999776	0.000000078719100
0.4062	0.38583975163390	0.385839830353001	0.00000166761338
0.5	0.46364760900080	0.463649276613144	0.000043044026501
0.6250	0.55859931534356	0.558642359370063	0.000043044026501
0.7188	0.62323229760101	0.623556289841569	0.000323992240555
0.8125	0.68231655487474	0.684190697399316	0.001874142524568
0.9062	0.73622997521529	0.745076036939866	0.008846061724573
1	0.78539816339744	0.820934620934621	0.035536457537173

It can be observed that the maximum error is 0.035536457537173.

The exact and approximate solutions are shown in Figure 3.6 (a) and the resulted error is shown in Figure 3.6 (b).

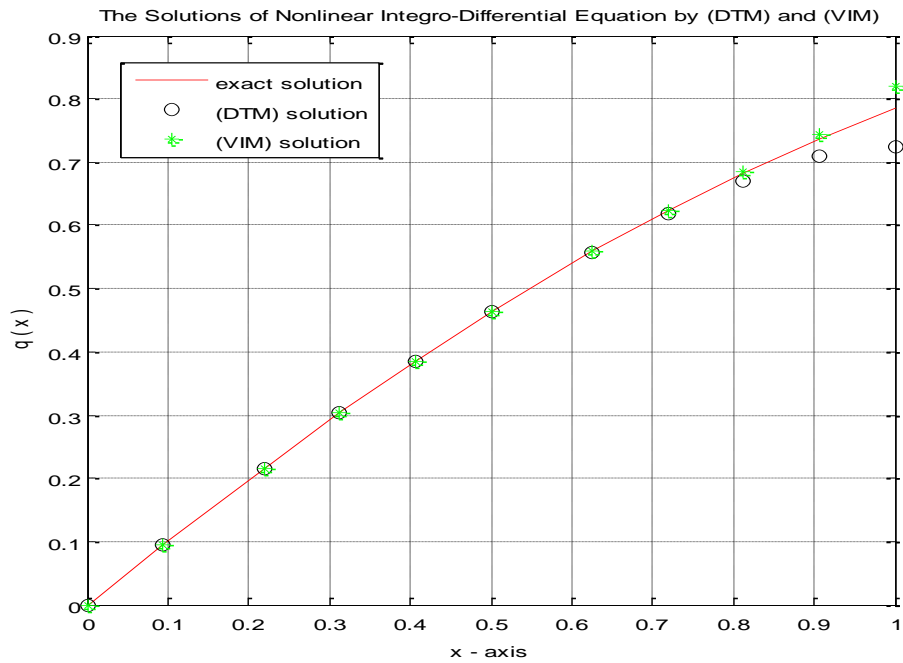


**Fig. 3.6 (a):** A comparison between the exact and approximate solutions in examples (3.2).



**Fig. 3.6 (b):** Absolute error between exact and numerical solutions in example (3.2).

The exact and approximate solutions of all methods are shown in Figure 3.7.



**Fig. 3.7:** A comparison between the exact and approximate solutions in examples (3.2).

### Example 3.3

Consider the Volterra nonlinear integro-differential equation :

$$\frac{d}{dx}q(x) = 1 - e^x + e^{2x} + \int_0^x e^{x-t}(1 - q^2(t))dt, \quad (3.5)$$

with the initial condition:

$$q(0) = 1 \quad (3.6)$$

The exact solution of equation (3.3) is :

$$q(x) = e^x.$$

### 3.6 The Numerical Realization of Equations (3.5) – (3.6) Using The Modified Laplace Adomian Decopostion Method (LADM).

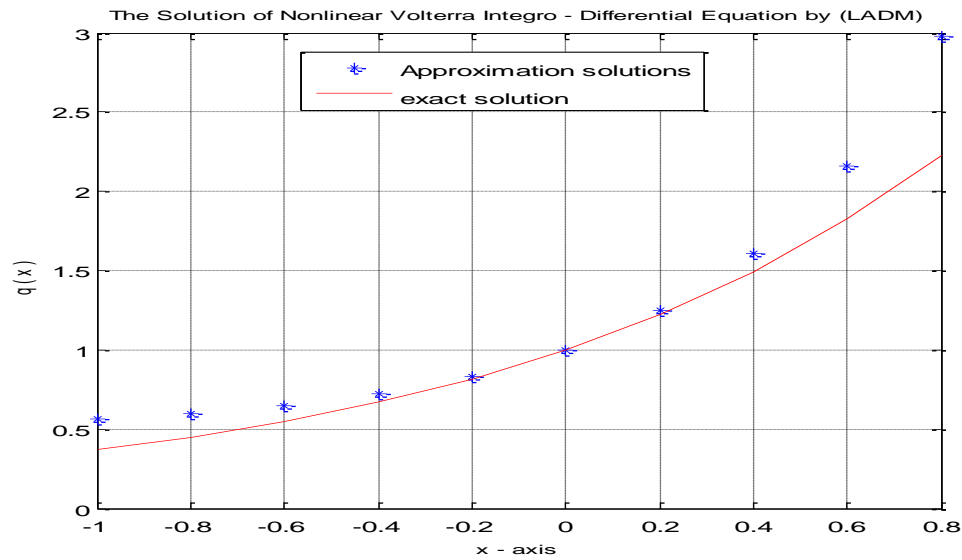
Using Algorithm (3.2) for equations (3.5) – (3.6), Table (3.6) contains both the exact and numerical solutions using the Laplace Adomian Decopostion Method for equations (3.5) – (3.6) .

**Table (3.6): The exact and numerical solutions using the Laplace Adomian Decopostion Method (LDTM) with  $n = 1$ .**

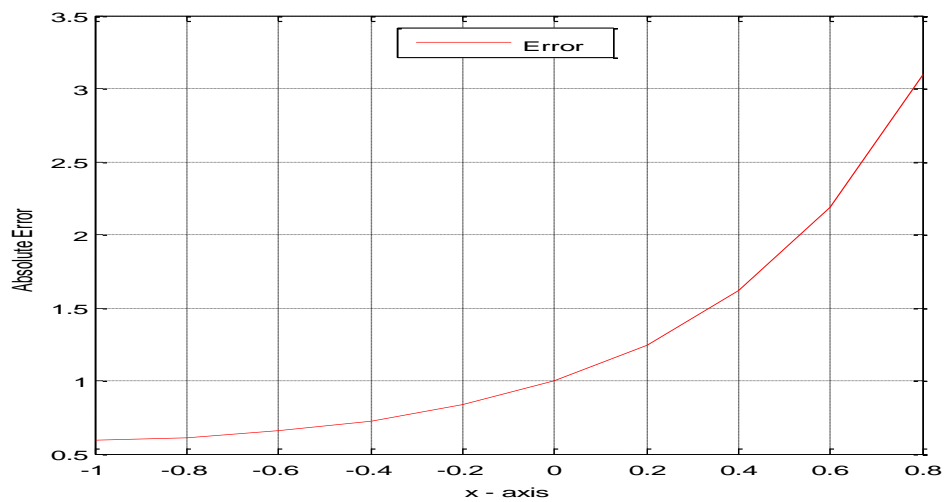
$x$	Exact solution $q(x) = e^x$	Numerical solution $q(x) = e^x$	Absolute error $=  q - q_n $
-1	0.367879441171442	0.567667641618306	0.199788200446864
-0.8	0.449328964117222	0.600948258997328	0.151619294880106
-0.6	0.548811636094027	0.650597105956101	0.101785469862075
-0.4	0.670320046035639	0.724664482058611	0.054344436022971
-0.2	0.818730753077982	0.835160023017820	0.016429269939838
0	1.000000000000000	1.000000000000000	0
0.2	1.221402758160170	1.245912348820635	0.024509590660465
0.4	1.491824697641270	1.612770464246234	0.120945766604964
0.6	1.822118800390509	2.160058461368274	0.337939660977765
0.8	2.225540928492468	2.976516212197558	0.750975283705090

It can be observed that the maximum error is 0.750975283705090.

The exact and approximate solutions are shown in Figure 3.8 (a) and the resulted error is shown in Figure 3.8 (b).



