جمهورية العراق

وزارة التعليم العالي والبحث العلمي

جامعة بغداد

كلية العلوم للبنات

جامعة بغداد

خوارزميات لحل معادلات فولتيرا التكاملية باستخدام دوال الثلمة الغير متعددة الحدود

رسالة تقدمت بها

ساره حميد حربي الخالدي

بكالويوس علوم رياضيات/ كلية العلوم للبنات / جامعة بغداد،٢٠٠٩

الى مجلس كلية العلوم للبنات-جامعة بغداد

هي جزء من متطلبات شهادة الماجستير في علوم الرياضيات

بأشراف

أ.م. د. منى منصور مصطف

شوال /١٤٣٤ **أب / ٢٠١٣**

المستخلص

في هذه الاطروحة محاولة لدراسة وتطويربعض الطرق العددية لمعالجة نوعان مختلفان من معادلات فولتيرا التكاملية الخطية وهما:

- **معادلات فولتيرا التكاملية الخطية من النوع الثاني.**
- **معادلات فولتيرا التكاملية الخطية ذات نواة ضعيفة شاذة.**

نوعان مختلفان من دوال الثلمة الغير متعددة الحدود(متضمنة: الخطية والتربيعية) طبقت لاول مرة لمعالجة المسألتين اعلاه.

كذلك معادلات فولتيرا التكاملية من النوع الاول تم تحويلها الى معادلات فولتيرا التكاملية من النوع الثاني وبعد ذلك تم معالجتها عدديا بنفس الطريقة.

علاوة على ذلك تم تقديم العديد من الامثلة لتوضيح الدقة والكفاءة وسهولة الاداء للطريقة المقترحة من جهة ولتاكيد رتبة التقارب من جهة اخرى.كذلك تم دراسة تحليل التقارب لهذه الطريقة.

واخيراً تم كتابة الخوارزميات والبرامج بلغة(MATLAB\ R12, 2012) لكل من الطريقتين اعلاه

1.1Introduction:

Integral Equation s occur in natural way in the course of obtaining mathematical solution to mixed boundary value problems of mathematical physics .of the many possible approaches to the reduction of a given mixed boundary problem to an integral equation[24].

This chapter is organized as follows: In section (1.2), we study the classification of integral equations, and some basic concepts are given. In section (1.3), the mathematical theory of the existence and uniqueness theorem for linear VIE's will be considered. In section (1.4), some analytical methods are considered to find the solution of linear VIE's of the second kind and VIE's with weakly singular kernel .In section (1.5), a discussion about using analytical method are given. In section (1.6), we give the aim of this thesis and finally in section (1.7) some test examples are given.

1.2 Classification of Integral Equations

We will mention some basic definitions for integral equations,

Definition (1.1): [44]

An integral equation is that equation in which the unknown function u(x) appears inside an integral sign .The most standard type of integral equation in u(x) is of the form:

$$\mathrm{h}(x)\mathrm{u}(x) = \mathrm{f}(x) + \lambda \int_{a(x)}^{b(x)} k(x,t)u(t)dt \ , x \in [\mathrm{a},\mathrm{b}] \qquad (1.1)$$

where $\mathrm{a}(x)$ and $\mathrm{b}(x)$are the limits of integration, λ is a constant parameter,

and $k(x,t)$is a known function of two the variables x and t, called the kernel of the integral equation. The functions f (x) and k(x, t) are given in advance. It is to be noted that the limits of integration determined as $a(x)$ and $b(x)$ and may be both variables, constants, or mixed.

Definition (1.2): [18]

An integral equation (1.1) is called **non-linear integral equation,** if the kernel $k(x,t)$ is given in the form $k\big(x,t,u(t)\big)$.

Definition (1.3): [9]

The linear integral equation (1.1) is called **homogenous**, if $f(x)=0$, otherwise it is called **non- homogenous**.

Definition (1.4): [24]

The equation (1.1) is called **linear integral equation of the first kind**, if $h(x)=0, while\ if\ h(x)=1$,it called **linear integral equation of the second kind,** otherwise it is called of the **third kind .**

Definition (1.5): [34]

The integral equation is called **Volterra integral equation**, when $a(x)=a$ and $b(x)=x$, where a is costant,that is:

$$h(x)u(x)=f(x)+\lambda\int_{a}^{x}k(x,t)u(t)dt\ ,x\in[a,b] \qquad (1.2)$$

Definition (1.6): [9]

The integral equation is called **Fredholm integral equation**, when $a(x) = a \ \ , and \ \ b(x) = b,$ where $a \ and \ b \ are$ constants ,which has a form:

$$\mathrm{h}(x)\mathrm{u}(x) = \mathrm{f}(x) + \lambda \int_a^b k(x,t)u(t)dt \ \ , x \in [\mathrm{a}, \mathrm{b}] \qquad (1.3)$$

Definition (1.7): [18]

If the kernel in integral equation (1.1) depends on the difference $(x - t)$, then it called difference kernel and the equation is called **integral equation of convolution type** .*i.e*, $k(x,t) = k(x - t)$

Here we can apply laplace transform to get the exact solution.

Definition (1.8): [18]

The kernel is called **degenerate** or (**sparable)** kernel, when the kernel May be decompose as follows:

$$k(x,t) = \sum_{k=1}^{n} a_k \ (x) b_k(t)$$

Definition (1.9): [44]

An integro differential equation is an equation involving derivative and integral togather with unknown function $u(x)$ which is of the form:

$$u^{(k)}(x) + \sum_{j=0}^{k-1} p_j(x)u^j(x) = f(x) + \int_{a(x)}^{b(x)} k(x,t)\, u(t)dt \qquad (1.4)$$

$where \ \ u^{(j)}(x) = \dfrac{d^j u}{dx^j}$

Definition (1.10): [34]

The integral $\int_a^b f(x)\,dx$ is called improper if.

(i) a=∞ or b=∞ or both

(ii) $f(x)$ is unbounded at one or more points of a $\leq x \leq$b (there points are called singular points)Moreover, it is called singular if the kernel k(x, t) becomes unbounded at one or more points in the interval of integration.

*Integral corresponds to (i) and (ii) are called improper integrals of the 1st and 2nd kind respectively

* Integral with both (i) and (ii) are called improper integrals of the 3rd kind.

Definition (1.11): [34]

If the kernel $k(x,t)$ is in the form $k(x,t) = \frac{H(x,t)}{(x-\varepsilon)^{\alpha}}$

Where H is bounded in D :$a \leq x \leq b$ and $a \leq t \leq b$ with $H(x,t) \neq 0$and $\alpha$is constant s.t $0 \leq \alpha \leq 1$ then the integral equation is called weakly singular. The equations of the form:

$$f(x) = \int_0^x \frac{u(t)}{(x-t)^{\alpha}}\,dt \qquad 0 < \alpha < 1. \qquad (1.5)$$

or of the second kind

$$u(x) = f(x) + \int_0^x \frac{u(t)}{(x-t)^{\alpha}}\,dt \qquad 0 < \alpha < 1. \qquad (1.6)$$

are called **generalized Abel's integral equation and weakly singular integral equations** respectively. For α = 1/2

$$f(x) = \int_0^x \frac{u(t)}{(x-t)^{\frac{1}{2}}}\,dt$$

is called the **Abel's singular integral equation** .We will focus our concern on equation of the form:

$$u(x) - \int_0^t \frac{t^{\mu-1}}{x^{\mu}}\, u(t)dt = f(x)\,, \quad x \in [0,T]$$

which is **Volterra integral equations of the second kind with weakly singular kernel**. Where u(t) is unknown function and f is known function. Where $0< \mu<1$ .However, there is a singularity at t=0 and s=0 for any positive value of t.

In this thesis we will consider the two following problems:

- Linear Volterra integral equation of the Second kind (VIE's) with λ =1,of the form:

$$u(x) = f(x) + \int_a^x k(x,t)u(t)dt \qquad (1.7)$$

- Linear Volterra integral equations of the Second kind with weakly singular kernel, of the form:

$$u(x) - \int_0^x \frac{t^{\mu-1}}{x^{\mu}}\, u(t)dt = f(x)\,, \quad x \in [0,T] \qquad (1.8)$$

1.3 Existence and Uniqueness:

In this section, we will try to impose a certain conditions in order to prove the existence and uniqeness theorem for integral equation to be applied to linear VIE's of the second kind. Before we prove existence and uniqueness, we present some definitions; background and review which we will need to prove the main results of this section.

Definition (1.12): [8]

Let$\{f_n(t)\}$ be a sequence of functions from an interval [a,b] to real numbers, then:

- $\{f_n(t)\}$ is **uniformly bounded on [a,b]** if there exists M such that n a positive integer and $t \in [a,b]$ imply$|f_n(t)| \leq M$.
- $\{f_n(t)\}$ is **equicontinuous** if for any $\epsilon > 0$ there exists $\delta > 0$**,**such that **:**

$[n$ is a positive integer,$t_1,\ t_2 \in$ [a, b]$and|t_1 - t_2| < \delta]$ imply $|f_n(t_1) - f_n(t_2)| < \epsilon$

Theorem (1.1): [8]

Let $(t_0\ , x_0) \in R^{n+1}$ and suppose there are positive constants a,b and M, such that D=$\{(t,x): |t - t_0| \leq b\}, G: D \to R^n$ is continuous, and $|G(t,x).| \leq M, if\ (t,x) \in D$ Then there is at least one solution $x(t) of$:

$$x' = G(t,x), x(t_0) = x_0 \qquad (1.9)$$

and $x(t)$ is define for $|t - t_0| \leq T$ with T=min$\{a, b, M\}$

Definition (1.13): [27]

Let $U \subset R^{n+1}$ and $G: U \to R^{n+1}$ we say that **G satisfies a local lipscitz condition with respect to x**, if for each compact subset M of U there is a constant k such that $(t, x_1) and\ (t, x_2)$ in M implies:

$$|G(t,x_1) - G(t,x_2)| \leq K|x_{1-}x_2| \qquad (1.10)$$

Theorem (1.2): [8]

Let the conditions of theorem (1.1) hold and suppose that there is a constant L such that for all $(t, x_1), (t, x_2) \ \epsilon D$ implies:

$$|G(t, x_1) - G(t, x_2)| \leq L|x_1 - x_2|$$

Then (1.9) has only one solution.

Definition (1.14): [27]

A pair (Ω, P) is a **metric space** if Ω is a non-empty set and p: Ω x Ω $\rightarrow [0, \infty)$ *such* that when y, z and u are in Ω, then:

$a)\ P(y, z) \geq 0\ and\ P(y, y) = 0.$

$b)\ P(y, z) = P(z, y).$

$c)\ P(y, z) \leq P(y, u) + P(u, z).$

Definition (1.15): [8]

Let (Ω, P) be a metric space and A: $\Omega \rightarrow \Omega$ the operator A is a **contraction operator** if there is an $\alpha \in (0,1)$ such that:

$$x \in \Omega\ and\ y \in \Omega\ imply\ P[A(x), A(y)] \leq \alpha P(x, y)$$

Theorem (1.3): (contractive mapping principle) [8]

Let (Ω, p) be a complete metric space and A: $\Omega \rightarrow \Omega$ a contraction operator. Then there is a unique $\emptyset \in \Omega$ with A ($\emptyset$) = $\emptyset$.

Theorem (1.4): [8]

Let a,b and L be positive number, and for some fixed $\alpha \in (0,1)$,define $c = \alpha b$. Suppose:

a) f is continuous on $[0, a]$, also intgrable and bonded and satisfy Lipshitz condition.

b) k is continuous on U={(t, s, x) : 0≤ $s, t \leq a$ and $|x - f(t)| \leq b\}$

c) k satisfies Lipshitz condition with respect to x on U

$$|k(t,s,x) - k(t,s,y)| \leq L|x - y|$$

if (t, s, x), (t, s, y) ∈ U. If M $= \max_t |k(t,s,x)|$,

then there is a unique solution of:

u(t)= f(t) +$\int_0^t k\big(t,s,u(s)\big)ds$

on [0,T],where T=min$\{a, \frac{b}{M}, c\}$.

1.4Analytical Methods for Solving VIE's:

In this section, some methods which have been used for solving linear VIE's of second kind and VIE'S with weakly singular kernel have been studied and illustrated by examples.

1.4.1 Solution of Linear VIE's of the Second Kind:[44]

We will first define Volterra integral equations of the second kind given by:

$$u(x) = f(x) + \int_a^x k(x,t)\, u(t)dt \quad a \leq x \leq b$$

The unknown function *u*(*x*), that will be determined, occurs inside and outside the integral sign. The kernel *k*(*x*, *t*) and the function *f*(*x*) are given continues functions.

1.4.1.1Adomain Decomposition Method:[44]

The Adomian decomposition method (ADM) was introduced and developed by George Adomian.The Adomian decomposition method consists of decomposing the unknown function *u*(*x*) of any equation into a

sum of an infinite number of components defined by the decomposition series:

$$u(x) = \sum_{n=0}^{\infty} u_n(x) \qquad (1.11)$$

or equivalently

$$u(x) = u_0(x) + u_1(x) + u_2(x) + \cdots$$

where the components $u_n(x), n \geq 0$ are to be determined in a recursive manner. The decomposition method concerns itself with finding the components individually; we substitute (1.11) into the Volterra integral equation to obtain

$$\sum_{n=0}^{\infty} u_n(x) = f(x) + \int_a^x k(x,t) \left(\sum_{n=0}^{\infty} u_n(t)\right) dt \quad (1.12)$$

The zeroth component $u_0(x)$ is identified by all terms that are not included under the integral sign. Consequently, the components $u_j(x)$, $j \geq 1$ of the unknown function $u(x)$ is completely determined by setting the recurrence relation:

$$u_0(x) = f(x),$$

$$u_{n+1}(x) = \int_a^x k(x,t)\, u_n(t)\, dt \quad , n \geq 0 \quad (1.13)$$

Example (1.1):[44]

To Solve the following Volterra integral equation:

$$u(x) = 1 - \int_0^x u(t)\, dt \qquad (1.14)$$

where $f(x) = 1 and\ k(x,t) \equiv -1$,

Substituting decomposition series (1.11) in to both side of VIE (1.14) gives,

$$\sum_{n=0}^{\infty} u_n(x) = 1 - \int_0^x \sum_{n=0}^{\infty} u_n(t)dt$$

We identify the zeroth component by all terms that are not included under the integral sign. Therefore, we obtain the following recurrence relation:

$$u_0(x) = 1,$$
$$u_{k+1}(x) = -\int_0^x u_k(t)\, dt \quad , k \geq 0$$

so that

$$u_0(x) = 1,$$

$$u_1(x) = -\int_0^x u_0(t)\, dt = \int_0^x 1\, dt = -x \ ,$$

$$u_2(x) = -\int_0^x u_1(t)\, dt = -\int_0^x -t\, dt = \frac{x^2}{2!} \ ,$$

$$u_3(x) = -\int_0^x u_2(t)\, dt = -\int_0^x \frac{t^2}{2!}\, dt = -\frac{x^3}{3!} \ ,$$

$$u_4(x) = -\int_0^x u_3(t)\, dt = -\int_0^x -\frac{t^3}{3!}\, dt = \frac{x^4}{4!} \ ,$$

And so on. Gives the series solution

$u(x) = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots = e^{-x}$.Which is the exact solution for equation (1.14).

1.4.1.2 The Successive Approximations Method:[9,44]

The successive approximations method, also called the Picard iteration method .This method solves any problem by finding successive approximations to the solution by starting with an initial guess, called the

zeroth approximation. As will be seen, the zeroth approximation is any selective real-valued function that will be used in a recurrence relation to determine the other approximations. The successive approximations method introduces the recurrence relation

$$u_n(x) = f(x) + \int_a^x k(x,t)\, u_{n-1}(t)\ dt \quad , n \geq 1 \qquad (1.15)$$

where the zeroth approximation $u_0(x)$can be any selective real valued function. We always start with an initial guess for $u_0(x)$,mostly we select 0, 1, *x for* $u_0(x)$, and by using (1.15), several successive approximations $u_k(x), k \geq 1$ will be determined as:

$$u_1(x) = f(x) + \int_a^x k(x,t)\, u_0(t)\ dt$$

$$u_2(x) = f(x) + \int_a^x k(x,t)\, u_1(t)\ dt$$

$$u_3(x) = f(x) + \int_a^x k(x,t)\, u_2(t)\ dt$$

$$\vdots$$

$$u_n(x) = f(x) + \int_a^x k(x,t)\, u_{n-1}(t)\, dt$$

The successive approximations method or the Picard iteration method will be illustrated by the following example.

