NUMERICAL SOLUTIONS OF NON-LINEAR VOLTERRA INTEGRAL EQUATIONS OF THE SECOND KIND

MEREWAN ABDEL SALEH

AUGUST 2015

NUMERICAL SOLUTIONS OF NON-LINEAR VOLTERRA INTEGRAL EQUATIONS OF THE SECOND KIND

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES OF
ÇANKAYA UNIVERSITY

BY

MEREWAN ABDEL SALEH

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF
MATHEMATICS AND COMPUTER SCIENCE

AUGUST 2015

Title of the Thesis: **Numerical Solutions of Non-Linear Volterra Integral Equations of the second Kind.**

Submitted by **Merewan Abdel SALEH**

Approval of the Graduate School of Natural and Applied Sciences, Çankaya University.

Prof. Dr. Taner ALTUNOK
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Billur KAYMAKÇALAN
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Assist. Prof. Dr. Borhan F. Juma
Co-Supervisor

Prof. Dr. Kenan TAŞ
Supervisor

Examination Date: 22.05.2014

Examining Committee Members

Prof. Dr. Billur KAYMAKÇALAN	(Çankaya Univ.)	
Prof. Dr. Kenan TAŞ	(Çankaya Univ.)	
Assoc. Prof. Dr. Fahd JARAD	(THK Univ.)	

STATEMENT OF NON-PLAGIARISM PAGE

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name : Merewan, SALEH
Signature :
Date : 10.08.2015

iii

ABSTRACT

NUMERICAL SOLUTIONS OF NON-LINEAR VOLTERRA INTEGRAL EQUATIONS OF THE SECOND KIND

Merewan Abdel SALEH
M.Sc., Department of Mathematics and Computer Science
Supervisor: Prof. Dr. Kenan TAŞ
Co- Supervisor: Assist. Prof. Dr. Borhan F. Juma
August 2015, 82 pages

In this thesis it will be shown how one can solve one of the types of integral equation, non-linear Volterra Integral Equations (VIEs) of the second kind, by certain methods: numerical integration (Trapezoidal rule and Simpson's rule), Runge-Kutta methods (classic third-order, optimal third-order, fourth-order, and classic fourth-order), classic spline functions (quadratic and cubic classic spline functions), and B-spline functions (first-order, second-order, third-order, and fourth-order). It will also be shown how one can convert one of the methods to another one.

These methods exist and are easy to use to solve differential equations and integration problems, but it is difficult to apply them to integral equations, especially non-linear integral equations, because the known function occurs into kernel of the integral. Therefore it is needed to modify these methods and techniques to apply them to non-linear integral equations.

Keywords: Non-Linear Equations, Second Kind Volterra Integral Equations, Runge-Kutta methods, B-spline functions.

ÖZ

LINEER-OLMAYAN İKİNCİ ÇEŞİT VOLTERRA INTEGRAL DENKLEMLERİNİN NÜMERİK ÇÖZÜMLERİ

Merewan Abdel SALEH
Yüksek Lisans, Matematik ve Bilgisayar Bilimleri Bölümü
Tez Yöneticisi: Prof. Dr. Kenan TAŞ
Eş Danışman: Yrd. Doç. Dr. Borhan F. Juma
Ağustos 2015, 82 sayfa

Bu tezde lineer-olmayan ikinci çeşit Volterra integral denklemlerinin nümerik integrasyon (Ikizkenar kuralı and Simpson kuralı), Runge-Kutta metodu (klasik üçüncü-derece, optimal üçüncü-derece, dördüncü-derece ve klasik dördüncü-derece), klasik şerit fonksiyonları (kuadratik ve kübik klasik şerit fonksiyonları) ve B-şerit fonksiyonları (birinci-derece, ikinci-derece, üçüncü-derece ve dördüncü-derece) yardımıyla nasıl çözülebileceği gösterilmiştir. Ayrıca methotlardan birinin diğerine nasıl çevrileceği de incelenmektedir.

Bu methotlar varlığı ve diferensiyel denklemler ile integral problemlerinin çözülmesinde kolaylıkla kullanılmakta olduğu bilinmektedir, ancak özellikle lineer-olmayan integral denklemlere uygulamasında çeşitli zorluklar ortaya çıkmaktadır, çünkü lineer olmayan terimler integralin çekirdeğinde mevcuttur. Bu nedenle söz konusu methotların ve teknikler üzerinde, lineer-olmayan integral denklemlere de uygulanabilecek biçimde, bazı düzenlemeler yapılma ihtiyacı olduğu görülmüştür.

Anahtar Kelimeler: Lineer-Olmayan Denklemler, İkinci Çeşit Volterra Integral Denklemleri, Runge-Kutta methotları, B-şerit fonksiyonları.

ACKNOWLEDGEMENTS

I would like to express my appreciation to my great supervisor, Prof. Dr. Kenan TAŞ, and Co-supervision Assist. Prof. Dr. Borhan F. Juma, who gave me unlimited support and valuable guidance; there are not enough words to express my thanks for you.

In addition, I want to express my deep gratitude to my dear father, mother, wife, and children (Vesteen and Laveen). Finally, thanks go to my sisters, and all of my friends for their endless and continuous encouragement and support throughout the years.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURES

LIST OF TABLES

TABLES

LIST OF ABBREVIATIONS

B1-Sp	First-Order B-Spline
B2-Sp	Second-Order B-Spline
B3-Sp	Third-Order B-Spline
B4-Sp	Fourth-Order B-Spline
Cu-Sp1	Cubic Classic Spline Using Trapezoidal Rule
Cu-Sp2	Cubic Classic Spline Using Classic Runge-Kutta Fourth Order Method
L.S.E.	Least Square Error
Qu-Sp1	Quadratic Classic Spline Using Trapezoidal Rule
Qu-Sp2	Quadratic Classic Spline Using Classic Runge-Kutta Fourth Order Method
RK3	Third-Order Runge-Kutta Method
RK4	Fourth-Order Runge-Kutta Method
SMP	Simpson's Rule
TRP	Trapezoidal Rule
VIDEs	Volterra Integro-Differential Equations
VIE	Volterra Integral Equation
VIEs	Volterra Integral Equations
VIEB1SP	Volterra Integral Equation Using First-Order B-Spline Function
VIEB2SP	Volterra Integral Equation Using Second-Order B-Spline Function
VIEB3SP	Volterra Integral Equation Using Third-Order B-Spline Function
VIEB4SP	Volterra Integral Equation Using Fourth-Order B-Spline Function

VIECRK3	Volterra Integral Equation Using Classic Third-Order Runge-Kutta Method
VIECRK4	Volterra Integral Equation Using Classic Fourth-Order Runge-Kutta Method
VIECSP1	Volterra Integral Equation Using Cubic Classic Spline Function (Using Trapezoidal Rule)
VIECSP2	Volterra Integral Equation Using Cubic Classic Spline Function (Using Classic Fourth Order Runge-Kutta Method)
VIELSP	Volterra Integral Equation Using Linear Classic Spline Function
VIEORK3	Volterra Integral Equation Using Optimal Third-Order Runge-Kutta Method
VIEQSP1	Volterra Integral Equation Using Quadratic Classic Spline Function (Using Trapezoidal Rule)
VIEQSP2	Volterra Integral Equation Using Quadratic Classic Spline Function (Using Classic Fourth Order Runge-Kutta Method)
VIERK4	Volterra Integral Equation Using Fourth-Order Runge-Kutta Method
VIESMP	Volterra Integral Equation Using Simpson's Rule
VIETRP	Volterra Integral Equation Using Trapezoidal Rule

CHAPTER 1

INTRODUCTION

1.1 Background

The mechanics problem of calculating the time a particle takes to slide under gravity down a given smooth curve from any point on the curve to its lower end leads to an exercise in integration. The time, $f(\mathrm{X})$ say, for the particle to descend from the height X is given by an expression of the form:

$$f(\mathrm{X}) = \int_0^x \frac{\mathrm{g}(x)}{(\mathrm{X}-x)^{1/2}}\,dx \qquad (0 \le \mathrm{X} \le b) \qquad (1.1)$$

where $\mathrm{g}(x)$ embodies the slope of the given curve. The converse problem is that in which the time of descent from height X is given. The particular curve that produces this time is less straightforward to find. It entails the determination of the function g from (1.1), $f(\mathrm{X})$ now being assigned for $0 \le \mathrm{X} \le b$. From this point of view, (1.1) is called an integral equation, this description expressing the fact that the function to be determined appears under an integral sign.

The equation (1.1) is of historical importance, and is attributed to Abel.

Let us start with Abel in the 1820s. His work on analysis is of continuing interest in integral equations. The names of many modem mathematicians, for example Cauchy, Fredholm, Hubert, Volterra, etc. are linked with this subject [1]. In 1913, Volterra's book "Leçons sur les équations intégral et intégro-différentielles" appeared. Since that date, an enormous amount of literature on the theory and on its applications, which include elasticity, semi-conductors, scattering theory, metallurgy, seismology, heat condition, fluid flow, population dynamics, chemical reactions, etc., has appeared [2].

There are two main reasons for this degree of attention. The first, and perhaps the most common, reason is that integral operators, conversions, and equations are suitable tools for studying differential equations. As a result, integral equation

techniques are better known to classical analysts, and important and elegant results were developed by them. The second, which is seen in the work of Abel on tautochrone curves, is that integral equations are used for natural mathematical models that include physically attractive cases [1].

Many readers will already have encountered integral equations, but perhaps only in a context where this terminology is not used. For example, the pair of equations:

$$\left.\begin{aligned} \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{ixt}\ \mathrm{g}(t)dt = f(x) \qquad -\infty < x < \infty, \\ \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-ixt}\ \mathrm{g}(t)dt = f(x) \qquad -\infty < x < \infty, \end{aligned}\right\} \tag{1.2}$$

which define the Fourier transform and its inverse, may be viewed as an integral equation and its solution. If $f(x)$ is regarded as known for $-\infty < x < \infty$ in the first equation, then the second equation provides the solution for $\mathrm{g}(x)$, also for $-\infty < x < \infty$.

The name "integral equation" first appeared in 1888 in a paper on elliptic partial differential equations by the German scientist Paul du Bois-Riemann. The name "Volterra integral equation" was first coined by the Romanian mathematician Traian Lalesco in 1908, seemingly following a suggestion from his teacher, the French mathematician Emile Picard. The terminology "integral equation of the (first, second, third) kind" was first used by the German mathematician David Hilbert in connection with his study of Fredholm integral equations [3].

Integral equations are encountered in different fields of science, and they have applications in, for example, flexibility, plasticity, approximation theory, heat and mass transfer, fluid dynamics, biomechanics, filtration theory, electrostatics, electrodynamics, game theory, control theory, queuing theory, electrical engineering, medicine, and economics. Exact (closed-form) solutions of integral equations play an important role in the proper understanding of the qualitative features of various phenomena and processes in different areas of natural science [4].

Newton-Cotes formulae (quadrature methods) are the most common tools used by engineers and many scientists to get approximate solutions for definite integrals when there is no way of solving these analytically. These procedures are based on the strategy of substituting a complicated function or scheduled data with an approximating function that it is not difficult to integrate [5-6].

The Runge-Kutta methods, because of their self-starting property, have a unique place amongst the classical types of method. They use only the information from the last step computed, and are therefore called single-step methods.

Spline functions were introduced in the 1940s in the context of approximation theory. Schoenberg introduced the name spline function in 1946. These functions have been used in geometric modeling since the 1970s, and in about 1974 the concept of spline basis functions was introduced. This enabled piecewise descriptions to be made with automatic retention of the continuity of derivatives. The piecewise descriptions that are enabled are made with automatic retention of the continuity of derivatives. The price paid for this is that there are surfaces defined by control points through which the surfaces do not pass.

There are many types of spline, such as classic splines, B-splines, Bezier splines, cardinal splines, Catmull-Rom or Overhauser splines, and uniform rational B-splines (NURBS). They are all based on different mathematical concepts, but they have one thing in common: the control points, or anchor points, are modifiable [7].

Splines provide an important category of functions for approximation in areas such as finite element methods with suitable numerical data, and the numerical solution of ordinary and partial differential equations.

A spline function is a function that consists of polynomial pieces joined with certain smoothness condition; it is used for data interpolation and can be used for finding the derivatives and integrals of functions [8].

Splines can be of any degree. Linear splines are simple straight line segments connecting two points. The linear splines co-yield (first-order) approximating polynomials. The slopes (first derivatives) and the curvature (second derivatives) are discontinuous at every data point. Quadratic splines yield (second-order) approximating polynomials. Quadratic splines mean that the slopes can be forced to be continuous at each data point, but the curvatures are still discontinuous. Cubic splines yield a (third degree) polynomial joining each pair of data points. The slopes and curvatures of cubic splines can be forced to be continuous at each data point [9].

Spline functions belong to the category of piecewise polynomial functions, satisfying continuity characteristics only slightly less severe than those of polynomials, and thus they are a normal generalization of polynomials. They are found to have very desirable properties such as “approximating, interpolating, and curve-fitting

functions"[10]. Because of their smoothing and interpolating properties, spline functions are gaining popularity in many fields such as engineering sciences, image processing and restoration [10], and data interpolation [11-12].

Saeed presented some computational methods for solving a system of linear Volterra integral and integro-differential equations [13]. Mahmoudi used the wavelet Galerkin method to solve non-linear integral equations [14]. Wazwaz used a modified decomposition method for the analytic treatment of non-linear integral equations and systems of non-linear integral equations [15]. Mustafa used spline functions to solve a system of Volterra integral equations [16], and Al-Nasir in 1999 used quadrature methods to solve Volterra integral equations of the second kind [17]. Al-Rawi also applied quadrature methods to solve the first kind of integral equation of the convolution type [18]. Saadati and colleagues used the trapezoidal rule to solve linear integro-differential equations [19]. Moreover, Al-Timeme in 2003 used quadrature rules to find the numerical solutions of the initial value problems to Volterra integro-differential equations of the second kind [20]. Al-Dahan solved a VIE using quadrature [21], and Al-Jawary also applied quadrature to solve a system of VIEs [22].

A large number of researchers and scientists have published books that are entirely devoted to non-linear integral equation methods and their applications [23-24].

Sastry employed spline functions and trapezoidal and Chebyshev series methods in 1973 to find the numerical solution of a linear Fredholm integral equation of the second kind [25]. In 2007 Saberi-Nadjaf and Heidari applied a modified trapezoidal quadrature method to find the numerical solution of linear Fredholm integral equations of the second kind [26]. Nik, Eshkuvatov, Yaghobifar and Hasan used the degenerate kernel method in 2008 to find the exact solution of singular Fredholm integral equations of the second kind. Moreover, they applied the Galerkin method with a Laguerre polynomial to get an approximate solution of a singular Fredholm integral equation of the second kind [27]. Rahbar and Hashemizadeh in 2008 applied a quadrature method to find the numerical solution of a linear Fredholm integral equation of the second kind [28]. Hacia and Kaczmarek have presented the conditions for the bounds of the solution to a system of linear Volterra integral equations [29]. Chen, Li and Ou studied the classification of the solutions to a system of integral equations [30]. Baker and Miller presented a spline collocation method for

the numerical solution of the system of integral equations on a polygon in R2 [31]. Biazar, Babolian, and Islam considered linear and non-linear systems of Volterra integral equations of the first kind [32], and used the Adomian decomposition method to solve them. Babolian, Biazar and Vahidi studied the application of the Adomian decomposition method to a system of linear Volterra integral equations of the second kind [33].

In 2004, Maleknejad and Shahrezaee used the Runge-Kutta method for the numerical solution of a system of Volterra integral equations [34]. Runge-Kutta methods are used to find numerical solutions of initial problems [35-36], and a second-order Runge-Kutta method can be used to treat linear Volterra integral equations of the second kind [37-38].

Many people have used spline functions to approximate the numerical solution to integral equations. For example, Al-Kahachi used linear, quadratic and cubic classic spline functions to treat linear Volterra integral equations of the first kind [39]. Al-salhi used classic spline functions (linear, quadratic and cubic) to find the numerical solution of non-linear Volterra integra1 equations of the second kind [40]. Al-Asadi used three types of classic spline functions for solving non-linear Volterra integral equations of the first kind [41]. Abdul Hameed used three different types of classic spline and B-spline functions to treat different orders of linear Fredholm integral equations [42]. Juma used a different type of classic spline function and B-spline function and a new type of spline function, called Catmull-Rom or Overhauser splines, to find the approximate numerical solution to a system of non-linear Volterra integral equations of the second kind [43]. Salam and Huda introduced numerical methods for solving linear Fredholm-Volterra integral equations of the second kind [44]. Malindzisa and Khumalo considered numerical solutions of a class of non-linear (non-standard) Volterra integral equations [45]. Netravali used a cubic spline approximation in C2 to construct the solution of a general Volterra integral equation of the second kind [46]. Al-Faour, Kadhim and Jaber presented algorithms, with the aid of the MATLAB language, for solving, numerically, a Hammerstein Volterra integral equation of the second kind of convolution type using linear, quadratic and cubic spline functions [47].

In this thesis, we shall investigate an approximation solution to non-linear VIEs of the second kind, because integral equations, especially non-linear integral equations,

are usually difficult to solve analytically. This follows work that has recently been done by different authors on non-linear integral equations.

1.2 Objectives

In general, this work focuses on the numerical or approximate solutions of non-linear VIEs of the second kind.

Therefore, we study and modify some of these numerical and approximate methods to treat non-linear VIEs. In this work the methods that are used to solve these equations are:

Numerical integration (trapezoidal rule and Simpson's rule).

Runge-Kutta methods (classic third-order, optimal third-order, fourth-order, and classic fourth-order).

Classic spline functions (quadratic and cubic classic spline functions).

B-spline functions (first-order, second-order, third-order, and fourth-order).

These methods exist and are easy to use to solve differential equations and integration, but it is difficult to apply them to integral equations, especially non-linear integral equations, because the known function into another function (kernel function) into integral. We therefore need to modify these methods and techniques so that they can be applied to non-linear integral equations.

A comparison between the exact solution and the approximate solution for all the methods is made, using least square errors.

All methods are illustrated by a written algorithm, and the accuracy of these methods is proved by showing how they solve some illustrative examples with excellent results.

Finally, we wrote the general computer programs for the examples presented using the MATLAB program, Version 7.0.

1.3 Organization of the Thesis

The work in this thesis is divided into five chapters:

Chapter 1 is an introduction to the history of numerical analysis and the objectives of this thesis.

Chapter 2 presents the classification of the non-linear integral equation.

In Chapter 3, multistep methods including the trapezoidal rule and Simpson's rule are used to solve non-linear VIEs of the second kind.

Chapter 4 develops the methods and applies the third-order Runge-Kutta method (RK3 Optimal and RK3 Kutta's) and the fourth-order Runge-Kutta method (RK4_Classic and RK4_Kutta's) to solve non-linear VIEs of the second kind.

Chapter 5 contains two sections. The first one covers quadratic and cubic classic spline functions. The second one covers B-spline functions (first-order, second-order, third-order, and fourth-order), which have been used to find the approximate solution to non-linear VIEs of the second kind. The required integrals in this method are calculated using the trapezoidal rule and the Runge-Kutta method, with initial values generated using Day's starting procedure.

Chapter 6 contains conclusions and gives the results, from all the chapters, that have been obtained by following all the methods presented.

CHAPTER 2

PRELIMINARY CONCEPTS

2.1 Classification of Integral Equations

We are concerned with integral equations in which the integration is with respect to the single real variable. The extension of the terminology and methods to higher order integral equations, where these appear, is straightforward.

The notation adopted in this section is as follows: the unknown function will be denoted by Φ or $\Phi(x)$. Every integral equation contains a function obtained from Φ by integration that is of the form $\int_a^b k\big(x,t,\Phi(t)\big)dt$, where k is called the "kernel". For example, in the integral equation

$$\Phi(x) = \int_0^1 |x-t|\Phi(t)dt + f(x) \qquad (0 \le x \le 1). \tag{2.1}$$

$k(x,t) = |x-t|$ is the kernel, and the function f, called the "free term", is also assumed to be known. The free term and the kernel will in general be complex-valued functions of real variables. A condition such as $(0 \le x \le 1)$ following an equation indicates that the equation holds for all values of x in the given interval, or "for all $x \in [0,1]$". We seek a solution $\Phi(x)$ satisfying the equation for all $x \in [0,1]$, for the integral equation mentioned above [48-49].