**Fig. 3.8 (a):** A comparison between the exact and approximate solutions in examples (3.3) .



**Fig. 3.8 (b):** Absolute error between exact and numerical solutions in example (3.3).

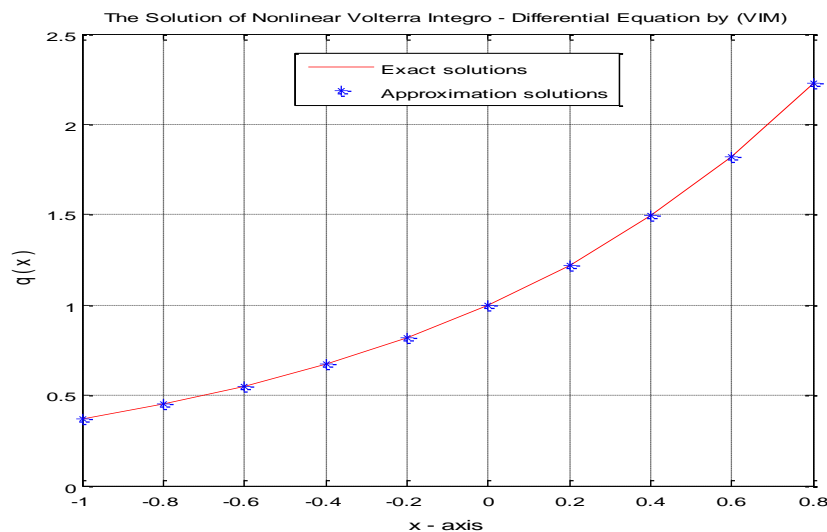
### 3.7 The Numerical Realization Of Equations (3.5) – (3.6) Using The Variational Iteration Method (VIM)

**Table (3.7): The exact and numerical solutions using the variational iteration method (VIM) with  $n = 4$ .**

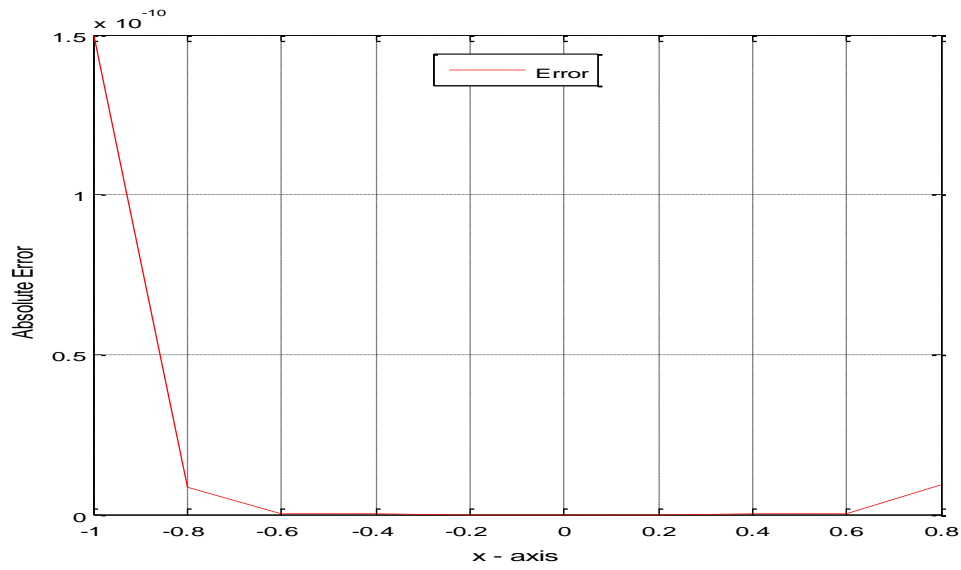
$x$	Exact solution $q(x) = e^x$	Numerical solution $q_n(x)$	Absolute error $=  q - q_n $
-1	0.367879441171442	0.367879441321282	0.000000000149839
-0.8	0.449328964117222	0.449328964125571	0.00000000008350
-0.6	0.548811636094027	0.548811636094228	0.00000000000201
-0.4	0.670320046035639	0.670320046035640	0.00000000000001
-0.2	0.818730753077982	0.818730753077982	0
0	1.000000000000000	1.000000000000000	0
0.2	1.221402758160170	1.221402758160170	0
0.4	1.491824697641270	1.491824697641269	0.00000000000001
0.6	1.822118800390509	1.822118800390290	0.00000000000219
0.8	2.225540928492468	2.225540928483107	0.00000000009361

It can be observed that the maximum error is 0.00000000009361.

The exact and approximate solutions are shown in Figure 3.9 (a) and the resulted error is shown in Figure 3.9 (b).

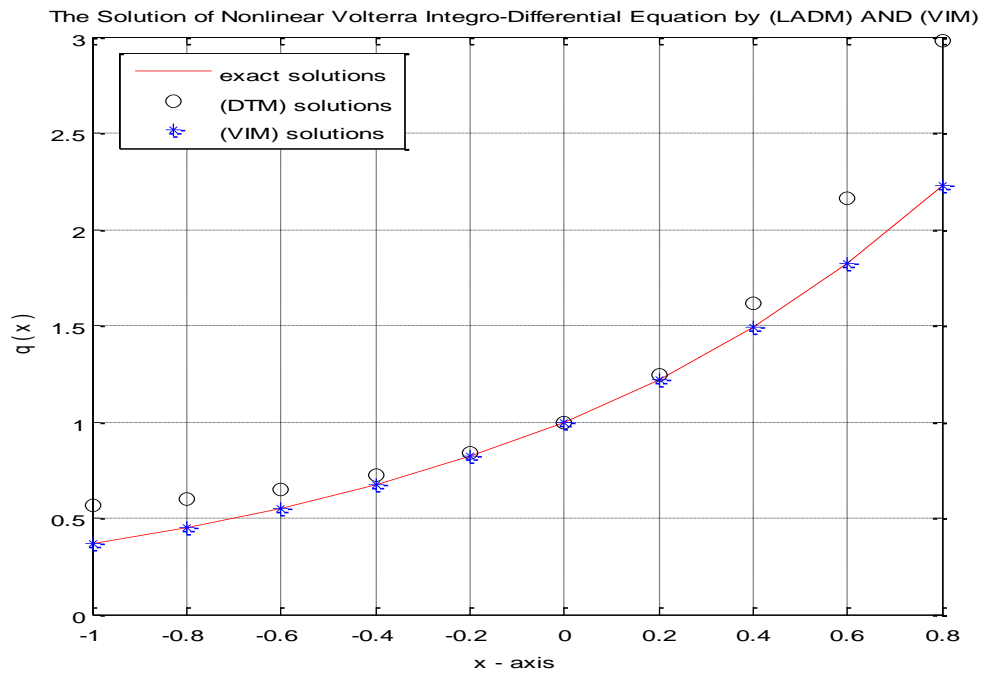


**Fig. 3.9 (a):** A comparison between the exact and approximate solutions in examples (3.3).



**Fig. 3.9 (b):** Absolute error between exact and numerical solutions in example (3.3).

The exact and approximate solutions of all methods are shown in Figure 3.7.



**Fig. 3.10:** A comparison between the exact and approximate solutions in examples (3.3).

### Example 3.4

Consider the Volterra nonlinear integro-differential equation :

$$\frac{d}{dx}q(x) = 1 + e^x - 2xe^x - e^{2x} + \int_0^x e^{x-t}q^2(t)dt, \quad (3.7)$$

with the initial condition:

$$q(0) = 2 \quad (3.8)$$

The exact solution of equation (3.4) is :

$$q(x) = 1 + e^x.$$

### 3.8 The numerical Realization of equations (3.7) – (3.8) Using The Variational Iteration Method (VIM)

Using Algorithm (3.3) for equations (3.7) – (3.8) . Table (3.6) contains both the exact and the numerical results using the Variational Iteration Method for equations (3.7) – (3.8).

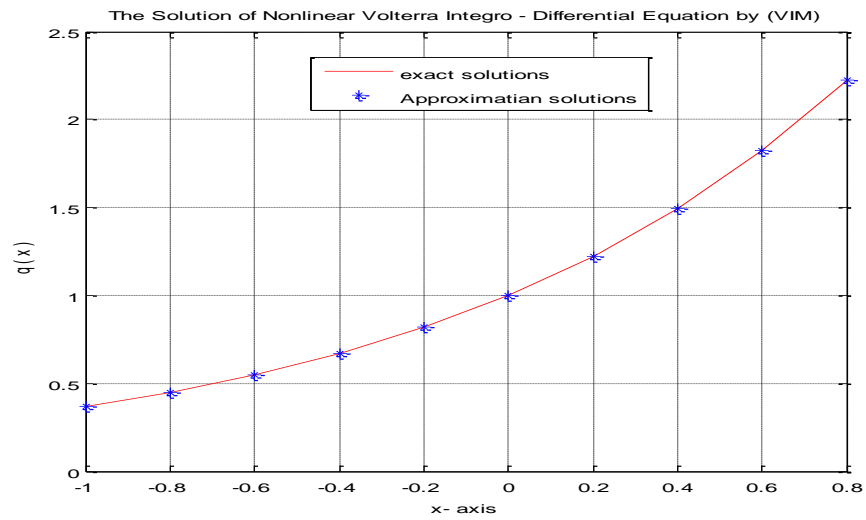
Table (3.8): The exact and numerical solutions using Variational Iteration Method (VIM) with  $n = 3$ .