Example (1.2): [44]

To solve the following Volterra integral equation by using the successive approximations method,

$$u(x) = -1 + e^x + \frac{1}{2}x^2 e^x - \frac{1}{2}\int_0^x tu(t)\ dt \qquad (1.16)$$

For the zeroth approximation $u_0(x)$, we select

$u_0(x) = 0,$

We next use the iteration formula

$$u_{n+1}(x) = -1 + e^x + \frac{1}{2}x^2e^x - \frac{1}{2}\int_0^x tu_n(t)(t)\ dt \quad , n \geq 0 \ (1.17)$$

Substituting $u_0(x)$ in equation (1.17), we obtain

$$u_1(x) = -1 + e^x + \frac{1}{2}x^2e^x$$

$$u_2(x) = -3 + \frac{1}{4}x^2 + e^x\left(3 - 2x + \frac{5}{4}x^2 - \frac{1}{4}x^3\right),$$

$$u_3(x) = x\left(1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots\right),$$

$u_{n+1}(x) = x(1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots) = xe^{-x}$. Which is the exact solution for equation (1.16).

1.4.1.3 The Laplace Transformation Method: [9,18,44]

The Laplace transformation method can be used for solving integral equation, It was stated that if the kernel depends on the difference $(x - t)$ .Then by taking Laplace transform for both sides of VIE's we find:

$$U(s) = F(s) + K(s)U(s) \qquad (1.18)$$

Where $U(s) = L\{u(x)\}, K(s) = L\{k(x)\}, F(s) = L\{f(x)\}$

The solution of $u(x)$ is obtained by taking the invers of Laplace transform of

$$U(s) = \frac{F(s)}{1 - K(s)} \qquad , K(s) \neq 0$$

Then we find

$$u(x) = L^{-1}\{\frac{F(s)}{1 - K(s)}\}$$

This method will be illustrated by example (1.3).

Example (1.3):[44]

To solve the following Volterra integral equation:

$$u(x) = 1 - \int_0^x (x - t)\, u(t)\ dt \qquad (1.19)$$

Where $f\ (x) = 1 and\ k(x, t) = (x - t)$, Taking Laplace transforms of both side of equation (1.19) gives:

$$L\{u(x)\} = L(1) - L\{(x)\ L(u)\},$$

So that

$$\text{U(s)} = \ \frac{1}{s} - \frac{1}{s^2} U(S)$$

$$\text{U(s)} = \frac{S}{1+S^2}$$

By taking the invers of Laplace transform, of U(s), we obtain that $u(x) = cosx$,which is the exact solution for equation (1.19).

1.4.2 Solution of Linear VIE's of the Second kind with weakly singular kernel[10,11,20]

We consider the second kind VIE's with weakly singular kernel

$$u(x) - \int_0^t \frac{t^{\mu-1}}{x^{\mu}}\, u(t)dt = f(x)\,, \quad x \in [0,T]$$

where $0< \mu<1$ and f is known function. However, there is a singularity at t=0 and s=0 for any positive value of t.

1.4.2.1 Analytic Method

In [10] the author gives suggestion for the analytic solution to solve linear VIE's of the second kind with weakly singular kernel.

Lemma1.1: [10]

(a) If $0 < \mu \leq 1$ and $f \in C^1$[0,t] (with f (0) = 0 if μ =1) then VIE's of the second kind with weakly singular kernel (1.8), has a family of solution $u \in C\,[0,t]$

$$u(t) = c_0 t^{1-\mu} + f(t) + \gamma + t^{1-u} \int_0^t s^{\mu-2}\big(f(s) - f(0)\big)ds, \qquad (1.20)$$

where

$$\gamma = \begin{cases} \frac{1}{\mu-1}\ f(0) & if\ \mu < 1, \\ 0 & if\ \mu = 1, \end{cases} \qquad (1.21)$$

and c_0 is an arbitrary constant. Out of family of solutions there is one particular solution $u \in C^1[0,t]$. Such a solution is unique and can be obtained from (1.20) by taking $c_0 = 0$.

$(\boldsymbol{b}) if \mu > 1 and\ f \in C^m\,[0,t], m \geq 0,$ then the unique solution $u \in C^m[0,t]\ is$:

$$u(t) = f(t) + t^{1-\mu} \int_0^t s^{\mu-2} f(s)ds \qquad (1.22)$$

We note that (1.22) can be obtained from (1.20) with $c_0 = 0$. Indeed, from it follow (1.20) that

$$c_0 = \lim_{t \to 0+} t^{\mu-1} u(t),$$

and this limit is zero when $\mu > 1$. In principle, if we know the value of c_0 we may use (1.20) to obtain the numerical approximations of the solution.

1.5 Discussion

In this chapter a simple review of VIE's, especially in section three. After that we adopt analytic method for solving linear VIE's including Adomain decomposition method, the successive approximations method and the Laplace transform method. Also solve VIE's with weakly singular kernel. There are other kinds of methods may be used to solve VIE's, which also numerical method.

There are many reasons that prove the necessity of numerical method:

1. Many problems cannot be solved using analytical methods.

2. Digital computers are not designed to solve problems when analytic methods are used.

3. When a function is given in tabular form.

4. New method is always needed to solve integral equations because no

single method work well for all such equations.

1.6 The Aim of this Thesis:

The main purpose of this thesis is to introduce new numerical method for a first time using non-polynomial spline functions for solving of the second kind linear VIE's and VIE's with weakly singular kernel, also ,we try to solve VIE's of the first kind with $k(x, x) \neq 0$.Finally, writing

successful programs for the given numerical methods by using MATLAB\ R12,2012 .

1.7 Test Examples

In this thesis, the following test examples will be considered

Test Example 1: Consider the VIE of the second kind [36]:

$$\emptyset(x) = x + \int_0^x (t - x)\emptyset(t)dt$$

With exact solution$\emptyset(x) = sinx$.

Test Example 2: Consider VIE of the second kind [43]:

$$y(x) = 1 + \int_0^x (t - x)y(t)dt$$

With exact solution $y(x) = cosx$.

Test Example 3: Consider the VIE of the second kind:

$$u(x) = 2x + 5 - 3e^x + \int_0^x e^{x-t}u(t)dt$$

With exact solution $u(x) = x + 2$.

Test Example 4: Consider the VIE of the second kind [36]:

$$u(x) = x3^x + \int_0^x 3^{x-t}u(t)dt$$

With exact solution $u(x) = 3^x(1 - e^{-x})$.

Test Example 5: Consider the VIE of the second kind [37]:

$$u(x) = 1 - x + \frac{x^2}{2} + \int_0^x (t - x)\, u(t)dt$$

With exact solution $u(x) = (1 - \sin(x))$.

Test Example 6: Consider the VIE of the second kind:

$$u(x) = \ x + e^x + x^2 - \frac{1}{2}x^4 - x^2e^x + \int_0^x x^2u(t)\ dt$$

The exact solution is $u(x) = x + e^x$.

Test Example 7: Consider the VIE of the first kind [22]:

$$\int_0^x cos(x - t)y(t)dt = sinx$$

With exact solution, $y(x) = 1$.

Test Example 8: Consider the VIE of the first kind [22]:

$$\int_0^x cos(x - t)y(t)dt = 1 - cosx$$

With exact solution, $y(x) = x$.

Test Example 9: Consider the VIE of the second kind with weakly singular kernel [15]:

$$u(x) - \int_0^x \frac{t^{\mu-1}}{x^\mu}\, u(t)dt = f(x)\,, \quad x \in [0, T]$$

Where f (x) =x+1, the exact solution is $u(x) = \frac{\mu}{\mu-1} + \frac{\mu+1}{\mu}\, x$.

Test Example 10: Consider the VIE of the second kind with weakly singular kernel:[10]

$$u(x) - \int_0^x \frac{t^{\mu-1}}{x^{\mu}}\, u(t)dt = f(x)\,, \quad x \in [0,T]$$

Where f (x) =x^2+ x +1,the exact solution is $u(x) = \frac{\mu}{\mu-1} + \frac{\mu+1}{\mu}\, x + \frac{\mu+2}{\mu+1}\, x^2$.

Test Example 11: Consider the VIE of the second kind with weakly singular kernel:

$$u(x) - \int_0^x \frac{t^{\mu-1}}{x^{\mu}}\, u(t)dt = f(x)\,, \quad x \in [0,T]$$

Where f (x) =0.71428571*x^3, and exact solution is u$(x) = x^3$.

Test Example 12: Consider the VIE of the second kind with Weakly Singular Kernel:

$$u(x) - \int_0^x \frac{t^{\mu-1}}{x^{\mu}}\, u(t)dt = f(x)\,, \quad x \in [0,T]$$

Wheref $= 0.714285714285714285714285714285 * x^3 -$ 0.600*x^2, ,and exact solution is $u(x) = x^3$-x^2.

Test Example13: Consider test example (10), with $\mu = 0.4$.

Test Example14: Consider test example (9), with $\mu = 0.6$.

2.1 Introduction

A polynomial is a mathematical expression involving a sum of powers in one or more variables multiplied by a coefficient [50]. Polynomials have long been the function most widely used to approximate other functions mainly ,because of their simple mathematical properties. However, it is well known that polynomial of high degree tend to oscillate strongly and in many cases they are liable to produce very poor approximation.

Spline functions are piecewise polynomials of degree n joined togather at the break points with n-1 continuous derivatives. The break points of splines are called knot [33].With spline functions, we combine low degree and hence weakly oscillating polynomial in such a way to obtain a function which is as smooth as possible in the sense that it has maximal continuity intervals without being globally a polynomial. Spline functions can be integrated and differentiated due to being piecewise polynomials and can be easily stored and implemented on digital computers [17].

A piecewise non-polynomial spline function is a blend of trigonometric, as well as, polynomial basis functions, which form a complete extended Chebyshev space. This approach ensures enhanced accuracy and general form to the existing spline function. A parameter is introduced in the trigonometric part of the spline function. The $C^{\infty}-$ differentiability of the trigonometric part of non-polynomial splines compensates for the loss of soomthes inherited by spline function. It is well known that the Bezier basis is a basis for the degree n algebraic polynomials.

$$s_n = span\{1, x, x^2, \dots, x^n\} \qquad (2.1)$$

A new basis, called the c-Bezier basis, is constructed in [13] , for the space

$$\lceil(n) = span\{1, x, x^2, \dots, x^{n-2}, cosx, sinx\} \qquad (2.2)$$

in which x^{n-1} and x^n in (2.1) replaced by $cosx$ and $sinx$ .There is a wide use to non-polynomial spline functions, see[4,12,16,32,38,46] . This chapter is organized as follows: In section (2.2), we define some types of musty used polynomial spline functions, first the classic polynomial spline function involving the linear, quadratic, cubic, and second we define the B-spline function. In section (2.3), we offer non-polynomial spline functions, then we derive linear and quadratic non-polynomial spline functions which is rudiment to this work.

2.2 Some types of polynomial spline functions:

Before we mention some type of polynomial spline function, we recall some basic definitions:

Definition (2.1): [13]

A function S is called a **spline of degree k** if:

1. The domain of S is an interval [a, b],
2. $S \epsilon C^{k-1}[a, b]$.

3. There are t_i (the knots of S) s.t $a = x_0 < x_1 < x_2 \dots < x_n = b$ and such that S is polynomial of degree at most k on each subinterval $[x_i, x_{i+1}]$.

Definition (2.2): [7]

A spline function is called **natural spline** if it satisfies the following another condition:

$$S_k^m(x_0) = S_k^m(x_n) = 0$$

Definition (2.3):[2]

A spline function is called **clamped spline** if it satisfies the additional condition:

$$S^{(m-1)}(x_0) = u^{(m-1)}(x_0)$$

and

$$S^{(m-1)}(x_n) = u^{(m-1)}(x_n) \ .$$

There are many other type of polynomial spline functions, for example M –spline, L-spline, G-spline, P-spline …,etc[50].

In this section, we define two type of spline functions which are the classic spline function and B-spline function.

2.2.1 Classic spline function:

In this section, we define the most usely spline function. The classic spline functions consist of three simple spline kinds for approximation and interpolation of data.

2.2.1.1 Linear classic spline function:

Definition (2.4): [48]

A function L is called **linear spline** if it satisfys:

1. There is a partion of the interval a=$x_0 < \cdots < x_n = b$, such that L is polynomial of degree 1 on each subinterval$[x_i, x_{i+1}]$.
2. L is continuous on [a ,b],*i.e,*.

$$L(x) = \begin{cases} l_0(x) & , \quad x \in [x_0, x_1] \\ l_1(x) & , \quad x \in [x_1, x_2] \\ \vdots & \\ l_{n-1}(x) & , \quad x \in [x_{n-1}, x_n] \end{cases} \qquad (2.3)$$

where x_0 , x_1 , … , x_n are called knots ,and each piece of L(x) has the form:

$$l_i(x) = a_i x + b_i \qquad (2.4)$$

Where a_i, b_i are the coefficients of liner classic spline function (2.4).

2.2.1.2 Quadratic classic spline

Definition (2.5): [46]

A function Q is called a **Quadratic spline** if it satisfys:

1. Q, $Q'$are continuous on [a,b].

2. Q is polynomial of degree at most 2 on each subinterval $[x_i, x_{i+1}]$, where

$a = x_0 < x_1 < \cdots < x_n = b$. Quadratic spline has a form:

$$q_i = a_i + b_i(x - x_i) + c_i(x - x_i)^2 \qquad (2.5)$$

2.2.1.3 Cubic classic spline:

Definition (2.6): [48]

A function S is called a **Cubic spline** if it satisfys:

1. S, S', S'' are continuous on [a,b].

2. S is polynomial of degree at most 3 on each subinterval $[x_i, x_{i+1}]$, where $a = x_0 < x_1 \ldots < x_n = b$. Cubic spline has a form:

$$s_i = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3 \qquad (2.6)$$

The x_i are the knots and assumed to be arranged in ascending order. The spline function S, which we are constructing, consists of n cubic polynomial pieces:

$$S(x) = \begin{cases} s_0(x) & , x \in [x_0, x_1] \\ s_1(x) & , x \in [x_1, x_2] \\ \vdots & \\ s_{n-1}(x) & , x \in [x_{n-1}, x_n] \end{cases} \qquad (2.7)$$

Here s_i denotes the cubic polynomial that will be used on the subinterval $[x_i, x_{i+1}]$. The interpolation condition is $s_i(x_i) = u_i$, $(0 \le i \le n)$.

2.2.2 B-Spline function:

Before we introduce the definition of B-spline function .we want to manifest a concept of Bezer curve which has widely employed in graphical application.

Definition (2.7):[32]

Let $p_0, p_1, \dots, p_n$ be $n+1$ points ordered in the plane. The oriented polygon formed by them is called the **characteristic polygon** or **Bezier polygon**. Let us introduce the Bernstein polynomials over the interval [0,1], defined as

$$b_{n,k}(t) = \binom{n}{k} t^k (1-t)^{n-k} = \frac{\mathrm{n!}}{\mathrm{k!}\,(n-k)!} t^k (1-\mathrm{t})^{n-k}$$

for n = 0,1,... and k=0,..,n , they may be obtained by the following recursive formula:

$$b_{n,0}(t) = (1-t)^n$$

$$b_{n,k}(t) = (1-t)\ \ b_{n-1,k}(t) + t\, b_{n-1,k-1}(t) \quad k = 1, \dots n,\ \ t \in [0,1]$$

It is easily seen that $b_{n,k} \in P_{n,}$ for $k = 0,1, \dots n$,where $P_{n,}$ is the space of all polynomial of degree n

Also, $\{ b_{n,k,} k = 0 \dots n\}$ provides a bases for P_n.The **Bezier curve** is defined as follows:

$$B_n(p_0, .., p_n, \mathrm{t}) = \sum_{k=0}^{n} p_k\ b_{n,k}(t)\ 0 \le t \le 1 \qquad (2.7)$$

The Bezier curves also is obtained by a pure geometric approach starting from the characteristic polygon. Indeed for any fixed $t \in [0,1]$, we define

$p_{i,1}(t) = (1-t)p_i + tp_{i+1}(t)$ for i= 0,1,..$n-1$

for t fixed , we can repeat the procedure by generating the new vertices $p_{i,2}(t)$,for i=0,1,…, $n-2$, and terminating as soon as the polygon comprises only the vertices $p_{0,n}(t)$and $p_{1,n-1}(t)$. It can be shown that:

$p_{0,n}(t) = (1-t) = p_{0,n-1}(t) + tp_{1,1-1}(t)$

That is, $p_{0,n}(t)$ is equal to the value of the of the Bezier curve B_n at the points corresponding to the fixed value of t.