The classification of integral equations centers on three essential characteristics that together describe their overall structure, and it is useful to set these down briefly before giving greater detail.

i. The "kind" of an equation refers to the location of the unknown function. Equations of the first kind have the unknown function present only under the integral sign; equations of the "second" and "third" kind also have the unknown function outside the integral.

ii. The historical descriptions “Fredholm” and “Volterra” are concerned with the integration interval. In a Fredholm equation the integration is over a finite interval with fixed end-points; in a Volterra equation the integral is indefinite.
iii. The adjective “singular” is sometimes used when the integration is improper, either because the interval is infinite or because the integrand is unbounded within the given interval. Obviously an integral equation can be singular on both counts.

The general forms of integral equation of the Volterra, Fredholm, first, second, third, linear, non-linear, homogenous, Hammerstein, convolution and symmetric kinds are as follows [50-51]:

$$h(x)\Phi(x) = f(x) + \int_a^{b(x)} k\big(x, t, \Phi(t)\big)dt \qquad x \in [a, b] \qquad (2.2)$$

Definition 2.1:
The integral equation (2.2) is said to be a Volterra integral equation (VIE) if $b(x) = x$.

Definition 2.2:
The integral equation (2.2) said to be a Fredholm integral equation if $b(x) = b$.

Definition 2.3:
The integral equation (2.2) said to be a linear integral equation if $k\big(x, t, \Phi(t)\big) = k(x, t)\ \Phi(t)$; otherwise, the equation is a non-linear integral equation.

Definition 2.4:
The integral equation (2.2) is called an integral equation of the first kind if $h(x) = 0$.

Definition 2.5:
The integral equation (2.2) is called an integral equation of the second kind if $h(x) = 1$.

Definition 2.6:

The integral equation (2.2) is called an integral equation of the third kind if $h(x)$ is an assigned function.

Definition 2.7:

The integral equation (2.2) is called a homogenous integral equation of the second kind if $h(x) = 1$ and $f(x) = 0$.

Definition 2.8:

The integral equation (2.2) is called a Hammerstein type of integral equation if $k\big(x, t, \Phi(t)\big) = k(x, t)\, H\big(t, \Phi(t)\big)$.

Definition 2.9:

The integral equation (2.2) is called a convolution type of integral equation if the kernel depends only on the difference $(x - t)$, i.e. if $k\big(x, t, \Phi(t)\big) = k\big(x - t, \Phi(t)\big)$.

Definition 2.10:

The integral equation (2.2) is called a symmetric type of integral equation if the kernel satisfies $k(x, t) = k(t, x)$.

Definition 2.11:

The kernel of the integral equation (2.2) is called a degenerate kernel for the integral equation if it can be written in the form:

$$\int_a^b f(x)dx = \frac{h}{2}k(x,t) = \sum_{k=1}^{n} a_k(x)b_k\,(t).$$

2.2 The Connection Between Differential and Integral Equations

The initial value problem is equivalent to a VIE, and the boundary value problem is equivalent to a Fredholm integral equation.

Sometimes the solutions to initial value and boundary value problems are not very simple, but if we reduce this problem to the integral equation we can solve them directly.

2.3 Initial Value Problems Reduced to VIEs

Suppose that Φ satisfies

$$\left.\begin{aligned} \Phi'(x) &= k\big(x, \Phi(x)\big) \qquad (0 < x < 1) \\ \Phi(0) &= \Phi_0 \end{aligned}\right\} \tag{2.3}$$

where the function k and the number Φ_0 are given. We assume that Φ is continuous in the closed interval [0,1] which, in particular, allows the initial condition to be interpreted sensibly. Integrating Eq. (2.3), gives

$$\Phi(x) = \int_0^x k\big(t, \Phi(t)\big)dt + \Phi_0 \qquad (0 \le x \le 1). \tag{2.4}$$

Conversely, if Φ is continuous function satisfying Eq. (2.4) then $\Phi(0) = \Phi_0$ and theintegral may be differentiated to give Eq. (2.3).

We can proceed in the same way for the second-order initial value problem:

$$\left.\begin{aligned} \Phi''(x) &= k\big(x, \Phi(x)\big) \qquad (0 < x < 1), \\ \Phi(0) &= \Phi_0, \quad \Phi'(0) = \Phi_0', \end{aligned}\right\} \tag{2.5}$$

where the number Φ_0' is additionally assigned. Again we must make a stipulation regarding continuity, and to avoid the need to raise this issue repeatedly we adopt the convention that, unless otherwise indicated, in a problem such as Eq. (2.5) Φ and its derivatives up to the highest order specified are extended at the end points of the interval to continuous functions on the closed interval.

One-time integration of Eq. (2.5) gives:

$$\Phi'(x) = \int_0^x k\big(t, \Phi(t)\big)dt + \Phi_0' \qquad (0 \le x \le 1),$$

Satisfying $\Phi'(0) = \Phi_0'$ and a second integration produces:

$$\Phi(x) = \iint_0^{x\ x} k\big(t, \Phi(t)\big)dt + \Phi_0' x + \Phi_0 \qquad (0 \le x \le 1), \tag{2.6}$$

The further constant of integration is chosen so that $\Phi(0) = \Phi_0$.

Suppose:

$$H(s) = \int_0^s k\big(t, \Phi(t)\big)dt\,,$$

Then

$$\Phi(x) = \int_0^x H(s)ds + \Phi_0' x + \Phi_0 \qquad (2.7)$$

Using integration methods on Eq. (2.7) we obtain

$$\Phi(x) = [s\,H(s)]_0^x - \int_0^x s\,H'(s)ds + \Phi_0' x + \Phi_0$$

$$= \int_0^x (x-t)k\big(t,\Phi(t)\big)dt + \Phi_0' x + \Phi_0 \qquad (0 \le x \le 1) \qquad (2.8)$$

Then Eq. (2.8) is the integral equation corresponding to Eq. (2.5). If Φ is a continuous solution of Eq. (2.8) then differentiation under the integral sign shows that Φ also satisfies Eq. (2.5) also, so the two problems for Φ correspond [13-16].

Example 2.1:

Reduce the initial value problem to a VIE in the following:

i. $$\left.\begin{matrix}\Phi'(x) = x^2 + \Phi^2(x) \\ \Phi(0) = 0\end{matrix}\right\} \quad (0 < x < 1) \qquad (2.9)$$

ii. $$\left.\begin{matrix}\Phi''(x) = x\Phi(x) \\ \Phi(0) = 1,\ \Phi'(x) = 0\end{matrix}\right\} \quad (0 < x < 1) \qquad (2.10)$$

Solution:

Applying Eq. (2.4) to Eq. (2.9) we get

$$\Phi(x) = \int_0^x \big(t^2 + \Phi^2(t)\big)dt\,, \qquad (0 \le x \le 1).$$

Applying Eq. (2.8) to Eq. (2.10) we get:

$$\Phi(x) = \int_0^x (x-t)t\,\Phi(t)dt + 1\,, \qquad (0 \le x \le 1).$$

2.4 Picard's Method of Successive Approximations

Consider the initial value problem given by the first-order non-linear differential equation $\frac{d\Phi}{dx} = f\big(x,\Phi(x)\big)$ with the initial condition $\Phi(a) = b$ at $x = a$. This initial

value problem can be transformed into a non-linear integral equation and is written as:

$$\Phi(x) = b + \int_a^x f\big(x, \Phi(x)\big)dx.$$

For a first approximation, we replace the $\Phi(x)$ in $f\big(x, \Phi(x)\big)$ by b, for a second approximation, we replace it by the first approximation, for the third by the second, and so on. We demonstrate this method below with examples [52].

Example 2.2:

Consider the first-order non-linear differential equation $\frac{d\Phi}{dx} = x + \Phi^2$, where $\Phi(0) = 0$ when $x = 0$. Determine the approximate analytical solution using Picard's method.

Solution

The given differential equation can be written in integral equation form as

$$\Phi(x) = \int_0^x \big(x + \Phi^2(x)\big)dx.$$

The zero'th approximation is $\Phi(x) = 0$.

First approximation: Put $\Phi(x) = 0$ in $x + \Phi^2(x)$, yielding

$$\Phi(x) = \int_0^x x dx = \frac{1}{2}x^2.$$

Second approximation: Put $\Phi(x) = \frac{x^2}{2}$ in $x + u^2$, yielding

$$\Phi(x) = \int_0^x \left(x + \frac{x^2}{4}\right) dx = \frac{x^2}{2} + \frac{x^5}{20}.$$

Third approximation: Put $\Phi = \frac{x^2}{2} + \frac{x^5}{20}$ in $x + \Phi^2$, giving

$$\Phi(x) = \int_0^x \left\{x + \left(\frac{x^2}{2} + \frac{x^5}{20}\right)^2\right\} dx$$

$$= \int_0^x \left(x + \frac{x^4}{4} + \frac{x^7}{20} + \frac{x^{10}}{400}\right) dx$$

$$= \frac{x^2}{2} + \frac{x^5}{20} + \frac{x^8}{160} + \frac{x^{11}}{4400}.$$

Proceeding in this manner, the fourth approximation can be written, after a rigorous algebraic manipulation, as

Fourth approximation:

$$\Phi(x) = \frac{x^2}{2} + \frac{x^5}{20} + \frac{x^8}{160} + \frac{7x^{11}}{8800} + \frac{3x^{14}}{49280} + \frac{87x^{17}}{23936000} + \frac{x^{20}}{7040000} + \frac{x^{23}}{445280000},$$

And so on. This is the solution of the problem in series form, and the series seems from its appearance to be convergent [52].

2.5 Existence of a Solution for Non-Linear VIEs

In this section, we will show an existence theorem for the solution of non-linear VIEs. Also, in what follows, we give a brief summary of the conditions under which a solution exists for this equation.

We will start by rewriting the non-linear VIE of the second kind as:

$$\Phi(x) = f(x) + \int_0^x k\big(x, t, \Phi(\mathrm{t})\big)dt. \tag{2.11}$$

It appears that the specific conditions under which a solution exists for the non-linear VIE are:

(i) The function $f(x)$ is integrable and bounded in $a \le x \le b$.

(ii) The function $f(x)$ satisfies the Lipschitz condition in the interval (a, b).

This means that

$$|f(x) - f(y)| < L|x - y|. \tag{2.12}$$

(iii) The function $k\big(x, t, \Phi(t)\big)$ is integrable and bounded $|k\big(x, t, \Phi(t)\big)| < K$ in $a \le x, t \le b$.

(iv) The function $k\big(x, t, \Phi(t)\big)$ satisfies the Lipschitz condition

$$|k(x, t, z) - k(x, t, z')| < M|z - z'|. \tag{2.13}$$

In this chapter, the focus will be on solving non-linear VIEs rather than proving theoretical concepts of convergence and existence. The theorems of uniqueness, existence, and convergence are important and necessary, and can be found in the literature. This text will concentrate on the determination of the solution $\Phi(x)$ of a non-linear VIE of the first and the second kind [53].

2.6 Some Fundamental Concepts

In this section, we consider some important theorems and concepts that are used in this thesis.

Theorem 2.1:

Suppose:

i. every component is continuous when $f(x)$ is continuous.

ii. $k\big(x,t,\Phi(t)\big)$ is a continuous function for $t,x \in [a,b]$ and $-\infty < \|\Phi\| < \infty$, and

iii. the kernel satisfies the Lipschitz condition

$$\big\|k\big(x,t,\Phi(t)\big) - k\big(x,t,\Psi(t)\big)\big\| \leq a\|\Phi(t) - \Psi(t)\|,$$

where a is Lipschitz constant and

$$\|k(x,t,\Phi)\| = \max_{1\leq i\leq m} |k_i(x,t,\Phi)|.$$

Then Eq. (2.2) has a unique continuous solution in $[a,b]$ [54-55].

We also need the following theorems in the sequel:

Theorem 2.2:

Suppose $x_0, x_1, \dots, x_n$ are distinct numbers in the interval $[a,b]$ and $f \in C^{n+1}[a,b]$. Then, for each x in $[a,b]$, a number $\varepsilon(x)$ in (a,b) exists with

$$f(x) = P(x) + \frac{f^{n+1}\big(\varepsilon(x)\big)}{(n+1)!}(x-x_0)(x-x_1)\dots(x-x_n),$$

where $P(x)$ is the interpolating polynomial given by Lagrange interpolation [56].

Theorem 2.3 (Weighted mean value theorem for integrals) [56]:

Suppose $f \in C[a,b]$, the Riemann integral of g exists on the interval $[a,b]$, and $\mathrm{g}(x)$ keeps the same sign on $[a,b]$. Then there exists a number c in (a,b) with:

$$\int_a^b f(x)\mathrm{g}(x)dx = f(c)\int_a^b \mathrm{g}(x)dx.$$

Theorem 2.4 (Intermediate value theorem) [56]:

If $f \in C[a,b]$ and K is any number between $f(a)$ and $f(b)$, then there exists a number c in (a,b) for which $(c) = K$.

Test Example 1:

Consider the non-linear VIE:

$$\Phi(x) = 1 + x^2 - xe^{x^2} + \int_0^x e^{x^2-t^2-1}\, e^{\Phi(t)} dt$$

with exact solution [53]:

$$\Phi(x) = 1 + x^2$$

Test Example 2:

Consider the non-linear VIE:

$$\Phi(x) = \cos(x) - \sin(x) - \frac{1}{4}\sin(2x) + \frac{1}{2}x - \frac{1}{2}x^2 + \int_0^x (x-t)\Phi^2(t) dt$$

with exact solution [53]:

$$\Phi(x) = \cos(x) - \sin(x)$$

Test Example 3:

Consider the non-linear VIE:

$$\Phi(x) = e^x - \frac{1}{9}e^{3x} + \frac{1}{9} + \frac{1}{3}x + \int_0^x (x-t)\Phi^3(t) dt$$

with exact solution [53]:

$$\Phi(x) = e^x$$

CHAPTER 3

QUADRATURE METHODS

3.1 Numerical Integration

The need often arises to evaluate the definite integral of a function that has no explicit antiderivative or whose antiderivative is difficult to obtain. The basic method involved in approximating $\int_a^b f(x)dx$ is called numerical quadrature and it uses a sum $\sum_{i=0}^{n} f(x_i)L_i(x)$ to approximate $\int_a^b f(x)dx$.

In this section, the methods of quadrature are built on the interpolation polynomials. The basic idea is to select a set of distinct nodes $\{x_0, \dots, x_n\}$ from the interval $[a, b]$. Then integrate the Lagrange interpolating polynomial:

$$P_n(x) = \sum_{i=0}^{n} f(x_i)L_i(x) \tag{3.1}$$

and its truncation error term over $[a, b]$ to obtain

$$\int_a^b f(x)dx = \int_a^b \sum_{i=0}^{n} f(x_i)L_i(x)\,dx + \int_a^b \prod_{i=0}^{n}(x - x_i)\frac{f^{(n+1)}\big(\xi(x)\big)}{(n+1)!}dx$$

$$= \sum_{i=0}^{n} a_i f(x_i) + \frac{1}{(n+1)!}\int_a^b \prod_{i=0}^{n}(x - x_i)f^{(n+1)}\big(\xi(x)\big)dx,$$

where $\xi(x)$ is in $[a, b]$ for each x and $a_i = \int_a^b L_i(x)\,dx$, for each $i = 0,1, \dots, n$.

The quadrature formula is, therefore,

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n} a_i f(x_i),$$

with the error given by

$$E(f) = \frac{1}{(n+1)!}\int_a^b \prod_{i=0}^{n}(x - x_i) f^{(n+1)}\big(\xi(x)\big)dx. \tag{3.2}$$

The procedures that give us the trapezoidal rule and Simpson's rule are produced by using first and second Lagrange polynomials with equally-spaced nodes. These two rules are presented in calculus courses [57].

3.2 Quadrature Rule

The quadrature rule uses a weighted sum of a finite number of sample values of the integrand function. Let $f(x)$ be a real-valued function of a real variable, defined on a finite interval $a \le x \le b$. We can seek to compute the value of the integral $\int_a^b f(x)\,dx$, thus

$$\int_a^b f(x)\,dx = \sum_{j=1}^{N} w_j\, f\big(x_j\big) + R[f] \tag{3.3}$$

$R[f]$ is the remainder (which is usually not known exactly), the quadrature rule $\{w_j, x_j\}_{j=1}^{N}$ exists in tabulated form and the real numbers x_j are the integration nodes that lie in the bounded interval and are constants called quadrature weights [22-58].
We seek to generate the quadrature rule that is defined by the number of nodes and the weight function, through the two algorithms presented here.

3.2.1 Trapezoidal Rule

To derive the trapezoidal rule for approximating $\int_a^b f(x)\,dx$, let $x_0 = a,\ x_1 = b,$ $h = b - a$ and use the linear Lagrange polynomial:

$$P_1(x) = \frac{(x - x_1)}{(x_0 - x_1)} f(x_0) + \frac{(x - x_0)}{(x_1 - x_0)} f(x_1). \tag{3.4}$$

Then:

$$\int_a^b f(x)\,dx = \int_{x_0}^{x_1} \left[\frac{(x - x_1)}{(x_0 - x_1)} f(x_0) + \frac{(x - x_0)}{(x_1 - x_0)} f(x_1)\right] dx + \frac{1}{2}\int_{x_0}^{x_1} f''\big(\xi(x)\big)(x - x_0)(x - x_1)dx. \tag{3.5}$$

The product $(x-x_0)(x-x_1)$ keeps the same sign on $[x_0, x_1]$, so the weighted mean value theorem (Theorem 2.3) for integrals can be applied to the error term to give, for some ξ in $(x_0 - x_1)$,

$$\int_{x_0}^{x_1} f''\big(\xi(x)\big)(x-x_0)(x-x_1)dx = f''(\xi)\int_{x_0}^{x_1}(x-x_0)(x-x_1)dx$$

$$= f''(\xi)\left[\frac{x^3}{3} - \frac{(x_1+x_0)}{2}x^2 + x_0x_1x\right]_{x_0}^{x_1}$$

$$= -\frac{h^3}{6}f''(\xi).$$

Consequently, Eq. (3.5) implies that

$$\int_a^b f(x)\,dx = \left[\frac{(x-x_1)^2}{2(x_0-x_1)}f(x_0) + \frac{(x-x_0)^2}{2(x_1-x_0)}f(x_1)\right]_{x_0}^{x_1} - \frac{h^3}{12}f''(\xi)$$

$$= \frac{(x_1-x_0)}{2}[f(x_0)+f(x_1)] - \frac{h^3}{12}f''(\xi). \qquad (3.6)$$

Using the notation $h = x_1 - x_0$ gives the following rule:

Trapezoidal Rule:

$$\int_a^b f(x)\,dx = \frac{h}{2}[f(x_0)+f(x_1)] - \frac{h^3}{12}f''(\xi). \qquad (3.7)$$

This Eq. (3.7) is called the trapezoidal rule because when f is a function with non-negative values, $\int_a^b f(x)\,dx$ is approximated by the area of a trapezoid, as described in Fig. 1.

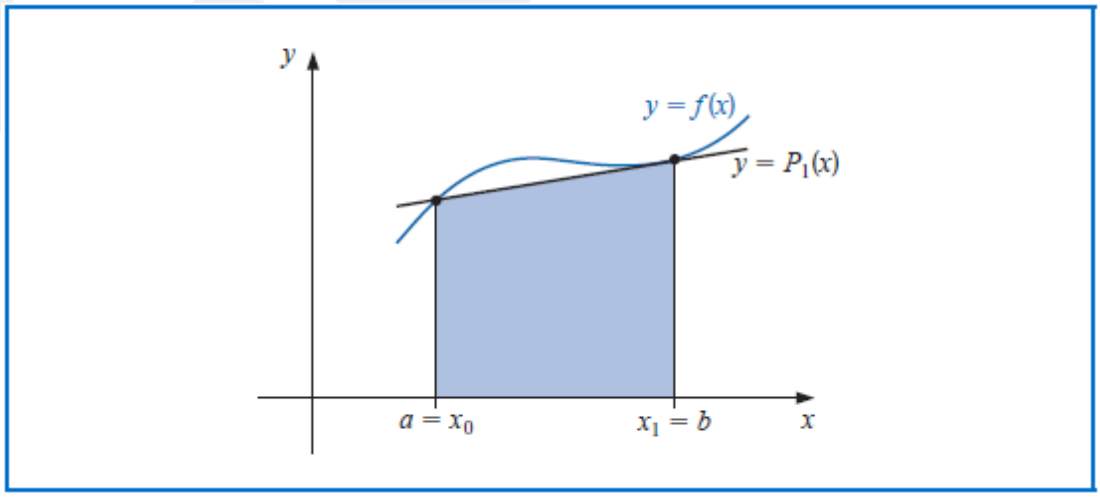

Figure 1 Trapezoidal rule [57]

The error term for the trapezoidal rule includes f'', so the trapezoidal rule when applied to any function gives the exact result if f'' is identical to 0, which means that f is any polynomial of degree one or less [57].