$x$	Exact solution $q(x) = 1 + e^x$	Numerical solution $q_n(x)$	Absolute error $=  q - q_n $
0	2.0000000000000000	2.0000000000000000	0
0.0938	2.098340055937721	2.098340054991735	0.000000000920105
0.2188	2.244582335327159	2.24458182938934	0.000005427229550
0.3125	2.366837941173796	2.366836647670151	0.000005427229550
0.4062	2.501102742986564	2.501096504099779	0.000038359154292
0.5	2.648721270700128	2.648699569298986	0.000185507539221
0.6250	2.868245957432222	2.868163173060590	0.001042650006852
0.7188	3.051969369391527	3.051777803836352	0.003142377825653
0.8125	3.253534787213209	3.253135200988730	0.008390931887913
0.9062	3.474900021868255	3.474130861914352	0.020388681771895
1	3.718281828459046	3.716892914345171	0.045913559787484

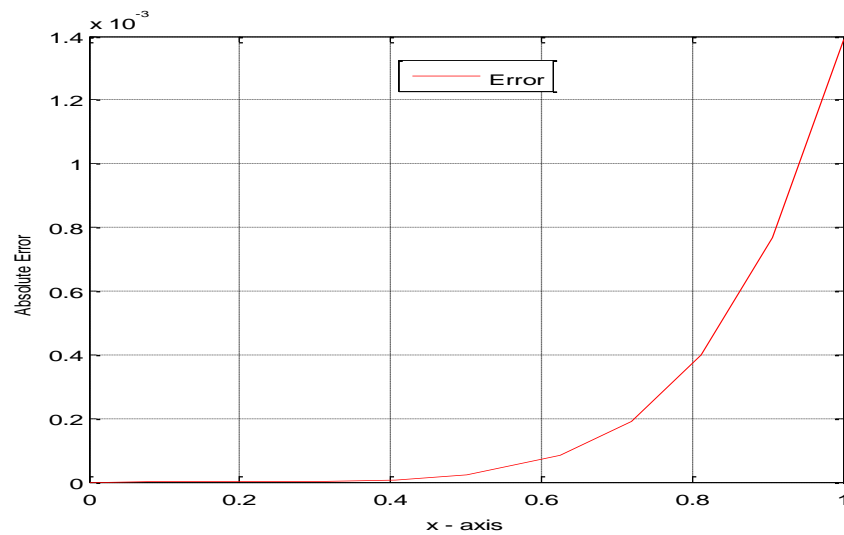


It can be observed that the maximum error is 0.045913559787484.

The exact and approximate solutions are shown in Figure 3.11 (a) and the resulted error is shown in Figure 3.11(b).



**Fig. 3.11 (a):** A comparison between the exact and approximate solutions in examples (3.4).



**Fig. 3.11 (b):** Absolute error between exact and numerical solutions in examples (3.4).

### 3.9 The Numerical Realization Of Equations (3.7) – (3.8) Using The Modified Laplace Adomian Decomposition Method (LADM).

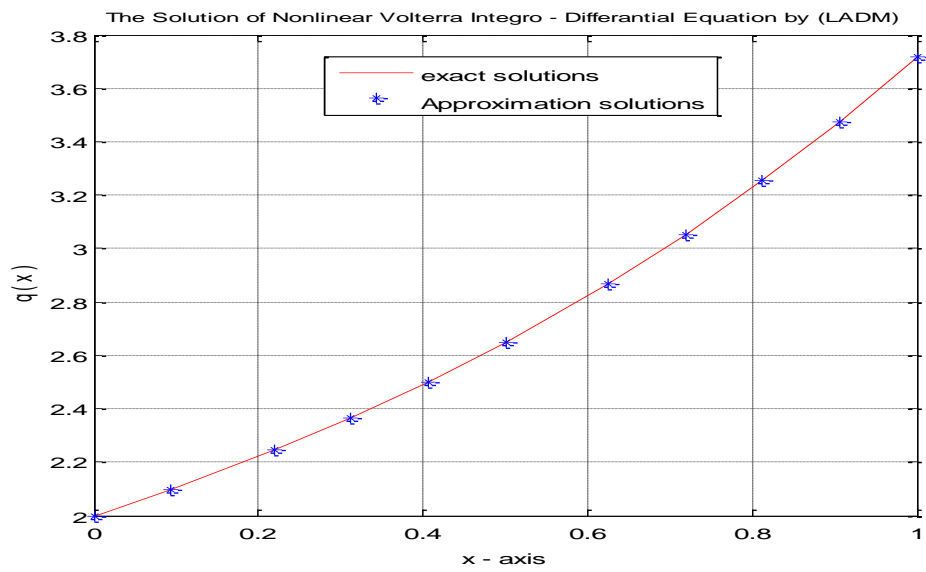
Using Algorithm (3.2) for equations (3.7) – (3.8) . Table (3.6) contains both the exact and the numerical results using the Variational Iteration Method for equations (3.7) – (3.8) .

Table (3.9): The exact and numerical solutions using Modified Laplace Adomian Decomposition Method (LADM) with  $n = 4$ .

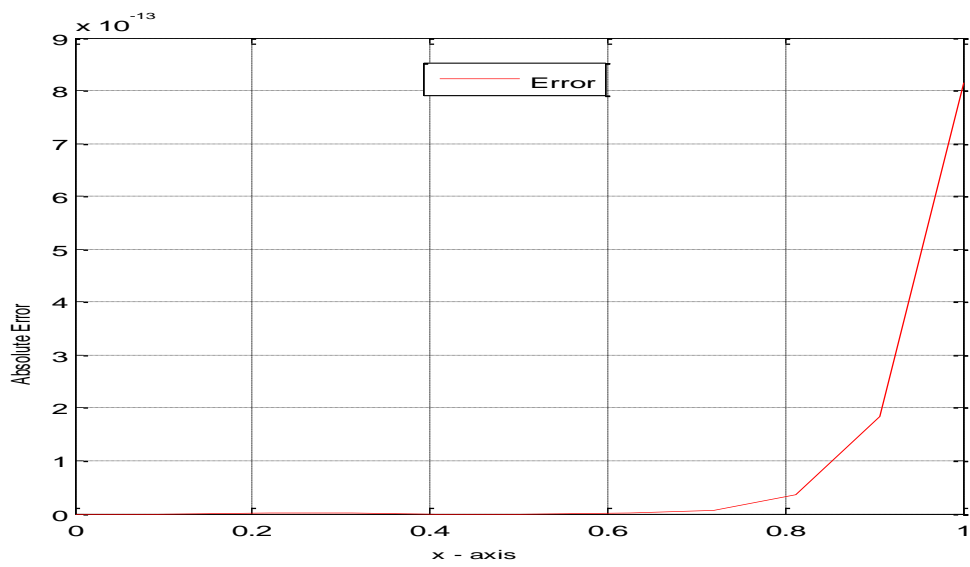
$x$	Exact solution $q(x) = 1 + e^x$	Numerical solution $q_n(x)$	Absolute error $=  q - q_n $
0	2.000000000000000	2.000000000000000	0
0.0938	2.098340055937721	2.098340055937721	0.000000000000000
0.2188	2.244582335327159	2.244582335327160	0.000000000000000
0.3125	2.366837941173796	2.366837941173797	0.000000000000000
0.4062	2.501102742986564	2.501102742986564	0
0.5	2.648721270700128	2.648721270700128	0
0.6250	2.868245957432222	2.868245957432221	0.000000000000001
0.7188	3.051969369391527	3.051969369391522	0.000000000000006
0.8125	3.253534787213209	3.253534787213173	0.000000000000036
0.9062	3.474900021868255	3.474900021868070	0.000000000000185
1	3.718281828459046	3.718281828458230	0.000000000000816

It can be observed that the maximum error is  $= 0.000000000000816$ .

The exact and approximate solutions are shown in Figure 3.12 (a) and the resulted error is shown in Figure 3.12 (b).



**Fig. 3.12(a):** A comparison between the exact and approximate solutions in examples (3.4).



**Fig 3.12 (b):** Absolute error between exact and numerical solutions in examples (3.4).

### 3.10 The Numerical Realization Of Equations (3.7) – (3.8) Using The Differential Transform Method With Adomian Polynomials (DTM)

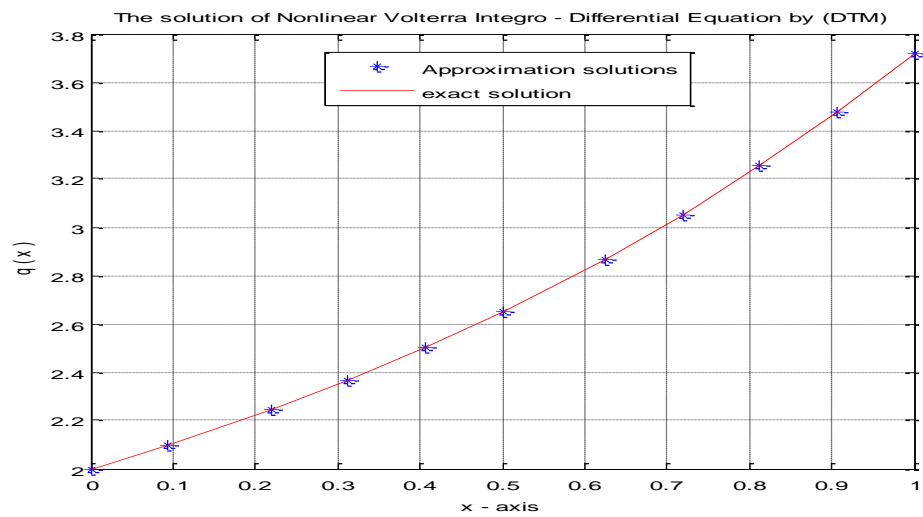
Using Algorithm (3.1) for equations (3.7) – (3.8) . Table (3.6) contains both the exact and the numerical results using the Variational Iteration Method for equations (3.7) – (3.8).