Definition (2.8):[32]

The **normalized B-spline** $B_{i,k+1}$ of degree k relative to the distinct nodes $x_i, \dots, x_{i+k+1}$ is defined as:

$$B_{i,k+1} = (x_{i+k+1} - x_i)g(x_i, \dots, x_{i+k+1}) \qquad (2.8)$$

where

$$g(t) = (t-x)_+^k = \begin{cases} (t-x)^k & if\ x \le t \\ 0 & otherwise \end{cases} \qquad (2.9)$$

By Newton form of the interpolating polynomial we have,(see [32]):

$$f[x_0, \dots, x_n] = \sum_{i=0}^{n} \frac{f(x_i)}{w'_{n+1}(x_i)} \qquad (2.10)$$

Where $w_{n+1} = \prod_0^n (x - x_i)$

Subtitling in (2. 8), we get:

$$B_{i,k+1}(x) = \left(x_{i+k+1} \quad -x_i \quad\right) \sum_{j=0}^{k+1} \frac{(x_{i+k}-x)_+^k}{\prod_{\substack{i=0 \\ i\neq j}}^{k+1}(x_{i+j} \quad -x_{i+1})}$$

The B- spline admits the following recursive formulation [a,b], see figure (2.6)

$$B_{i,1}(x) = \begin{cases} 1 & if \quad x \in \quad [x_i, x_{i+1}] \\ 0 & other\ wise \end{cases}$$

$$B_{i,k+1}(x) = \frac{\mathrm{x} - x_i}{x_{i+k} - x_i} B_{i,k}(x) + \frac{x_{i+k+1} - x}{x_{i+k+1} \quad -x_{i+1}} B_{i+1,k}(x)$$

compared to Bezier curve; a B-spline curve has a few distinct features

(a) The degree of the curve is independent from the total number of the control point.

(b) It is made out of several curve segments that are joined smoothly.

(c) It is locally propagates.

2.3Non- Polynomial Spline Function:

Consider the partition $\Delta = \{t_0, t_1, t_2, \dots, t_n\}$ of *[a,b]*$\subset \mathbb{R}$. Let S(Δ) denote the set of piecewise polynomials on subinterval $\mathrm{I_i} = [\mathrm{t_i}, \mathrm{t_{i+1}}]$ of partition Δ. Let *u(t)* be the exact solution, this new method provides an approximation not only for $u(t_i)$ at the knots but also $u^{(n)}(t_i)$,n=1,2,…, at every point in the range of integration .Also ,C^∞ the differentiability of the trigonometric part of non-polynomial splines compensates for loss of smoothness inherited by polynomial [4,38,39] .The non-polynomial spline function, obtained by the segment $P_i(t)$.Each non- polynomial spline of n order $P_i\ (t)$ has the form*:*

$$P_i(t) = a_i cosk(t - t_i) + b_i sink(t - t_i) + \cdots + y_i(t - t_i\ )^{n-1} + z_i$$

where $a_i, b_i, \dots, y_i$ and z_i constants and k is the frequency of the trigonometric functions which will be used to raise the accuracy of the method.

In this section we introduce different types of non-polynomial spline functions, linear non-polynomial spline function, the span of linear is x^3, and quadratic non –polynomial spline function, the span of quadratic is x^4. The main advantage of non-polynomial spline function is to obtain method for solving VIE's with higher accuracy.

2.3.1 Linear Non-Polynomial Spline Function

The form of the linear non-polynomial spline function is:

$$P_i(t) = a_i cosk(t - t_i) + b_i sink\,(t - t_i) + c_i(t - t_i) + d_i$$

$$i = 0, .., n \qquad (2.10)$$

where $a_i, b_i, c_i,$ and d_i are constants to be determined . In order to obtain the value of $a_i, b_i, c_i,$ and d_i , we differentiate equation (2. 10) three times with respect to t ,then we get:

$$\left.\begin{aligned} p_i'(t) &= -ka_i sink(t - t_i) + kb_i cosk\,(t - t_i) + c_i \\ p_i''(t) &= -k^2 a_i cosk(t - t_i) - k^2 b_i sink\,(t - t_i) \\ p_i'''(t) &= k^3 a_i sink(t - t_i) - k^3 b_i cosk(t - t_i) \end{aligned}\right\} \quad (2.11)$$

Hence replace t by t_i in the relation (2.10) and (2.11) yields:

$$P_i(t_i) = a_i + d_i$$

$$p_i'(t_i) = kb_i + c_i$$

$$p_i''(t_i) = -k^2 a_i$$

$$p_i'''(t_i) = -k^3 b_i$$

From the above equations, the values of $a_i, b_i, c_i,$ and d_i are obtained as follows:

$$a_i = -\frac{1}{k^2} p_i''(t_i) \qquad (2.12)$$

$$b_i = -\frac{1}{k^3} p_i'''(t_i) \qquad (2.13)$$

$$c_i = p_i'(t_i) + k b_i \qquad (2.14)$$

$$d_i = P_i(t_i) + a_i \qquad (2.15) \quad \text{for } i=0,1,\ldots, n$$

2.3.2 Quadratic Non –Polynomial Spline Function

The form of the quadratic non-polynomial spline function is:

$$Q_i(t) = a_i cosk(t - t_i) + b_i sink(t - t_i) + c_i(t - t_i) + d_i(t - t_i)^2 + e_i \qquad (2.16)$$

where a_i, b_i, c_i, d_i and e_i are constants to be determined .In order to obtain the values of a_i, b_i, c_i, d_i and e_i ,we differentiate equation (2.16) four times with respect to t, and then we get the following equations:

$$\begin{aligned} Q_i'(t) &= -k a_i sink(t - t_i) + k b_i cosk\,(t - t_i) + c_i + 2\, d_i(t - t_i) \\ Q_i''(t) &= -k^2 a_i cosk(t - t_i) - k^2 b_i \sin k(t - t_i) + 2\, d_i \\ Q_i'''(t) &= k^3 a_i sink(t - t_i) - k^3 b_i cos\, k(t - t_i) \\ Q_i^{(4)}(t) &= k^4 a_i cosk(t - t_i) + k^4 b_i sink\,(t - t_i) \end{aligned} \qquad (2.17)$$

Hence replace t by t_i in the relation (2.16) and (2.17) yields:

$$Q_i(t_i) = a_i + e_i$$

$$Q_i'(t_i) = k b_i + c_i$$

$$Q_i''(t_i) = -k^2 a_i + 2 d_i$$

$Q_i'''(\mathrm{t_i}) = -\mathrm{k}^3\mathrm{b_i}$

$Q_i^{(4)}(\mathrm{t_i}) = \mathrm{k}^4 a_i$

We obtain the values of $\mathrm{a_i}, \mathrm{b_i}, \mathrm{c_i}, \mathrm{d_i}$ and e_i from the above relations as follows:

$$a_i = \frac{1}{k^4} Q_i^{(4)}(\mathrm{t_i}) \qquad (2.18)$$

$$\mathrm{b_i} = -\frac{1}{k^3} Q_i'''(\mathrm{t_i}) \qquad (2.19)$$

$$\mathrm{c_i} = Q_i'(\mathrm{t_i}) - \mathrm{kb_i} \qquad (2.20)$$

$$d_i = 1/2[Q_i''(\mathrm{t_i}) + \mathrm{k}^2\mathrm{a_i}] \qquad (2.21)$$

$$e_i = Q_i(t_i) - a_i \qquad (2.22)$$

for $i=0,1,\ldots,n$.

3.1 Introduction

Integral equations find special applicability within scientific and mathematical disciplines. To check the numerical method, it is applied to solve different test problems with known exact solutions and the numerical solutions obtained confirm the validity of the numerical method. Many types of equations do not have an analytical solution. Therefore, these problems should be solved by using numerical techniques. In numerical methods, computer codes and more powerful processors are required to achieve accurate results.

Volterra integral equation arises in many scientific application such as the population dynamic, spread of epidemics [44]. Also there are many works concerned about Volterra integral equation, see [5,10,14,22,25, 29,36,37,43].

In this chapter, both linear and quadratic of non-polynomial spline functions have been applied to find the numerical solution of VIE's .New algorithms have been proposed for the first time which is essential in this work. The rest of this chapter is organized as follows:

In section 3.2 linear VIE's of the second kind have been solved using linear and quadratic non-polynomial spline functions in subsection (3.2.1) and (3.2.2) respectively . In section 3.3, will be introduced the solution of linear VIE's of the first kind with $k(x,x) \neq 0$.In section 3.4, we used linear and quadratic non-polynomial spline functions to solve VIE's with weakly singular kernel in subsection (3.4.1) and (3.4.2) respectively. In section 3.5, numerical examples are given for illustrations and compassion between the method has been made and polynomial spline function and other known methods. And finally, in section 3.6, including a discussion for this

.

Algorithm developed for solving those equations using MATLAB\ R12,2012 language programming.(see appendix (c)).

3.2 Solution of Linear VIE's of the Second Kind:

In this section, the linear and quadratic non-polynomial spline functions have been used to find the numerical solution of linear VIE's of the second kind, which has the form:

$$u(x) = f(x) + \int_a^x k(x,t)u(t)dt \qquad x \in [a,b] \tag{3.1}$$

where $k(x,t)$and $f(x)$ are known functions and continuous in C^4 [a,b] , but $u(x)$ is unknown function. To solve the equation (3.1), we need to differentiate equation (3.1) four times with respect to x , by using Libenze formula we realize:

$$u'(x) = f'(x) + \int_a^x \frac{\partial k(x,t)}{\partial x}\, u(t)dt + k(x,x)\ \ u(x) \tag{3.2}$$

$$u''(x) = f''(x) + \int_a^x \frac{\partial^2 k(x,t)}{\partial x^2}\, u(t)dt\ + \left(\frac{\partial k(x,t)}{\partial x}\right)_{t=x}\ \ u(x) + \frac{dk(x,x)}{dx}\, u(x) + +k(x,x)u'(x) \tag{3.3}$$

$$u'''(x) = f'''(x) + \int_a^x \frac{\partial^3 k(x,t)}{\partial x^3}u(t)dt + \left(\frac{\partial^2 k(x,t)}{\partial x^2}\right)_{t=x} u(x) + \frac{d}{dx}\left(\frac{\partial k(x,t)}{\partial x}\right)_{t=x}\ u(x) + \left(\frac{\partial k(x,t)}{\partial x}\right)_{t=x} u'(x) + \ \frac{d^2k(x,x)}{dx^2}u(x) + \ 2\frac{dk(x,x)}{dx}u'(x) + k(x,x)u''(x) \tag{3.4}$$

$$u^{(4)}(x) = f^{(4)}(x) + \int_a^x \frac{\partial^4 k(x,t)}{\partial x^4} u(t)dt + \left(\frac{\partial^3 k(x,t)}{\partial x^3}\right)_{t=x} u(x) +$$

$$\frac{d}{dx}\left(\frac{\partial^2 k(x,t)}{\partial x^2}\right)_{t=x} u(x) + \left(\frac{\partial^2 k(x,t)}{\partial x^2}\right)_{t=x} u'(x) +$$

$$\frac{d^2}{dx^2}\left(\frac{\partial k(x,t)}{\partial x}\right)_{t=x} u(x) + \frac{d}{dx}\left(\frac{\partial k(x,t)}{\partial x^2}\right)_{t=x} u'(x)$$

$$+ \frac{d}{dx}\left(\frac{\partial k(x,t)}{\partial x}\right)_{t=x} u'(x) + \left(\frac{\partial k(x,t)}{\partial x}\right)_{t=x} u''(x)$$

$$+ \frac{d^3 k(x,x)}{dx^3} u(x) + 3\frac{d^2 k(x,x)}{dx^2} u'(x) + 3\frac{dk(x,x)}{dx} u''(x) + k(x,x)u'''(x) \quad (3.5)$$

To complete our procedure for solving VIE's. we substitute x=a in equations(3.1) - (3.5) , then we get :

$$u_0 = u(a) = f(a) \quad (3.6)$$

$$u_0' = u'(a) = f'(a) + k(a,a)u(a) \quad (3.7)$$

$$u_0'' = u''(a) = f''(a) + \left(\left(\frac{\partial k(x,t)}{\partial x}\right)_{t=x}\right)_{x=a} \mathrm{u(a)} + \left(\frac{\partial k(x,t)}{\partial x}\right)_{x=a} \frac{dk(x,x)}{dx} u(a)$$

$$+k(a,a)u'(a) \quad (3.8)$$

$$u_0''' = u'''(a) = f'''(a) + \left(\left(\frac{\partial^2 k(x,t)}{\partial x^2}\right)_{t=x}\right)_{x=a} u(a)$$

$$+\left[\frac{d}{dx}\left[\frac{\partial k(x,t)}{\partial x}\right]_{t=x}\right]_{x=a} u(a) + \left[\left[\frac{\partial k(x,t)}{\partial x}\right]_{t=x}\right]_{x=a} u'(a)$$

$$+\left(\frac{d^2 k(x,x)}{dx^2}\right)_{x=a} u(a) + 2\left(\frac{dk(x,x)}{dx}\right)_{x=a} u'(a)$$

$$+k(a,a)u''(a) \quad (3.9)$$

$$u_0^{(4)} = u^{(4)}(a) = f^4(a) + \left[\left[\frac{\partial^3 k(x,t)}{\partial x^3}\right]_{t=x}\right]_{x=a} u(a) +$$

$$\left[\frac{d}{dx}\left(\frac{\partial^2 k(x,t)}{\partial x^2}\right)_{t=x}\right]_{x=a} u(a) + \left(\frac{\partial^2 k(x,t)}{\partial x^2}\right)_{t=x} u'(a) +$$

$$\left[\frac{d^2}{dx^2}\left[\frac{\partial k(x,t)}{\partial x}\right]_{t=x}\right]_{x=a} u(a) + 2\left[\frac{d}{dx}\left[\frac{\partial k(x,t)}{\partial x}\right]_{t=x}\right]_{x=a} u'(a)$$

$$+\left(\left[\frac{\partial k(x,t)}{\partial x}\right]_{t=x}\right)_{x=a} u''(a) + \left(\frac{d^3 k(x,x)}{dx^3}\right)_{x=a} u(a) + 3\left(\frac{d^2 k(x,x)}{dx^2}\right)_{x=a} u'(a) +$$

$$3\left(\frac{dk(x,x)}{dx}\right)_{x=a} u''(a) + k(a,a)u'''(a) \qquad (3.10)$$

Now, we try to solve equation (3.1) using linear and quadratic non-polynomial spline functions

3.2.1 Using Linear Non-Polynomial Spline Function:

We approximate the solution of linear VIE's of the second kind (3.1) by using linear non-polynomial spline function (2.10). We introduce a method of solution in algorithm (VIE2NPS1):

The Algorithm (VIE2NPS1):

Step 1: Set h = (b-a) /n, $t_i = t_0$ +ih , i=0,…,n,(where t_0 =a, t_n =b) and $u_0 = f(a)$.

Step 2: Evaluate a_0, b_0, c_0 and d_0 by substituting (3.6)-(3.9) in equations (2.12) - (2.15).

Step 3: Calculate $p_0(t)$using step2 and equation (2.10).

Step 4: Approximate $u_1 = p_0(t_1)$

Step 5: For i=1 to n-1 do the following steps:

Step6: Evaluate a_i, b_i, c_i and d_i by using equations (2.12)-(2.15) and replacing. $u(t_i), u'(t_i), u''(t_i)$ and $u'''(t_i)$ *by* $p_i(t_i)$, $p_i'(t_i), p_i''(t_i)$ *and* $p_i'''(t_i)$.

Step 7: Calculate $p_i(t)$ using step 6 and equation (2.10).

Step 8: Approximate $u_{i+1} = p_i(t_{i+1})$

3.2.2 Using Quadratic Non-Polynomial Spline Function:

In order to approximate the solution of linear VIE's of second kind (3.1) by using quadratic non-polynomial spline function (2.16). We present a method of solution in algorithm (VIE2NPS2):

The Algorithm (VIE2NPS2):

Step 1: Set h = (b-a) /n, $t_i = t_0$ +ih , i=0,…,n,(where t_0 =a , t_n =b)

and $u_0 = f(a)$.

Step 2: Evaluate a_0, b_0, c_0, d_0 and e_0 by substituting (3.6)- (3.10) in equations (2.18)- (2.22).

Step 3: Calculate $p_0(t)$using step2 and equation (2.16).