Theorem 3.1: (Composite trapezoidal rule)

Let $f \in C^2[a,b], h = (b-a)/n,$ and $x_j = a + jh,$ for each $j = 0,1,\dots,n.$ There exists a $\mu \in (a,b)$ for which the composite trapezoidal rule for n subintervals can be written, together with its error term, as follows [57]:

$$\int_a^b f(x)dx = \frac{h}{2}\left[f(a) + 2\sum_{j=1}^{n-1} f(x_j) + f(b)\right] - \frac{b-a}{12}h^2 f''(\mu). \qquad (3.8)$$

Theorem 3.2 (Trapezoidal rule error estimate, single subinterval):

Let $f \in C^2([a,b])$ and let p_1 interpolate f at a and b. Define $T_1(f) = I(p_1)$. Then [59] there exists $\eta \in [a,b]$ such that:

$$I(f) - T_1(f) = -\frac{1}{12}(b-a)^3 f''(\eta).$$

where

$$I(f) = \int_a^b f(x)dx.$$

Theorem 3.3 (Trapezoidal rule error estimate, uniform grid):

Let $f \in C^2([a,b])$ and let $T_n(f)$ be the n subinterval trapezoidal rule approximation to $I(f)$, by a uniform grid.

There exists $\xi_h \in [a,b]$, depending on h, such that [59]:

$$I(f) - T_n(f) = -\frac{b-a}{12}h^2 f''(\xi_h).$$

Theorem 3.4 (Trapezoidal rule error estimate, non-uniform grid):

Let $f \in C^2([a,b])$ and let $T_n(f)$ be the n subinterval trapezoidal rule approximation to $I(f)$ using the non-uniform grid defined by

$$a = x_0 < x_1 < x_2 < \cdots < x_{n-1} < x_n = b,$$

with $h_i = x_i - x_{i-1}$ and $h = max_i h_i$. Then [59]:

$$|I(f) - T_n(f)| \leq \frac{b-a}{12}h^2 \max_{x\in[a,b]} |f''|.$$

3.2.2 Simpson's Rule

We get Simpson's rule by integrating over the closed interval $[a, b]$ and using the second Lagrange polynomial together with equally-spaced nodes $x_0 = a, x_2 = b$, and $x_1 = a + h$, where $h = (b - a)/2$. Simpson's rule is given in Fig. 2.

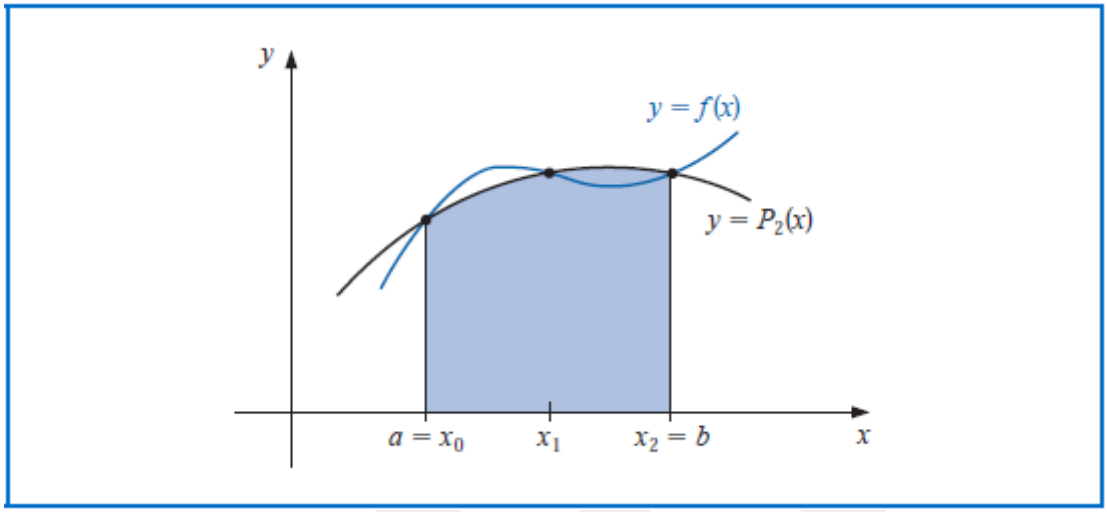

Figure 2 Simpson's rule [57]

Therefore

$$\int_a^b f(x)\,dx = \int_{x_0}^{x_2} \left[\frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} f(x_0) + \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} f(x_1) + \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)} f(x_2) \right] dx + \int_{x_0}^{x_2} \frac{(x - x_0)(x - x_1)(x - x_2)}{6} f^{(3)}\big(\xi(x)\big)\,dx.$$

In this way, Simpson's rule also offers only an $O(h^4)$ error term including $f^{(3)}$. If we approach the problem in another way, a higher-order term including $f^{(4)}$ can be derived.

To show this alternative method, assume that f is expanded in the third Taylor polynomial about x_1. Then for each x in $[x_0, x_2]$, a number $\xi(x)$ in (x_0, x_2) exists with

$$f(x) = f(x_1) + f'(x_1)(x - x_1) + \frac{f''(x_1)}{2}(x - x_1)^2 + \frac{f'''(x_1)}{6}(x - x_1)^3 + \frac{f^{(4)}(\xi(x))}{24}(x - x_1)^4$$

and

$$\int_{x_0}^{x_2} f(x)dx = \left[f(x_1)(x - x_1) + \frac{f'(x_1)}{2}(x - x_1)^2 + \frac{f''(x_1)}{6}(x - x_1)^3 + \frac{f'''(x_1)}{24}(x - x_1)^4\right]_{x_0}^{x_2} + \frac{1}{24}\int_{x_0}^{x_2} f^{(4)}(\xi(x))(x - x_1)^4 dx. \quad (3.9)$$

Because $(x - x_1)^4$ is never negative on $[x_0, x_2]$, the weighted mean value theorem (Theorem 2.3) for integrals means that:

$$\frac{1}{24}\int_{x_0}^{x_2} f^{(4)}(\xi(x))(x - x_1)^4 dx = \frac{f^{(4)}(\xi_1)}{24}\int_{x_0}^{x_2} (x - x_1)^4 dx = \frac{f^{(4)}(\xi_1)}{120}(x - x_1)^5\Big]_{x_0}^{x_2},$$

for some number ξ_1 in (x_0, x_2).

However, $h = x_2 - x_1 = x_1 - x_0$, so

$$(x_2 - x_1)^2 - (x_0 - x_1)^2 = (x_2 - x_1)^4 - (x_0 - x_1)^4 = 0,$$

whereas

$$(x_2 - x_1)^3 - (x_0 - x_1)^3 = 2h^3 \text{ and } (x_2 - x_1)^5 - (x_0 - x_1)^5 = 2h^5.$$

Consequently, Eq. (3.9) can be rewritten as:

$$\int_{x_0}^{x_2} f(x)dx = 2hf(x_1) + \frac{h^3}{3}f''(x_1) + \frac{f^{(4)}(\xi_1)}{60}h^5.$$

If we now replace $f''(x_1)$ by the approximation, we have:

$$\int_{x_0}^{x_2} f(x)dx = 2hf(x_1) + \frac{h^3}{3}\left\{\frac{1}{h^2}[f(x_0) - 2f(x_1) + f(x_2)] - \frac{h^2}{12}f^{(4)}(\xi_2)\right\} + \frac{f^{(4)}(\xi_1)}{60}h^5$$

$$= \frac{h}{3}[f(x_0) + 4f(x_1) + f(x_2)] - \frac{h^5}{12}\left[\frac{1}{3}f^{(4)}(\xi_2) - \frac{1}{5}f^{(4)}(\xi_1)\right].$$

Alternative methods show that the values ξ_1 and ξ_2 in this expression can be replaced by a common value ξ in (x_0, x_2). This gives Simpson's rule.

Simpson's Rule:

$$\int_{x_0}^{x_2} f(x)dx = \frac{h}{3}[f(x_0) + 4f(x_1) + f(x_2)] - \frac{h^5}{90}f^{(4)}(\xi). \qquad (3.10)$$

In Simpson's rule the error term includes $f^{(4)}$, and when applied to any polynomial of degree three or less it gives exact results [57].

Theorem 3.5: (Composite Simpson's rule)

Let $f \in C^4[a,b], n$ be even, $h = (b-a)/n$, and $x_j = a + jh$, for each $j = 0,1,\dots,n$. There exists a $\mu \in (a,b)$ for which the composite Simpson's rule for n subintervals can be written, together with its error term, in the following way [57]:

$$\int_a^b f(x)dx = \frac{h}{3}\left[f(a) + 2\sum_{j=1}^{(n/2)-1} f(x_{2j}) + 4\sum_{j=1}^{n/2} f(x_{2j-1}) + f(b)\right] - \frac{b-a}{180}h^4 f^{(4)}(\mu). \qquad (3.11)$$

Theorem 3.6 (Simpson's rule error estimate, single subinterval):

Let $f \in C^4([a,b])$ and let p_1 interpolate f at a and b. Define $T_1(f) = I(p_1)$. Then there exists $\eta \in [a,b]$ such that [59]:

$$I(f) - T_1(f) = -\frac{1}{180}(b-a)^5 f^{(4)}(\eta).$$

where

$$I(f) = \int_a^b f(x)dx.$$

Theorem 3.7 (Simpson's rule error estimate, uniform grid):

Let $f \in C^4([a,b])$ and let $T_n(f)$ be the n subinterval Simpson's rule approximation to $I(f)$, using a uniform grid.

Then there exists $\xi_h \in [a,b]$, depending on h, such that [59]:

$$I(f) - T_n(f) = -\frac{b-a}{180}h^4 f^{(4)}(\xi_h).$$

Theorem 3.8 (Simpson's rule error estimate, non-uniform grid):

Let $f \in C^2([a,b])$ and let $T_n(f)$ be the n subinterval Simpson's rule approximation to $I(f)$, using the non-uniform grid defined by:

$$a = x_0 < x_1 < x_2 < \cdots < x_{n-1} < x_n = b,$$

with $h_i = x_i - x_{i-1}$ and $h = max_i h_i$. Then [59]:

$$|I(f) - T_n(f)| \leq \frac{b-a}{180} h^4 \max_{x \in [a,b]} |f^{(4)}|.$$

3.3 Solving Non-Linear VIEs Using Quadrature Methods

The goal of using the quadrature method of integration is to approximate the integral equation over the suitable interval by evaluating the function at a finite number of sample points.

3.3.1 Trapezoidal Rule for Solving Non-Linear VIEs

$x_0 = t_0 = 0, h = (b-a)/n = b/n$

$x_r = x_0 + r * h, t_r = t_0 + r * h$

$$\Phi(x_0) = f(x_0)$$

$$\Phi(x_r) = f(x_r) + \left(\frac{h}{2}\right)[K\big(x_r, t_0, \Phi(t_0)\big) + 2\sum_{j=1}^{r-1} K\left(x_r, t_j, \Phi\big(t_j\big)\right)$$

$$+K\big(x_r, t_r, \Phi_1(t_r)\big)] \tag{3.12}$$

where $r = 1,2,3, \dots, n$.

To evaluate $\Phi_1(t_r)$ we use Day's starting procedure:

$$\Phi_1(x_1) = f(x_1) + h * K\big(x_1, t_0, \Phi(t_0)\big) \tag{3.13}$$

$$\Phi_1(x_r) = f(x_r) + \left(\frac{r}{r-1}\right) * h * \sum_{j=1}^{r-1} K\left(x_r, t_j, \Phi\big(t_j\big)\right) \tag{3.14}$$

where $r = 2,3,4, \dots, n$.

3.3.2 Simpson's Rule for Solving Non-Linear VIEs

$x_0 = t_0 = 0,\ h = (b-a)/n = b/n$

$x_r = x_0 + r * h,\ t_r = t_0 + r * h$

$$\Phi(x_0) = f(x_0)$$

$$\Phi(x_1) = f(x_1) + \left(\frac{h}{3}\right) * \begin{bmatrix} K\big(x_1, t_0, \Phi(t_0)\big) + 4K\left(x_1, \frac{t_1}{2}, \Phi_3(t_1)\right) \\ +K\big(x_1, t_1, \Phi_2(t_1)\big) \end{bmatrix} \quad (3.15)$$

where we use Day's starting procedure to evaluate $\Phi_2(x_1)$ and $\Phi_3(x_1)$ as follows:

$$\Phi_1(x_1) = f(x_1) + h * K\big(x_1, t_0, \Phi(t_0)\big)$$

$$\Phi_2(x_1) = f(x_1) + \left(\frac{h}{2}\right) * \big[K\big(x_1, t_0, \Phi(t_0)\big) + K\big(x_1, t_1, \Phi_1(t_1)\big)\big]$$

$$\Phi_3(x_1) = \frac{f(x_1)}{2} + \left(\frac{h}{4}\right) * \begin{bmatrix} K\left(\frac{x_1}{2}, t_0, \Phi(t_0)\right) \\ +K\left(\frac{x_1}{2}, \frac{t_1}{2}, \frac{\Phi(t_0)}{2} + \frac{\Phi_2(t_1)}{2}\right) \end{bmatrix} \quad (3.16)$$

If r is even:

$$\Phi(x_r) = f(x_r) + \left(\frac{h}{3}\right) * [K\big(x_r, t_0, \Phi(t_0)\big) + \sum_{j=1}^{r-1} W_{rj}(x_r, t_j, \Phi(t_j))$$

$$+K\big(x_r, t_r, \Phi_1(t_r)\big)] \quad (3.17)$$

$W_{rj} = 4$ if $j = 1,3,5, \ldots$

$W_{rj} = 2$ if $j = 2,4,6, \ldots$

If r is odd:

$$\Phi(x_r) = f(x_r) + \left(\frac{h}{3}\right)\sum_{j=0}^{r-3} W_{rj} K\left(x_r, t_j, \Phi\big(t_j\big)\right)$$

$$+\left(\frac{3}{8}\right) * h[K\big(x_r, t_{r-3}, \Phi(t_{r-3})\big) + 3K\big(x_r, t_{r-2}, \Phi(t_{r-2})\big)$$

$$+3K\big(x_r, t_{r-1}, \Phi(t_{r-1})\big) + K\big(x_r, t_r, \Phi_1(t_r)\big)]. \quad (3.18)$$

where $W_{r0} = W_{r,r-3} = 1$

and $W_{rj} = 4$ if $j = 1,3,5, \ldots$ and $W_{rj} = 2$ if $j = 2,4,6, \ldots$

$$\Phi_1(x_r) = f(x_r) + \left(\frac{r}{r-1}\right) * h \sum_{j=1}^{r-1} K(x_r, t_j, \Phi(t_j)). \qquad (3.19)$$

where $r = 2,3,4, \dots, n$.

3.4 Numerical Algorithm

3.4.1 Non-Linear VIEs Using the Trapezoidal Rule (Non-Linear VIETRP)

Step (1):

a- Assume $h = \frac{b-a}{n}, n \in N$.

b- Set $\Phi_0 = f_0$.

Step (2):

Compute Φ_{11} using Day's starting procedure (Eq. (3.13)).

Step (3):

Compute Φ_1 using steps 1 and 2 and the trapezoidal rule (Eq. (3.12)).

Step (4):

Compute $\Phi_{1r}; r = 2,3, \dots, n$, using Day's starting procedure (Eq. (3.14)).

Step (5):

Compute $\Phi_r ; r = 2,3, \dots, n$, using steps 1, 3, and 4 and the composite trapezoidal rule (Eq. (3.12)).

3.4.2 Non-Linear VIEs Using Simpson's Rule (Non-Linear VIESMP)

Step (1):

a- Assume $h = \frac{b-a}{n}, n \in N$.

b- Set $\Phi_0 = f_0$.

Step (2):

Compute Φ_{11}, Φ_{21} using Day's starting procedure (Eq. (3.16)).

Step (3):

Compute Φ_1 using steps 1 and 2 and Simpson's rule (Eq. (3.15)).

Step (4):

Compute $\Phi_{1r}\,; r = 2,3, \dots, n,$ using Day's starting procedure (Eq. (3.19)).

Step (5):

Compute $\Phi_r\,; r = 2,3, \dots, n,$ using steps 1, 3, and 4 and Simpson's rule (Eq. (3.17)) if n is even and the composite Simpson's rule (Eq. (3.18)) if n is odd.

3.5 Numerical Examples

We will show here how the quadrature methods (the trapezoidal rule and Simpson's rule) can be used to give numerical results for non-linear VIEs with variable coefficients. The results of these two methods are given using algorithms for a non-linear VIE using the trapezoidal rule (non-linear VIETRP) and a non-linear VIE using Simpson's rule (non-linear VIESMP) respectively.

Test Example 1:

Consider the non-linear VIE:

$$\Phi(x) = 1 + x^2 - xe^{x^2} + \int_0^x e^{x^2 - t^2 - 1}\, e^{\Phi(t)} dt$$

with the exact solution [53]:

$$\Phi(x) = 1 + x^2$$

The results for Test Example 1 using the trapezoidal rule (TRP) and Simpson's rule (SMP) are shown in Table 1.

Table 1 Comparison Between the Exact Solution and the Numerical Solution of $\Phi(x)$ Taking $h = 0.1$ for Test Example 1 Using TRP and SMP

x	Exact Solution	Approximate Solution of $\Phi(x)$ by	
		TRP	SMP
0.0	1.000000000000000	1.000000000000000	1.000000000000000
0.1	1.010000000000000	1.010000000000000	1.010000000000000
0.2	1.040000000000000	1.040000000000000	1.040000000000000
0.3	1.090000000000000	1.090000000000000	1.090000000000000
0.4	1.160000000000000	1.160000000000000	1.160000000000000
0.5	1.250000000000000	1.250000000000000	1.250000000000000
0.6	1.360000000000000	1.360000000000000	1.360000000000000
0.7	1.490000000000000	1.490000000000000	1.490000000000000
0.8	1.640000000000000	1.640000000000000	1.640000000000000
0.9	1.810000000000000	1.810000000000001	1.810000000000000
1.0	2.000000000000000	2.000000000000001	1.999999999999999
L.S.E.		**1.133987551255e-030**	**9.8607613152626e-031**

Test Example 2:

Consider the non-linear VIE:

$$\Phi(x) = \cos(x) - \sin(x) - \frac{1}{4}\sin(2x) + \frac{1}{2}x - \frac{1}{2}x^2 + \int_0^x (x-t)\Phi^2(t)dt$$

with the exact solution [53]:

$$\Phi(x) = \cos(x) - \sin(x)$$

The results for the Test Example 2 using the TRP rule and the SMP rule are shown in Table 2.

Table 2 Comparison Between the Exact Solution and the Numerical Solution of $\Phi(x)$ Taking $h = 0.1$ for Test Example 2 Using TRP and SMP

x	Exact Solution	Approximate Solution of $\Phi(x)$ by	
		TRP	SMP
0.0	1.000000000000000	1.000000000000000	1.000000000000000
0.1	0.895170748631198	0.895503415932432	0.895503415932432
0.2	0.781397247046180	0.782061925148484	0.781401679708307
0.3	0.659816282464266	0.660810400022110	0.660162284715298
0.4	0.531642651694235	0.532960540950866	0.531666054716707
0.5	0.398157023286170	0.399788834522724	0.398523473229770
0.6	0.260693141514643	0.262623884040472	0.260746589873102
0.7	0.120624500046797	0.122833251191109	0.121016886395488
0.8	-0.020649381552357	-0.018190048971533	-0.020559370631043
0.9	-0.161716941356819	-0.159041257256578	-0.161295986748409
1.0	-0.301168678939757	-0.298317644132700	-0.301041643201665
L.S.E.		**3.5882724173280e-005**	**7.23504602860014e-007**

Test Example 3:

Consider the non-linear VIE:

$$\Phi(x) = e^x - \frac{1}{9}e^{3x} + \frac{1}{9} + \frac{1}{3}x + \int_0^x (x-t)\Phi^3(t)dt$$

with the exact solution [53]:

$$\Phi(x) = e^x$$

The results for the Test Example 3 using the TRP rule and the SMP rule are shown in Table 3.