**Table (3.8): The exact and numerical solutions using Differential Transform Method with Adomian Polynomials (DTM) with  $n = 11$ .**

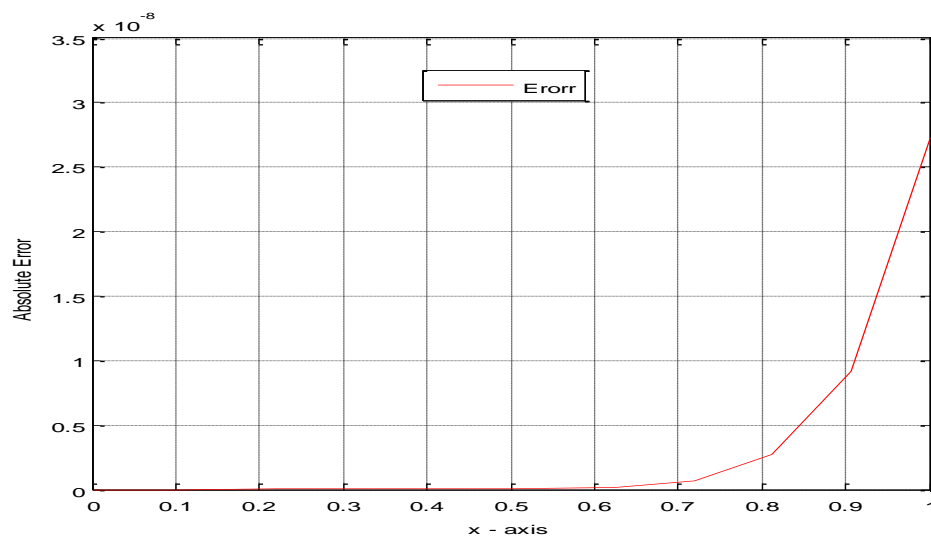
$x$	Exact solution $q(x) = 1 + e^x$	Numerical solution $q_n(x)$	Absolute error $=  q - q_n $
0	2.0000000000000000	2.0000000000000000	0
0.0938	2.098340055937721	2.098340055937721	0
0.2188	2.244582335327159	2.244582335327160	0.0000000000000001
0.3125	2.366837941173796	2.366837941173725	0.0000000000000071
0.4062	2.501102742986564	2.501102742985276	0.0000000000001288
0.5	2.648721270700128	2.648721270687366	0.0000000000012763
0.6250	2.868245957432222	2.868245957282028	0.0000000000150195
0.7188	3.051969369391527	3.051969368686468	0.0000000000705060
0.8125	3.253534787213209	3.253534784476825	0.0000000002736384
0.9062	3.474900021868255	3.474900012701995	0.000000009166260
1	3.718281828459046	3.718281801146385	0.000000027312661

It can be observed that the maximum error is 0.000000027312661.

The exact and approximate solutions are shown in Figure 3.13 (a) and the resulted error is shown in Figure 3.13 (b).

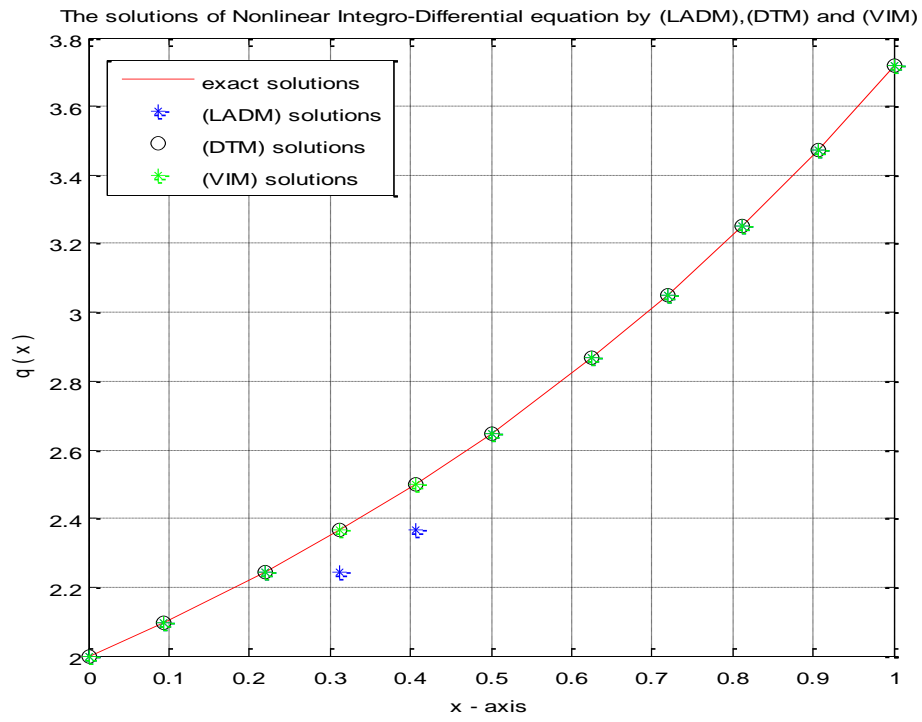


**Fig. 3.13 (a):** A comparison between the exact and approximate solutions in examples (3.4).



**Fig. 3.13 (b):** Absolute error between exact and numerical solutions in examples (3.4).

The exact and approximate solutions of all methods are shown in Figure 3.14.



**Fig. 3.14:** A comparison between the exact and approximate solutions in examples (3.4).

## **Conclusions**

Computational methods have been used to solve nonlinear Volterra integro - differential equation . The numerical methods are implemented in a form of algorithms to solve some numerical examples using Matlab software.

Based on our numerical results, one sees clearly that the Variational Iteration Method (VIM) is the most effective numerical technique for solving nonlinear Volterra integro - differential equation.

## References

- [1] I. Arikoglu, **Solution of boundary value problems for integro - differential equations by using differential transform method**, Appl. Math. Comput. 168 (2005), pp 1145-1158.
- [2] I. Arikoglu, **Solution of integral and integro-differential equation systems by using differential transform method**, Comput. Math. Appl. 65 (2008) 2411-2417.
- [3] T. Batiha, M. Norani and I. Hashim, *Numerical Solutions Of The Nonlinear Integro – Differential Equations*, Int. J. Open Problems Compt. Math., Vol. 1, No. 1, June 2008.
- [4] S. Behiry, **Nonlinear Integro - differential Equations by Differential Transform Method with Adomian Polynomials**, Math. Sci. Lett. 2, No. 3, 209-221 (2013).
- [5] S. Behiry, **Differential Transform Method for Nonlinear Initial - Value Problems by Adomian Polynomials**, General Required Courses Department, Jeddah Community College, king Abdulaziz University, Jaddah 21589, Kingdom of Saudi Arabia, Vol. 4, no. 8, (2012) 581-781.
- [6] A. Bielecki, **A remark on the Banach method - Cacciopoli Tikhnov in the theory of the different ordinary ordinations**, Bull Acad. Polon. Sci. If st. Sci. Math. Phys. Astr. 4 (1956), 261-264.



- [7] A. Borhanifar, Reza Abazari , **Differential transform method for a class of nonlinear integro - differential equations with derivative type kernel**, Department of Mathematics, University of Mohaghegh Ardabili, Ardabil, Iran . November 16, 2011.
- [8] P. Darania, E. Abadian, A. Oskoi, **Linearization method for solving nonlinear integral equations**, Math. Probl. Eng. (2006) Vol. 99 No. 3 2015, 277-287.
- [9] P. Darania, K. Ivaz, **Numerical solution of nonlinear Volterra-Fredholm integro - differential equations**, Appl. Math. Comput. 56 (2008) 2197-2209.
- [10] A. Elsaid, **Fractional differential transform method combined with Adomian polynomials**, Appl. Math. Comput. 218 (2012) 6899-6911.
- [11] J. He, **A new approach to nonlinear partial differential equations**, Commun. Nonlin. Sci. Numer. Simul. 2 (1997) 230-235.
- [12] J. He, **Approximate analytical solution of Blasius' equation**, Commun. Nonlinear. Sci. Numer. Simul. 3 (1998) 260–263.
- [13] J. He, **Non - perturbative methods for strongly nonlinear problems**, Berlin: Dissertation. Dererlag im Internet GmbH, (2006).

- [14] J. He, *Variational iteration method akind of non-linear analytical, technique: some examples*. **Int. J. Non-Linear Mech.** 34 (1999) 699–708.
- [15] J. He, **Variational iteration method for autonomous ordinary differential systems**. *Appl. Math. Comput.* 114 (2000) 115-123.
- [16] J. He, *Some asymptotic methods for strongly nonlinear equations*. **Int. J. Modern Phys. B** 20 (2006) 1141–1199.
- [17] M. Hussain and Majid Khan, **modified Laplace decomposition method**, *Applied Mathematical Sciences*, Vol. 4, 2010, no. 36, 1769-1783.
- [18] M. Inokuti, H. Sekine and T. Mura, **General use of the Lagrange multiplier in nonlinear mathematical physics**. In: Nemat Nassed S, editor. *Variational Method in the Mechanics of Solids*. Pergamon Press, (1978).
- [19] M. Krasnoselskii, **Topological Methods in the Theory of Nonlinear Integral Equations**, Pergamon Press Oxford, 1964.
- [20] T. Kulik and C. Tisdell, *Volterra integral equations on time scales: Basic qualitative and quantitative results with applications to initial value problems on unbounded domains*, **Int. J. Difference Equ.** Appl Vol.3, (2008).No. 1 (2008), 103–133.