Step 4: Approximate $u_1 = p_0(t_1)$

Step 5: For i=1 to n-1 do the following steps:

Step6: Evaluate a_i, b_i, c_i, d_i and e_i by using equations (2.18)-(2.22)

and replacing $u(t_i), u'(t_i), u''(t_i), u'''(t_i) and\ u^{(4)}(t_i)$

$by\ p_i(t_i)\ , p_i'(t_i),, p_i''(t_i),, p_i'''(t_i)\ and\ p_i^{(4)}(t_i)$.

Step 7: Calculate $p_i(t)$ using step 6 and equation (2.16).

Step 8: Approximate $u_{i+1} = p_i(t_{i+1})$

3.3Solution of VIE's of the FIRST Kind:

In this section, we introduce the solution of VIE's of the first kind which has a form:

$$g(x) = \int_a^x r(x,t)\, u(t)dt \quad , \quad x \in [a,b] \tag{3.11}$$

where $r(x,t)$and $g(x)$ are known functions and continuous in C^4 [a,b] , but $u(x)$ is unknown function. When $r(x,x) \neq 0$, we differentiat equations (3.11) one time with respect to x.Therefore, we get conversion to the second kind, *i.e*;

$$u(x) = \frac{1}{r(x,x)}\,[g'(x) - \int_a^x \frac{\partial r(x,t)}{\partial x}\, u(t)dt\,] \tag{3.12}$$

then we use the non-polynomial spline function and algorithms (VIE2NPS1) and (VIE2NPS2) to solve(3.12).

where $k(x,t) = \frac{1}{r(x,x)} * \frac{\partial r(x,t)}{\partial x}$ and $f(x) = \frac{1}{r(x,x)}\, g'(x)$

3.4 Linear VIE's of the Second Kind with Weakly Singular kernel:

In this section, the linear and quadratic non-polynomial spline functions will be used to compute the numerical solution of linear VIE's of second kind with weakly singular kernel, which is:

$$u(x) - \int_0^x \frac{t^{\mu-1}}{x^{\mu}}\, u(t)dt = f(x)\,, \quad x \in [0,T] \tag{3.13}$$

where $0< \mu<1$ and f is known function. There is a singularity at x=0 and t=0 for any positive value of x. In order to solve (3.13), we multiply both sides of (3.13) by x^{μ} yields to:

$$x^{\mu}\, u(x) - \int_0^t t^{\mu-1}\ u(t)dt = f(x)\, x^{\mu} \qquad (3.14)$$

Hence differentiation (3.14) with respect to x, we get:

$$x^{\mu}u'(x) + \mu x^{\mu-1}u(x) - \frac{1}{x^{1-\mu}}u(x) = \mu x^{\mu-1}f(x) + f'(x)\ \ x^{\mu} \qquad (3.15)$$

And multiplication both sides of (3.15) by $x^{1-\mu}$ yields ,

$$x\, u'(x) + (\mu - 1)u(x) = \mu f(x) + x\, f'(x)$$

Remark 1: Note that $\lim_{x\to 0} \int_0^x \frac{t^{\mu-1}}{x^{\mu}}\ u(t)dt = \frac{u(0)}{\mu}$ .Therefore**,** if $u(0) \neq 0$

we have $u(0) \neq f(0)$, more precisely $u_0 = \frac{\mu}{\mu-1}\mathrm{f}(0)$.

Hence equation (3.13) may be converted into the following form[15]:

$$x\, u'(x) + (\mu - 1)u(x) = \mu f(x) + x\, f'(x) \qquad (3.16)$$

With

$$u_0 = \frac{\mu}{\mu-1}\mathrm{f}(0) \qquad (3.17)$$

Hence differentiate equation (3,16) four times with respect to x, we get:

$$\begin{aligned}
&x\ \mathrm{u}''(x) + \mu\ \mathrm{u}'(x) = (\mu+1)\mathrm{f}'(x)\ + x\ \mathrm{f}''(x)\\
&x\ \mathrm{u}'''(x) + (\mu+1)\mathrm{u}''(x) = (\mu+2)\mathrm{f}''(x) + x\ \mathrm{f}'''(x)\\
&x\ \mathrm{u}^{(4)}(x) + (\mu+2)\mathrm{u}'''(x) = (\mu+3)\mathrm{f}'''(x) + x\ \mathrm{f}^{(4)}(x)\\
&x\, u^{(5)}(x) + (\mu+3)u''''(x) = (\mu+4)f^{(4)}(x) + x\, f^{(5)}(x)
\end{aligned} \qquad (3.18)$$

Hence replace x by a in the relation, yields:

$$u'_0 = \frac{\mu+1}{\mu} f'(a)$$

$$u''_0 = \frac{\mu+2}{\mu+1} f''(a) \qquad (3.19)$$

$$u'''_0 = \frac{\mu+3}{\mu+2} f'''(a)$$

$$u_0^{(4)} = \frac{\mu+4}{\mu+3} f^{(4)}(a)$$

3. *4.1Using* Linear *Non-Polynomial Spline Function*

In order to approximate the solution of linear VIE's of the second kind with weakly singular kernel (3.13) by using linear non-polynomial spline function (2.10).We present a method of solution in algorithm **(**VIE2WSKNPS1):

The Algorithm: (VIE2WSKNPS1):

Step 1: Set h=(b-a)/n; $t_i = t_0 + ih, i = 0,1, \dots, n$, (where$t_0 = a, t_n =$ b)and $u_0 = \frac{\mu}{\mu-1} f(a)$

Step 2: Evaluate $a_0, b_0, c_0,$ and $d_0$by substituting (3.17)and (3.19) in equations (2.12)- (2.15).

Step 3: Calculate $P_0(t)$ using step 2 and equations (2.10).

Step 4: Approximate $u_1 \approx P_0(t_1)$.

Step 5: For i=1 to n-1 do the following steps:

Step 6: Evaluate a_i, b_i, c_i , and d_i using equations (2.12)- (2.15) and replacing u(t) and its derivatives by $P_i(t)$ and its derivative's

Step 7: Calculate $P_i(t)$ using step 6 and equations (2.10).

Step 8: Approximate $u_{i+1} = p_i(t_{i+1})$

3.4.2Using Quadratic *Non-Polynomial Spline Function*

We approximate the solutions of linear VIE's of the second kind with weakly singular kernel by using quadratic non-polynomial spline function (2.16). In the following algorithm (VIE2WSKNPS2):

The Algorithm (VIE2WSKNPS2):

Step 1: Set h=(b-a)/n; $t_i = t_0 + ih, i = 0,1, \dots, n$, (where $t_0 = a, t_n = b$) and $u_0 = \frac{\mu}{\mu-1} f(a)$

Step 2: Evaluate a_0, b_0, c_0, d_0 and e_0 by substituting (3.17) and (3.19) in equations (2.18) - (2.22).

Step 3: Calculate $P_0(t)$ using step 2 and equations (2.16).

Step 4: Approximate $u_1 \approx P_0(t_1)$.

Step 5: For i=1 to n-1 do the following steps:

Step 6: Evaluate $a_i, b_i, c_i , d_i and\ e_i$ substituting in equations (2.18) - (2.22). and replacing u(t) and its derivatives by $P_i(t)$ and its derivative's

Step 7: Calculate $P_i(t)$ using step 6 and equations (2.16).

Step 8: Approximate $u_{i+1} = P_i(t_{i+1})$

3.5Numerical Examples

Test Example (1): Consider the VIE of the second kind [36]

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

With exact solution $\emptyset(x) = sin\, x$. Table (3.1) presents a comparison between the exact and numerical solution using linear and quadratic non-polynomial spline functions, where $P_i(x)$ denotes the approximate solution using non-polynomial spline function, with h=0.1

Table 3.1: Exact and numerical solution of test example (1)

x	Exact solution	$P_i(x)$	
		linear	quadratic
0	0	0	0
0.1	0.099833416646828	0.099833416646828	0.099833416646828
0.2	0.198669330795061	0.198669330795061	0.198669330795061
0.3	0.295520206661340	0.295520206661340	0.295520206661340
0.4	0.389418342308651	0.389418342308650	0.389418342308650
0.5	0.479425538604203	0.479425538604203	0.479425538604203
0.6	0.564642473395035	0.564642473395035	0.564642473395035
0.7	0.644217687237691	0.644217687237691	0.644217687237691
0.8	0.717356090899523	0.717356090899523	0.717356090899523
0.9	0.783326909627483	0.783326909627483	0.783326909627483
1	0.841470984807897	0.841470984807896	0.841470984807896

Table (3.2) present a comparison between the error in our methods and other method in [36], where Error =|exact value –numerical value| and $\|\boldsymbol{err}\|_\infty = \boldsymbol{max}\,|\boldsymbol{Error}|$.

Table 3.2: comparison between the error with reference [36]

x	$Error$ In linear	$Error$ In quadratic	$Error$**obtain in[36]**
0	0	0	0
0.1	0	0	2.0508063e-012
0.2	0	0	8.3558996e-013
0.3	5.5511151231258e-17	5.5511151231258e-17	2.0756558e-013
0.4	1.11022302462516e-16	1.11022302462516e-16	2.4960310e-013
0.5	1.11022302462516e-16	1.11022302462516e-16	3.6565473e-013
0.6	1.11022302462516e-16	1.11022302462516e-16	1.5317015e-013
0.7	1.11022302462516e-16	1.11022302462516e-16	1.1908461e-013
0.8	2.2244604925031e-16	2.2244604925031e-16	2.5211375e-013
0.9	3.33066907387547 e-16 .	3.33066907387547 e-16 .	2.8431391e-013
1	4.44089209850063e-16	4.44089209850063e-16	8.8095244e-013
$\|\|err\|\|_\infty$	**4.44089209850063e-16**	**4.44089209850063e-16**	**2.050806e-012**

Table (3.3) present a comparison between error obtain using linear and quadratic non-polynomial spline functions and polynomial spline function including (1^{st} order and 2^{nd} order [see Appendix (A): Algorithm (**VIE2PS1**) and Algorithm (**VIE2PS1**)]), with h=0.1.

Table 3.3: Comparison between error obtain using polynomial and non-polynomial spline functions

x	Non-polynomial spline		Polynomial spline	
	$Error$ In linear	$Error$ In quadratic	$Error$ **In 1st order**	$Error$ **In 2nd order**
0	0	0	0	0
0.1	0	0	0.0001665833531	0.000166583353
0.2	0	0	0.0986693307950	0.098669330795
0.3	5.55111512312e-17	5.55111512312e-17	0.0955202066613	0.095520206661
0.4	1.110223024625e-16	1.110223024625e-16	0.1894183423086	0.189418342308
0.5	1.110223024625e-16	1.110223024625e-16	0.1794255386042	0.179425538604
0.6	1.110223024625e-16	1.110223024625e-16	0.2646424733950	0.264642473395
0.7	1.110223024625e-16	1.110223024625e-16	0.2442176872376	0.244217687237
0.8	2.224460492503e-16	2.224460492503e-16	0.3173560908995	0.317356090899
0.9	330669073875 e-16	3.330669073875 e-16	0.2833269096274	0.283326909627
1	4.440892098500e-16	4.440892098500e-16	0.341470984807	0.341470984807
$\|\mathbf{err}\|_\infty$	**4.440892098500e-16**	**4.440892098500e-16**	**0.341470984807**	**0.341470984807**

Test Example (2): Consider the VIE of the second kind [43]:

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

With exact solution$y(x) = cosx$. Tables (3.4) present a comparison between the exact and numerical solution of linear and quadratic non-polynomial spline functions where $P_i(x)$ denote the approximate solution using non-polynomial spline function, with h=0.1

Table 3.4: Exact and numerical solution of test example (2)

x	Exact solution	$P_i(x)$	
		linear	quadratic
0	1.000000000000000	1.000000000000000	1.000000000000000
0.1	0.995004165278026	0.995004165278026	0.995004165278026
0.2	0.980066577841242	0.980066577841242	0.980066577841242
0.3	0.955336489125606	0.955336489125606	0.955336489125606
0.4	0.921060994002885	0.921060994002885	0.921060994002885
0.5	0.877582561890373	0.877582561890373	0.877582561890373
0.6	0.825335614909678	0.825335614909678	0.825335614909678
0.7	0.764842187284489	0.764842187284488	0.764842187284488
0.8	0.696706709347165	0.696706709347165	0.696706709347165
0.9	0.621609968270664	0.621609968270664	0.621609968270664
1	0.540302305868140	0.540302305868139	0.540302305868139

Table (3.5) present a comparison between the error in our methods and other method in [43], where error =|exact value –numerical value| and $\|\boldsymbol{err}\|_\infty = \boldsymbol{max}\,|\boldsymbol{Error}|$.

Table 3.5: comparison between the error with reference [43]

x	***Error* in *linear***	***Error* in** quadratic	***Error* obtain in [43]**
0	0	0	-
0.1	0	0	-
0.2	0.1110223024e-16	0.1110223024e-16	0
0.3	0.1110223024e-16	0.1110223024e-16	-
0.4	0.2220446049e-16	0.2220446049e-16	0
0.5	0.2220446049e-16	0.2220446049e-16	-
0.6	0.2220446049e-16	0.2220446049e-16	8.993e-015
0.7	0.3330669073e -16	0.3330669073e -16	-
0.8	0.3330669073e -16	0.3330669073e -16	5.031e-013
0.9	0.3330669073e -16	0.3330669073e -16	-
1	444089209850e-16	444089209850e-16	1.142e-011
$\|\boldsymbol{err}\|_\infty$	**444089209850e-16**	**444089209850e-16**	**1.142e-011**

.

Table (3.6) present a comparison between error obtain using linear and quadratic non-polynomial spline functions and polynomial spline function including (1^{st} order and 2^{nd} order [see Appendix (A)] : Algorithm (**VIE2PS1**) and Algorithm (**VIE2PS1**)]),) with h=0.1.

Table 3.6: Comparison between error obtain using polynomial and non-polynomial spline functions

x	Non-polynomial spline		Polynomial spline	
	Error in linear	***Error in*** quadratic	***Error*** **In 1^{st} order**	***Error*** **In$2^{n/}$order**
0	0	0	0	0
0.1	0	0	0.0049958347219	0.00000416527802
0.2	0.1110223024e-16	0.1110223024e-16	0.0199334221587	0.00493342215875
0.3	0.1110223024e-16	0.1110223024e-16	0.044663510874	0.01466351087439
0.4	0.2220446049e-16	0.2220446049e-16	0.078939005997	0.02893900599711
0.5	0.2220446049e-16	0.2220446049e-16	0.1224174381096	0.04741743810962
0.6	0.2220446049e-16	0.2220446049e-16	0.1746643850903	0.06966438509032
0.7	0.3330669073e -16	0.3330669073e -16	0.2351578127155	0.09515781271551
0.8	0.3330669073e -16	0.3330669073e -16	0.3032932906528	0.12329329065283
0.9	0.3330669073e -16	0.3330669073e -16	0.3783900317293	0.15339003172933
1	444089209850e-16	444089209850e-16	0.4596976941318	0.18469769413186
$\|\mathbf{err}\|_\infty$	**444089209850e-16**	**444089209850e-16**	**0.4596976941318**	**0.18469769413186**

Test Example (3): Consider the VIE of the second kind :

$$u(x) = 2x + 5 - 3e^x + \int_0^x e^{x-t}u(t)dt \qquad 0 \le x \le 1$$

With exact solution $u(x) = x + 2$. Tables (3.7) present a comparison between the exact and numerical solution of linear and quadratic non-polynomial spline functions ,where $P_i(x)$ denote the approximate solution using non-polynomial spline function, with h=0.1

Table 3.7: Exact and numerical solution of test example (3)

x	Exact solution	$P_i(x)$	
		linear	quadratic
0	2.000000000000000	2.000000000000000	2.000000000000000
0.1	2.100000000000000	2.100000000000000	2.100000000000000
0.2	2.200000000000000	2.200000000000000	2.200000000000000
0.3	2.300000000000000	2.300000000000000	2.300000000000000
0.4	2.400000000000000	2.400000000000000	2.400000000000000
0.5	2.500000000000000	2.500000000000000	2.500000000000000
0.6	2.600000000000000	2.600000000000001	2.600000000000000
0.7	2.700000000000000	2.700000000000001	2.700000000000000
0.8	2.800000000000000	2.800000000000001	2.800000000000000
0.9	2.900000000000000	2.900000000000001	2.900000000000000
1	3.000000000000000	3.000000000000001	3.000000000000000

Table (3.8) present a comparison between the error in our methods where error =|exact value –numerical value| and $\|\boldsymbol{err}\|_\infty = \boldsymbol{max}\,|\boldsymbol{Error}|$

.