Table 3 Comparison Between the Exact Solution and the Numerical Solution of $\Phi(x)$ Taking $h = 0.1$ for Test Example 3 Using TRP and SMP

x	Exact Solution	Approximate Solution of $\Phi(x)$ by	
		TRP	SMP
0.0	1.000000000000000	1.000000000000000	1.000000000000000
0.1	1.105170918075648	1.104631050567203	1.104631050567203
0.2	1.221402758160170	1.220201707317676	1.221361312606863
0.3	1.349858807576003	1.347805803973578	1.349437821469184
0.4	1.491824697641270	1.488622662846719	1.491672337076845
0.5	1.648721270700128	1.643907118474721	1.648359013054650
0.6	1.822118800390509	1.814963952586527	1.821753761407806
0.7	2.013752707470477	2.003092223895529	2.013323270569754
0.8	2.225540928492468	2.209470447807359	2.224772651827378
0.9	2.459603111156950	2.434922684136788	2.458791587689015
1.0	2.718281828459046	2.679437681142294	2.716631947412773
L.S.E.		**0.002580467257583**	**4.913444656575e-006**

CHAPTER 4

RUNGE-KUTTA METHOD

4.1 Runge-Kutta Method

The derivation follows closely that of the ordinary differential equation. We recall the general p-stage Runge-Kutta method for the initial value problem:

$$\Phi'(x) = f\big(x, \Phi(x)\big) \tag{4.1}$$

$$\Phi(a) = \Phi_0$$

Is given by

$$\Phi_{i+1} = \Phi_i + h \sum_{j=0}^{p-1} w_j \, m_j^i \tag{4.2}$$

where

$$m_0^i = f(a + ih, \Phi_i)$$

$$m_r^i = f\left(a + (i + c_r)h, \Phi_i + h \sum_{j=0}^{r-1} a_{rj} \, m_j^i\right) \tag{4.3}$$

$$\sum_{j=0}^{r-1} a_{rj} = \begin{cases} a_r & r = 1,2, \dots, p-1 \\ 1 & r = p \end{cases} \tag{4.4}$$

with Φ_r an approximation to the solution at $x = x_r = a + rh$. The second argument of m_r^i may be regarded as an approximation to $\Phi(a + (i + a_r)h)$ and we rewrite Eq. (4.2) as:

$$\Phi_{i+1} = \Phi_i + h \sum_{j=0}^{p-1} w_j k\big(x_i + c_j h, \Phi_{i+aj}\big) \tag{4.5}$$

The parameters a_{pj} and c_j are chosen, in practice, to yield a final approximation of the specified order [37].

4.1.1 Third-Order Runge-Kutta Methods

The general form of a third-order Runge-Kutta method (RK3) is

$$m_1 = hf(x_i, y_i)$$
$$m_2 = hf(x_i + c_2 h, y_i + a_{21} m_1)$$
$$m_3 = hf(x_i + c_3 h, y_i + a_{31} m_1 + a_{32} m_2)$$
$$y_{i+1} = y_i + w_1 m_1 + w_2 m_2 + w_3 m_3$$

The array of parameters has the following form:

$\boldsymbol{0}$			
$\boldsymbol{c_2}$	$\boldsymbol{a_{21}}$		
$\boldsymbol{c_3}$	$\boldsymbol{a_{31}}$	$\boldsymbol{a_{32}}$	
	$\boldsymbol{w_1}$	$\boldsymbol{w_2}$	$\boldsymbol{w_3}$

The first row gives the parameters needed to compute m_1 (which is always $m_1 = hf(x_i, y_i)$ so c_1 is always 0), the second row corresponds to m_2, the third row gives the parameters to compute m_3, and the row below the line gives the weights used to form y_{i+1}.

The parameter arrays for two third-order methods are as follows:

The parameters for the RK3 Kutta's Classic method are as follows:

$$c_1 = 0, c_2 = \frac{1}{2}, a_{21} = \frac{1}{2}, c_3 = 1, a_{31} = -1, a_{32} = 2, w_1 = \frac{1}{6}, w_2 = \frac{2}{3}, w_3 = \frac{1}{6}$$

The parameters for the RK3 Optimal method are as follows:

$$c_1 = 0, c_2 = \frac{1}{2}, a_{21} = \frac{1}{2}, c_3 = \frac{3}{4}, a_{31} = 0, a_{32} = \frac{3}{4}, w_1 = \frac{2}{9}, w_2 = \frac{3}{9}, w_3 = \frac{4}{9}$$

4.1.2 Fourth-Order Runge-Kutta Methods

The Runge-Kutta methods described in the previous sections are only the most common and simplest forms in a very extensive field of study. For the fourth-order Runge-Kutta method (RK4),

$$m_1 = hf(x_i, y_i)$$

$$m_2 = hf(x_i + c_2h, y_i + a_{21}m_1)$$
$$m_3 = hf(x_i + c_3h, y_i + a_{31}m_1 + a_{32}m_2)$$
$$m_4 = hf(x_i + c_4h, y_i + a_{41}m_1 + a_{42}m_2 + a_{43}m_3)$$
$$y_{i+1} = y_i + w_1m_1 + w_2m_2 + w_3m_3 + w_4m_4$$

The parameters may be shown in an array:

$\mathbf{0}$				
$\boldsymbol{c_2}$	$\boldsymbol{a_{21}}$			
$\boldsymbol{c_3}$	$\boldsymbol{a_{31}}$	$\boldsymbol{a_{32}}$		
$\boldsymbol{c_4}$	$\boldsymbol{a_{41}}$	$\boldsymbol{a_{42}}$	$\boldsymbol{a_{43}}$	
	$\boldsymbol{w_1}$	$\boldsymbol{w_2}$	$\boldsymbol{w_3}$	$\boldsymbol{w_4}$

The entries in the first column are the coefficients of h; the other elements in each row are the coefficients of m_2, m_3, and m_4 that are used to determine the value of y used to compute the next m. Finally, the entries below the line are the weights used in the linear combination of the m's to compute the value of the next y.

The parameter arrays for two fourth-order methods are as follows:

The parameters for the Classic fourth-order Runge-Kutta method (RK4_Classic) are as follows:

$$c_1 = 0, c_2 = \frac{1}{2}, a_{21} = \frac{1}{2}, c_3 = \frac{1}{2}, a_{31} = 0, a_{32} = \frac{1}{2}, c_4 = 1, a_{41} = 0, a_{42} = 0,$$
$$a_{43} = 1$$
$$w_1 = \frac{1}{6}, w_2 = \frac{1}{3}, w_3 = \frac{1}{3}, w_4 = \frac{1}{6}$$

The parameters for the fourth-order Runge-Kutta method (RK4_Kutta's) are as follows:

$$c_1 = 0, c_2 = \frac{1}{3}, a_{21} = \frac{1}{3}, c_3 = \frac{2}{3}, a_{31} = -\frac{1}{3}, a_{32} = 1, c_4 = 1, a_{41} = 1, a_{42} = -1,$$
$$a_{43} = 1, w_1 = \frac{1}{8}, w_2 = \frac{3}{8}, w_3 = \frac{3}{8}, w_4 = \frac{1}{8}$$

4.2 Solution of Non-Linear VIEs Using the Runge-Kutta Method

The method defined in Eq. (4.5) can be extended and gives a category of Runge-Kutta methods for the solution of non-linear VIEs of the second kind, as follows:

$$\Phi(x) = f(x) + \int_0^x k\big(x, t, \Phi(t)\big)\, dt \,; x \in [a, b]. \tag{4.6}$$

Setting $x = x_i$ in (4.6) we have

$$\Phi(x_i) = f(x_i) + \int_a^{a+ih} k\big(a + ih, t, \Phi(t)\big)\, dt, i = 1,2, \dots, n. \tag{4.7}$$

4.2.1 Classic Third-Order Runge-Kutta Method (RK3 Kutta's)

$$\Phi_0 = f_0$$

$$m_0^i = \Phi_i \tag{4.8}$$

$$m_1^i = L_i\left(x_i + \frac{1}{2}h\right) + \frac{1}{2}hk\left(x_i + \frac{1}{2}h, t_i, m_0^i\right) \tag{4.9}$$

$$m_2^i = L_i(x_i + h) + h\begin{Bmatrix} -k\big(x_i + h, t_i, m_0^i\big) \\ +2k\left(x_i + h, t_i + \frac{1}{2}h, m_1^i\right) \end{Bmatrix} \tag{4.10}$$

$$\Phi_{i+1} = L_i(x_i + h) + \frac{1}{6}h\begin{Bmatrix} k\big(x_i + h, t_i, m_0^i\big) \\ +4k\left(x_i + h, t_i + \frac{1}{2}h, m_1^i\right) \\ +k\big(x_i + h, t_i + h, m_2^i\big) \end{Bmatrix} \tag{4.11}$$

$$L_i(x) = f(x) + \frac{1}{6}h\sum_{j=0}^{i-1}\begin{Bmatrix} k\big(x, t_j, m_0^j\big) + 4k\left(x, t_j + \frac{1}{2}h, m_1^j\right) \\ +k\big(x, t_j + h, m_2^j\big) \end{Bmatrix} \tag{4.12}$$

where $i = 0,1,2, \dots, n$.

and

$$L_0(x) = f(x) \tag{4.13}$$

4.2.2 Optimal Third-Order Runge-Kutta Method (RK3 Optimal)

$$\Phi_0 = f_0$$

$$m_0^i = \Phi_i \tag{4.14}$$

$$m_1^i = L_i\left(x_i + \frac{1}{2}h\right) + \frac{1}{2}hk\left(x_i + \frac{1}{2}h, t_i, m_0^i\right) \tag{4.15}$$

$$m_2^i = L_i\left(x_i + \frac{3}{4}h\right) + \frac{3}{4}h\left\{k\left(x_i + \frac{3}{4}h, t_i + \frac{1}{2}h, m_1^i\right)\right\} \tag{4.16}$$

$$\Phi_{i+1} = L_i(x_i + h) + \frac{1}{9}h\begin{Bmatrix} 2k\left(x_i + h, t_i, m_0^i\right) \\ +3k\left(x_i + h, t_i + \frac{1}{2}h, m_1^i\right) \\ +4k\left(x_i + h, t_i + \frac{3}{4}h, m_2^i\right) \end{Bmatrix} \tag{4.17}$$

$$L_i(x) = f(x) + \frac{1}{9}h\sum_{j=0}^{i-1}\begin{Bmatrix} 2k\left(x, t_j, m_0^j\right) \\ +3k\left(x, t_j + \frac{1}{2}h, m_1^j\right) \\ +4k\left(x, t_j + \frac{3}{4}h, m_2^j\right) \end{Bmatrix} \tag{4.18}$$

where $i = 1,2, \ldots, n$.

and

$$L_0(x) = f(x). \tag{4.19}$$

4.2.3 Classic Fourth-Order Runge-Kutta Method (RK4_Classic)

$$\Phi_0 = f_0$$

$$m_0^i = \Phi_i \tag{4.20}$$

$$m_1^i = L_i\left(x_i + \frac{1}{2}h\right) + \frac{1}{2}hk\left(x_i + \frac{1}{2}h, t_i, m_0^i\right) \tag{4.21}$$

$$m_2^i = L_i\left(x_i + \frac{1}{2}h\right) + \frac{1}{2}h\left\{k\left(x_i + \frac{1}{2}h, t_i + \frac{1}{2}h, m_1^i\right)\right\} \tag{4.22}$$

$$m_3^i = L_i(x_i + h) + h\left\{k\left(x_i + h, t_i + \frac{1}{2}h, m_2^i\right)\right\} \tag{4.23}$$

$$\Phi_{i+1} = L_i(x_i + h) + \frac{1}{6}h\begin{Bmatrix} k\left(x_i + h, t_i, m_0^i\right) \\ +2k\left(x_i + h, t_i + \frac{1}{2}h, m_1^i\right) \\ +2k\left(x_i + h, t_j + \frac{1}{2}h, m_2^i\right) \\ +k\left(x_i + h, t_i + h, m_3^i\right) \end{Bmatrix} \tag{4.24}$$

and

$$L_i(x) = f(x) + \frac{1}{6}h\sum_{j=0}^{i-1}\begin{Bmatrix} k\left(x, t_j, m_0^i\right) \\ +2k\left(x, t_j + \frac{1}{2}h, m_1^i\right) \\ +2k\left(x, t_j + \frac{1}{2}h, m_2^i\right) \\ +k\left(x, t_j + h, m_3^i\right) \end{Bmatrix} \tag{4.25}$$

where $i = 1,2, \ldots, n.$

and

$$L_0(x) = f(x) \tag{4.26}$$

4.2.4 Fourth-Order Runge-Kutta Method (RK4_Kutta's)

$$\Phi_0 = f_0$$

$$m_0^i = \Phi_i \tag{4.27}$$

$$m_1^i = L_i\left(x_i + \frac{1}{3}h\right) + \frac{1}{3}hk\left(x_i + \frac{1}{3}h, t_i, m_0^i\right) \tag{4.28}$$

$$m_2^i = L_i\left(x_i + \frac{2}{3}h\right) + \frac{2}{3}h\begin{Bmatrix} -\frac{1}{3}\,k\left(x_i + \frac{2}{3}h, t_i, m_0^i\right) \\ +k\left(x_i + \frac{2}{3}h, t_i + \frac{1}{3}h, m_1^i\right) \end{Bmatrix} \tag{4.29}$$

$$m_3^i = L_i(x_i + h) + h\begin{Bmatrix} k\left(x_i + h, t_i, m_0^i\right) \\ -k\left(x_i + h, t_i + \frac{1}{3}h, m_1^i\right) \\ +k\left(x_i + h, t_i + \frac{2}{3}h, m_2^i\right) \end{Bmatrix} \tag{4.30}$$

$$\Phi_{i+1} = L_i(x_i + h) + \frac{1}{8}h\begin{Bmatrix} k\left(x_i + h, t_i, m_0^i\right) \\ +3k\left(x_i + h, t_i + \frac{1}{3}h, m_1^i\right) \\ +3k\left(x_i + h, t_j + \frac{2}{3}h, m_2^i\right) \\ +k\left(x_i + h, t_i + h, m_3^i\right) \end{Bmatrix} \tag{4.31}$$

and

$$L_i(x) = f(x) + \frac{1}{8}h\sum_{j=0}^{i-1}\begin{Bmatrix} k\left(x, t_j, m_0^i\right) \\ +3k\left(x, t_j + \frac{1}{3}h, m_1^i\right) \\ +3k\left(x, t_j + \frac{2}{3}h, m_2^i\right) \\ +k\left(x, t_j + h, m_3^i\right) \end{Bmatrix} \tag{4.32}$$

where $i = 1,2, \dots, n.$

and

$$L_0(x) = f(x) \tag{4.33}$$

4.3 Numerical Algorithm

4.3.1 Classic Third-Order Runge-Kutta Method (RK3 Kutta's)

Step (1):

a- Assume $h = \frac{b-a}{n}, n \in N.$

b- Set $\Phi_0 = \Phi(x_0) = f_0 = f(x_0)$.

Step (2):

Set $x_0 = a = 0,$ then find $L_0(x_0), L_0\left(x_0 + \frac{1}{2}h\right), L_0(x_0 + h)$ using Eq.(4.13).

Step (3):

Put $i = 0$ to find m_0^0, m_1^0, m_2^0 using Eq. (4.8), (4.9), and (4.10).

Step (4):

Compute Φ_1 using steps 1, 2, and 3 and Eq. (4.11).

Step (5):

For $i = 1$ to n do steps 6, 7, 8 and 9.

Step (6):

$x_i = x_0 + i * h.$

Step (7):

Compute $L_i(x_i), L_i\left(x_i + \frac{1}{2}h\right), L_i(x_i + h)$ using Eq. (4.12).

Step (8):

Compute m_0^i, m_1^i, m_2^i using Eq. (4.8), (4.9), and (4.10).

Step (9):

Compute Φ_{i+1} using steps 6, 7 and 8 and Eq. (4.11).

4.3.2 Optimal Third-Order Runge-Kutta Method (RK3 Optimal)

Step (1):

a- Assume $h = \frac{b-a}{n}, n \in N.$

b- Set $\Phi_0 = \Phi(x_0) = f_0 = f(x_0)$.

Step (2):

Set $x_0 = a = 0$, then find $L_0(x_0), L_0\left(x_0 + \frac{1}{2}h\right), L_0\left(x_0 + \frac{3}{4}h\right), L_0(x_0 + h)$ using Eq. (4.19).

Step (3):

Put $i = 0$ to find m_0^0, m_1^0, m_2^0 using Eq. (4.14), (4.15), and (4.16).

Step (4):

Compute Φ_1 using steps 1, 2, and 3 and Eq. (4.17).

Step (5):

For $i = 1$ to n do steps 6, 7, 8 and 9.

Step (6):

$x_i = x_0 + i * h.$

Step (7):

Compute $L_i(x_i), L_i\left(x_i + \frac{1}{2}h\right), L_i\left(x_i + \frac{3}{4}h\right), L_i(x_i + h)$ using Eq. (4.18).

Step (8):

Compute m_0^i, m_1^i, m_2^i using Eq. (4.14), (4.15), and (4.16).

Step (9):

Compute Φ_{i+1} using steps 6, 7 and 8 and Eq. (4.17).

4.3.3 Classic Fourth-Order Runge-Kutta Method (RK4_Classic)

Step (1):

a- Assume $h = \frac{b-a}{n}, n \in N.$

b- Set $\Phi_0 = \Phi(x_0) = f_0 = f(x_0)$.

Step (2):

Set $x_0 = a = 0$, then find $L_0(x_0), L_0\left(x_0 + \frac{1}{2}h\right), L_0(x_0 + h)$ using Eq. (4.26).

Step (3):

Put $i = 0$ to find m_0^0, m_1^0, m_2^0 , m_3^0 using Eq. (4.20), (4.21), (4.22) and (4.23).

Step (4):

Compute Φ_1 using steps 1, 2, and 3 and Eq. (4.24).

Step (5):

For $i = 1$ to n do steps 6, 7, 8 and 9.

Step (6):

$x_i = x_0 + i * h.$

Step (7):

Compute $L_i(x_i), L_i\left(x_i + \frac{1}{2}h\right), L_i(x_i + h)$ using Eq. (4.25).

Step (8):

Compute $m_0^i,\ m_1^i,\ m_2^i,\ m_3^i$ using Eq. (4.20), (4.21), (4.22) and (4.23).

Step (9):

Compute Φ_{i+1} using steps 6, 7 and 8 and Eq. (4.24).

4.3.4 Fourth-Order Runge-Kutta Method (RK4_Kutta's)

Step (1):

a- Assume $h = \frac{b-a}{n}, n \in N.$

b- Set $\Phi_0 = \Phi(x_0) = f_0 = f(x_0)$.

Step (2):

Set $x_0 = a = 0$, then find $L_0(x_0), L_0\left(x_0 + \frac{1}{3}h\right), L_0\left(x_0 + \frac{2}{3}h\right), L_0(x_0 + h)$ using Eq. (4.33).

Step (3):

Put $i = 0$ to find $m_0^0,\ m_1^0,\ m_2^0,\ m_3^0$ using Eq. (4.27), (4.28), (4.29) and (4.30).

Step (4):

Compute Φ_1 using steps 1, 2, and 3 and Eq. (4.31).

Step (5):

For $i = 1$ to n do steps 6, 7, 8 and 9.

Step (6):

$x_i = x_0 + i * h.$

Step (7):

Compute $L_i(x_i), L_i\left(x_i + \frac{1}{3}h\right), L_i\left(x_i + \frac{2}{3}h\right), L_i(x_i + h)$ using Eq. (4.32).

Step (8):

Compute $m_0^i, m_1^i, m_2^i, m_3^i$ using Eq. (4.27), (4.28), (4.29) and (4.30).