- [21] P. Kythe, P. Puri, **Computational Methods for Linear Integral Equations**, University of New Orleans, New Orleans 1992.
- [22] J. Manafianheris , **Solving the integro - differential equations using the modified Laplace Adomian decomposition method**, Journal of Mathematical Extension , Vol. 6, (2012). No. 1, 65-79.
- [23] A. Mahmood and L. Sadoon, **Existence of a Solution of a Cartain Volterra - Fredholm Integro Differential Equations**, J. Edu. & Sci., Vol.(25), (2012). No.(3), p (62- 67).
- [24] K. Maleknejed, F. Mirzaee, *Numerical solution of integro - differential equations by using rationalized Haar functions method*, Kybernetes **Int. J. Syst. Math.** 35 (2006) 1735-1744.
- [25] B. Pachpatte, *Applications Of The Leray – Schauder Alternative To Some Volterra Integral and Integro - Differential Equations*, **Indian J. pure appl. Math., Inform.** 26 (12) : 1161-1168, December 1995.
- [26] A. Andrzel Granas, *On the Leray–Schauder Alternative, Topological Methods in the Nonlinear Analysis*, **Journal of the Juliuss Schauder center** ,Vol. 2, (1993). 225-231.
- [27] B. Pachpatte, **Inequalities for Differential and Integral Equations**, Academic Press, New York, 1998.

- [28] B. Pachpatte, **Integral and Finite Difference Inequalities and Applications**, North-Holland Mathematics Studies, Vol. 205, Elsevier Science B.V. Amsterdam, 2006.
- [29] B. Pachpatte, **On Cartain Volterra Integral and Integro-Differential Equations**, Facta Universitatis, Ser. Math. Inform. Vol. (23), p (1- 12), (2008).
- [30] C. Pitts, **Introduction to metric spaces**, Oliver and Boyd, Edinburgh, (1972).
- [31] M. Rashed, **Numerical solution of functional differential, integral and integro-differential equations**, Appl. Numer. Math.156 (2004) 485- 492.
- [32] M. Razzaghi, S. Yousefi, **Legendre wavelets method for nonlinear Volterra-Fredholm integral equations**, Math. Comput. Simul. 70 (2005) 1-8.
- [33] M. Reihani, Z. Abadi, ***Rationalized Haar functions method for solving Fredholm and Volterra integral equations***, J. Comput. Appl. Math. 200 (2007) 12-20.
- [34] N. Sweilam, **Fourth order integro-differential equations using variational iteration method**. Comp. Math. Appl. (2007) no..54 (7-8) 1086-1091.

- [35] H. Tidke, **Existence of global solutions to nonlinear mixed Volterra-Fredholm integro differential equations with nonlocal conditions**, Jelec. Diff. Eqs., Vol. (2009), No. (55), p (1- 7), (2009).
- [36] A. Wazwaz, **The combined Laplace transform - Adomian decomposition method for handling nonlinear Volterra integro – differential equations** , Appl. Math. Comput. 216 (2010) 1304 -1309.
- [37] A. Wazwaz, **Linear and Nonlinear Integral Equations: Methods and Applications**, Springer Heidelberg, Dordrechi London, (2011).
- [38] J. Zhao, RM. Corless, **Compact finite difference method for integro - differential equations** ,Appl. Math. Comput. 177 (2006) 271- 288.
- [39] J. Zhou, **Differential Transformation and Its Applications for Electrical Circuits**, Huazhong University Press, Wuhan, China 1986.
- [40] S. Yalcinbas ,**Taylor polynomial solution of nonlinear Volterra-Fredholm integral quations**, Appl. Math. Comput. 127 (2002) 195-206.

## Appendix

### Matlab Code for Differential Transform Method (DTM) for Example

#### 3.1

- `clc`
- `clear all`
- `format long`
- `%The general form of nonlinear vollterra integro differential equation`
- `%Display`
- `disp(sprintf('Farah Abu-Thabet'))`
- `disp(sprintf('Laplace Method'))`
- `%% in put data`
- `u(1)=1;`
- `u(2)=0;`
- `u(3)=1;`
- `u(4)=0;`
- `%%A0`
- `G(1)=0;`

```

➤ G(2)=0;

➤ G(3)=1;

➤ G(4)=0;

➤ G(5)=1;

➤ %%nonlinear adomian polynomial

➤ %%for i=4

➤ for k=2:6

➤ u(k+3)=(1/((k+1)*(k+2)))*(6*u(k+1)+(8/k)*G(k-1));

➤ end

➤ %%solution of problem

➤ syms x

➤ yapp=0;

➤ for i=1:9

➤ yapp=yapp+(u(i)*x^(i-1));

➤ end

➤ yapp

➤ %%Exact solution

```

```

➤ x=[0 0.0938 0.2188 0.3125 0.4062 0.5 0.6250 0.7188 0.8125 0.9062
    1];

➤ M1=x.^8./24 + x.^6./6 + x.^4./2 + x.^2 + 1;

➤ exact=exp(x.^2);

➤ error=abs(M1-exact);

➤ plot(x,M1,x,exact,'m-.')

➤ %%print

➤ disp(sprintf(' TABLE(1) :) '))

➤ [x' exact' M1' error']

#####

```

### **Matlab Code for Differential Transform Method (DTM) for Example 3.2**

```

➤ clc

➤ clear all

➤ format long

➤ %The general form of nonlinear vollterra integro differential
    equation

➤ %Display

```



- `disp(sprintf('Farah Abu-Thabet'))`
- `disp(sprintf('Transform'))`
- `%% in put data`
- `u(1)=0;`
- `u(2)=1;`
- `%% A0`
- `G(1)=1;`
- `G(2)=1;`
- `G(3)=1;`
- `G(4)=-1/6;`
- `G(5)=1/24;`
- `%% nonlinear adomian polynomial`
- `for k=2:3`
- `u(k+1)=(-1*(k-2)*u(k-1)/(k))+(1/(6*(k+1))*[((-1^k)*(6-11*k+6*k^2-`  
`k^3)/factorial(k))]);`
- `End`
- `u(5)=-2*u(3)/4;`

- $u(6) = -3*u(4)/5 + 1/30 * ((-6/\text{factorial}(4)) + G(1)/4);$
- $u(7) = -4*u(5)/6 + 1/36 * ((24/\text{factorial}(5)) + G(2)/5);$
- $u(8) = -5*u(6)/7 + 1/42 * ((-60/\text{factorial}(6)) + G(3)/6);$
- $u(9) = -6*u(7)/8 + 1/48 * ((120/\text{factorial}(7)) + G(4)/7);$
- $u(10) = -7*u(8)/9 + 1/54 * ((-210/\text{factorial}(8)) + G(5)/8);$
- %%solution of problem
- syms x
- yapp=0;
- for i=1:7
- yapp=yapp+(u(i)\*x^(i-1));
- end
- yapp
- %%Exact solution
- $x = [0 \ 0.0938 \ 0.2188 \ 0.3125 \ 0.4062 \ 0.5 \ 0.6250 \ 0.7188 \ 0.8125 \ 0.9062$   
1];
- $M1 = x.^5./5 - x.^3./3 - x.^7./7 + x;$
- exact=atan(x);

- error=abs(M1-exact);
- plot(x,M1,x,exact,'m-.')
- %%print
- disp(sprintf(' TABLE(1) :) '))
- [x' exact' M1' error']

#####

### **Matlab Code for Differential Transform Method (DTM) for Example 3.4**

- clc
- clear all
- format long
- %The general form of nonlinear vollterra integro differential equation
- %Display
- disp(sprintf('Farah Abu-Thabet'))
- disp(sprintf('Transform'))
- %% in put data
- u(1)=0;

- $u(2)=1;$
- %%A0
- $G(1)=0;$
- $G(2)=0;$
- $G(3)=1;$
- $G(4)=1;$
- $G(5)=7/12;$
- $G(6)=1/2;$
- %%non linear adomian polynomial
- $u(3)=(1/\text{factorial}(2))+(1/\text{factorial}(2))*(G(1)/\text{factorial}(0));$
- $u(4)=(1/\text{factorial}(3))+(1/\text{factorial}(3))*((G(2)/\text{factorial}(1))-$   
 $(G(1)/(\text{factorial}(1)*\text{factorial}(0))));$
- $u(5)=(1/\text{factorial}(4))+(1/\text{factorial}(4))*((G(4)/\text{factorial}(3))-$   
 $(G(3)/(\text{factorial}(2)*\text{factorial}(2)))+(G(2)/(\text{factorial}(3)*\text{factorial}(1))));$
- $u(6)=(1/\text{factorial}(5))+(1/\text{factorial}(5))*((G(5)/\text{factorial}(4))-$   
 $(G(4)/(\text{factorial}(2)*\text{factorial}(3)))+(G(3)/(\text{factorial}(3)*\text{factorial}(2))));$