Table 3.8: comparison between the error linear and quadratic using Non-polynomial spline function

x	***Error linear***	***Error*** quadratic
0	0	0
0.1	0	0
0.2	0	0
0.3	4.440892098500630e-16	0
0.4	4.440892098500630e-16	0
0.5	4.440892098500630e-16	0
0.6	4.440892098500630e-16	0
0.7	4.440892098500630e-16	0
0.8	8.881784197001252e-16	0
0.9	8.881784197001252e-16	0
1	8.881784197001252e-16	0
$\|\boldsymbol{err}\|_\infty$	**8.881784197001252e-16**	**0**

.

Table (3.9) present a comparison between error obtain using non-polynomial spline function including linear and quadratic polynomial spline function including (1^{st} order and 2^{nd} order [see Appendix

(A)]: Algorithm (**VIE2PS1**) and Algorithm (**VIE2PS1**)]),)) with h=0.1.

Table3.9: Comparison between error obtain using polynomial and non-polynomial spline functions

x	**Non-polynomial spline**		**Polynomial spline**	
	Error linear	***Error*** quadratic	***Error*** **In 1st order**	***Error*** **In 2norder**
0	0	0	0	0
0.1	0	0	0	0
0.2	0	0	0.10000000000	0.10000000000
0.3	4.4408920985e-16	0	0.10000000000	0.10000000000
0.4	4.4408920985e-16	0	0.20000000000	0.20000000000
0.5	4.4408920985e-16	0	0.20000000000	0.20000000000
0.6	4.4408920985e-16	0	0.30000000000	0.30000000000
0.7	4.4408920985e-16	0	0.30000000000	0.30000000000
0.8	8.8817841970e-16	0	0.39999999999	0.39999999999
0.9	8.8817841970e-16	0	0.39999999999	0.39999999999
1	8.8817841970e-16	0	0.50000000000	0.50000000000
$\|err\|_\infty$	**8.8817841970e-16**	**0**	**0.50000000000**	**0.50000000000**

Test Example (4): Consider the VIE of the second kind [36]:

$u(x) = x3^x + \int_0^x -3^{x-t}u(t)dt \;\; 0 \le x \le 1$

With exact solution $u(x) = 3^x(1 - e^{-x})$.Tables (3.10) present a comparison between the exact and numerical solution using linear and quadratic non-polynomial spline functions where $P_i(x)$ denote the approximate solution using non-polynomial spline functions, with h=0.1

Table 3.10: Exact and numerical solution of test example (4)

x	Exact solution	$P_i(x)$	
		linear	**quadratic**
0	0	0	0
0.1	0.106213163030966	0.106201860724981	0.106212914772260
0.2	0.225812709291563	0.225627932943987	0.225804620934631
0.3	0.360363539107348	0.359408023974225	0.360301017594445
0.4	0.511612377368213	0.508528520521745	0.511344245618246
0.5	0.681508888598327	0.673822533031535	0.680676279903068
0.6	0.872229243985166	0.855961571523340	0.870121524139087
0.7	1.086202425097018	1.055448836085519	1.081568581971016
0.8	1.326139582081997	1.272614189066096	1.316951386694774
0.9	1.595066801044777	1.507610859197218	1.578229878035268
1	1.896361676485673	1.760413910584314	1.867370419076533

Table (3.11) present a comparison between the error in our methods where error =|exact value –numerical value| and $\|\boldsymbol{err}\|_\infty = \boldsymbol{max}\,|\boldsymbol{Error}|$

.

. **Table 3.11: comparison between the error with reference [36]**

x	***Error linear***	***Error* quadratic**	***Error obtain in* [36]**
0	0	0	inf-0.0074113
0.1	1.130230598539450e-05	2.482587062729857e-07	1.1600483e-002
0.2	1.847763475755215e-04	8.088356932056673e-06	2.8608994e-002
0.3	9.555151331228085e-04	6.252151290275787e-05	2.2232608e-002
0.4	3.083856846467614e-03	2.681317499672042e-04	1.0103823e-002
0.5	7.686355566791314e-03	8.326086952588074e-04	1.7285379e-002
0.6	1.626767246182581e-02	2.107719846078826e-03	6.5041788e-003
0.7	3.075358901149872e-02	4.633843126001569e-03	8.3481474e-003
0.8	5.352539301590098e-02	9.188195387223042e-03	5.7238171e-003
0.9	8.745594184755889e-02	1.683692300950868e-02	1.1187893e-003
1	1.359477659013595e-01	2.899125740913999e-02	1.1830649e-002
$\|\boldsymbol{err}\|_\infty$	**1.359477659013595e-01**	**2.899125740913999e-02**	**2.8608994e-002**

Table (3.12) present a comparison between error obtain using non-polynomial spline function including linear and quadratic polynomial spline function including (1^{st} order and 2^{nd} order [see Appendix

(A)]: Algorithm (**VIE2PS1**) and Algorithm (**VIE2PS1**)]) with h=0.1.

Table 3.12: Comparison between error obtain using polynomial and non-polynomial spline functions

x	Non-polynomial spline		Polynomial spline	
	$Error$ $linear$	$Error$ $quadratic$	$Error$ In 1st order	$Error$ In 2nd order
0	0	0	0	0
0.1	1.1302305985394e-05	2.4825870627298e-07	0.0062131630309	0.0002270401442
0.2	1.8477634757552e-04	8.0883569320566e-06	0.1258127092915	0.1078543406315
0.3	9.5551513312280e-04	6.2521512902757e-05	0.1603635391073	0.1244468017872
0.4	3.0838568464676e-03	2.6813174996720e-04	0.3116123773682	0.2517511485014
0.5	7.6863555667913e-03	8.3260869525880e-04	0.3815088885983	0.2917170452981
0.6	1.6267672461825e-02	2.1077198460788e-03	0.5722292439851	0.4465206633648
0.7	3.0753589011498e-02	4.6338431260015e-03	0.6862024250970	0.5185909842699
0.8	5.3525393015900e-02	9.1881953872230e-03	0.9261395820819	0.7106391581614
0.9	8.7455941847558e-02	1.6836923009508e-02	1.0950668010447	0.8256912711441
1	1.3594776590135e-01	2.8991257409139e-02	1.3963616764856	1.0671249177182
$\|\|err\|\|_\infty$	**1.3594776590135e-01**	**2.8991257409139e-02**	**1.3963616764856**	**1.0671249177182**

Test Example (5): Consider the VIE of the second kind [37]:

$$u(x) = 1 - x + \frac{x^2}{2} + \int_0^x t - x\, u(t)dt \quad 0 \le x \le 1$$

With exact solution $u(x) = (1 - \sin(x))$**.**Tables (3.13) present a comparison between the exact and numerical solution using linear and quadratic non-polynomial spline functions, where $P_i(x)$ denote the approximate solution using non-polynomial spline functions, with h=0.1

Table 3.13: Exact and numerical solution of test example (5)

x	Exact solution	$P_i(x)$	
		linear	**quadratic**
0	1.000000000000000	1.000000000000000	1.000000000000000
0.1	0.900166583353172	0.900166583353172	0.900166583353172
0.2	0.801330669204939	0.801330669204939	0.801330669204939
0.3	0.704479793338660	0.704479793338660	0.704479793338660
0.4	0.610581657691349	0.610581657691350	0.610581657691350
0.5	0.520574461395797	0.520574461395797	0.520574461395797
0.6	0.435357526604965	0.435357526604965	0.435357526604965
0.7	0.355782312762309	0.355782312762309	0.355782312762309
0.8	0.282643909100477	0.282643909100477	0.282643909100477
0.9	0.216673090372517	0.216673090372517	0.216673090372517
1	0.158529015192104	0.158529015192104	0.158529015192104

Table (3.14) present a comparison between the error in our methods and other method in [37], where error =|exact value –numerical value| and $\|\boldsymbol{err}\|_\infty = \boldsymbol{max}\,|\boldsymbol{Error}|$.

Table 3.14: comparison between the error with reference [37]

x	***Error in linear***	***Error* in quadratic**	***Error obtain in* [37]**
0	0	0	-
0.1	0	0	-
0.2	0	0	-
0.3	0	0	-
0.4	2.220446049250310e-16	2.220446049250310e-16	-
0.5	1.110223024625160e-16	1.110223024625160e-16	-
0.6	1.110223024625160e-16	1.110223024625160e-16	-
0.7	1.110223024625160e-16	1.110223024625160e-16	-
0.8	2.220446049250310e-16	2.220446049250310e-16	-
0.9	3.330669073875471e-16	3.330669073875471e-16	-
1	4.440892098500626e-16	4.440892098500626e-16	-
$\|\boldsymbol{err}\|_\infty$	**4.440892098500626e-16**	**4.440892098500626e-16**	**3.6208210e-04**

Table (3.15) present a comparison between error obtain using non-polynomial spline function including linear and quadratic polynomial spline function including (1^{st} order and 2^{nd} order [see Appendix(A): Algorithm (**VIE2PS1**) and Algorithm (**VIE2PS1**)]) with h=0.1.

Table 3.15: Comparison between error obtain using polynomial and non-polynomial spline functions

x	Non-polynomial spline		Polynomial spline	
	$Error$ $in\ linear$	$Error$ in quadratic	$Error$ $in1^{st}$ order	$Error$ $in2^{nd}$ order
0	0	0	0	0
0.1	0	0	0.0001665833531	0.0001665833531
0.2	0	0	0.0986693307950	0.0986693307950
0.3	0	0	0.0955202066613	0.0955202066613
0.4	2.2204460492.e-16	2.2204460492.e-16	0.1894183423086	0.1894183423086
0.5	1.1102230246e-16	1.1102230246e-16	0.1794255386042	0.1794255386042
0.6	1.1102230246e-16	1.1102230246e-16	0.2646424733950	0.2646424733950
0.7	1.1102230246e-16	1.1102230246e-16	0.2442176872376	0.2442176872376
0.8	2.22044604925e-16	2.22044604925e-16	0.3173560908995	0.3173560908995
0.9	3.3306690738e-16	3.3306690738e-16	0.2833269096274	0.2833269096274
1	4.4408920985e-16	4.4408920985e-16	0.3414709848078	0.3414709848078
$\|err\|_\infty$	**4.4408920985e-16**	**4.4408920985e-16**	**0.3414709848078**	**0.3414709848078**

Test Example (6): Consider the VIE of the second kind:

$$u(x) = x + e^x + x^2 - \frac{1}{2}x^4 - x^2e^x + \int_0^x x^2u(t)\ dt \quad 0 \le x \le 1$$

With exact solution $u(x) = x + e^x$ .Tables (3.16) present a comparison between the exact and numerical solution using linear and quadratic non-polynomial spline functions, where $P_i(x)$ denote the approximate solution using non-polynomial spline functions, with h=0.

Table 3.16: Exact and numerical solution of test example (6)

x	Exact solution	$P_i(x)$	
		$linear$	quadratic
0	1.000000000000000	1.000000000000000	-
0.1	1.205170918075648	1.205162418075146	1.205145756963043
0.2	1.421402758160170	1.421264091363697	1.420997779998731
0.3	1.649858807576003	1.649143304213054	1.647797347710631
0.4	1.891824697641270	1.889520663688464	1.885276687676925
0.5	2.148721270700128	2.142991899505424	2.132661651943934
0.6	2.422118800390509	2.410021911695286	2.388679452056574
0.7	2.713752707470476	2.690940125477821	2.651571376339868
0.8	3.025540928492468	2.985937199753312	2.919110362364652
0.9	3.359603111156950	3.295063122101852	3.188623249019196
1	3.718281828459046	3.618226709323964	3.457017485851407

Table (3.17) present a comparison between the error in our methods where error =|exact value –numerical value| and $\|\boldsymbol{err}\|_\infty = \boldsymbol{max}\,|\boldsymbol{Error}|$.

Table 3.17: comparison between the error using Non-polynomial spline functions

x	$\boldsymbol{Error}$ $\boldsymbol{in\ linear}$	$\boldsymbol{Error}$ $\boldsymbol{in}$ **quadratic**
0	0	-
0.1	0.000008500000502	0.000025161112605
0.2	0.000138666796473	0.000404978161439
0.3	0.000715503362949	0.002061459865372
0.4	0.002304033952806	0.006548009964346
0.5	0.005729371194704	0.016059618756194
0.6	0.012096888695223	0.033439348333935
0.7	0.022812581992656	0.062181331130608
0.8	0.039603728739156	0.106430566127816
0.9	0.064539989055098	0.170979862137754
1	0.100055119135082	0.261264342607639
$\|\boldsymbol{err}\|_\infty$	**0.100055119135082**	**0.261264342607639**

Table (3.18) present a comparison between error obtain using linear and quadratic non-polynomial and polynomial spline function including (1st order and 2nd order [see Appendix (A)): Algorithm (**VIE2PS1**) and Algorithm (**VIE2PS1**)]) with h=0.1

Table 3.18: Comparison between error obtain using polynomial and non-polynomial spline functions

x	Non-polynomial spline		Polynomial spline	
	$Error$ $in\ liear$	$Error$ in quadratic	$Error$ In 1st order	$Error$ In 2norder
0.1	0	-	0.00517091807	0.00017091807
0.2	0.000008500000502	0.000025161112605	0.22140275816	0.20640275816
0.3	0.000138666796473	0.000404978161439	0.24985880757	0.21985880757
0.4	0.000715503362949	0.002061459865372	0.49182469764	0.44182469764
0.5	0.002304033952806	0.006548009964346	0.54872127070	0.47372127070
0.6	0.005729371194704	0.016059618756194	0.82211880039	0.71711880039
0.7	0.012096888695223	0.033439348333935	0.91375270747	0.77375270747
0.8	0.022812581992656	0.062181331130608	1.22554092849	1.04554092849
0.9	0.039603728739156	0.106430566127816	1.35960311115	1.13460311115
1	0.064539989055098	0.170979862137754	1.71828182845	1.44328182845
$\|err\|_\infty$	**0.100055119135082**	**0.261264342607639**	**1.71828182845**	**1.44328182845**

Test Example (7): Consider the VIE of the first kind [22]:

$$\int_0^x cos(x-t)y(t)dt = sinx \ 0 \le x \le 1$$

And Exact solution, $y(x) = 1$. To solve this equation we differentiate the equation above one time with respect to x,to obtain :

$$y(x) = cosx + \int_0^x sin(x-t)y(t)dt$$

Which is second kind VIE's, with $f(x) = cosx \ and \ k(x,t) = \sin(x-t)$.

Tables (3.19) present a comparison between the exact and numerical solution using linear and quadratic non-polynomial spline functions, where $P_i(x)$ denote the approximate solution using non-polynomial spline functions, with h=0.1.

Table 3.19: Exact and numerical solution of test example (7)

x	Exact solution	$P_i(x)$	
		linear	**quadratic**
0	1	1	1
0.1	1	1	1
0.2	1	1	1
0.3	1	1	1
0.4	1	1	1
0.5	1	1	1
0.6	1	1	1
0.7	1	1	1
0.8	1	1	1
0.9	1	1	1
1	1	1	1

Table (3.20) present a comparison between the error in our methods and other method in [22], where error =|exact value –numerical value| and $\|\boldsymbol{err}\|_\infty = \boldsymbol{max}\,|\boldsymbol{Error}|$.

Table 3.20: comparison between the error with reference [22]

x	***Error in linear***	***Error*** **quadratic**	***Error obtain in*** **[22]**
0	0	0	
0.1	0	0	1.00166
0.2	0	0	
0.3	0	0	1.00166
0.4	0	0	
0.5	0	0	1.00166
0.6	0	0	
0.7	0	0	
0.8	0	0	
0.9	0	0	
1	0	0	
$\|\boldsymbol{err}\|_\infty$	**0**	**0**	

Table (3.21) present a comparison between error obtain using linear and quadratic non-polynomial and polynomial spline function including (1st order and 2nd order [see Appendix(A): Algorithm (**VIE2PS1**) and Algorithm (**VIE2PS1**)]) with h=0.1.