Step (9):

Compute Φ_{i+1} using steps 6, 7 and 8 and Eq. (4.31).

4.4 Numerical Examples

We will show here the numerical results for the Runge-Kutta methods (classic third-order Runge-Kutta, optimal third-order Runge-Kutta, classic fourth-order Runge-Kutta, and fourth-order Runge-Kutta) for non-linear VIEs with variable coefficients. The algorithms for the programming for these methods are given in sections (4.4.1), (4.4.2), (4.4.3), and (4.4.4) respectively.

Test Example 1:

Consider the non-linear VIE:

$$\Phi(x) = 1 + x^2 - xe^{x^2} + \int_0^x e^{x^2 - t^2 - 1}\, e^{\Phi(t)} dt$$

with the exact solution [53]:

$$\Phi(x) = 1 + x^2$$

The results for the Test Example 1 using the classic third-order Runge-Kutta method (RK3 Kutta's), the optimal third-order Runge-Kutta method (RK3 Optimal), the fourth-order Runge-Kutta method (RK4_Kutta's) and the classic fourth-order Runge-Kutta method (RK4_Classic) are shown in Table 4 and Table 5.

Table 4 Comparison Between the Exact Solution and the Numerical Solution of $\Phi(x)$ Taking $h = 0.1$ for Test Example 1 Using the RK3 Kutta's and RK3 Optimal Methods

x	Exact Solution	Approximate Solution of $\Phi(x)$ by	
		RK3 Kutta's	RK3 Optimal
0.0	1.000000000000000	1.000000000000000	1.000000000000000
0.1	1.010000000000000	1.010000000000000	1.010000000000000
0.2	1.040000000000000	1.040000000000000	1.040000000000000
0.3	1.090000000000000	1.090000000000000	1.090000000000000
0.4	1.160000000000000	1.160000000000000	1.160000000000000
0.5	1.250000000000000	1.250000000000000	1.250000000000000
0.6	1.360000000000000	1.360000000000000	1.360000000000000
0.7	1.490000000000000	1.490000000000000	1.490000000000000
0.8	1.640000000000000	1.640000000000000	1.640000000000000
0.9	1.810000000000000	1.809999999999999	1.810000000000000
1.0	2.000000000000000	2.000000000000000	2.000000000000000
L.S.E.		**7.39557098644698e-031**	**4.93038065763132e-031**

Table 5 Comparison Between the Exact Solution and the Numerical Solution of $\Phi(x)$ Taking $h = 0.1$ for Test Example 1 Using the RK4_Classic and RK4_Kutta's Methods

x	Exact Solution	Approximate Solution of $\Phi(x)$ by	
		RK4_Classic	RK4_Kutta's
0.0	1.000000000000000	1.000000000000000	1.000000000000000
0.1	1.010000000000000	1.010000000000000	1.010000000000000
0.2	1.040000000000000	1.040000000000000	1.040000000000000
0.3	1.090000000000000	1.090000000000000	1.090000000000000
0.4	1.160000000000000	1.160000000000000	1.160000000000000
0.5	1.250000000000000	1.250000000000000	1.250000000000000
0.6	1.360000000000000	1.360000000000000	1.360000000000000
0.7	1.490000000000000	1.490000000000000	1.490000000000000
0.8	1.640000000000000	1.640000000000000	1.640000000000000
0.9	1.810000000000000	1.810000000000000	1.810000000000000
1.0	2.000000000000000	2.000000000000000	2.000000000000000
L.S.E.		**4.9303806576313e-031**	**9.8607613152626e-032**

Test Example 2:

Consider the non-linear VIE:

$$\Phi(x) = \cos(x) - \sin(x) - \frac{1}{4}\sin(2x) + \frac{1}{2}x - \frac{1}{2}x^2 + \int_0^x (x-t)\Phi^2(t)dt$$

with the exact solution [53]:

$$\Phi(x) = \cos(x) - \sin(x)$$

The results for the Test Example 2 using the classic third-order Runge-Kutta method (RK3 Kutta's), the Optimal third-order Runge-Kutta method (RK3 Optimal), the fourth-order Runge-Kutta method (RK4_Kutta's) and the classic fourth-order Runge-Kutta method (RK4_Classic) are shown in Table 6 and Table 7.

Table 6 Comparison Between the Exact Solution and the Numerical Solution of $\Phi(x)$ Taking $h = 0.1$ for Test Example 2 Using the RK3 Kutta's and RK3 Optimal Methods

x	Exact Solution	Approximate Solution of $\Phi(x)$ by	
		RK3 Kutta's	RK3 Optimal
0.0	1.000000000000000	1.000000000000000	1.000000000000000
0.1	0.895170748631198	0.895178813279648	0.895173065601988
0.2	0.781397247046180	0.781410679432610	0.781402003566974
0.3	0.659816282464266	0.659832906296005	0.659823669221341
0.4	0.531642651694235	0.531660837416181	0.531652913811702
0.5	0.398157023286170	0.398175673950091	0.398170436655657
0.6	0.260693141514643	0.260711633831809	0.260709978889317
0.7	0.120624500046797	0.120642580743281	0.120644987890706
0.8	-0.020649381552357	-0.020631738702726	-0.020625113113437
0.9	-0.161716941356819	-0.161699707084951	-0.161688911783665
1.0	-0.301168678939757	-0.301151952990972	-0.301137108839271
L.S.E.		**2.757312101152e-009**	**3.4423208371321e-009**

Table 7 Comparison Between the Exact Solution and the Numerical Solution of $\Phi(x)$ Taking $h = 0.1$ for Test Example 2 Using the RK4_Classic and RK4_Kutta's Methods

x	Exact Solution	Approximate Solution of $\Phi(x)$ by	
		RK4_Classic	RK4_Kutta's
0.0	1.000000000000000	1.000000000000000	1.000000000000000
0.1	0.895170748631198	0.895170906506840	0.895170580141165
0.2	0.781397247046180	0.781397535731821	0.781396900415589
0.3	0.659816282464266	0.659816685392011	0.659815754539819
0.4	0.531642651694235	0.531643161234677	0.531641943671594
0.5	0.398157023286170	0.398157639457795	0.398156138602344
0.6	0.260693141514643	0.260693870955257	0.260692083995984
0.7	0.120624500046797	0.120625355148306	0.120623272525844
0.8	-0.020649381552357	-0.020648383536493	-0.020650777962576
0.9	-0.161716941356819	-0.161715779492503	-0.161718507191050
1.0	-0.301168678939757	-0.301167330370196	-0.301170415442933
L.S.E.		**6.33780023356121e-012**	**1.175360243510608e-012**

Test Example 3:

Consider the non-linear VIE:

$$\Phi(x) = e^x - \frac{1}{9}e^{3x} + \frac{1}{9} + \frac{1}{3}x + \int_0^x (x-t)\Phi^3(t)dt$$

with the exact solution [53]:

$$\Phi(x) = e^x$$

The results for the Test Example 3 using the classic third-order Runge-Kutta method (RK3 Kutta's), the optimal third-order Runge-Kutta method (RK3 Optimal), the fourth-order Runge-Kutta method (RK4_Kutta's) and the classic fourth-order Runge-Kutta method (RK4_Classic) are shown in Table 8 and Table 9.

Table 8 Comparison Between the Exact Solution and the Numerical Solution of $\Phi(x)$ Taking $h = 0.1$ for Test Example 3 Using the RK3 Kutta's and RK3 Optimal Methods

x	Exact Solution	Approximate Solution of $\Phi(x)$ by	
		RK3 Kutta's	RK3 Optimal
0.0	1.000000000000000	1.000000000000000	1.000000000000000
0.1	1.105170918075648	1.105183610018104	1.105184399432768
0.2	1.221402758160170	1.221439406458402	1.221434561704785
0.3	1.349858807576003	1.349938964011672	1.349916826608440
0.4	1.491824697641270	1.491982470545614	1.491921863885297
0.5	1.648721270700128	1.649016579684664	1.648879008354339
0.6	1.822118800390509	1.822657897568498	1.822373183338301
0.7	2.013752707470477	2.014726307637915	2.014165729138771
0.8	2.225540928492468	2.227296683009181	2.226221729754752
0.9	2.459603111156950	2.462787128199351	2.460749241955830
1.0	2.718281828459046	2.724123987225707	2.720262287140248
L.S.E.		**4.8710009480530e-005**	**5.97350220452353e-006**

Table 9 Comparison Between the Exact Solution and the Numerical Solution of $\Phi(x)$ Taking $h = 0.1$ for Test Example 3 Using the RK4_Classic and RK4_Kutta's Methods

x	Exact Solution	Approximate Solution of $\Phi(x)$ by	
		RK4_Classic	RK4_Kutta's
0.0	1.000000000000000	1.000000000000000	1.000000000000000
0.1	1.105170918075648	1.105169797081169	1.105171136451842
0.2	1.221402758160170	1.221400042233600	1.221403416994237
0.3	1.349858807576003	1.349853669392728	1.349860310160499
0.4	1.491824697641270	1.491815685448746	1.491827774165584
0.5	1.648721270700128	1.648705805925621	1.648727245008598
0.6	1.822118800390509	1.822092224854712	1.822130085143481
0.7	2.013752707470477	2.013706432539967	2.013773738704628
0.8	2.225540928492468	2.225458747002958	2.225579964904265
0.9	2.459603111156950	2.459453644710708	2.459675780341661
1.0	2.718281828459046	2.718002593522533	2.718418310815509
L.S.E.		**1.102692065138610e-007**	**2.604964081210461e-008**

CHAPTER 5

SPLINE FUNCTIONS

5.1 Spline Interpolation

A spline is a function composed of simple functions glued together. Splines are used to approximate complex functions and shapes. Because it is composed of different functions, a spline is different from a polynomial interpolation, which consists of a single well-defined function that approximates a given shape; mathematical splines are normally piecewise polynomial functions where the polynomial pieces correspond to the interval between the points that hold the physical spline fixed. A set of knots defines the intervals [60].

5.2 First and Second Degree Splines

Splines make use of partitions, which lead to an interval being cut into a number of subintervals.

Definition 5.1: Partition.

The interval [a,b] is partitioned into an ordered sequence $\{x_i\}_{i=0}^{n}$ such that

$$a = x_0 < x_1 < \cdots < x_{n-1} < x_n = b$$

The numbers x_i are called "knots".

A linear spline is a spline of degree 1, and is a function that is linear on each subinterval determined by a partition.

Definition 5.2: Linear Splines.

A function L is a spline of degree 1 on $[a, b]$ if

1. The domain of S is $[a, b]$.
2. S is continuous on $[a, b]$.

3. There is a partition $\{x_i\}_{i=0}^{n}$ of $[a, b]$ such that on each $[x_i, x_{i+1}], L$ is a linear polynomial.

A linear spline is defined completely by its values at the knots. For example, with the following:

x	x_0	x_1	...	x_n
y	y_0	y_1	...	y_n

there is only one linear spline with these values at the knots and it is on each given subinterval.

For a spline with this data, the linear polynomial on each subinterval is in the form:

$$L_i(x) = y_i + \frac{y_{i+1} - y_i}{x_{i+1} - x_i}(x - x_i). \tag{5.1}$$

Note that if $x \in [x_i, x_{i+1}]$, then $x - x_i > 0$, but $x - x_{i-1} \leq 0$. Therefore if we want to evaluate $L(x)$, we seek the largest i so that $x - x_i > 0$, then evaluate $L_i(x)$.

5.2.1 First Degree Spline Accuracy

In the same way as for polynomial functions, splines are also used to interpolate tabulated data as well as functions. In the latter case this means that if the spline is being used to interpolate the function f, we can say that this is equivalent to interpolating the data:

x	x_0	x_1	...	x_n
y	$f(x_0)$	$f(x_1)$	...	$f(x_n)$

We will consider how to find a bound for the error on a single interval of the partition, with the use of a little calculus. Assume $p(x)$ is the linear polynomial

interpolating $f(x)$ at the endpoints of the subinterval $[x_i, x_{i+1}]$, then for $x \in [x_i, x_{i+1}]$,

$$|f(x) - p(x)| \leq max\{|f(x) - f(x_i), |f(x) - f(x_{i+1})\}.$$

That is, $|f(x) - p(x)|$ is no larger than the "maximum variation" of $f(x)$ on this interval.

In particular, if $f'(x)$ exists and is bounded by M_1 on $[x_i, x_{i+1}]$, then

$$|f(x) - p(x)| \leq \frac{M_1}{2}(x_{i+1} - x_i).$$

Similarly, if the second derivative $f''(x)$ is exists and bounded by M_2 on the subinterval $[x_i, x_{i+1}]$, then

$$|f(x) - p(x)| \leq \frac{M_2}{8}(x_{i+1} - x_i)^2.$$

5.2.2 Second Degree Splines

We can similarly define piecewise quadratic splines, or splines of degree 2.

Definition 5.4: Quadratic Splines.

A function Q is a quadratic spline on $[a, b]$ if

1. The domain of Q is $[a, b]$.
2. Q is continuous on $[a, b]$.
3. Q_0 is continuous on (a, b).
4. There is a partition $\{x_i\}_{i=0}^{n}$ of $[a, b]$ such that on $[x_i, x_{i+1}], Q$ is a polynomial of degree at most 2.

5.2.3 Computing Second Degree Splines

Assume the data

x	x_0	x_1	...	x_n
y	y_0	y_1	...	y_n

are given. Let $z_i = Q_i'(x_i)$, and assume that the additional condition to define the quadratic spline is given by identifying z_0. We will seek to compute the form of $Q_i(x)$.

Because $Q_i(x_i) = y_i, Q_i'(x_i) = z_i, Q_i'(x_{i+1}) = z_{i+1}$, we see that we can define

$$Q_i(x) = \frac{z_{i+1} - z_i}{2(x_{i+1} - x_i)}(x - x_i)^2 + z_i(x - x_i) + y_i. \tag{5.2}$$

Use this at x_{i+1}:

$$y_{i+1} = Q_i(x_{i+1}) = \frac{z_{i+1} - z_i}{2(x_{i+1} - x_i)}(x_{i+1} - x_i)^2 + z_i(x_{i+1} - x_i) + y_i,$$

$$y_{i+1} - y_i = \frac{z_{i+1} + z_i}{2}(x_{i+1} - x_i) + z_i(x_{i+1} - x_i),$$

Thus we can determine, from the data alone, z_{i+1} from z_i:

$$z_{i+1} = 2\frac{y_{i+1} - y_i}{x_{i+1} - x_i} - z_i. \tag{5.3}$$

5.3 Natural Cubic Splines

We can expand the definition to a spline of degree k, if we remember the definitions of linear and quadratic splines.

Definition 5.6: Splines of degree k.

A function S is a spline of degree k on $[a, b]$ if

1. The domain of S is $[a, b]$.
2. $S, S', S'', \ldots, S^{(k-1)}$ are continuous on (a, b).
3. There is a partition $\{x_i\}_{i=0}^n$ of $[a, b]$ such that on $[x_i, x_{i+1}]$, S is a polynomial of degree $\leq k$.

We would expect that a spline of degree k has $k - 1$ degrees of freedom, as we show here. A spline of degree k is defined by $n(k + 1)$ parameters if the partition has $n + 1$ knots. The following given data:

x	x_0	x_1	...	x_n
y	y_0	y_1	...	y_n

provide $2n$ equations. The continuity of $S, S', S'', \ldots, S^{(k-1)}$ at the $n-1$ internal knots gives $(k-1)(n-1)$ equations. This is a total of $n(k+1)-(k-1)$ equations. Consequently, we get $k-1$ more unknowns than equations. Thus, except for some singularity, and we must add $k-1$ constraints for a unique definition of the spline. These are the degrees of freedom. Usually k is selected to be equal to 3. These are cubic splines. We must add two extra constraints to define the spline. The normal or usual choice is to make:

$$S''(x_0) = S''(x_n) = 0. \tag{5.4}$$

This yields the natural cubic spline.

5.3.1 Why Natural Cubic Splines?

We show that natural cubic splines are an excellent choice because they are the interpolant of the minimal H^2 semi norm. The natural result following this theorem states this in more readily understandable terms.

Theorem 5.7:

Assume the function f has two continuous derivatives, and S is the natural cubic spline interpolating f at the knots $a = x_0 < x_1 < \cdots < x_n = b$. Then:

$$\int_a^b [S''(x)]^2\,dx \le \int_a^b [f''(x)]^2\,dx$$

Proof: We let $\mathrm{g}(x) = f(x) - S(x)$. Then $\mathrm{g}(x)$ is zero on the $(n+1)$ knots x_i. The derivatives are linear, meaning that

$$f''(x) = S''(x) + \mathrm{g}''(x).$$

Then

$$\int_a^b [f''(x)]^2\,dx = \int_a^b [S''(x)]^2\,dx + \int_a^b [\mathrm{g}''(x)]^2\,dx + \int_a^b 2S''(x)\mathrm{g}''(x)\,dx.$$

Moreover, the last integral is zero. Integrating by parts we obtain

$$\int_a^b 2S''(x)\mathrm{g}''(x)\,dx = S''\mathrm{g}'|_a^b - \int_a^b S'''\mathrm{g}'dx = -\int_a^b S'''\mathrm{g}'dx,$$

because $S''(a) = S''(b) = 0$. Then note that S is a polynomial of degree ≤ 3 on each interval, thus $S'''(x)$ is a piecewise constant function that takes the value c_i on each interval $[x_i, x_{i+1}]$. Thus

$$\int_a^b S'''\mathrm{g}'dx = \sum_{i=0}^{n-1}\int_a^b c_i\mathrm{g}'dx = \sum_{i=0}^{n-1} c_i\mathrm{g}\,|_{x_i}^{x_{i+1}} = 0,$$

with the last equality following because $\mathrm{g}(x)$ is equal to zero at the knots.

Corollary 5.8: The natural cubic spline is the best twice-continuously differentiable interpolant for a twice-continuously differentiable function, under the measure given by the theorem.

Proof: Assume f to be twice-continuously differentiable and assume S to be the natural cubic spline interpolating $f(x)$ at some given nodes $\{x_i\}_{i=0}^n$. Also assume that $R(x)$ is some twice-continuously differentiable function that interpolates $f(x)$ at these nodes. This leads us to the fact that $S(x)$ interpolates $R(x)$ at these nodes. We apply the theorem to obtain

$$\int_a^b [S''(x)]^2\,dx \leq \int_a^b [R''(x)]^2\,dx$$

5.4 B-Splines

The B-splines form a basis for spline functions, and take their name from this. We presuppose the existence of an infinite number of knots.

$$\ldots < x_2 < x_1 < x_0 < x_1 < x_2 < \cdots,$$

with

$$\lim_{k\to-\infty} x_k = -\infty$$

and

$$\lim_{k\to\infty} x_k = \infty.$$

The B-splines of degree 0 are defined as single blocks.

$$B_i^0 = \begin{cases} 1 & x_i \le x \le x_{i+1} \\ 0 & otherwise \end{cases} \tag{5.5}$$

The zero degree B-splines are continuous from the right side, and they are nonzero only on one subinterval $[x_i, x_{i+1})$; they add up to 1.

We want justify the description of B-splines as basis splines. If S is a spline of degree 0 on the given knots and is continuous from the right side then:

$$S(x) = \sum_i S(x_i) B_i^0(x). \tag{5.6}$$

As can be seen, the basis splines work in the same way that Lagrange polynomials worked for polynomial interpolation.

The B-splines of degree k are defined recursively:

$$B_i^k(x) = \left(\frac{x - x_i}{x_{i+k} - x_i}\right) B_i^{k-1}(x) + \left(\frac{x_{i+k+1} - x}{x_{i+k+1} - x_{i+1}}\right) B_{i+1}^{k-1}(x). \tag{5.7}$$

The B-splines speedily become unwieldy. We choose the case when $k = 1$. The B-spline $B_i^1(x)$ is

- Piecewise linear.
- Continuous.
- Nonzero only on (x_i, x_{i+2}).
- 1 at x_{i+1}.

These B-splines are sometimes called hat functions. Imagine wearing a hat shaped like this!