- $u(7)=(1/\text{factorial}(6))+(1/\text{factorial}(6))*((G(6)/\text{factorial}(5))-$   
 $(G(5)/(\text{factorial}(2)*\text{factorial}(4)))+(G(4)/(\text{factorial}(3)*\text{factorial}(3)))-$   
 $(G(3)/(\text{factorial}(2)*\text{factorial}(4)))));$
- %%solution of problem
- syms x
- yapp=0;
- for i=1:7
- for j=1:8
- yapp=yapp+(x^2+3)+(taylor(1+exp(x), x, 'Order', 11));
- end
- end
- yapp;
- %%Exact solution
- x=[0 0.0938 0.2188 0.3125 0.4062 0.5 0.6250 0.7188 0.8125 0.9062  
1];
- $M1=x.^{10}/3628800 + x.^9/362880 + x.^8/40320 + x.^7/5040 +$   
 $x.^6/720 + x.^5/120 + x.^4/24 + x.^3/6 + x.^2/2 + x + 2;$
- exact=1+exp(x);

- `error=abs(M1-exact);`
- `plot(x,M1,'*',x,exact,'r')`
- `%%print`
- `disp(sprintf(' TABLE(1) :) '))`
- `[x' exact' M1' error']`

#####

### **Matlab Code for Modified Laplace Adomian Decomposition method (MLAD) for Example 3.1**

- `clc`
- `clear all`
- `format long`
- `%The general form of nonlinear vollterra integro differential  
equation`
- `%Display`
- `disp(sprintf('Farah Abu-Thabet'))`
- `disp(sprintf('Laplace Method'))`
- `%In put data`
- `syms t x s`

- %%for i=1
- $A=2/3$ ;
- %%for i=2
- $U=A*\log(A)*t$ ;
- $F=\text{int}(U,t,0,x)$ ;
- $A1=\text{laplace}(F)$ ;
- $B=(8*A1/(s^2-6))$ ;
- $C=\text{ilaplace}(B)$ ;
- $D=((1/3)*\cosh(\sqrt{6}*x))+(\text{subs}(C,t,x))$ ;
- %%for i=2
- $U1=t*(D*\log(A)+D)$ ;
- $F1=\text{int}(U1,t,0,x)$ ;
- $A2=\text{laplace}(F1)$ ;
- $B1=(8*A2/(s^3-6*s))$ ;
- $C1=\text{ilaplace}(B1)$ ;
- $D1=(\text{subs}(C1,t,x))$ ;
- %% approximation(1)

- `x=[0 0.0938 0.2188 0.3125 0.4062 0.5 0.6250 0.7188 0.8125 0.9062`  
`1];`
- `M1=`  
`(22152124404789197.*cosh(6.^(1/2).*x))./81064793292668928 +`  
`(4869473359433779.*x.^2)./27021597764222976 +`  
`4869473359433779/81064793292668928;`
- `exact=exp(x.^2);`
- `error=abs(M1-exact);`
- `plot(x,M1,x,exact,'m')`
- `%%print`
- `disp(sprintf(' TABLE(1) :) '))`
- `[x' exact' M1' error']`

#####

### **Matlab Code for Modified Laplace Adomian Decomposition method (MLAD) for Example 3.2**

- `clc`
- `clear all`
- `format long`



- %The general form of nonlinear vollterra integro differential equation
- %Display
- disp(sprintf('Farah Abu-Thabet'))
- disp(sprintf('Laplace Method'))
- %In put data
- syms t x s
- %%for i=1
- A=0.5+0.5\*exp(2\*x);
- %%for i=2
- F=(A)^2;
- A1=laplace(F);
- B=(A1/(s\*s-s));
- C=ilaplace(B);
- D=(subs(C,t,x))\*-1;
- %%for i=3
- K=((A)\*(D)\*2);

- `A2=laplace(K);`
- `B1=(A2/(s*(s-1)));`
- `C1=ilaplace(B1);`
- `D1=-1*subs(C1,t,x);`
- `%% approximation(1)`
- `x=[-1 -0.8 -0.6 -0.4 -0.2 0 0.2 0.4 0.6 0.8];`
- `M1=0.5+0.5.*exp(2.*x);`
- `exact=exp(x);`
- `error1=abs(M1-exact);`
- `plot(x,M1,x,exact,'m')`
- `%%print`
- `disp(sprintf(' TABLE(1) :) '))`
- `[x' exact' M1' error']`
- `%% approximation(2)`
- `x=[-1 -0.8 -0.6 -0.4 -0.2 0 0.2 0.4 0.6 0.8];`
- `M2= x./4 - exp(2.*x)./4 - exp(4.*x)./48 + exp(x)./3 - 1/16;`
- `exact=exp(x);`

- `error=abs(M2-exact);`
- `plot(x,M2,x,exact,'m')`
- `%%print`
- `disp(sprintf(' TABLE(2) :) '))`
- `[x' exact' M2' error']`
- `%% approximation(3)`
- `x=[-1 -0.8 -0.6 -0.4 -0.2 0 0.2 0.4 0.6 0.8 ]`
- $$M3 = \frac{(3.*x).}{16} + \frac{(11.*\exp(2.*x)).}{32} - \frac{\exp(3.*x).}{18} + \frac{(13.*\exp(4.*x)).}{576} + \frac{\exp(6.*x).}{1440} - \frac{(31.*\exp(x)).}{90} - \frac{(x.*\exp(2.*x)).}{8} - \frac{(x.*\exp(x)).}{3} + x.^2./8 + 19/576;$$
- `error=abs(M3-exact);`
- `plot(x,M3,x,exact)`
- `%%print`
- `disp(sprintf(' TABLE(3) :) '))`
- `[x' exact' M3' error']`

#####

## Matlab Code for Modified Laplace Adomian Decomposition method (MLAD) for Example 3.4

- `clc`
- `clear all`
- `format long`
- `%The general form of nonlinear vollterra integro differential  
equation`
- `%Display`
- `disp(sprintf('Farah Abu-Thabet'))`
- `disp(sprintf('Laplace Method'))`
- `%In put data`
- `syms t x s`
- `%%for i=1`
- `A=3;`
- `%%for i=2`
- `F=(A)^2;`
- `A1=9/s;`
- `B=(A1/(s*s-s));`

```

➤ C=ilaplace(B);

➤ D=(subs(C,t,x));

➤ %%for i=3

➤ K=((A)*(D)*2);

➤ A2=laplace(K);

➤ B1=(A2/(s*(s-1)));

➤ C1=ilaplace(B1);

➤ D1=-exp(2*x)-exp(x)*((2*x)-1)+subs(C1,t,x);

➤ %%for i=4

➤ K1=((A)*(D1)*2)+(D^2);

➤ A3=laplace(K1);

➤ B2=(A3/(s*(s-1)));

➤ C2=ilaplace(B2);

➤ D2=subs(C2,t,x);

➤ %%for i=5

➤ K2=((D)*(D1)*2)+((A)*2);

➤ A4=laplace(K2);

```

➤  $B3=(A4/(s*(s-1)))$ ;

➤  $C3=ilaplace(B3)$ ;

➤  $D3=subs(C3,t,x)$ ;

➤ %% approximation(1)

a)  $x=[0 \ 0.0938 \ 0.2188 \ 0.3125 \ 0.4062 \ 0.5 \ 0.6250 \ 0.7188 \ 0.8125 \ 0.9062$   
1];

(a)  $M1=x.^4/24 + x.^3/6 + x.^2/2 + x + 2$ ;

(b)  $exact=1+\exp(x)$ ;

(c)  $error=abs(M1-exact)$ ;

(d)  $plot(x,M1,'*',x,exact,'r')$

(e) %%print

(f) $disp(sprintf(' \text{TABLE}(1) :') )$

(g) [x' exact' M1' error']

b) %% approximation(2)

i)  $x=[0 \ 0.0938 \ 0.2188 \ 0.3125 \ 0.4062 \ 0.5 \ 0.6250 \ 0.7188 \ 0.8125$   
0.9062 1];

➤  $M2= x.^7/5040 + x.^6/720 + x.^5/120 + x.^4/24 + x.^3/6 + x.^2/2$   
 $+ x + 2$ ;

➤ `exact=1+exp(x);`

(a) `error=abs(M2-exact);`

(b) `plot(x,M2,x,exact,'m')`

(c) `%%print`

(d) `disp(sprintf(' TABLE(2) : ) '))`

(i) `[x' exact' M2' error']`

1. `%% approximation(3)`

ii) `x=[0 0.0938 0.2188 0.3125 0.4062 0.5 0.6250 0.7188 0.8125  
0.9062 1];`

➤ `M3= x.^9/362880 + x.^8/40320 + x.^7/5040 + x.^6/720 +  
x.^5/120 + x.^4/24 + x.^3/6 + x.^2/2 + x + 2;`

➤ `exact=1+exp(x);`

(a) `error=abs(M3-exact);`

(b) `plot(x,M3,x,exact)`

(c) `%%print`

(d) `disp(sprintf(' TABLE(3) : ) '))`

(i) `[x' exact' M3' error']`

(ii) `%% approximation(4)`

ii) `x=[0 0.0938 0.2188 0.3125 0.4062 0.5 0.6250 0.7188 0.8125  
0.9062 1];`

➤ `M4=x.^11/39916800 + x.^10/3628800 + x.^9/362880 +  
x.^8/40320 + x.^7/5040 + x.^6/720 + x.^5/120 + x.^4/24 + x.^3/6  
+ x.^2/2 + x + 2;`