Table 3.21: Comparison between error obtain using polynomial and non-polynomial spline functions

x	Non-polynomial spline		Polynomial spline	
	$Error$ $in\ linear$	$Error$ quadratic	$Error$ In 1st order	$Error$ In 2nd order
0	0	0	0	0
0.1	0	0	0	0
0.2	0	0	0	0
0.3	0	0	0	0
0.4	0	0	0	0
0.5	0	0	0	0
0.6	0	0	0	0
0.7	0	0	0	0
0.8	0	0	0	0
0.9	0	0	0	0
1	0	0	0	0
$\|\|err\|\|_\infty$	**0**	**0**	**0**	**0**

Test Example (8): Consider the VIE of the first kind [22]:

$\int_0^x cos(x-t)y(t)dt = 1 - cosx\ 0 \le x \le 1$

And Exact solution, $y(x) = x$. To solve this equation we differentiate the equation above one time with respect to x,therefore ,we have :

$$y(x) = sinx + \int_0^x sin(x-t)y(t)dt$$

Which is second kind VIE's, with $f(x) = sinx\ and\ k(x,t) = \sin(x-t)$

Tables (3.22) present a comparison between the exact and numerical solution using linear and quadratic non-polynomial spline functions, where $P_i(x)$ denote the approximate solution using non-polynomial spline functions, with h=0.1.

Table 3.22: Exact and numerical solution of test example (8)

x	Exact solution	$P_i(x)$	
		linear	**quadratic**
0	0	0	0
0.1	0.100000000000000	0.100000000000000	0.100000000000000
0.2	0.200000000000000	0.200000000000000	0.200000000000000
0.3	0.300000000000000	0.300000000000000	0.300000000000000
0.4	0.400000000000000	0.400000000000000	0.400000000000000
0.5	0.500000000000000	0.500000000000000	0.500000000000000
0.6	0.600000000000000	0.600000000000000	0.600000000000000
0.7	0.700000000000000	0.700000000000000	0.700000000000000
0.8	0.800000000000000	0.800000000000000	0.800000000000000
0.9	0.900000000000000	0.900000000000000	0.900000000000000
1	1.000000000000000	1.000000000000000	1.000000000000000

Table (3.23) present a comparison between the error in our methods and other method in [22]. where error =|exact value –numerical value| and $\|\boldsymbol{err}\|_\infty = \boldsymbol{max}\,|\boldsymbol{Error}|$ **,** with h=0.01**.**

Table 3.23: comparison between the error with reference [22]

x	***Error in linear***	***Error in* quadratic**	***Error* obtain in[22]**
0.15	0	2.775557561562900e-17	0.15006
0.45	2.442490654175330e-15	0	0.45019
0.75	4.440892098500601e-16	0	0.75031
$\|\boldsymbol{err}\|_\infty$	6.661338147750939e-16	1.110223024625157e-16	

Table (3.24) present a comparison between error obtain using non-polynomial spline function including linear and quadratic and polynomial spline function including (1^{st} order and 2^{nd} order [see Appendix

(A): Algorithm (**VIE2PS1**) and Algorithm (**VIE2PS1**)]) with h=0.1.

Table 3.24: Comparison between error obtain using polynomial and non-polynomial spline functions

x	Non-polynomial spline		Polynomial spline	
	Error in linear	***Error in* quadratic**	***Error* In 1st order**	***Error* In 2nd order**
0	0	0	0	0
0.1	0	0	0	0
0.2	0	0	0.10000000000	0.10000000000
0.3	0	5.5510e-17	0.10000000000	0.10000000000
0.4	0	0	0.20000000000	0.20000000000
0.5	0	0	0.20000000000	0.20000000000
0.6	0	0	0.30000000000	0.30000000000
0.7	0	0	0.30000000000	0.30000000000
0.8	1.1100e-16	0	0.40000000000	0.40000000000
0.9	1.1100e-16	0	0.40000000000	0.40000000000
1	1.1100e-16	0	0.50000000000	0.50000000000
$\|\|err\|\|_\infty$	1.1100e-16	5.5510e-17	0.50000000000	0.50000000000

Test Example (9): Consider the VIE of second kind with Weakly Singular Kernel [15]:

$$u(x) - \int_0^x \frac{t^{\mu-1}}{x^{\mu}}\, u(t)dt = f(x) \qquad 0 \le x \le 1$$

Where f (x) =x+1 and $= 0.5$, with $u(x) = \frac{\mu}{\mu-1} + \frac{\mu+1}{\mu}\, x$. Tables (3.25) present a comparison between the exact and numerical solution using linear and quadratic non-polynomial spline functions, where $P_i(x)$ denote the approximate solution non-polynomial spline functions, with h=0.1

Table 3.25: Exact and numerical solution of test example (9)

x	Exact solution	$P_i(x)$	
		linear	**quadratic**
0	-1.000000000000000	-1.000000000000000	-1.000000000000000
0.1	-0.700000000000000	-0.700000000000000	-0.700000000000000
0.2	-0.400000000000000	-0.400000000000000	-0.400000000000000
0.3	-0.100000000000000	-0.100000000000000	-0.100000000000000
0.4	0.200000000000000	0.200000000000000	0.200000000000000
0.5	0.500000000000000	0.500000000000000	0.500000000000000
0.6	0.800000000000000	0.800000000000000	0.800000000000000
0.7	1.100000000000000	1.100000000000000	1.100000000000000
0.8	1.400000000000000	1.400000000000000	1.400000000000000
0.9	1.700000000000000	1.700000000000000	1.700000000000000
1	2.000000000000000	2.000000000000000	2.000000000000000

Table (3.26 present a comparison between the error in our methods and other method in [15]where error =|exact value –numerical value| and $\|\boldsymbol{err}\|_\infty = \boldsymbol{max}\,|\boldsymbol{Error}|$

Table 3.26: comparison between the error with reference [15]

x	***Error in linear***	**Error quadratic**	***Error* obtain in [15]**
0.08	2.220446049250300e-16	2.220446049250300e-16	2.1e-0.4
0.16	4.440892098500600e-16	4.440892098500600e-16	5.3e-0.4
0.24	5.551115123125801e-16	5.551115123125801e-16	1.3e-0.3
0.48	5.551115123125801e-16	5.551115123125801e-16	1.3e-0.3
0.64	9.992007221626401e-16	9.992007221626401e-16	8.0e-0.4
0.80	1.332267629550190e-15	1.332267629550190e-15	6.1e-0.4
0.96	2.220446049250310e-15	2.220446049250310e-15	5.1e-0.4
1	2.220446049250310e-15	2.220446049250310e-15	4.9e-0.4
$\|\boldsymbol{err}\|_\infty$	**2.220446049250313e-15**	**2.220446049250313e-15**	

Table (3.27) present a comparison between error obtain using non-polynomial spline function including linear and quadratic polynomial spline function including (1^{st} order and 2^{nd} order [see Appendix(A): Algorithm (**VIE2PS1**) and Algorithm (**VIE2PS1**)]) with h=0.1.

Table 3.27: Comparison between error obtain using polynomial and non-polynomial spline functions

x	Non-polynomial spline		Polynomial spline	
	$Error$ $in\ linear$	$Error$ in quadratic	$Error$ In 1st order	$Error$ In 2nd order
0	0	0	0	0
0.1	0	0	0	0
0.2	0	0	0	0
0.3	0	0	0	0
0.4	0	0	0	0
0.5	0	0	2.220044604 0e-16	2.220044604 0e-16
0.6	2.220044604 0e-16	2.220044604 0e-16	4.441089209 0e-16	4.441089209 0e-16
0.7	4.441089209 0e-16	4.441089209 0e-16	6.66133814 0e-16	6.66133814 0e-16
0.8	6.66133814 0e-16	6.66133814 0e-16	0	0
0.9	0	0	2.22044604 0e-16	2.22044604 0e-16
1	2.22044604 0e-16	2.22044604 0e-16	4.441089209 0e-16	4.441089209 0e-16
$\|\|err\|\|_\infty$	**6.66133814775e-16**	**6.66133814775e-16**	**6.66133814 0e-16**	**6.66133814 0e-16**

Test Example (10): Consider the VIE of second with Weakly Singular Kernel [10]:

$$u(x) - \int_0^x \frac{t^{\mu-1}}{x^{\mu}}\, u(t)dt = f(x) \qquad 0 \le x \le 1$$

Where f (x) $=x^2+x$ +1 and $\mu = 0.5$, with $u(x) = \frac{\mu}{\mu-1} + \frac{\mu+1}{\mu}\, x + \frac{\mu+2}{\mu+1}\, x^2$. Tables 3.28 present a comparison between the exact and numerical solution of non-polynomial spline function including linear and quadratic , where $P_i(x)$ denote the approximate solution use non-polynomial spline functions, with h=0.1.

Table 3.28: Exact and numerical solution of test example (10)

x	Exact solution	$P_i(x)$	
		linear	**quadratic**
0	-1.000000000000000	-1.000000000000000	-1.000000000000000
0.1	-0.683333333333333	-0.683347217593419	-0.683333333333333
0.2	-0.333333333333333	-0.333555259470805	-0.333333333333333
0.3	0.050000000000000	0.048878369581314	0.050000000000000
0.4	0.466666666666667	0.463130019990385	0.466666666666667
0.5	0.916666666666667	0.908058127032092	0.916666666666667
0.6	1.400000000000000	1.382214616967740	1.400000000000000
0.7	1.916666666666666	1.883859375718374	1.916666666666667
0.8	2.466666666666667	2.410977635509450	2.466666666666666
0.9	3.050000000000001	2.961300105764454	3.050000000000000
1	3.666666666666667	3.532325647106203	3.666666666666667

Table (3.29) present a comparison between the error in our methods and other method in [10], where error =|exact value –numerical value| and $\|err\|_\infty = max\ |Error|$ **,with h=0.01.**

Table 3.29: comparison between the error with reference [10]

x	***Error in linear***	***Error in* quadratic**	***Error obtain in* [10]**
$\|err\|_\infty$	1.291328686053168e-01	5.329070518200751e-15	4.03e-1

Table (3.30) present a comparison between error obtain using non-polynomial spline function including linear and quadratic and polynomial spline function including (1^{st} order and 2^{nd} order [see Appendix

(A): Algorithm (**VIE2PS1**) and Algorithm (**VIE2PS1**)]) with h=0.1.

Table 3.30: Comparison between error obtain using polynomial and non-polynomial spline functions

x	Non-polynomial spline		Polynomial spline	
	$Error$ $in\ linear$	$Error$ in quadratic	$Error$ In 1st order	$Error$ In2nd order
0	0	0	0	0
0.1	0	0	1.6666666666666e-02	0
0.2	1.3884260085417e-05	5.551115120e-17	6.6666666666666e-02	5.551115120e-17
0.3	2.2192613747207e-04	2.2204460490e-16	1.5000000000000e-01	2.2204460490e-16
0.4	1.1216304186860e-03	5.551115120e-17	2.6666666666666e-01	5.551115120e-17
0.5	3.5366466762822e-03	0	4.1666666666666e-01	0
0.6	8.6085396345747e-03	0	5.9999999999999e-01	0
0.7	1.7785383032259e-02	4.440892090e-16	8.1666666666666e-01	4.440892090e-16
0.8	3.2807290948292e-02	8.881784190e-16	1.0666666666666e+00	8.881784190e-16
0.9	5.5689031157217e-02	8.881784190e-16	1.3500000000000e+00	8.881784190e-16
1	8.8699894235547e-02	4.440892090e-16	1.6666666666666e+00	4.440892090e-16
$\|\|err\|\|_\infty$	**8.8699894235547e-02**	**8.881784190e-16**	**1.6666666666666e+00**	**8.881784190e-16**

Test Example (11): Consider the VIE of second with Weakly Singular Kernel:

$$u(x) - \int_0^x \frac{t^{\mu-1}}{x^{\mu}}\, u(t)dt = f(x) \qquad 0 \le x \le 1$$

Where f (x) =0.71428571*x^3 , $\mu = 0.5$,with exact solution $u(x) = x^3$. Tables 3.31 present a comparison between the exact and numerical solution using linear and quadratic non-polynomial spline functions, where $P_i(x)$ denote the approximate solution using non-polynomial spline functions, with h=0.1.

Table 3.31: Exact and numerical solution of test example (11)

x	Exact solution	$P_i(x)$	
		$linear$	quadratic
0	0	0	0
0.1	0.001000000000000	0.000999500113034	0.000999500113034
0.2	0.008000000000000	0.007984015181729	0.007984015181729
0.3	0.027000000000000	0.026878759870690	0.026878759870690
0.4	0.064000000000000	0.063489945767158	0.063489945767158
0.5	0.125000000000000	0.123446767634102	0.123446767634102
0.6	0.216000000000000	0.212145158356918	0.212145158356918
0.7	0.343000000000000	0.334693874565692	0.334693874565692
0.8	0.512000000000000	0.495863451627685	0.495863451627685
0.9	0.729000000000000	0.700038538034872	0.700038538034872
1	1.000000000000000	0.951174085445580	0.951174085445580

Table 3.32 present a comparison between the error in our methods where error =|exact value –numerical value| and $\|\boldsymbol{err}\|_\infty = \boldsymbol{max}\ |\boldsymbol{Error}|$

Table 3.32: comparison between the error using Non-polynomial spline function

x	$Error$ $in\ linear$	$Error$ in quadratic
0.1	0	0
0.2	4.998869659316330e-07	4.998869659316330e-07
0.3	1.598481827124777e-05	1.598481827124777e-05
0.4	1.212401293096826e-04	1.212401293096826e-04
0.5	5.100542328419638e-04	5.100542328419638e-04
0.6	1.553232365897550e-03	1.553232365897550e-03
0.7	3.854841643081697e-03	3.854841643081697e-03
0.8	8.306125434307476e-03	8.306125434307476e-03
0.9	1.613654837231471e-02	1.613654837231471e-02
1	2.896146196512817e-02	2.896146196512817e-02
$\|\boldsymbol{err}\|_\infty$	**2.896146196512817e-02**	**2.896146196512817e-02**

Table (3.33) present a comparison between error obtain using non-polynomial spline function including linear and quadratic and polynomial spline function including (1^{st} order and 2^{nd} order [see Appendix(A): Algorithm (**VIE2PS1**) and Algorithm (**VIE2PS1**)]) with h=0.1

Table 3.33: Comparison between error obtain using polynomial and non-polynomial spline functions

x	Non-polynomial spline		Polynomial spline	
	$Error$ $in\ linear$	$Error$ in quadratic	$Error$ In 1^{st} order	$Error$ In 2^{nd} order
0.1	0	0	0.00100000000	0.00100000000
0.2	4.998869659316330e-07	4.998869659316330e-07	0.00800000000	0.00800000000
0.3	1.598481827124777e-05	1.598481827124777e-05	0.02700000000	0.02700000000
0.4	1.212401293096826e-04	1.212401293096826e-04	0.06400000000	0.06400000000
0.5	5.100542328419638e-04	5.100542328419638e-04	0.12500000000	0.12500000000
0.6	1.553232365897550e-03	1.553232365897550e-03	0.21600000000	0.21600000000
0.7	3.854841643081697e-03	3.854841643081697e-03	0.34300000000	0.34300000000
0.8	8.306125434307476e-03	8.306125434307476e-03	0.51200000000	0.51200000000
0.9	1.613654837231471e-02	1.613654837231471e-02	0.72900000000	0.72900000000
1	2.896146196512817e-02	2.896146196512817e-02	1.00000000000	1.00000000000
$\|\boldsymbol{err}\|_\infty$	**2.896146196512817e-02**	**2.896146196512817e-02**	1.00000000000	1.00000000000

Test Example (12): Consider the VIE of second with weakly singular kernel:

$$u(x) - \int_0^x \frac{t^{\mu-1}}{x^{\mu}}\, u(t)dt = f(x) \qquad 0 \le x \le 1$$

Wheref= $0.71428571*x^3 - 0.6*x^2$ $\mu = 0.5$,with exact solution $u(x) = x^3$-x^2.Tables 3.34 present a comparison between the exact and numerical solution using linear and quadratic non-polynomial spline functions ,where $P_i(x)$ denote the approximate solution non-polynomial spline functions, whit h=0.