The good thing about the hat functions is that they can be used through an analogy. The hat functions play a similar role to polynomial interpolation and the Lagrange functions because:

$$B_i^1(x_j) = \begin{cases} 1 & (i+1) = j \\ 0 & (i+1) \neq j \end{cases} \tag{5.8}$$

Then if we want to inset data such as the following with splines of degree 1,

x	x_0	x_1	...	x_n
y	y_0	y_1	...	y_n

we can directly set

$$S(x) = \sum_{i=0}^{n} y_i B_{i-1}^1(x). \tag{5.9}$$

5.5 Solution of Non-Linear VIEs Using Classic Spline Functions

In this section, we use classic spline functions (quadratic and cubic spline functions) to seek and find the numerical solutions of non-linear VIEs of the form:

$$\Phi(x) = f(x) + \int_0^x k\left(x, t, \Phi(t)\right)dt \qquad ; x \in [a, b] \tag{5.10}$$

where f is assumed to be continuous on x, and k denotes the given continuous function.

5.5.1 Using the Linear Classic Spline Function $L(t)$

A linear interpolation $L(t)$ with the knots $t_0, t_1, \dots, t_n$ in the interval $[t_i, t_{i+1}]$ is given by the formula:

$$L(t) = A_i(t)L_i + B_i(t)L_{i-1} \tag{5.11}$$

where

$A_i(t) = \frac{t_{i+1}-t}{h}$ and $B_i(t) = \frac{t-t_i}{h}$

Substituting Eq. (5.11) into Eq. (5.10) gives

$L_i; i = 1,2, \dots, n$ with $t = t_r$ and $r = 1,2, \dots, n$, so we get:

$$L_r = f_r + \sum_{z=0}^{r-1} \int_{x_z}^{x_{z+1}} k\,(x, t, [A_z(t)L_z + B_z(t)L_{z+1}])dt\,;$$

$$r = 1,2,\ldots,n. \tag{5.12}$$

where $L_r = L(t_r)$ and $f_r = f(t_r)$, and the iterated integrals in Eq. (5.12) are calculated using the trapezoidal rule or the Runge-Kutta method.

The following algorithm can be used to solve non-linear VIEs using a linear classic spline function:

Algorithm for non-linear VIEs using a linear classic spline function (non-linear VIELSP):

Step (1):

a- Assume $h = \frac{b}{n}, n \in N$.

b- Set $L_0 = f_0$.

Step (2):

To compute L_r use step 1 and put $r = 1$ in Eq. (5.12), and then use the non-linear VIETRP algorithm.

Step (3):

As in step 2, and using Eq. (5.12), compute $L_r; r = 2,3, \ldots, n$.

5.5.2 Using the Quadratic Classic Spline Function $Q(t)$

A quadratic classic spline $Q(t)$ with the knots t_0, t_1, t_n in the interval $[t_i, t_{i+1}]$ can be written as:

$$Q(t) = A_i(t)Q_i + B_i(t)Q_{i+1} + D_i(t)\frac{dQ_i}{dt} \tag{5.13}$$

where

$A_i(t) = 1 - \left(\frac{t-t_i}{h}\right)^2, B_i(t) = 1 - A_i(t)$ and $D_i(t) = \frac{(t-t_i)(t_{i-1}-t)}{h}$

Putting Eq. (5.13) into Eq. (5.10) gives

$Q_i; i = 1,2, \ldots, n$, with $t = t_r$ and $r = 1,2,..,n$, we get:

$$Q_r = f_r + \sum_{z=0}^{r-2} \int_{x_z}^{x_{z-1}} K\left(x_r, t, \left[A_z(t)Q_z + B_z(t)Q_{z+1} + D_z(t)\frac{dQ_z}{dt}\right]\right) dt$$

$$+ \int_{x_{z-1}}^{x_z} K\left(x_r, t, \left[A_{r-1}(t)Q_{z-1} + B_{r-1}(t)Q_z + D_r(t)\frac{dQ_z}{dt}\right]\right) dt \qquad (5.14)$$

where $Q_r = Q(t_r)$ and $f_r = f(t_r)$

Now, for $r = 1$ we need to calculate $\frac{dQ_0}{dt}$. We can find this value by differentiating Eq. (5.10) once with respect to x, from which we get:

$$Q'(x) = f'(x) + \int_0^x \frac{dK(x,t,Q(t))}{dx} dt + K(x,x,Q(x))$$

Putting $t = a$, we obtain:

$$Q_0' = f_0'(x) + K(a,a,Q_0) \qquad (5.15)$$

But, for $r = 2,3,\ldots,n$ we calculate $\frac{dQ_r}{dt}$ from the equation below:

$$\frac{dQ_r}{dt} = \frac{dQ_{r-1}}{dt} + \frac{2(Q_r - Q_{r-1})}{h} \qquad (5.16)$$

The following algorithm can be used to solve a system of non-linear VIEs using a quadratic classic spline function:

Algorithm for non-linear VIEs using a quadratic classic spline function and the trapezoidal rule (non-linear VIEQSP1):

Step (1):

a- Assume $h = \frac{b}{n}, n \in N$.

b- Set $Q_0 = f_0$.

Step (2):

a- To calculate $\frac{dQ_0}{dt}$, use step 1 with Eq. (5.15).

b- To compute Q_1 use steps 1 and 2a, put $r = 1$ in Eq. (5.14), and use the non-linear VIETRP algorithm.

Step (3):

a- Use steps 1 and 2 to find $\frac{dQ_1}{dt}$, putting $r = 1$ in Eq. (5.16).

b- Put $r = 2$ in Eq. (5.14), find Q_2, and use the non-linear VIETRP algorithm.

Step (4):

In the same way as in step 3, and by using Eq. (5.16) and also Eq. (4.14), compute $\frac{dQ_2}{dt}, Q_3 \frac{dQ_3}{dt}, Q_4$, ..., and so on.

Algorithm for non-linear VIEs using a quadratic classic spline function and the classic fourth-order Runge-Kutta method (non-linear VIEQSP2):

Step (1):

a- Assume $h = \frac{b}{n}, n \in N$.

b- Set $Q_0 = f_0$.

Step (2):

a- To calculate $\frac{dQ_0}{dt}$, use step 1 with Eq. (5.15).

b- To compute Q_1 use steps 1 and 2a, put $r = 1$ in Eq. (5.14), and use the RK4_Classic algorithm.

Step (3):

a- Use steps 1 and 2 to find $\frac{dQ_1}{dt}$, putting $r = 1$ in Eq. (5.16).

b- Put $r = 2$ in Eq. (5.14) to find Q_2, and use the RK4_Classic algorithm.

Step (4):

In the same way as in step 3, and by using Eq. (5.16) and also Eq. (4.14), compute $\frac{dQ_2}{dt}, Q_3 \frac{dQ_3}{dt}, Q_4$, ..., and so on.

5.5.3 Using the Cubic Classic Spline Function $S(t)$

A cubic classic spline $S(t)$ with the knots $t_0, t_1, \dots, t_n$ in the interval $[t_i, t_{i+1}]$ can be written as:

$$S(t) = A_i(t)S_i + B_i(t)S_{i+1} + C_i(t)\frac{dS_i}{dt} + D_i(t)\frac{dS_{i+1}}{dt} \tag{5.17}$$

$$A_i(t) = 1 - 3\left(\frac{t-t_i}{h}\right)^2 + 2\left(\frac{t-t_i}{h}\right)^3, B_i(t) = 1 - A_i(t)$$

where

$$C_i(t) = (t-t_i)\left(\frac{t-t_{i+1}}{h}\right)^2 \text{ and } D_i(t) = (t-t_i)\left(\frac{t-t_{i+1}}{h}\right)^2$$

Substituting Eq. (5.17) in Eq. (5.10) gives

$S_i; i = 1,2, \dots, n$, with $t = t_r$ and $r = 1,2,..,n$, we get:

$$S_r = f_r + \sum_{z=0}^{r-2} \int_{x_z}^{x_{z-1}} K\left(x_r, t, \begin{bmatrix} A_z(t)S_z + B_z(t)S_{z+1} \\ +C_z(t)\frac{dS_z}{dt} + D_z(t)\frac{dS_{z+1}}{dt} \end{bmatrix}\right) dt$$
$$+ \int_{x_{z-1}}^{x_z} K\left(x_r, t, \begin{bmatrix} A_{r-1}(t)S_{z-1} + B_{r-1}(t)S_z \\ +C_r(t)\frac{dS_{z-1}}{dt} + D_r(t)\frac{dS_z}{dt} \end{bmatrix}\right) dt \tag{5.18}$$

where

$S_r = S(t_r)$ and $f_r = f(t_r)$

We calculate $\frac{dS_r}{dt}$ by the equation as:

$$\frac{dS_{r-1}}{dt} = -\frac{dS_{r-1}}{dt} - 4\frac{dS_r}{dt} + \frac{3(S_{r-2} - S_{r-1})}{h} \tag{5.19}$$

The following algorithm can be used to solve a system of non-linear VIEs using a cubic spline function.

Algorithm for non-linear VIEs using a cubic classic spline function and the trapezoidal rule (non-linear VIECSP1):

Step (1):

a- Assume $h = \frac{b}{n}, n \in N$.

b- Set $S_0 = f_0$.

Step (2):

a- To calculate $\frac{dS_0}{dt}$, use step 1 with Eq. (5.15).

b- To compute S_1 use steps 1 and 2-a, put $r = 1$ in Eq. (5.18), and use the non-linear VIETRP algorithm.

Step (3):

a- Use steps 1 and 2 to find $\frac{dS_1}{dt}$ by putting $r = 2$ in Eq. (5.19).

b- Put $r = 2$ in Eq. (5.19) to find S_2, and use the non-linear VIETRP algorithm.

Step (4):

In the same way as in step 3, and by using Eq. (5.18) and Eq. (5.19). compute $\frac{dS_2}{dt}, S_3, \frac{dS_3}{dt} S_4$, ..., and so on.

Algorithm for non-linear VIEs using the cubic classic spline function and the classic fourth-order Runge-Kutta Method (non-linear VIECSP2):

Step (1):

a- Assume $h = \frac{b}{n}, n \in N$.

b- Set $S_0 = f_0$.

Step (2):

a- To calculate $\frac{dS_0}{dt}$, use step 1 with Eq. (5.15).

b- To compute S_1 use steps 1 and 2-a, put $r = 1$ in Eq. (5.18), and use the RK4_Classic algorithm.

Step (3):

a- Use steps 1 and 2 to find $\frac{dS_1}{dt}$, by putting $r = 2$ in Eq. (5.19).

b- Put $r = 2$ in Eq. (5.19) to find S_2, and use the RK4_Classic algorithm.

Step (4):

In the same way as in step 3, and by using Eq. (5.18) and Eq. (5.19), compute $\frac{dS_2}{dt}, S_3, \frac{dS_3}{dt} S_4$, ..., and so on.

5.6 Solution of Non-Linear VIEs Using B-Spline Functions

In this section, we use first-, second-, third- and fourth-order B-spline functions to seek and find the numerical solution of non-linear VIEs in the form:

$$\Phi(x) = f(x) + \int_0^x k\left(x, t, \Phi(t)\right)dt \ ; x \in I \in [a, b] \tag{5.20}$$

where f is assumed to be continuous on I, and k denotes a given continuous function. First, we derive a new formula that is essential for our work.

Theorem 5.1: (Fundamental theorem).

This is the generalized form of the basic theorem in integral calculus, Leibnitz's generalized formula [46].

$$\frac{d}{dx}\int_{\alpha(x)}^{\beta(x)} F(x,y)dy = \int_{\alpha(x)}^{\beta(x)} \frac{\partial F(x,y)}{\partial x}dy + F\big(x,\beta(x)\big)\frac{d\beta(x)}{dx} - F\big(x,\alpha(x)\big)\frac{d\alpha(x)}{dx} \tag{5.21}$$

Remark 5.1:

1- Leibnitz's generalized formula can be written as follows:

$$\frac{d}{dx}\int_{\alpha(x)}^{\beta(x)} F\big(x,y,\Phi(x)\big)dy = \int_{\alpha(x)}^{\beta(x)} \frac{\partial F\big(x,y,\Phi(x)\big)}{\partial x}dy + F\left(x,\beta(x),\Phi\big(\beta(x)\big)\right)\frac{d\beta(x)}{dx} - F\left(x,\alpha(x),\Phi\big(\beta(x)\big)\right)\frac{d\alpha(x)}{dx}$$

2- if $\alpha(x) = 0$ and $\beta(x) = x$ then Leibnitz's generalized formula can be written as:

$$\frac{d}{dx}\int_0^x F\big(x,y,\Phi(y)\big)dy = \int_0^x F\big(x,y,\Phi(y)\big) + F\big(x,x,\Phi(x)\big) \qquad (5.22)$$

Theorem 5.2:

For all $n \geq 0$

$$\frac{d^n}{dx^n}\int_0^x k\big(x,t,\Phi(t)\big)dt = \int_0^x \frac{d^n}{dx^n}k\big(x,t,\Phi(t)\big)dt + \sum_{j=1}^{n-1}\frac{d^{j-1}}{dx^{j-1}}\left(\frac{d^{n-j}}{dx^{n-j}}k\big(x,t,\Phi(t)\big)|_{t=x}\right) + \frac{d^{n-1}}{dx^{n-1}}k\big(x,x,\Phi(x)\big) \qquad (5.23)$$

Proof:

Mathematical induction is used to prove this theorem.

In order to establish the validity of this theorem, the following steps are needed:

(i) prove that the theorem is true for $n = 1$,

i.e.

$$\frac{d}{dx}\int_0^x k\big(x,t,\Phi(t)\big)dt = \int_0^x \frac{d}{dx}k\big(x,t,\Phi(t)\big)dt + k\big(x,x,\Phi(x)\big)$$

This has already been shown in Remark (5.1) (2).

(ii) For fixed $n = k$, assume Eq. (5.23) is true, i.e.,

$$\frac{d^k}{dx^k}\int_0^x k\big(x,t,\Phi(t)\big)dt = \left\{\begin{array}{l}\displaystyle\int_0^x \frac{d^k}{dx^k}k\big(x,t,\Phi(t)\big)dt \\ \displaystyle+\sum_{j=1}^{k-1}\frac{d^{j-1}}{dx^{j-1}}\left(\frac{d^{k-j}}{dx^{k-j}}k\big(x,t,\Phi(t)\big)|_{t=x}\right) \\ \displaystyle+\frac{d^{k-1}}{dx^{k-1}}k\big(x,x,\Phi(x)\big)\end{array}\right\} \qquad (5.24)$$

Then we need to prove that Eq. (5.23) is true for $n = k + 1$, that is we want to prove that:

$$\frac{d^{k+1}}{dx^{k+1}}\int_0^x k\big(x,t,\Phi(t)\big)dt=\left\{\begin{array}{l}\displaystyle\int_0^x \frac{d^{k+1}}{dx^{k+1}}k\big(x,t,\Phi(t)\big)dt\\ \displaystyle+\sum_{j=1}^{n-1}\frac{d^{j-1}}{dx^{j-1}}\left(\frac{d^{k+1-j}}{dx^{k+1-j}}k\big(x,t,\Phi(t)\big)|_{t=x}\right)\\ \displaystyle+\frac{d^k}{dx^k}k\big(x,x,\Phi(x)\big)\end{array}\right\}$$

Differentiating Eq. (5.24) with respect to x yields:

$$\frac{d}{dx}\frac{d^k}{dx^k}\int_0^x k\big(x,t,\Phi(t)\big)dt=\left\{\begin{array}{l}\displaystyle\frac{d}{dx}\int_0^x \frac{d^k}{dx^k}k\big(x,t,\Phi(t)\big)dt\\ \displaystyle+\frac{d}{dx}\sum_{j=1}^{n-1}\frac{d^{j-1}}{dx^{j-1}}\left(\frac{d^{k-j}}{dx^{k-j}}k\big(x,t,\Phi(t)\big)|_{t=x}\right)\\ \displaystyle+\frac{d}{dx}\frac{d^k}{dx^k}k\big(x,x,\Phi(x)\big)\end{array}\right\}$$

With the aid of Remark (5.1) (2), the following is obtained:

$$\frac{d^{k+1}}{dx^{k+1}}\int_0^x k\big(x,t,\Phi(t)\big)dt=\left\{\begin{array}{l}\displaystyle\int_0^x \frac{d^{k+1}}{dx^{k+1}}k\big(x,t,\Phi(t)\big)dt+\frac{d^k}{dx^k}k\big(x,t,\Phi(t)\big)|_{t=x}\\ \displaystyle+\sum_{j=1}^{n-1}\frac{d^{j-1}}{dx^{j-1}}\left(\frac{d^{k-j}}{dx^{k-j}}k\big(x,t,\Phi(t)\big)|_{j=x}\right)\\ \displaystyle+\frac{d^k}{dx^k}k\big(x,x,\Phi(x)\big)\end{array}\right\}$$

Therefore,

$$\frac{d^{k+1}}{dx^{k+1}}\int_0^x k\big(x,t,\Phi(t)\big)dt=\left\{\begin{array}{l}\displaystyle\int_0^x \frac{d^{k+1}}{dx^{k+1}}k\big(x,t,\Phi(t)\big)dt\\ \displaystyle+\sum_{j=1}^{n-1}\frac{d^{j-1}}{dx^{j-1}}\left(\frac{d^{k-j}}{dx^{k-j}}k\big(x,t,\Phi(t)\big)|_{t=x}\right)\\ \displaystyle+\frac{d^k}{dx^k}k\big(x,x,\Phi(x)\big)\end{array}\right\}$$

Hence Eq. (5.23) is true for $n=k+1$, so it is valid for all n.

If we differentiate Eq. (5.20) n times with respect to x, using Eq. (5.23) we have:

$$\frac{d^n}{dx^n}\Phi(x) = \frac{d^n}{dx^n}f(x) + \left\{ \begin{array}{l} \int_0^x \frac{d^n}{dx^n}k\big(x,t,\Phi(t)\big)dt \\ +\sum_{j=1}^{n-1}\frac{d^{j-1}}{dx^{j-1}}\left(\frac{d^{n-j}}{dx^{n-j}}k\big(x,t,\Phi(t)\big)|_{t=x}\right) \\ +\left(\frac{d^{n-1}}{dx^{n-1}}k\big(x,x,\Phi(x)\big)\right) \end{array} \right\} \quad (5.25)$$

then putting $x = 0$ gives:

$$\Phi_0^{(n)} = f_0^{(n)} + \left\{ \begin{array}{l} \sum_{j=1}^{n-1}\frac{d^{j-1}}{dx^{j-1}}\left(\frac{d^{n-j}}{dx^{n-j}}k\big(x,t,\Phi(t)\big)|_{t=x}\right)|_{x=0} \\ +\frac{d^{n-1}}{dx^{n-1}}k\big(x,x,\Phi(x)\big)|_{x=0} \end{array} \right\} \quad (5.26)$$

$$\Phi_0' = f_0' + k\big(0,0,\Phi(0)\big) \quad (5.27)$$

Therefore, if we put $n = 2$, and substitute $x = 0$, we have:

$$\Phi_0'' = f_0'' + \left[\begin{array}{l} \left.\frac{\partial k\big(x,t,\Phi(t)\big)}{\partial x}\right|_{t=x=0} + \left[\left.\frac{\partial k\big(x,x,\Phi(x)\big)}{\partial x}\right|_{x=0}\right. \\ +k\big(0,0,\Phi(0)\big)\Phi'(0) \end{array} \right] \quad (5.28)$$

In a similar way, if we put $n = 3$ then the third derivative of Eq. (5.20), after setting $x = 0$, becomes:

$$\Phi_0''' = f_0''' + \left[\begin{array}{l} \left.\frac{\partial^2 k\big(x,t,\Phi(t)\big)}{\partial x^2}\right|_{t=x=0} + \frac{\partial}{\partial x}\left(\left[\left.\frac{\partial k\big(x,x,\Phi(t)\big)}{\partial x}\right|_{t=x}\right)|_{x=0}\right. \\ +\left(\frac{\partial^2}{\partial x^2}\Big(k\big(x,t,\Phi(t)\big)\Big)|_{t=x}\right)|_{x=0} + \frac{\partial}{\partial x^2}k\big(x,x,\Phi(x)\big) \end{array} \right] \quad (5.29)$$

Now, we treat Eq. (5.20) using first-, second-, third-, and fourth-order B-spline functions as follows:

5.6.1 Using a First-Order B-Spline Function $B^1(s)$

For $0 \leq s \leq 1$, the first-order B-spline function formula is given by:

$$B^1(s) = b_0(1-s) + b_1 s \quad ; 0 \leq s \leq 1$$

b_0 and b_1 are control points. Now, if we take $s = 0$ and $s = 1$, we get $b_0 = B^1(0)$ and $b_1 = B^1(1)$ respectively. Therefore, we have:

$$B^1(s) = B^1(0)(1-s) + B^1(1)s \quad ; 0 \leq s \leq 1 \tag{5.30}$$

The numerical solution of a system of non-linear VIEs using a first-order B-spline function is calculated by following the algorithm below:

Algorithm for non-linear VIE using a first-order B-spline function (non-linear VIEB1SP):

Step (1):

a- Put $h = (b-a) + n \quad ; n \in N$.

b- Set $b_0 = f_0$.