➤ `exact=1+exp(x);`

(a) `error=abs(M4-exact);`

(b) `plot(x,M4,x,exact)`

(c) `%%print`

(d) `disp(sprintf(' TABLE(4) : ) '))`

(i) `[x' exact' M4' error']`

(ii) `%% approximation(5)`

ii) `x=[0 0.0938 0.2188 0.3125 0.4062 0.5 0.6250 0.7188 0.8125  
0.9062 1];`

➤ `M5=x.^14/87178291200 + x.^13/6227020800 + x.^12/479001600  
+ x.^11/39916800 + x.^10/3628800 + x.^9/362880 + x.^8/40320 +  
x.^7/5040 + x.^6/720 + x.^5/120 + x.^4/24 + x.^3/6 + x.^2/2 + x +  
2;`

➤ `exact=1+exp(x);`



(a) `error1=abs(M5-exact);`

(b) `plot(x,M5,x,exact)`

(c) `%%print`

(d) `disp(sprintf(' TABLE(5) : ) '))`

(i) `[x' exact' M5' error']`

ii) `%%Plot All Iteration with Exact Solution`

➤ `plot(x,M1,'*',x,M2,'B',x,M3,'R',x,M4,'Y',x,M5,'G',x,exact,'P')`

#####

### **Matlab Code for Variational Iteration Method (VIM) for Example**

#### **3.1**

➤ `clc`

➤ `clear all`

➤ `format long`

➤ `%The general form of nonlinear vollterra integro differential  
equation`

➤ `%Display`

➤ `disp(sprintf('Farah Abu-Thabet'))`

➤ `disp(sprintf('Variation Iterative Method'))`

- %In put data
- N=4;
- syms x
- syms r
- syms t
- y(1)=1;
- %%for i=1
- A=int(8\*r\*1\*log(1),r,0,t);
- B=int(((t-x)\*(0-6\*1+4)-A),t,0,x);
- U1=y(1)+(B);
- B=expand(U1);
- %%for i=2
- A2=int(((subs(U1,x,r))\*8\*r\*log((subs(U1,x,r))),r,0,t);
- B2=int((t-x)\*(diff(diff(subs(U1,x,t)))-6\*(subs(U1,x,t))+4)-A2,t,0,x);
- U2=U1-B2;
- C=expand(U2);
- %% approximation(1)

- `x=[0 0.0938 0.2188 0.3125 0.4062 0.5 0.6250 0.7188 0.8125 0.9062`  
`1];`
- `M1=x.^2 + 1;`
- `exact=exp(x.^2);`
- `error=abs(M1-exact);`
- `plot(x,M1,x,exact)`
- `%%print`
- `disp(sprintf(' TABLE(1) :) '))`
- `[x' exact' M1' error']`
- `%%%%%%%%%`  
`%%%%%%%%%`
- `%% approximation(2)`
- `x=[0 0.0938 0.2188 0.3125 0.4062 0.5 0.6250 0.7188 0.8125 0.9062`  
`1];`
- `M2=(16*pi)/15 - (32.*x)/15 - (log(x - i)*16*i)/15 + (log(x +`  
`i)*16*i)/15 + 2.*x.*log(x - i) + 2.*x.*log(x + i) + x.^2 -`  
`(58.*x.^3)/45 - x.^4/2 - (9.*x.^5)/25 + (4.*x.^3.*log(x.^2 + 1))/3 +`  
`(2.*x.^5.*log(x.^2 + 1))/5 + 1;`
- `exact=exp(x.^2);`

- `error=abs(M2-exact);`
- `plot(x,M2,x,exact)`
- `%%print`
- `disp(sprintf(' TABLE(2) :) '))`
- `[x' exact' M2' error']`
- `%%all plot`
- `plot(x,M1,x,exact,'m')`

#####

### **Matlab Code for Variational Iteration Method (VIM) for Example 3.2**

- `clc`
- `clear all`
- `format long`
- `%The general form of nonlinear vollterra integro differential equation`
- `%Display`
- `disp(sprintf('Farah Abu-Thabet'))`
- `disp(sprintf('Variation Iterative Method'))`

➤ %In put data

➤ N=4;

➤ syms x

➤ syms r

➤ syms t

➤ y(1)=0;

➤ %%for i=1

a)  $A = \int(r^3, r, 0, t);$

b)  $B = \int((-1)*((0-(t^3+3*t^2+6*t+6)*\exp(-1*t)))+A), t, 0, x);$

c)  $U1 = y(1) + (B);$

d)  $B = \text{expand}(U1);$

➤ %%for i=2

a)  $A2 = \int((r^3*\exp(-1*(\text{subs}(U1, x, r))))), r, 0, t);$

b)  $B2 = \int((-1)*((6*(t^2+1)*(diff(\text{subs}(U1, x, t)) - (t^3+3*t^2+6*t+6)*\exp(-1*t)))+A2), t, 0, x);$

i)  $U2 = U1 - B2;$

c)  $C = \text{expand}(U2);$

```

d) %%for i=3

e) A3=int((r^3*exp(-1*(subs(U2,x,r)))),r,0,t);

f) B3=int((-1)*((6*(t^2+1)*(diff(subs(U2,x,t))-
    (t^3+3*t^2+6*t+6)*exp(-1*t)))+A3),t,0,x);

i) U3=U2-B3;

g) L=expand(U3);

i) %% approximation(1)

➤ x=[0 0.0938 0.2188 0.3125 0.4062 0.5 0.6250 0.7188 0.8125 0.9062
    1];

➤ M1=-1*x.^3/3+x;

i) exact= atan(x);

ii) error=abs(M1-exact);

iii) plot(x,M1,x,exact)

iv) %%print

v) disp(sprintf(' TABLE(1) :) '))

vi) [x' exact' M1' error']

%% approximation(2)

```

```
➤ x=[0 0.0938 0.2188 0.3125 0.4062 0.5 0.6250 0.7188 0.8125 0.9062
1];
```

```
➤ M2= x.^9/9 - x.^7/7 + x.^5/5 - x.^3/3 + x;
```

```
➤ exact= atan(x);
```

```
➤ error=abs(M2-exact);
```

```
i) plot(x,M2,x,exact)
```

```
ii) %%print
```

```
iii) disp(sprintf(' TABLE(2) :) '))
```

```
iv) [x' exact' M2' error']
```

```
%% approximation(3)
```

```
➤ x=[0 0.0938 0.2188 0.3125 0.4062 0.5 0.6250 0.7188 0.8125 0.9062
1];
```

```
i) M3=x.^13/13 - x.^11/11 + x.^9/9 - x.^7/7 + x.^5/5 - x.^3/3 + x;
```

```
➤ exact= atan(x);
```

```
i) error1=abs(M3-exact);
```

```
ii) plot(x,M3,x,exact)
```

```
iii) %%print
```

```
iv) disp(sprintf(' TABLE(3) :) '))
```

v) [x' exact' M3' error']

➤ %%all plot

➤ plot(x,M1,'R',x,M2,'Y',x,exact,'B',x,exact,'M')

#####

## **Matlab Code for Variational Iteration Method (VIM) for Example**

### **3.3**

➤ clc

➤ clear all

➤ format long

➤ %The general form of nonlinear vollterra integro differential  
equation

➤ %Display

➤ disp(sprintf('Farah Abu-Thabet'))

➤ disp(sprintf('Variation Iterative Method'))

➤ %In put data

➤ N=4;

➤ syms x

➤ syms r



➤ syms t

➤ y(1)=1;

➤ %%for i=1

i)  $A = \text{int}(0, r, 0, t);$

ii)  $B = \text{int}((0 - 1 + \exp(t) - \exp(2*t) - A), t, 0, x);$

iii)  $U1 = y(1) - (B);$

iv)  $B = \text{expand}(U1);$

➤ %%for i=2

➤  $A2 = \text{int}((\exp(t-r) * (1 - (\text{subs}(U1, x, r))^2)), r, 0, t);$

(1)  $B2 = \text{int}(\text{diff}(\text{subs}(U1, x, t) - 1 + \exp(t) - \exp(2*t) - A2), t, 0, x);$

(2)  $U2 = U1 - B2;$

(3)  $C = \text{expand}(U2);$

(4) %%for i=3

(5)  $A3 = \text{int}((\exp(t-r) * (1 - (\text{subs}(U2, x, r))^2)), r, 0, t);$

(6)  $B3 = \text{int}(\text{diff}(\text{subs}(U2, x, t) - 1 + \exp(t) - \exp(2*t) - A3), t, 0, x);$

(7)  $U3 = U2 - B3;$

(8)  $L = \text{expand}(U3);$

➤ %% approximation(1)

a)  $x = [-1 \ -0.8 \ -0.6 \ -0.4 \ -0.2 \ 0 \ 0.2 \ 0.4 \ 0.6 \ 0.8]$ ;

(a)  $M1 = x.^4/24 + x.^3/6 + x.^2/2 + x + 0.9999$ ;

(b)  $exact = \exp(x)$ ;

(c)  $error = \text{abs}(M1 - exact)$ ;

(d)  $\text{plot}(x, M1, x, exact)$

(e) %%print

(f)  $\text{disp}(\text{sprintf}(' \text{TABLE}(1) : )')$

(g)  $[x' \ exact' \ M1' \ error']$

(h) %% approximation(2)

b)  $x = [-1 \ -0.8 \ -0.6 \ -0.4 \ -0.2 \ 0 \ 0.2 \ 0.4 \ 0.6 \ 0.8]$ ;