Table 3.34: Exact and numerical solution of test example (12)

x	Exact solution	$P_i(x)$	
		linear	**quadratic**
0	0	0	0
0.1	-0.009000000000000	-0.008992169330915	-0.009000499886966
0.2	-0.032000000000000	-0.031882829135788	-0.032015984818271
0.3	-0.063000000000000	-0.062448261878098	-0.063121240129310
0.4	-0.096000000000000	-0.094388066227072	-0.096510054232842
0.5	-0.125000000000000	-0.121388108585153	-0.126553232365898
0.6	-0.144000000000000	-0.137183611823726	-0.147854841643082
0.7	-0.147000000000000	-0.135621750865331	-0.155306125434308
0.8	-0.128000000000000	-0.110723129677985	-0.144136548372315
0.9	-0.081000000000000	-0.056741525423801	-0.109961461965129
1	0	0.03177869718185	-0.048825914554421

Table 3.35 present a comparison between the error in our methods where error =|exact value –numerical value| and $\|\boldsymbol{err}\|_\infty = \boldsymbol{max}\ |\boldsymbol{Error}|$

Table 3.35: comparison between the error using Non-polynomial spline function

x	***Error in linear***	***Error in* quadratic**
0.1	0	0
0.2	7.830669085488740e-06	4.998869659322835e-07
0.3	1.171708642117922e-04	1.598481827127379e-05
0.4	5.517381219020817e-04	1.212401293097382e-04
0.5	1.611933772927773e-03	5.100542328420749e-04
0.6	3.611891414847301e-03	1.553232365897550e-03
0.7	6.816388176274280e-03	3.854841643082030e-03
0.8	1.137824913466856e-02	8.306125434307754e-03
0.9	1.727687032201508e-02	1.613654837231515e-02
1	2.425847457619934e-02	2.896146196512861e-02
$\|\boldsymbol{err}\|_\infty$	**2.425847457619934e-02**	2.896146196512861e-02

Table (3.36) present a comparison between error obtain using non-polynomial spline function including linear and quadratic and polynomial spline function including (1st order and 2nd order [see Appendix(A) : Algorithm (**VIE2PS1**) and Algorithm (**VIE2PS1**)]) with h=0.1.

Table 3.36: Comparison between error obtain using polynomial and non- polynomial spline functions

x	Non-polynomial spline		Polynomial spline	
	Error in linear	***Error in* quadratic**	***Error* In 1st order**	***Error* In2nd order**
0.1	0	0	9.0000000000000e-03	1.0000000000000e-03
0.2	7.8306690854887e-06	4.9988696593228e-07	3.2000000000000e-02	8.0000000000000e-03
0.3	1.1717086421179e-04	1.5984818271273e-05	6.3000000000000e-02	2.7000000000000e-02
0.4	5.5173812190208e-04	1.2124012930973e-04	9.6000000000000e-02	6.4000000000000e-02
0.5	1.6119337729277e-03	5.1005423284207e-04	1.2500000000000e-01	1.2500000000000e-01
0.6	3.6118914148473e-03	1.5532323658975e-03	1.4400000000000e-01	2.1600000000000e-01
0.7	6.8163817627428e-03	3.8548416430820e-03	1.4700000000000e-01	3.4300000000000e-01
0.8	1.1378249134668e-02	8.3061254343077e-03	1.2800000000000e-01	5.1200000000000e-01
0.9	1.7276870322015e-02	1.6136548372315e-02	8.0999999999999e-02	7.2900000000000e-01
1	2.4258474576199e-02	2.8961461965128e-02	0	1.0000000000000e+00
$\|err\|_\infty$	**2.4258474576199e-02**	**2.8961461965128e-02**	**1.4700000000000e-01**	**1.0000000000000e+00**

Test Example (13): Consider the VIE of second with Weakly Singular Kernel:

$$u(x) - \int_0^x \frac{t^{\mu-1}}{x^{\mu}}\, u(t)dt = f(x) \qquad 0 \le x \le 1$$

Where f (x) $=x^2$+x+1 and $\mu = 0.4$, with $u(x) = \frac{\mu}{\mu-1} + \frac{\mu+1}{\mu}\, x + \frac{\mu+2}{\mu+1}\, x^2$. Tables 3.37 present a comparison between the exact and numerical solution using linear and quadratic non-polynomial spline functions, where $P_i(x)$ denote the approximate solution use non-polynomial spline functions, with h=0.1.

Table 3.37: Exact and numerical solution of test example (13)

x	Exact solution	$P_i(x)$	
		linear	**quadratic**
0	-0.666666666666667	-0.666666666666667	-0.666666666666667
0.1	-0.299523809523809	-0.299538090477041	-0.299523809523810
0.2	0.101904761904762	0.101676495020505	0.101904761904762
0.3	0.537619047619048	0.536465370616971	0.537619047619047
0.4	1.007619047619048	1.003981353894871	1.007619047619047
0.5	1.511904761904762	1.503050263994914	1.511904761904762
0.6	2.050476190476191	2.032182653643010	2.050476190476190
0.7	2.623333333333333	2.589588691215089	2.623333333333333
0.8	3.230476190476191	3.173196044143053	3.230476190476189
0.9	3.871904761904762	3.780670584976770	3.871904761904760
1	4.547619047619047	4.409439713213997	4.547619047619046

Table 3.35 present a comparison between the error in our methods where error =|exact value –numerical value| and $\|\boldsymbol{err}\|_\infty = \boldsymbol{max}\ |\boldsymbol{Error}|$

Table 3.38: comparison between the error using Non-polynomial spline function

x	***Error in linear***	***Error in* quadratic**
0	0	0
0.1	1.1102230246251e-16	1.1102230246251e-16
0.2	1.4280953231138e-05	1.6653345369377e-16
0.3	2.2826688425670e-04	2.7755575615628e-16
0.4	1.1536770020773e-03	5.5511151231257e-16
0.5	3.6376937241766e-03	4.4408920985006e-16
0.6	8.8544979098479e-03	2.2204460492503e-16
0.7	1.8293536833180e-02	4.4408920985006e-16
0.8	3.3744642118243e-02	4.4408920985006e-16
0.9	5.7280146333138e-02	1.7763568394002e-15
1	9.1234176927991e-02	1.3322676295501e-15
$\|\boldsymbol{err}\|_\infty$	**9.1234176927991e-02**	**1.7763568394002e-15**

Table (3.39) present a comparison between error obtain using linear and quadratic non-polynomial spline functions and polynomial spline function including (1st order and 2nd order [see Appendix(A): Algorithm (**VIE2PS1**) and Algorithm (**VIE2PS1**)]) with h=0.1.

Table 3.39: Comparison between error obtain using polynomial and non-polynomial spline functions

x	Non-polynomial spline		Polynomial spline	
	Error in linear	***Error in* quadratic**	***Error* In 1st order**	***Error* In2nd order**
0	0	0	0	0
0.1	1.1102230246251e-16	1.1102230246251e-16	1.7142857142857e-02	1.6653345369377e-16
0.2	1.4280953231138e-05	1.6653345369377e-16	6.8571428571428e-02	2.7755575615628e-16
0.3	2.2826688425670e-04	2.7755575615628e-16	1.5428571428571e-01	5.5511151231257e-16
0.4	1.1536770020773e-03	5.5511151231257e-16	2.7428571428571e-01	4.4408920985006e-16
0.5	3.6376937241766e-03	4.4408920985006e-16	4.2857142857142e-01	2.2204460492503e-16
0.6	8.8544979098479e-03	2.2204460492503e-16	6.1714285714285e-01	4.4408920985006e-16
0.7	1.8293536833180e-02	4.4408920985006e-16	8.3999999999999e-01	4.4408920985006e-16
0.8	3.3744642118243e-02	4.4408920985006e-16	1.0971428571428e+00	1.7763568394002e-15
0.9	5.7280146333138e-02	1.7763568394002e-15	1.3885714285714e+00	1.3322676295501e-15
1	9.1234176927991e-02	1.3322676295501e-15	1.7142857142857e+00	1.7763568394002e-15
$\|err\|_\infty$	**9.1234176927991e-02**	**1.7763568394002e-15**	**1.7142857142857e+00**	**1.7763568394002e-15**

Test Example (14): Consider the VIE of second with Weakly Singular Kernel:

$$u(x) - \int_0^x \frac{t^{\mu-1}}{x^{\mu}}\, u(t)dt = f(x) \qquad 0 \le x \le 1$$

Where f (x) =x+1 and $\mu = 0.6$, whit $u(x) = \frac{\mu}{\mu-1} + \frac{\mu+1}{\mu}\, x$. Tables (3.40) present a comparison between the exact and numerical solution using linear and quadratic non-polynomial spline functions, where $P_i(x)$ denote the approximate solution non-polynomial spline functions, with h=0.1

Table 3.40: Exact and numerical solution of test example (14)

x	Exact solution	$P_i(x)$	
		linear	**quadratic**
0	-1.500000000000000	-1.500000000000000	-1.500000000000000
0.1	-1.233333333333333	-1.233333333333333	-1.233333333333333
0.2	-0.966666666666667	-0.966666666666666	-0.966666666666666
0.3	-0.700000000000000	-0.700000000000000	-0.700000000000000
0.4	-0.433333333333333	-0.433333333333333	-0.433333333333333
0.5	-0.166666666666667	-0.166666666666666	-0.166666666666666
0.6	0.100000000000000	0.100000000000001	0.100000000000001
0.7	0.366666666666666	0.366666666666667	0.366666666666667
0.8	0.633333333333333	0.633333333333334	0.633333333333334
0.9	0.900000000000000	0.900000000000001	0.900000000000001
1	1.166666666666667	1.166666666666667	1.166666666666667

Table (3.41) present a comparison between the error in our methods where error =|exact value –numerical value| and $\|\boldsymbol{err}\|_\infty = \boldsymbol{max}\ |\boldsymbol{Error}|$

Table 3.41: comparison between the error using Non polynomial spline function

x	***Error*** **In linear**	**Error** In quadratic
0	0	0
0.1	2.220446049250313e-16	2.220446049250313e-16
0.2	4.440892098500626e-16	4.440892098500626e-16
0.3	4.440892098500626e-16	4.440892098500626e-16
0.4	3.330669073875470e-16	3.330669073875470e-16
0.5	5.551115123125783e-16	5.551115123125783e-16
0.6	6.661338147750939e-16	6.661338147750939e-16
0.7	7.771561172376096e-16	7.771561172376096e-16
0.8	8.881784197001252e-16	8.881784197001252e-16
0.9	7.771561172376096e-16	7.771561172376096e-16
1	8.881784197001252e-16	8.881784197001252e-16
$\|\boldsymbol{err}\|_\infty$	**8.881784197001252e-16**	**8.881784197001252e-16**

Table (3.42) present a comparison between error obtain using linear and quadratic non-polynomial spline functions and polynomial spline function including (1^{st} order and 2^{nd} order [see Appendix(A): Algorithm (**VIE2PS1**) and Algorithm (**VIE2PS1**)]) with h=0.1.

Table 3.42: Comparison between error obtain using polynomial and non- polynomial spline functions

x	Non-polynomial spline		Polynomial spline	
	$Error$ $in\ linear$	$Error$ in quadratic	$Error$ In 1st order	$Error$ In2nd order
0	0	0	0	0
0.1	2.2204460492503e-16	2.2204460492503e-16	4.4408920985006e-16	4.4408920985006e-16
0.2	4.4408920985006e-16	4.4408920985006e-16	4.4408920985006e-16	4.4408920985006e-16
0.3	4.4408920985006e-16	4.4408920985006e-16	3.3306690738754e-16	3.3306690738754e-16
0.4	3.3306690738754e-16	3.3306690738754e-16	5.5511151231257e-16	5.5511151231257e-16
0.5	5.5511151231257e-16	5.5511151231257e-16	6.6613381477509e-16	6.6613381477509e-16
0.6	6.6613381477509e-16	6.6613381477509e-16	7.7715611723760e-16	7.7715611723760e-16
0.7	7.7715611723760e-16	7.7715611723760e-16	8.8817841970012e-16	8.8817841970012e-16
0.8	8.8817841970012e-16	8.8817841970012e-16	7.7715611723760e-16	7.7715611723760e-16
0.9	7.7715611723760e-16	7.7715611723760e-16	8.8817841970012e-16	8.8817841970012e-16
1	8.8817841970012e-16	8.8817841970012e-16	8.8817841970012e-16	8.8817841970012e-16
$\|\|err\|\|_\infty$	**8.8817841970012e-16**	**8.8817841970012e-16**	**8.8817841970012e-16**	**8.8817841970012e-16**

3. 6.Discussion

In this chapter, we have introduced numerical methods for approximating the solution of three type of integral equations: witch are the VIE's of the 2nd kind, VIE's of the 1st kind and VIE's with weakly singular kernel using linear and quadratic non-polynomial spline functions.

Also we compared our method with polynomial spline of (1st order and 2nd order).We concludes the following remarks:

- The quadratic non-polynomial spline gives better accuracy than linear non-polynomial spline.
- The 2nd order polynomial spline gives better accuracy than 1st order polynomial spline.
- The quadratic non-polynomial spline gives better accuracy than 2nd order polynomial spline.

4.1 Introduction

In this chapter, certain conditions in order to prove the stability and convergence of non-polynomial spline function method will be imposed. This chapter is organized as follows: In section (4.2), we introduce stability of non-polynomial spline function method. In section (4.3), convergence of non-polynomial spline functions are proved, which consist of linear and quadratic non-polynomial spline functions. And finally, in section 4.4, conclusion for this chapter is given.

4.2. Stability Analysis:

Before introducing stability of non-polynomial spline function method, we present Von neumman condition for stability.

Lemma 4.1: The Von Neumman Necessary Condition for Stability [23]:

Let the eigenvalues of a matrix G be$\lambda_1, .\lambda_2, \dots, \lambda_k$ The spectral radius of G is

$$r_1 = \max_{1\le i\le k} |\lambda_i|.$$

Suppose the $\lambda's$ ordered such that $|\lambda_1|=r_1$and let $v^{(1)}$ be the corresponding eigenvector .Then

$$||G||=\max_V \frac{||Gv||}{||v||} \geq \frac{\left||Gv^{(1)}|\right|}{\left||v^{(1)}|\right|} = r_1,$$

or generally the spectral radius is the lower bound for the bound of a matrix. If G is raised to any power n, each of its eigenvalues gets raised to the same power, and therefore the spectral radius G^n is ${r_1}^n$.Therefore

$$||G^n|| \geq {r_1}^n$$

This is the **Von Neumman necessary condition for stability**.

Lemma 4.2: The Von Neumman Sufficient Condition for Stability [23]:

Define B is the triangular matrix, such that i=j as:

$$B=\begin{bmatrix} \lambda^{(1)} & 0 & 0 & & 0 \\ b_{21} & \lambda^{(2)} & 0 & \dots & 0 \\ b_{31} & b_{32} & \lambda^{(3)} & & 0 \\ & & \vdots & & \\ b_{n1} & b_{n2} & \cdots & & \lambda^{(P)} \end{bmatrix}$$

Whose diagonal elements are the eigenvalues such that $b_{ij} = 0 for\ i < j$. we assume that :

$$\max_{i>1} |\lambda^{(i)}| = \gamma < 1,$$

and call

$$\max(\ |\lambda^{(1)}|,1) = \lambda^{*}.$$

If λ^{*} is less than or equal 1, then the Von Neumman condition is sufficient as well as necessary for stability.

Now, we try to prove that non-polynomial spline functions method are stable.