Step (2):

Use the non-linear VIETRP algorithm to get an initial value for b_1.

Step (3):

a- $s_r = a + rh \quad ; r = 0,1,2, \dots, n$.

b- Calculate $B_r^1 \quad ; r = 0,1, \dots, n$, by substituting s_r in Eq. (5.30) and using steps 1, 2, and 3-a.

5.6.2 Using a Second-Order B-Spline Function $B^2(s)$

For $0 \leq s \leq 1$, the second-order B-spline function formula is given by:

$$B^2(s) = b_0(1-s)^2 + 2b_1(1-s)s + b_2 s^2 \quad ; 0 \leq s \leq 1 \tag{5.31}$$

The control points b_0 and b_1 can be found by substituting $s = 0$ and $s = 1$ in $B^2(s)$; that is, $b_0 = B^2(0)$ and $b_2 = B^2(1)$ respectively.

Therefore, we have:

$$B^2(s) = b_0(0)(1-s)^2 + 2b_1(1-s)s + B^2(1)s^2 \quad ; 0 \le s \le 1$$

To evaluate b_1, we differentiate Eq. (5.31) once with respect to s, to get:

$$\frac{dB^2(s)}{ds} = -2b_0(1-s) + 2b_1[s(-1) + (1-s)] + 2b_2 s$$

Putting $s = 0$ gives:

$$\frac{dB^2(0)}{ds} = -2b_0 + 2b_1$$

This shows that:

$$b_1 = \frac{1}{2}\frac{dB^2(0)}{ds} + b_0 \tag{5.32}$$

The numerical solution of a system of non-linear VIEs using a second-order B-spline function can be calculated by following the algorithm below:

Algorithm for non-linear VIE using a second-order B-spline function (non-linear VIEB2SP):

Step (1):

a- Put $h = (b-a) + n \quad ; n \in N$.

b- Set $b_{j0} = f_{j0} \quad ; j = 1,2,\dots,m.$

Step (2):

Use the non-linear VIETRP algorithm to get an initial value for $b_{j1} \quad ; j = 1,2,\dots,m.$

Step (3):

a- Use step 1 and Eq. (5.27) to calculate $\frac{dB^2(0)}{ds} \quad ; j = 1,2,\dots,m.$

b- Compute $b_{j1}; j = 1,2,\dots,m$ using steps 1 and 3-a and Eq. (5.32).

Step (4):

a- $s_r = a + rh \quad ; r = 1,2,\dots,n.$

b- Calculate $B_{ir}^2 \quad ; i = 1,2,\dots,m\ , r = 0,1,\dots,n,$ by substituting s_r in Eq. (5.31) and following steps 1, 2, 3 and 4-a.

5.6.3 Using a Third-Order B-Spline Function $B^3(s)$

The third-order B-spline function formula can be written as:

$$B^3(s) = b_0(1-s)^3 + 3b_1(1-s)^2 s + 3b_2(1-s)s^2 + b_3 s^3; \quad (5.33)$$

where $0 \leq s \leq 1$ and b_0 and b_1 are control points. Now, if we take $s = 0$ and $s = 1$, we get $b_0 = B^3(0)$ and $b_3 = B^3(1)$ respectively.

Therefore, we have:

$$B^3(s) = B^3(0)(1-s)^3 + 3b_1(1-s)^2 s + 3b_2(1-s)s^2 + b_3 s^3 \quad ; 0 \leq s \leq 1$$

To obtain b_1, we differentiate Eq. (5.33) once with respect to s, to get:

$$\frac{dB^3(s)}{ds} = -3b_0(1-s)^2 + 3b_1[(1-s)^2 - 2(1-s)s] + 3b_2[2(1-s)s - s^2] + 3b_3 s^2$$

Therefore, putting $s = 0$ yields:

$$\frac{dB^3(0)}{ds} = -3b_0 + 3b_1$$

This shows that:

$$b_1 = \frac{1}{3}\frac{dB^3(0)}{ds} + b_0 \quad (5.34)$$

In the same way, to obtain b_2, we differentiate Eq. (5.33) twice with respect to s, and then set $s = 0$, so we have:

$$b_2 = \frac{1}{6}\frac{d^2B^3(0)}{ds^2} - b_0 + 2b_1 \quad (5.35)$$

The numerical solution for a system of non-linear VIEs using a third-order B-spline function is calculated using the following algorithm:

Algorithm for non-linear VIE using a third-order B-spline function (non-linear VIEB3SP):

Step (1):

a- Put $h = (b - a) + n \quad ; n \in N.$

b- Set $b_{j0} = f_{j0} \quad ; j = 1,2, \dots, m.$

Step (2):

Use the non-linear VIETRP algorithm to get an initial value for $b_{j3} \quad ; j = 1,2, \dots, m.$

Step (3):

a- Using step 1 and Eq. (5.27), calculate $\frac{dB_j^3(0)}{ds} \quad ; j = 1,2, \dots, m.$

b- Compute $b_{j1}; j = 1,2, \dots, m$ using steps 1 and 3-a and Eq. (5.34).

Step (4):

a- Use steps 1 and 3 and Eq. (5.28) to calculate $\frac{d^2B_j^3(0)}{ds^2} \quad ; j = 1,2, \dots, m.$

b- Compute $b_{j2}; j = 1,2, \dots, m$ using steps 1, 3, and 4-a and Eq. (5.35).

Step (5):

a- $s_r = a + rh \quad ; r = 1,2, \dots, n.$

b- Calculate $B_{ir}^2 \quad ; i = 1,2, \dots, m \, , r = 0,1, \dots, n,$ by substituting s_r in Eq. (5.33) with steps 1, 2, 3, 4 and 5-a.

5.6.4 Using a Fourth-Order B-Spline Function $B^4(s)$

The fourth-order B-spline function formula can be written as:

$$P(s) = b_0(1-s)^4 + 4b_1 s(1-s)^3 \quad + 6b_2 s^2(1-s)^2 + 4b_3 s^3(1-s) + b_4 s^4; \qquad 0 \le s \le 1 \tag{5.36}$$

Therefore, in a similar way to what we have done before, we have:

$b_0 = B^4(0)$ and $b_4 = B^4(1)$

$$, b_1 = \frac{1}{4}\frac{dB^4(0)}{ds} + b_0 \tag{5.37}$$

$$, b_2 = \frac{1}{12}\frac{d^2B^4(0)}{ds^2} - b_0 + 2b_1 \tag{5.38}$$

and

$$b_3 = \frac{1}{24}\frac{d^3B^4(0)}{ds^3} - 3b_1 + 3b_2 + b_0 \tag{5.39}$$

The numerical solution of a system of non-linear VIEs using a fourth-order B-spline function can be calculated using the following algorithm:

Algorithm for non-linear VIE using a fourth-order B-spline function (non-linear VIEB4SP):

Step (1):

a- Put $h = (b - a) + n \quad ; n \in N$.

b- Set $b_{j0} = f_{j0} \quad ; j = 1,2, \dots, m$.

Step (2):

Use the non-linear VIETRP algorithm to get an initial value for b_{j4} ; $j = 1,2, \dots, m$.

Step (3):

a- Using step 1 and Eq. (5.27), calculate $\frac{dB_j^4(0)}{ds} \quad ; j = 1,2, \dots, m$.

b- Compute $b_{j4} \quad ; j = 1,2, \dots, m$ using steps 1 and 3-a and Eq. (5.37).

Step (4):

a- Use steps 1 and 3 and Eq. (5.28) to calculate $\frac{dB_j^4(0)}{ds} \quad ; j = 1,2, \dots, m$.

b- Compute $b_{j2} \quad ; j = 1,2, \dots, m$ using steps 1, 3, and 4-a and Eq. (5.38).

Step (5):

a- Use steps 1, 3 and 4 and Eq. (5.29) to calculate $\frac{d^3B_j^4(0)}{ds^3} \quad ; j = 1,2, \dots, m$.

b- Compute $b_{j3}; j = 1,2, \dots, m$ using steps 1, 3, 4 and 5-a and Eq. (5.39).

Step (6):

a- $s_r = a + rh \quad ; r = 1,2, \dots, n$.

b- Calculate B_{ir}^{4} $; i = 1,2, \dots, m, r = 0,1, \dots, n,$ by substituting s_r in Eq. (5.36) with steps 1, 2, 3, 4, 5 and 6-a.

5.7 Numerical Examples

We will show here some numerical results for a non-linear VIE with variable coefficients using spline functions. We use a quadratic classic spline with the trapezoidal rule (Qu-Sp1), a quadratic classic spline with the classic Runge-Kutta fourth order method (Qu-Sp2), a cubic classic spline with the trapezoidal rule (Cu-Sp1), a cubic classic spline with the classic Runge-Kutta fourth order method (Cu-Sp2), a first-order B-spline (B1-Sp), a second-order B-spline (B2-Sp), a third-order B-spline (B3-Sp), and a fourth-order B-spline (B4-Sp). The results for these methods are obtained using the non-linear VIEQSP1, non-linear VIEQSP2, non-linear VIECSP1, non-linear VIECSP2, non-linear VIEB1SP, non-linear VIEB2SP, non-linear VIEB3SP and non-linear VIEB4SP algorithms, respectively.

Test Example 1:

Consider the non-linear VIE:

$$\Phi(x) = 1 + x^2 - xe^{x^2} + \int_0^x e^{x^2 - t^2 - 1}\, e^{\Phi(t)} dt$$

with exact solution is [53]:

$$\Phi(x) = 1 + x^2$$

The results for the Test Example 1 using a first-order B-spline (B1-Sp), a second-order B-spline (B2-Sp), a third-order B-spline (B3-Sp), a fourth-order B-spline (B4-Sp), a quadratic spline using the trapezoidal rule (Qu-Sp1), a quadratic spline using the classic fourth-order Runge-Kutta method (Qu-Sp2), a cubic spline using the trapezoidal rule (Cu-Sp1) and a cubic spline using the classic fourth-order Runge-Kutta method (Cu-Sp2) are shown in Table 10 and Table 11.

Table 10 Comparison Between the Exact Solution and the Numerical Solution of $\Phi(x)$ Taking $h = 0.1$ for Test Example 1 Using the B1-Sp, B2-Sp, B3-Sp and B4-Sp Methods

x	Exact Solution	Approximate Solution of $\Phi(x)$ by			
		B1-Sp	B2-Sp	B3-Sp	B4-Sp
0.0	1.00000000000000	1.00000000000000	1.00000000000000	1.00000000000000	1.00000000000000
0.1	1.01000000000000	1.00954751665413	1.00972802145786	1.00960385461642	1.00963272419513
0.2	1.04000000000000	1.03841256663433	1.03887120013197	1.03851826061492	1.03867610980779
0.3	1.09000000000000	1.08683740346050	1.08736290334356	1.08689890479471	1.08734387606348
0.4	1.16000000000000	1.15508650335173	1.15518911790652	1.15484782694624	1.15573286687932
0.5	1.25000000000000	1.24353192285836	1.24243140769938	1.24250207700386	1.24389233805206
0.6	1.36000000000000	1.35279253324880	1.34936552677955	1.35016969073185	1.35194336021879
0.7	1.49000000000000	1.48399174192264	1.47666655605281	1.47857024212186	1.48032717979529
0.8	1.64000000000000	1.63928383040757	1.62583418229499	1.62927340290580	1.63033766579197
0.9	1.81000000000000	1.82304682844170	1.80011967318101	1.80548587459046	1.80523114929133
1.0	2.00000000000000	2.04495782732929	2.00677978916307	2.01346865603370	2.01245077188606
L.S.E.		**0.00235869078550**	**7.238601547e-004**	**6.388492649e-004**	**4.940517859e-004**

Table 11 Comparison Between the Exact Solution and the Numerical Solution of $\Phi(x)$ Taking $h = 0.1$ for Test Example 1 Using the Qu-Sp1, Qu-Sp2, Cu-Sp1 and Cu-Sp2 Methods

x	Exact Solution	Approximate Solution of $\Phi(x)$ by			
		Qu-Sp1	Qu-Sp2	Cu-Sp1	Cu-Sp2
0.0	1.00000000000000	1.00000000000000	1.00000000000000	1.00000000000000	1.00000000000000
0.1	1.01000000000000	1.01000000000000	1.01000000000000	1.01000000000000	1.01000000000000
0.2	1.04000000000000	1.03795946129038	1.03992676536745	1.04000000000000	1.04000000000000
0.3	1.09000000000000	1.08634087790346	1.08984177388826	1.09000000000000	1.09000000000000
0.4	1.16000000000000	1.15364554619616	1.15973082579604	1.16000000000000	1.16000000000000
0.5	1.25000000000000	1.24213589314259	1.24957740122872	1.25000000000000	1.25000000000000
0.6	1.36000000000000	1.34875008895409	1.35934865245530	1.36000000000000	1.36000000000000
0.7	1.49000000000000	1.47881895066659	1.48899574844859	1.49000000000000	1.49000000000000
0.8	1.64000000000000	1.62635149275074	1.63842206091325	1.64000000000000	1.64000000000000
0.9	1.81000000000000	1.80793176642011	1.80745874964826	1.81000000000000	1.81000000000000
1.0	2.00000000000000	2.01386478710978	1.99575796821871	2.00000000000000	2.00000000000000
L.S.E.		**7.541442571e-004**	**2.865689683e-005**	**3.4512664603e-031**	**4.9303806576e-031**

Test Example 2:

Consider the non-linear VIE:

$$\Phi(x) = \cos(x) - \sin(x) - \frac{1}{4}\sin(2x) + \frac{1}{2}x - \frac{1}{2}x^2 + \int_0^x (x-t)\Phi^2(t)dt$$

with exact solution [53]:

$$\Phi(x) = \cos(x) - \sin(x)$$

The results for the Test Example 2 using a first-order B-spline function (B1-Sp), a second-order B-spline function (B2-Sp), a third-order B-spline function (B3-Sp), a fourth-order B-spline function (B4-Sp), a quadratic spline using the trapezoidal rule (Qu-Sp1), a quadratic spline using the classic fourth-order Runge-Kutta method (Qu-Sp2), a cubic spline using the trapezoidal rule (Cu-Sp1) and a cubic spline using the classic fourth-order Runge-Kutta method (Cu-Sp2) are shown in Table 12 and Table 13.

Table 12 Comparison Between the Exact Solution and the Numerical Solution of $\Phi(x)$ Taking $h = 0.1$ for Test Example 2 Using the B1-Sp, B2-Sp, B3-Sp and B4-Sp Methods

x	Exact Solution	Approximate Solution of $\Phi(x)$ by			
		B1-Sp	B2-Sp	B3-Sp	B4-Sp
0.0	1.00000000000	1.00000000000	1.00000000000	1.00000000000	1.00000000000
0.1	0.89517074863	0.89550341593	0.89550341593	0.89550341593	0.89550341593
0.2	0.78139724704	0.78285059681	0.78244505133	0.78243845380	0.78226412052
0.3	0.65981628246	0.66294736562	0.66154907541	0.66151976221	0.66103336173
0.4	0.53164265169	0.53678938067	0.53378530771	0.53371149640	0.53292021833
0.5	0.39815702328	0.40546348869	0.40032817494	0.40018611988	0.39925671423
0.6	0.26069314151	0.27015769324	0.26252050571	0.26228627446	0.26148174267
0.7	0.12062450004	0.13217144628	0.12183869430	0.12149003216	0.12105536548
0.8	-0.02064938155	-0.00707657235	-0.02014455299	-0.02062149867	-0.02058163140
0.9	-0.16171694135	-0.14604573590	-0.16180357831	-0.16239639446	-0.16198375768
1.0	-0.30116867893	-0.28307754417	-0.30151221430	-0.30214663621	-0.30158792701
L.S.E.		**0.00107190652**	**1.8709909e-005**	**1.7200026e-005**	**6.2440025e-006**

Table 13 Comparison Between the Exact Solution and the Numerical Solution of $\Phi(x)$ Taking $h = 0.1$ for Test Example 2 Using the Qu-Sp1, Qu-Sp2, Cu-Sp1 and Cu-Sp2 Methods

x	Exact Solution	Approximate Solution of $\Phi(x)$ by			
		Qu-Sp1	Qu-Sp2	Cu-Sp1	Cu-Sp2
0.0	1.00000000000	1.0000000000	1.0000000000	1.00000000000	1.00000000000
0.1	0.89517074863	0.89770341593	0.89517439208	0.89530541593	0.89517063858
0.2	0.78139724704	0.78419804320	0.78140063462	0.78170825718	0.78139711546
0.3	0.65981628246	0.66314247204	0.65981926910	0.66034675291	0.65981615993
0.4	0.53164265169	0.53505069118	0.53164517386	0.53243643952	0.53164258198
0.5	0.39815702328	0.40196425404	0.39815907220	0.39925778843	0.39815706123
0.6	0.26069314151	0.26448859021	0.26069478388	0.26214313612	0.26069335103
0.7	0.12062450004	0.12466291904	0.12062585553	0.12246306049	0.12062495223
0.8	-0.02064938155	-0.01669674477	-0.02064814274	-0.01838763997	-0.02064861045
0.9	-0.16171694135	-0.15769625604	-0.16171561971	-0.15900419713	-0.16171577238
1.0	-0.30116867976	-0.29727430607	-0.30116706366	-0.29798617876	-0.30116703345
L.S.E.		**1.2910178e-004**	**5.4655149e-011**	**3.0323596e-005**	**4.9678347e-012**

Test Example 3:

Consider the non-linear VIE:

$$\Phi(x) = e^x - \frac{1}{9}e^{3x} + \frac{1}{9} + \frac{1}{3}x + \int_0^x (x-t)\Phi^3(t)dt$$

with exact solution [53]:

$$\Phi(x) = e^x$$

The results for the Test Example 3 using a first-order B-spline function (B1-Sp), a second-order B-spline function (B2-Sp), a third-order B-spline function (B3-Sp), a fourth-order B-spline function (B4-Sp), a quadratic spline function using the trapezoidal rule (Qu-Sp1), a quadratic spline function using the classic fourth-order Runge-Kutta method (Qu-Sp2), a cubic spline function using the trapezoidal rule (Cu-Sp1) and a cubic spline function using the classic fourth-order Runge-Kutta method (Cu-Sp2) are shown in Table 14 and Table 15.