(a)  $M2 = x.^9/362880 + x.^8/40320 + x.^7/5040 + x.^6/720 + x.^5/120 + x.^4/24 + x.^3/6 + x.^2/2 + x + 1$ ;

(b)  $exact = \exp(x)$ ;

(c)  $error = \text{abs}(M2 - exact)$ ;

(d)  $\text{plot}(x, M2, x, exact)$

(e) %%print

(f) disp(sprintf(' TABLE(2) :) '))

(g) [x' exact' M2' error']

ii) %% approximation(3)

c) x=[-1 -0.8 -0.6 -0.4 -0.2 0 0.2 0.4 0.6 0.8 ];

(a)  $M3 = x.^{12}/479001600 + x.^{11}/39916800 + x.^{10}/3628800 +$   
 $x.^9/362880 + x.^8/40320 + x.^7/5040 + x.^6/720 + x.^5/120 +$   
 $x.^4/24 + x.^3/6 + x.^2/2 + x + 1;$

(b) exact=exp(x);

(c) error=abs(M3-exact);

(d) plot(x,M3,x,exact)

(e) %%print

(f) disp(sprintf(' TABLE(3) :) '))

(g) [x' exact' M3' error']

➤ %%all plot

➤ plot(x,M1,'R',x,M2,'Y',x,exact,'B',x,exact,'M')

#####

## Matlab Code for Variational Iteration Method (VIM) for Example

### 3.4

- `clc`
- `clear all`
- `format long`
- `%The general form of nonlinear vollterra integro differential equation`
- `%Display`
- `disp(sprintf('Farah Abu-Thabet'))`
- `disp(sprintf('Variation Iterative Method'))`
- `%In put data`
- `N=4;`
- `syms x`
- `syms r`
- `syms t`
- `y(1)=2;`
- `%%for i=1`
- `A=int(exp(t-r)*((y(1))^2),r,0,t);`

```

➤ B=int((0-1-exp(t)+2*t*exp(t)+exp(2*t)-A),t,0,x);

➤ U1=y(1)-(B);

➤ A=expand(U1);

➤ %% for 2

➤ A2=int(((subs(U1,x,r))^2)*(exp(t-r)),r,0,t);

➤ B2=int((subs(diff(U1),x,t))-1-exp(t)+2*t*exp(t)+exp(2*t)-A2,t,0,x);

➤ U2=U1-B2;

➤ B=expand(U2);

➤ %% for 3

➤ A3=int(((subs(U2,x,r))^2)*(exp(t-r)),r,0,t);

➤ B3=int((subs(diff(U2),x,t))-1-exp(t)+2*t*exp(t)+exp(2*t)-A3,t,0,x);

➤ U3=U2-B3;

➤ C= expand(U3);

➤ %% approximation(1)

➤ x=[0 0.0938 0.2188 0.3125 0.4062 0.5 0.6250 0.7188 0.8125 0.9062
1];

➤ M1=7*exp(x) - exp(2.*x)/2 - 3.*x - 2.*x.*exp(x) - 9/2;

```

```

➤ exact=1+exp(x);

➤ error1=abs(M1-exact);

➤ plot(x,M1,x,exact)

➤ %%print

➤ disp(sprintf(' TABLE(1) :) '))

➤ [x' exact' M1' error']

➤ %% approximation(2)

➤ x=[0 0.0938 0.2188 0.3125 0.4062 0.5 0.6250 0.7188 0.8125 0.9062 1];

➤ M2=52*exp(2.*x) - (257.*x)/4 - (13*exp(3.*x))/9 + exp(4.*x)/48 +
    (2*exp(x))/3 - (37.*x.*exp(2.*x))/2 + (x.*exp(3.*x))/3 -
    24.*x.^2.*exp(1.*x) + 4.*x.^3.*exp(x) + 2.*x.^2.*exp(2.*x) -
    17.*x.*exp(x) - (45.*x.^2)/2 - 3.*x.^3 - 7091/144;

➤ exact=1+exp(x);

➤ error=abs(M2-exact);

➤ plot(x,M2,x,exact)

➤ %%print

➤ disp(sprintf(' TABLE(2) :) '))

➤ [x' exact' M2' error']

```

➤ %% approximation(3)

➤ x=[0 0.0938 0.2188 0.3125 0.4062 0.5 0.6250 0.7188 0.8125 0.9062 1];

➤ M3=(23854975\*exp(2.\*x))/576 - (1755133633.\*x)/20736 -  
 (4810751\*exp(3.\*x))/7776 + (86570981\*exp(4.\*x))/248832 -  
 (4698473\*exp(5.\*x))/480000 + (319343\*exp(6.\*x))/1944000 -  
 (65\*exp(7.\*x))/42336 + exp(8.\*x)/129024 +  
 (2933236952771\*exp(1.\*x))/54432000 - (12187981.\*x.\*exp(2.\*x))/288  
 + (533779.\*x.\*exp(3.\*x))/432 - (4766885.\*x.\*exp(4.\*x))/20736 +  
 (381511.\*x.\*exp(5.\*x))/72000 - (1351.\*x.\*exp(6.\*x))/21600 +  
 (x.\*exp(7.\*x))/3024 + (733435.\*x.^2.\*exp(x))/144 - 1433.\*x.^3.\*exp(x)  
 + (52903.\*x.^4.\*exp(x))/72 + (128.\*x.^5.\*exp(x))/5 + 18.\*x.^6.\*exp(x)  
 - (24.\*x.^7.\*exp(x))/7 + (1404235.\*x.^2.\*exp(2.\*x))/72 -  
 (60869.\*x.^2.\*exp(3.\*x))/54 - (74527.\*x.^3.\*exp(2.\*x))/12 +  
 (232073.\*x.^2.\*exp(4.\*x))/3456 + (19967.\*x.^3.\*exp(3.\*x))/54 +  
 (2831.\*x.^4.\*exp(2.\*x))/2 -  
 ➤ (7609.\*x.^2.\*exp(5.\*x))/7200 - (8441.\*x.^3.\*exp(4.\*x))/864 -  
 (469.\*x.^4.\*exp(3.\*x))/9 - 174.\*x.^5.\*exp(2.\*x) +  
 (7.\*x.^2.\*exp(6.\*x))/1080 + (3.\*x.^3.\*exp(5.\*x))/40 +  
 (5.\*x.^4.\*exp(4.\*x))/9 + (8.\*x.^5.\*exp(3.\*x))/3 + 8.\*x.^6.\*exp(2.\*x) -  
 (2214919.\*x.\*exp(x))/216 - (23678779.\*x.^2)/576 - (101187.\*x.^3)/8 -  
 (252809.\*x.^4)/96 - (7347.\*x.^5)/20 - (63.\*x.^6)/2 - (9.\*x.^7)/7 -  
 482735066144893/5080320000;

- `exact=1+exp(x);`
- `error=abs(M3-exact);`
- `plot(x,M3,x,exact)`
- `%%print`
- `disp(sprintf(' TABLE(3) :) '))`
- `[x' exact' M3' error']`
- `%%all plot`
- `plot(x,M1,'R',x,M2,'Y',x,M3,'G',x,exact,'B')`



جامعة النجاح الوطنية

كلية الدراسات العليا

## الطرق العددية لحل معادلة فولتيرا التكاملية التفاضلية الغير خطية

إعداد

فرح خالد شحادة أبو ثابت

إشراف

أ. د. ناجي قطناني

قدمت هذه الأطروحة استكمالاً لمتطلبات الحصول على درجة الماجستير في الرياضيات  
المحوسبة، بكلية الدراسات العليا، في جامعة النجاح الوطنية، نابلس - فلسطين.

2019

ب

## الطرق العددية لحل معادلة فولتيرا التكاملية التفاضلية الغير خطية

إعداد

فرح خالد شحادة أبو ثابت

إشراف

أ.د. ناجي قطناني

### الملخص

في هذه الأطروحة ركزنا على حل معادلة فولتيرا التكاملية التفاضلية الغير خطية لأنها تحتوي على مجموعة واسعة من التطبيقات في الفيزياء الرياضية، والهندسة، والميكانيكا، والكيمياء، وعلم الفلك، وعلم الأحياء، والاقتصاد، ونظرية الإمكانات.

بعد ان قدمنا بعض التعاريف والأساسيات التي نحتاجها، ركزنا اهتمامنا بشكل أساسي على الطرق العددية لحل معادلة فولتيرا التكاملية التفاضلية الغير خطية. هذه الطرق هي: طريقة التحويل التفاضلي مع كثيرات الحدود الأدمية (DTM) طريقة تحليل لابلاس أدوميان، (LADM) وطريقة التكرار المتغير (VIM).

حيث سيتم عرض الإطار الرياضي لهذه الطرق العددية مع خصائص التقارب الخاصة بها. حيث سيتم توضيح كفاءة هذه الطرق العددية من خلال بعض الأمثلة العددية.

تظهر النتائج العددية بوضوح أن طريقة التكرار المتغير هي واحدة من أقوى التقنيات العددية لحل معادلة فولتيرا التكاملية التفاضلية الغير خطية بالمقارنة مع التقنيات العددية الأخرى بناءً على الأمثلة المستخدمة.