Table 14 Comparison Between the Exact Solution and the Numerical Solution of $\Phi(x)$ Taking $h = 0.1$ for Test Example 3 Using the B1-Sp, B2-Sp, B3-Sp and B4-Sp Methods

x	Exact Solution	Approximate Solution of $\Phi(x)$ by			
		B1-Sp	B2-Sp	B3-Sp	B4-Sp
0.0	1.00000000000	1.00000000000	1.00000000000	1.00000000000	1.00000000000
0.1	1.10517091807	1.10463105056	1.10463105056	1.10463105056	1.10463105056
0.2	1.22140275816	1.21884114861	1.21909610395	1.21983534667	1.22020019606
0.3	1.34985880757	1.34374046551	1.34474726603	1.34736770147	1.34889270994
0.4	1.49182469764	1.48064733010	1.48314061662	1.48865831212	1.49240183731
0.5	1.64872127070	1.63117368523	1.63618050774	1.64480656317	1.65165959340
0.6	1.82211880039	1.79737030901	1.80643037432	1.81677641176	1.82677416056
0.7	2.01375270748	1.98198638105	1.99772569549	2.00597972270	2.01739473258
0.8	2.22554092849	2.18896180282	2.21632606302	2.21567988364	2.22406658786
0.9	2.45960311115	2.42443120170	2.47303293055	2.45414582790	2.45202476215
1.0	2.71828182845	2.69895926002	2.78716234634	2.74144298204	2.72100448830
L.S.E.		**0.00504718333**	**0.00329272718**	**7.8672636e-004**	**1.135924e-004**

Table 15 Comparison Between the Exact Solution and the Numerical Solution of $\Phi(x)$ Taking $h = 0.1$ for Test Example 3 Using the Qu-Sp1, Qu-Sp2, Cu-Sp1 and Cu-Sp2 Methods

x	Exact Solution	Approximate Solution of $\Phi(x)$ by			
		Qu-Sp1	Qu-Sp2	Cu-Sp1	Cu-Sp2
0.0	1.00000000000	1.00000000000	1.00000000000	1.00000000000	1.00000000000
0.1	1.10517091807	1.10219105056	1.10516304816	1.10493709056	1.10517020043
0.2	1.22140275816	1.21698345635	1.22138888613	1.22097228348	1.22140090849
0.3	1.34985880757	1.34298736975	1.34983514736	1.34927136553	1.34985519157
0.4	1.49182469764	1.48215979121	1.49178497650	1.49112184900	1.49181812589
0.5	1.64872127070	1.63408536299	1.64865473190	1.64794543752	1.64870952218
0.6	1.82211880039	1.80144476048	1.82200734022	1.82131156772	1.82209770742
0.7	2.01375270747	1.98219659081	2.01356486247	2.01295191104	2.01371436169
0.8	2.22554092849	2.17992548557	2.22522242930	2.22477546678	2.22547008214
0.9	2.45960311115	2.38797809342	2.45905727276	2.45888331677	2.45946977189
1.0	2.71828182845	2.61085942906	2.71733545688	2.71758083582	2.71802558339
L.S.E.		**0.02055693482**	**1.3495289e-006**	**4.5693109e-006**	**9.057363e-008**

CHAPTER 6

CONCLUSION

6.1 Discussion

In this thesis, Picard's iteration method is introduced to find the solution of a non-linear VIE of the second kind.

In addition to the previous methods, there are other approaches that are important in finding solutions. These are the approximate or numerical methods. The importance of such methods lies in the fact that many problems cannot be solved using Picard's iteration method, and it is very convenient to solve them using the numerical or approximate methods as a result of the great developments in computational methods used in computers.

This proves the need to find and use numerical or approximate methods.

Some numerical methods were used to find approximate solutions to non-linear VIEs of the second kind.

A comparison was made between these methods by calculating the least square error (L.S.E.) between the calculated solution and the exact solution of the VIEs.

Multistep methods, Runge-Kutta methods and spline functions were used to find the solution of non-linear VIEs of the second kind, and the exact results were presented. These results are compared below.

Tables 16, 17, and 18 show a comparison between the results obtained from solving Test Examples 1, 2, and 3 using the multistep methods, Runge-Kutta methods and spline functions.

The approximate or numerical solution of a non-linear VIE of the second kind is found, using some numerical methods. Algorithms are built, examples are solved and good results are obtained. A comparison is also made between these methods depending on the least square error (L.S.E.), which is calculated from the numerical solution against the exact solution.

The approximate solution of the non-linear integral equation Eq. (2.2) was obtained from multistep methods; these were described in chapter 3, sections 3, 4, and 5. For each method a computer program was written and several examples were solved.

From our results, the following notes are drawn:

1- Simpson's 1/3 rule requires that the interval be subdivided into an even number of subdivisions, and Simpson's 3/8 rule requires it to be subdivided into a number of subdivisions that is a multiple of three.

2- In order to approximate the solution of the Volterra equation at the points t_i, Simpson's 1/3 rule cannot define t_i for all i, so we need to use a combination of Simpson's 1/3 rule and Simpson's 3/8 rule.

In this thesis a single-step Runge-Kutta method of the second, third and fourth orderswas successfully used to find the numerical solutions of a non-linear VIE of the second kind. Examples were considered in this context. We note that the fourth-order Runge-Kutta method gives a better approximation to the exact solution than the other orders of Runge-Kutta methods.

Quadratic and cubic spline functions were introduced to find the numerical solutions to non-linear VIEs of the second kind. Several examples were applied for illustration and good results were achieved.

The following points were identified:

1-We found that a cubic spline function is superior to a quadratic spline function at $n = 10$ in all the examples.

2-As n increases, the error term decreases when quadratic and cubic spline functions are used, with no ended n or even an optimal solution be handled. This is true for all examples.

3-The interpretation of this paradox may lie in the fact that the mathematical treatment is insufficient for integral problems with this method.

4-In all examples, the Runge-Kutta method is better than the trapezoidal rule for evaluating the integral involved.

5-In all examples, the fourth-order Runge-Kutta method (RK4_Kutta's) is better than all the other methods.

This thesis introduces numerical methods for approximating the solution of a system of non-linear VIEs; these solutions use classic spline and B-spline functions of

different types. We have tried to emphasize some important ideas while maintaining a reasonable level of complexity. For one thing, we have always used a uniform step size.

This method for solving non-linear VIEs has some advantages and some disadvantages. We conclude with the following remarks:

1. The cubic classic spline function gives better accuracy than the quadratic classic spline function.
2. The fourth-order B-spline function gives better accuracy than the first-, second-, and third-order B-spline functions.
3. As the degree of the B-spline function increases, the error term is decreased.

The results for the least square error (L.S.E.) for the different methods for Test Example 1 are shown in Table 16.

Table 16 Comparison of the Error Between the Methods for Test Example 1

Methods	L.S.E.
TRP	1.133987551255e-030
SMP	9.8607613152626e-031
RK3 Kutta's	7.39557098644698e-031
RK3 Optimal	4.93038065763132e-031
RK4_Classic	4.9303806576313e-031
RK4_Kutta's	9.8607613152626e-032
Qu-Sp1	7.541442571e-004
Qu-Sp2	2.865689683e-005
Cu-Sp1	3.4512664603e-031
Cu-Sp2	4.9303806576e-031
B1-Sp	0.00235869078550
B2-Sp	7.238601547e-004
B3-Sp	6.388492649e-004
B4-Sp	4.940517859e-004

The results for the least square error (L.S.E.) for the different methods for Test Example 2 are shown in Table 17.

Table 17 Comparison of the Error Between the Methods for Test Example 2

Methods	L.S.E.
TRP	3.5882724173280e-005
SMP	7.23504602860014e-007
RK3 Kutta's	2.757312101152e-009
RK3 Optimal	3.4423208371321e-009
RK4_Classic	6.33780023356121e-012
RK4_Kutta's	1.175360243510608e-012
Qu-Sp1	1.2910178e-004
Qu-Sp2	5.4655149e-011
Cu-Sp1	3.0323596e-005
Cu-Sp2	4.9678347e-012
B1-Sp	0.00107190652
B2-Sp	1.8709909e-005
B3-Sp	1.7200026e-005
B4-Sp	6.2440025e-006

The results for the least square error (L.S.E.) for the different methods for Test Example 3 are shown in Table 18.

Table 18 Comparison of the Error Between the Methods for Test Example 3

Methods	L.S.E.
TRP	0.002580467257583
SMP	4.913444656575e-006
RK3 Kutta's	4.8710009480530e-005
RK3 Optimal	5.97350220452353e-006
RK4_Classic	1.102692065138610e-007
RK4_Kutta's	2.604964081210461e-008
Qu-Sp1	0.02055693482
Qu-Sp2	1.3495289e-006
Cu-Sp1	4.5693109e-006
Cu-Sp2	9.057363e-008
B1-Sp	0.00504718333
B2-Sp	0.00329272718
B3-Sp	7.8672636e-004
B4-Sp	1.135924e-004

6.2 Recommendations

Our recommendations for future work are as follow:

1. Another type of spline function, such as cardinal spline functions, could be used to compute a numerical solution for a system of non-linear VIEs and a system of higher-order non-linear Volterra integro-differential equations (VIDEs).

2. Another type of spline function, such as G-spline functions, could be used to compute a numerical solution for a system of non-linear VIEs and a system of higher-order non-linear VIDEs.

3. Another order of B-spline functions could be used to solve a system of non-linear VIEs and a system of higher-order non-linear VIDEs.

4. Other methods of numerical integration such as the Richardson and Romberg methods could be used.

REFERENCES

1. **Linz P., (1985),** "*Analytical and Numerical Methods for Volterra Equations*", Society for Industrial and Applied Mathematics (SIAM), Philadelphia, pp. 10-96.

2. **Baker C. T. H., (2000),** "*Volterra Integral Equations*", Manchester Centre for Computational Mathematics, Numerical Analysis Report, Manchester, no. 366.

3. **Saveljeva D., (2006),** "*Quadratic and Cubic Spline Collocation for Volterra Integral Equations*", Doctoral Thesis, University of Tartu, Estonia, pp. 15-102.

4. **Polyanin A. D., Manzhirov A. V., (1998),** "*Handbook of Integral Equations*", CRC Press LLC, Florida, pp. 676-693.

5. **Atkinson K. E., (1989),** "*An Introduction to Numerical Analysis*", John Wiley & Sons, New York, pp. 135-260.

6. **Burden R. L., Faires J. D., (1997),** "*Numerical Analysis*", Sixth Edition, Brooks / Cole Publishing Company, New York, pp. 173-205.

7. **Lutterkort D., (1999),** "*Computing Linear Envelopes for Uniform B-Spline Curves*", http://www.watzmann.net/lutter/papers/#a3, (Data Download Date: 02.03.1999).

8. **Cheney W., Kincaid D., (2004),** "*Numerical Mathematics and Computing*", Fifth edition, Brooks / Cole Publishing Company, California, pp. 386-455.

9. **Hoffman J. D., (2001),** "*Numerical Methods for Engineers and Scientists*", Second edition, Marcel Dekker CRC Press, New York, pp. 421-499.

10. **Greville T. N. E., (1969),** "*Theory and Applications of Spline Functions*", Wisconsin University Academic Press, New York, pp. 157-207.

11. **Hou H. S., Andrews H. C., (1978),** "*Cubic Splines for Image Interpolating and Digital Filtering*", IEEE. Terns. Speech and Signal Processing, Indiana, vol. ASSP-26, pp. 508-517.

12. **Kekre H. B., Sahasrabudhe S. C., (1982),** "*Raised Cosine Function for Image Data Interpolation*", Computer & Electrical Engineering-CEE, Bombay, vol. 9, no. 3.4, pp. 131-152.

13. **Saeed R. K., (2006),** "*Computational Methods for Solving System of Linear Volterra Integral and Integro-Differential Equations*", Doctoral Thesis, University of Salahaddin, Erbil, pp. 35-65.

14. **Mahmoudi Y., (2005),** "*Wavelet Galerkin Method for Numerical Solution of Nonlinear Integral Equation*", Applied Mathematics and Computation, vol. 167, no 2, pp. 1119-1129.

15. **Wazwaz A. M., (2005),** "*The Modified Decomposition Method for Analytic Treatment of Non-linear Integral Equations and Systems of Non-linear Integral Equations*", International Journal of Computer Mathematics, vol. 82, pp. 1107-1115.

16. **Mustafa M. M., (2004),** "*Numerical Algorithms for Treating System of Volterra Integral Equations Using Spline Functions*", Doctoral Thesis, University of Al-Mustansiriya, Baghdad, pp. 33-67.

17. **Al-Nasir R. H., (1999),** "*Numerical Solutions of Volterra Integral Equations of the Second Kind*", Master's Thesis, University of Technology, Baghdad, pp. 8-19.

18. **Al-Rawi S. N., (1995),** "*Numerical Solution of First Kind Integral Equations of Convolutions Type*", Master's Thesis, University of Technology, Baghdad, pp. 15-41.

19. **Saadati R., Raftari B., Adibi H., Vaezpour S. M., Shakeri S., (2008),** "*A Comparison Between the Variational Iteration Method and Trapezoidal Rule for Solving Linear Integro Differential Equations*", World Applied Sciences Journal, Pakistan, vol. 4, no. 3, pp. 321-325.

20. **Al-Timeme A. J. S., (2003),** "*Approximated Methods for First Order Volterra Integro-differential Equations*", Master's Thesis, University of Technology, Baghdad, pp. 28-48.

21. **Al-Dahan O. K., (1996),** "*Approximated Solution of Volterra Integral Equations*", Master's Thesis, University of Technology, Baghdad, pp. 10-27.

22. **Al-Jawary M. A. W., (2005),** "*Numerical Methods for System of Integral Equations*", Master's Thesis, University of Baghdad, Baghdad, pp. 22-35.

23. **Atkinson K. E., (1997),** "*The Numerical Solution of Integral Equations of the Second Kind*", Cambridge University Press, New York, pp. 1-381.

24. **Miller R. K., (1971),** "*Non-Linear Volterra Integral Equations*", W.A. Benjamin, California, pp. 312-335.

25. **Seth B. R., F. N. A., (1973), "***Numerical Solution of Non-Singular Fredholm Integral Equations of the Second Kind*", By Sastry S. S., Indian Journal of Pure and Applied Mathematics, vol. 6, pp. 773-782.

26. **Saberi-Nadjaf J., Heidari M., (2007),** "*Solving Linear Integral Equations of the Second Kind with Repeated Modified Trapezoid Quadrature Method*", Applied Mathematics and Computation, pp. 980-985.

27. **Nik N., Eshkuvatov Z., Yaghobifar M., Hasan M., (2008),** "*Numerical Solution of Infinite Boundary Integral Equation by Using Galerkin Method with Laguerre Polynomials*", World Academy of Science Engineering and Technology, Paris, vol. 42, p. 951.

28. **Rahbar S., Hashemizadeh E., (2008),** "*A Computational Approach to the Fredholm Integral Equation of the Second Kind*", World Congress on Engineering, London, vol. II, pp. 1-4.

29. **Hacia L., Kaczmarek L., (1995),** "*On the Bounds of a System of Volterra Integral Equations*", Demonstration Mathematical, vol. XXVIII, no. 4, pp. 945-952.

30. **Chen W., Li C., Ou B., (2000),** "*Classification of Solutions for a System of Integral Equations*", Journal of Integral Equations and Applications, vol. 151, pp. 141-147.

31. **Baker C. T. H., Miller G. F., (1982),** "*Treatment of Integral Equations by Numerical Methods*", LTD Academic Press, London, pp. 26-75.

32. **Biazar J., Babolian E., Islam R., (2003),** "*Solution of a System of Volterra Integral Equations of the First Kind by Adomian Method*", Applied Mathematics and Computation, vol. 139, pp. 249-258.

33. **Babolian E., Biazar J., Vahidi A. R., (2004),** "*On the Decomposition Method for System of Linear Volterra Integral Equations*", Applied Mathematics and Computation, vol. 147, pp. 19-27.

34. **Maleknejad K., Shahrezaee M., (2004),** "*Using Runge-Kutta Method for Numerical Solution of the System of Volterra Integral Equation*", Applied Mathematics and Computation, vol. 149, pp. 399-410.

35. **Hall G., Watt J. M., (1976),** "*Modern Numerical Methods for Ordinary Differential Equations*", Oxford University Press, California, pp. 79-142.

36. **Rice J. R., (1983),** "*Numerical Method, Software, and Analysis*", Mathematics and Computer Science, McGraw-Hill, New York, pp. 365-451.

37. **Delves L. M., Mohamed J. L., (1985),** "*Computational Methods for Integral Equations*" Cambridge University Press, Cambridge, pp. 266-370.

38. **Delves L. M., Walsh J., (1974),** "*Numerical Solution of Integral Equations*", Oxford University Clarendon Press, London, pp. 245-310.

39. **Al-Kahachi A. M., (2000),** "*Approximate Method for Solving Volterra Integral Equations of the First Kind*", Master's Thesis, University of Technology, Baghdad, pp. 6-25.

40. **Al-Salhi B. F., (2000),** "*Numerical Solution of Non-linear Volterra Integral Equations of the Second Kind*", Master's Thesis, University of Technology, Baghdad, pp. 11-29.

41. **Al-Asadi Z., (2000),** "*Algorithms for Solving Non-linear Volterra Integral Equations of First Kind*", Master's Thesis, Al-Mustansiriya University, Baghdad, pp. 8-33.

42. **Abdul Hameed F. T., (2002),** "*Numerical Solution of Fredholm Integro-differential Equation Using Spline Functions*", Master's Thesis, University of Technology, Baghdad, pp. 13-50.

43. **Juma B. F., (2005),** "*On Approximate Solutions so a System of Non-linear Volterra Integral Equations*". Doctoral Thesis, University of Technology, Baghdad, pp. 28-78.

44. **Salam J. M., Huda H. O., (2010),** "*Solutions for Linear Fredholm-Volterra Integral Equations of the Second Kind Using the Repeated Corrected Trapezoidal and Simpson's Methods*", University of Baghdad, vol. 13 (2), pp. 194-204.

45. **Malindzisa H. S., Khumalo M., (2014),** "*Numerical Solutions of a Class of Nonlinear Volterra Integral Equations*", Abstract and Applied Analysis, Hindawi Publishing Corporation, South Africa, vol. 2014, pp.1-8.

46. **Netravali A. N., (1973),** "*Spline Approximation to the Solution of the Volterra Integral Equation of the Second Kind*", Mathematics of Computation, vol. 27, no. 121, pp. 99-105.

47. **Al-Faour O. M., Kadhim A. J., Jaber A. S., (2008),** "*The Numerical Solution of Hammerstein-Volterra Integral Equations Using Spline Functions*", Department of Applied Sciences, University of Technology, Baghdad, vol. 19, no. 2, pp 106-119.

48. **Gardi N. A. S., (1999),** "*New Techniques in the Numerical Solutions of Fredholm and Volterra Integral Equations*", Doctoral Thesis, University of Salahddin, Erbil, pp. 54-96.

49. **Porter D., Stirling D. S. G., (1990),** "*Integral Equations, a Practical Treatment, from Spectral Theory to Applications*", Cambridge Texts in Applied Mathematics, Cambridge University Press, New York, pp. 121-207.

50. **Jerri A. J., (1985),** "*Introduction to Integral Equations with Applications*", Pure and Applied Mathematics, Marcel Dekker, New York, pp. 1-164.

51. **Chambers L. I. G., (1976),** "*Integral Equations A Short Course*", International Textbook Company Limited, London, pp. 114-166.

52. **Rahman M., (2007),** "*Integral Equations and their Applications*", WIT Press, Canada and Mexico, pp. 67-68

53. **Wazwaz A. M., (2011),** "*Linear and Nonlinear Integral Equations Methods and Applications*", Higher Education Press, Springer, Beijing, pp. 388-628.

54. **Karoui A., (2005),** "*Existence and Approximation Solutions of Nonlinear Integral Equations*", Journal of Inequalities and Applications, vol. 5, pp. 569-581.

55. **Linz P., (1969),** "*A Method for Solving Nonlinear Volterra Integral Equations of the Second Kind*", Mathematics of Computation, vol. 23, no. 107, pp. 595-599.

56. **Burden R. L., Faires J. D., (2001),** "*Numerical Analysis*", Brooks / Cole, Canada, vol. 1, pp. 259-711.

57. **Burden R. L., Faires J. D., (2011),** "*Numerical Analysis*", Ninth Edition, Brooks / Cole, Boston, pp. 193-206.

58. **Ibrahim G. H., (2006),** "*Numerical Treatments of System of Fredholm Integral Equations*", Master's Thesis, University of Technology, Baghdad, pp 12-30.

59. **Epperson J. F., (2013),** "*An Introduction to Numerical Methods and Analysis*", John Wiley & Sons, Hoboken, pp. 68-74.

60. **Pav S. E., (2012),** "*Numerical Methods Course Notes*", Department of Mathematics, University of California, San Diego, pp. 83-89.

APPENDIX A

CURRICULUM VITAE

PERSONAL INFORMATION

Surname, Name: Saleh, Merewan

Date and Place of Birth: 28 September 1981, Kirkuk

Marital Status: Married

Phone: +964 770 233 94 58

Email: marewan_812004abdal@yahoo.com

EDUCATION

Degree	Institution	Year of Graduation
M.Sc.	Çankaya University, Mathematics and Computer Science	2015
B.Sc.	Erbil University, Mathematical Education	2004
High School	Youm Al-Nakhwa High School	2000

WORK EXPERIENCE

Year	Place	Position
August 2006 -2012	Kirkuk University	Teaching Assistant

FOREIN LANGUAGES

Arabic, Kurdish, English

HOBBIES

Travel, Reading, Swimming, Football