Using equations (2.12)-(2.15),we get the following matrix:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & \mathrm{k} & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{a}_i \\ \mathbf{b}_i \\ \mathbf{c}_i \\ \mathbf{d}_i \end{bmatrix} = \begin{bmatrix} -\frac{1}{k^2}\mathbf{p_i''(t_i)} \\ -\frac{1}{\mathrm{k}^3}\mathbf{p_i'''(t_i)} \\ \mathbf{p_i'(t_i)} \\ \mathbf{p_i(t_i)} \end{bmatrix}$$

And from equations (2.18)-(2.22),we get:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & \mathrm{k} & 1 & 0 & 0 \\ -k^2/2 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{a}_i \\ \mathbf{b}_i \\ \mathbf{c}_i \\ \mathbf{d}_i \\ \mathbf{e}_i \end{bmatrix} = \begin{bmatrix} \frac{1}{k^4}\mathbf{Q}_i^{(4)}\mathbf{(t_i)} \\ -\frac{1}{k^3}\mathbf{Q_i'''(t_i)} \\ \mathbf{Q_i'(t_i)} \\ \mathbf{\frac{1}{2}Q_i''(t_i)} \\ \mathbf{Q}_i\mathbf{(t_i)} \end{bmatrix}$$

In general for n is even, we get the following matrix:

$$\begin{bmatrix} 1 & 0 & 0 & \ldots & 0 & 0 \\ 0 & 1 & 0 & & 0 & 0 \\ 0 & \mathrm{k} & 1 & & 0 & 0 \\ & & & \ldots & & \\ \vdots & & & & & \\ 0 & \frac{1}{(\mathrm{n}-1)!} & \ldots & & & \\ \frac{k^{(\mathrm{n}!)}}{\mathrm{n}!} & & & \ldots & & \\ 1 & 0 & \ldots & & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{a}_i \\ \mathbf{b}_i \\ \mathbf{c}_i \\ \mathbf{d}_i \\ \mathbf{e}_i \\ \mathbf{m}_i \\ \vdots \\ \mathbf{z}_i \end{bmatrix} = \begin{bmatrix} \frac{1}{k^{(\mathrm{n}+2)}}\mathbf{s_i^{(n+2)}(t_i)} \\ \vdots \\ \frac{k^{(\mathrm{n}!)}}{\mathrm{n}!}\mathbf{s_i^n(t_i)} \\ \mathbf{s_i(t_i)} \end{bmatrix}$$

And for n is odd, we get

$$\begin{bmatrix} 1 & 0 & 0 & \dots & 0 & 0 \\ 0 & 1 & & & 0 & 0 \\ 0 & k & 1 & & 0 & 0 \\ & & \dots & & & \\ \vdots & & & & & \\ \frac{1}{(n-1)!} & 0 & & & & \\ & & & & & \\ 0 & \frac{k^{(n!)}}{n!} & & \dots & & \\ 1 & 0 & \dots & 0 & 1 & \end{bmatrix} \begin{bmatrix} \mathbf{a}_i \\ \mathbf{b}_i \\ \mathbf{c}_i \\ \mathbf{d}_i \\ \mathbf{e}_i \\ \mathbf{m}_i \\ \vdots \\ \mathbf{z}_i \end{bmatrix} = \begin{bmatrix} \frac{1}{k^{(n+1)}} \mathbf{s}_\mathbf{i}^{(n+1)}(\mathbf{t}_\mathbf{i}) \\ \vdots \\ \frac{k^{(n!)}}{n!} \mathbf{s}_\mathbf{i}^{n}(\mathbf{t}_\mathbf{i}) \\ \mathbf{s}_\mathbf{i}(\mathbf{t}_\mathbf{i}) \end{bmatrix}$$

So the maximum of eigenvalues is1, by the lemma's (4.1) and (4.2) we achieve the stability of the method.

4.3 Convergence Analysis:

We present some definitions, as a background which will be needed to prove the convergence analysis of non-polynomial spline function.

Definition (4.1): [21]

The **linear k-step methods** is define as :

$$\sum_{j=0}^{k} \alpha_j y_{n+j} = h^2 \sum_{j=0}^{k} \beta_j y''_{n+j} \qquad (4.1)$$

Where $\alpha_k = 1 and\ \alpha_0\ and\ \beta_0$ do not both vanish

Definition (4.2):[21]

With the linear multistep method (4.1) we associate **the linear difference operator**

$$\Omega[y(x); h] = \sum_{j=0}^{k} [\alpha_j\ y(x + jh) - h^2 \beta_j y''(x + jh)]$$

Where $y(x)$is an arbitrary function, continuously twice differentiable on an interval [a,b]. If we assume that $y(x)$ has as many higher derivatives as we require, then using the Taylor expanding about the point x, we obtain:

$$\mathfrak{L}[y(x);h] = c_0 y(x) + c_1 h y'(x) + \cdots + c_q h^q y^{(q)}(x) + \cdots$$

Where

$$c_0 = \sum_{j=0}^{k} \alpha_j$$

$$c_1 = \sum_{j=0}^{k} j\alpha_j$$

and

$$c_q = \frac{1}{q!}\sum_{j=0}^{k} j^q \alpha_j - \frac{1}{(q-2)!}\sum_{j=0}^{k} j^{(q-2)}\beta_j \quad , q = 2,3, \ldots$$

Definition (4.3):[21]

We say that the method has **order P** if:

$$c_0 = c_1 = \ldots = c_p = c_{p+1} = 0, c_{p+2} \neq 0$$

$c_{p+2}$is **the error constant ,** and$c_{p+2}h^2 y^{(p+2)}(x_n)$ *is* **the principal local truncation error** at the point $\boldsymbol{x_n}$.

Definition (4.4): [21]

The method is said to be **consistent** if it has order at least one. If we define the first and second characteristic polynomials

$$\rho(r) = \sum_{j=0}^{k} \alpha_j r^j \ , \qquad \sigma(r) = \sum_{j=0}^{k} \beta_j r^j$$

it is easily verified that method (4.1) is consistent if and only if

$$\rho(1) = \rho'(1) = 0 \ , \ \ \rho''(1) = 2\sigma(1)$$

Definition (4.5): [21]

The linear multistep method (4.1) is said to be **zero-stable** if no root of the first characteristic polynomial has modulus greater than one, and every roots of modulus one has multiplicity not greater than two.

Theorem4.1:[21]

The necessary and sufficient conditions for a linear multi-step method to be convergent is consistent and zero-stable.

4.3.1 The linear Non-Polynomial Spline Function:

Consider a partition $\Delta = \{t_0, t_1, t_2, \dots, t_n\}$ of [a,b]$\subset \mathbb{R}$. Let S(Δ) denote the set of piecewise polynomials on the subinterval $I_i = [t_i, t_{i+1}]$ of partition Δ.

The form of the linear non-polynomial spline is:

$$P_i(x) = a_i \cos k(x - x_i) + b_i \sin k(x - x_i) + c_i(x - x_i) + d_i \qquad (4.1)$$

Where $a_i, b_i, c_i,$ and d_i are constants to be determined . In order to obtain the value of $a_i, b_i, c_i,$ and d_i , we differentiate equation (4.1) two times with respect to x to get:

$$\left.\begin{aligned} P_i'(x) &= -k a_i \sin k(x - x_i) + k b_i \cos k(x - x_i) + c_i \\ P_i''(x) &= -k^2 a_i \cos k(x - x_i) - k^2 b_i \sin k(x - x_i) \end{aligned}\right\} \quad (4.2)$$

By substituting $x = x_i$ in (4.2) and (4.1), we obtain the following equations:

$$\left.\begin{aligned} P_i(x_i) &= a_i + d_i \\ P_i'(x_i) &= k b_i + c_i \\ P_i''(x_i) &= -k^2 a_i \end{aligned}\right\} \qquad (4.3)$$

The following relations are defined as:

$$\left.\begin{aligned} &P_i(x_{i+1}) = Z_{i+1} \\ &P_i(x_i) = Z_i \\ &P_i''(x_{i+1}) = S_{i+1} \\ &P_i''(x_i) = S_i \end{aligned}\right\} \qquad (4.4)$$

From equations (4.1)-(4.4) ,we get the following relations:

$$Z_i = a_i + d_i$$

$$Z_{i+1} = a_i \cos\vartheta + b_i \sin\vartheta + c_i h + d_i$$

$$S_i = -k^2 a_i$$

$$S_{i+1} = -k^2 a_i \cos\vartheta - k^2 b_i \sin\vartheta \quad , \text{where } \vartheta = kh$$

We obtain the values of $a_i, b_i, c_i,$ and d_i from the above realations as follows:

$$a_i = \frac{-h^2 S_i}{\vartheta^2} \qquad (4.5)$$

$$b_i = \frac{h^2(\cos\vartheta S_i - S_{i+1})}{\vartheta^2 \sin\vartheta} \qquad (4.6)$$

$$c_i = \frac{(Z_{i+1} - Z_i)}{h} + \frac{h(S_{i+1} - S_i)}{\vartheta^2} \qquad (4.7)$$

$$d_i = \frac{h^2}{\vartheta^2} S_i + Z_i \qquad (4.8)$$

Using continuity conditions for non-polynomial spline of the first derivative to get the following consistency relation:

$$kb_i + c_i = -ka_{i-1} \sin\vartheta + k\, b_{i-1} \cos\vartheta + c_{i-1} \qquad (4.9)$$

Using equations (4.5)-(4.9), we get:

$$\frac{h^2 k(\cos\vartheta S_i - S_{i+1})}{\vartheta^2 \sin\vartheta} + \frac{(Z_{i+1} - Z_i)}{h} + \frac{h(S_{i+1} - S_{i+1})}{\vartheta^2} = \frac{h^2 k}{\vartheta^2} S_{i-1} \sin\vartheta + \frac{h^2 k(\cos\vartheta S_{i-1} - S_i)}{\vartheta^2 \sin\vartheta} \cos\vartheta + \frac{(Z_i - Z_{i-1})}{h} + \frac{h(S_i - S_{i-1})}{\vartheta^2}$$

After slight rearranging, the last equation reduced to [42]:

$$Z_{i+1} - 2Z_i + Z_{i-1} = \gamma S_{i+1} + \alpha S_i + \gamma S_{i-1}, \qquad (4.10)$$

where $\gamma = \frac{h^2}{\vartheta \sin\vartheta} - \frac{h^2}{\vartheta^2}$, and $\alpha = -\frac{2h^2\cos\vartheta}{\vartheta^2\sin\vartheta} + \frac{h^2}{\vartheta^2}$

$$Z_{i+1} - 2Z_i + Z_{i-1} = h^2\left[\left(\frac{1}{\vartheta \sin\vartheta} - \frac{1}{\vartheta^2}\right)S_{i+1} + \left(-\frac{2\cos\vartheta}{\vartheta^2\sin\vartheta} + \frac{1}{\vartheta^2}\right)S_i + \left(\frac{1}{\vartheta \sin\vartheta} - \frac{1}{\vartheta^2}\right)S_{i-1}, \right. \quad (4.11)$$

Theorem (4.2): The linear non-polynomial spline function is convergent if it satisfy the following condition:

$$1 - 2\gamma - \alpha = 0$$

Proof: First from (4.11), we get:

$$\rho(r) = (r-1)(r-1) = (r-1)^2$$

that is, by definition (4.5) ,the linear non-polynomial spline function is zero stable,

Second, using definition (4.2),we have:

$$c_0 = 0\,, c_1 = 0\,, c_2 = 1 - 2\gamma - \alpha = 0$$

So by definition (4.3), the linear non-polynomial spline function is consistent.

Therefore ,using theorem (4.1) the method is convergent.

4.3.2 The Quadratic Non-Polynomial Spline Function:

Consider the partition $\Delta = \{t_0, t_1, t_2, \ldots, t_n\}$ of $[a,b] \subset \mathbb{R}$. Let S(Δ) denotes the set of piecewise polynomials on the subintervals $I_i = [t_i, t_{i+1}]$ of partition Δ.

The form of the quadratic non-polynomial spline is:

$$Q_i(x) = a_i \cos k\,(x - x_i) + b_i \sin k(x - x_i) + c_i(x - x_i) + d_i(x - x_i)^2 + e_i \quad (4.12)$$

where a_i, b_i, c_i, d_i and e_i are constants. In order to obtain the value of these constants we differentiate equation (4.12) four times with respect to x, therefore, we get:

$$\left.\begin{aligned} Q_i'(x) &= -ka_i sink(x-x_i) + kb_i cosk\,(x-x_i) + c_i + 2d_i(x-x_i) \\ Q_i''(x) &= -k^2 a_i cosk(x-x_i) - k^2 b_i sin\mathrm{k}(x-x_i) + 2d_i \\ Q_i'''(x) &= \mathrm{k}^3 a_i sin\mathrm{k}(x-x_i) - \mathrm{k}^3 b_i cos\mathrm{k}(x-x_i) \\ Q_i^{(4)}(x) &= k^4 a_i cosk(x-x_i) + k^4 b_i sin\mathrm{k}(x-x_i) \end{aligned}\right\} \quad (4.13)$$

From the equations (4.12) and (4.13), we obtain the following relations :

$$\left.\begin{aligned} Q_i(x_i) &= a_i + e_i \\ Q_i'(x_\mathrm{i}) &= \mathrm{k}b_\mathrm{i} + c_\mathrm{i} \\ Q_i''(x_\mathrm{i}) &= -\mathrm{k}^2 a_i + 2d_i \\ Q_i'''(x_\mathrm{i}) &= -\mathrm{k}^3 b_\mathrm{i} \\ Q_i^{(4)}(x_\mathrm{i}) &= \mathrm{k}^4 a_i \end{aligned}\right\} \quad (4.14)$$

The following relations are defined as:

$$Q_i(x_{i+j}) = Y_{i+j}$$

$$Q_i'(x_{i+j}) = D_{i+j}$$

$$Q_i''(x_{i+j}) = M_{i+j}$$

$$Q_i'''(x_{i+j}) = T_{i+j}$$

$$Q_i^{(4)}(x_{i+j}) = S_{i+j}$$

From equations (4.12)-(4.14),we get the values of a_i, b_i, c_i, d_i and e_i as follow:

$$a_i = -\frac{S_{i+1}}{k^4 \cos(\frac{\vartheta}{2})} + \frac{\tan(\frac{\vartheta}{2})}{\mathrm{k}^3} T_i \quad (4.15)$$

$$b_\mathrm{i} = -\frac{T_i}{k^3} \quad (4.16)$$

$$c_i = D_i + \frac{T_i}{k^2} \quad (4.17)$$

$$di = \frac{M_{i+1/2}}{2} + \frac{S_{i+1/2}}{2k^2} \quad (4.18)$$

$$e_i = Y_{i+1\backslash 2} - \left(\frac{1}{k^4} + \frac{h^2}{8k^2}\right) S_{i+\left(\frac{1}{2}\right)} - \frac{h^2 M_{i+1/2}}{8} - \frac{hT_i}{2k^2} - \frac{hD_i}{2} \quad (4.19)$$

Where $\vartheta = kh$.

In the same way, as in the linear non-polynomial spline function using continuity conditions for non-polynomial spline of $Q_i(x)$ and it's first, second, third derivatives to get the following consistency relation:

$$\frac{h}{2}(D_i + D_{i-1}) = Y_{i+1/2} - Y_{i-1/2} - \frac{h^2}{8}\left(M_{i+1/2} + 3M_{i-1/2}\right) + \left(\frac{tan(\theta/2)}{k^3} - \frac{h}{2k^2}\right)(T_i + T_{i-1}) + \left(\frac{1}{k^4 cos(\theta/2)} - \frac{h^2}{8k^2} - \frac{1}{k^4}\right) S_{i+1/2} + \left(\frac{-cos(\theta)}{k^4 cos(\theta/2)} - \frac{3h^2}{8k^2} + \frac{1}{k^4}\right) S_{i-1/2} \quad (4.20)$$

$$D_i - D_{i-1} = hM_{i-\frac{1}{2}} + \left(\frac{-2sin\left(\frac{\theta}{2}\right)}{k^3} + \frac{h}{k^2}\right) S_{i-\frac{1}{2}} \quad (4.21)$$

$$\frac{tan(\theta/2)}{k}(T_i + T_{i-1}) = M_{i+1/2} - M_{i-1/2} + \left(\frac{-1}{k^2 cos(\theta/2)} + \frac{1}{k^2}\right) S_{i+1/2} + \left(\frac{cos\theta}{k^3 cos(\theta/2)} - \frac{1}{k^2}\right) S_{i-1/2} \quad (4.22)$$

$$T_i - T_{i-1} = \frac{2sin(\theta/2)}{k} S_{i-1/2} \quad (4.23)$$

Equations (4.20)-(4.22)yield the following equations,

$$hD_i = \left(Y_{i+1/2} - Y_{i-1/2}\right) + \left(\frac{1}{k^2} - \frac{h}{2k\ tan(\theta/2)} - \frac{h^2}{8}\right)\left(M_{i+1/2} - M_{i-1/2}\right) + \left(\frac{h}{2k^3 sin(\theta/2)} - \frac{h}{2k^3\ tan(\theta/2)} - \frac{h^2}{8k^2}\right)\left(S_{i+1/2} - S_{i-1/2}\right) \quad (4.24)$$

Likewise from Equations. (4.22) and (4.23) it follows that,

$$T_i = \frac{k}{2tan(\theta/2)}\left(M_{i+1/2} - M_{i-1/2}\right) + \left(\frac{1}{2k\ tan(\theta/2)} - \frac{1}{2k\ sin(\theta/2)}\right) * \left(S_{i+\frac{1}{2}} - S_{i-\frac{1}{2}}\right) \quad (4.25)$$

Eliminating of T's from Equations,(4.23) and (4.25), D's from equation (4.21) and (4.24) ,yield:

$$\frac{k}{2tan(\theta/2)}\left(-M_{i+\frac{1}{2}}+2M_{i-1/2}-M_{i-3/2}\right)=\left(\frac{1}{2k\,tan(\theta/2)}-\frac{1}{2k\,sin(\theta/2)}\right)\left(S_{i-3/2}+S_{i+1/2}\right)+\left(\frac{1}{k\,sin(\theta/2)}-\frac{1}{k\,tan\left(\frac{\theta}{2}\right)}-\frac{2\,sin(\theta/2)}{k}\right)S_{i-1/2} \quad (4.26)$$

$$\frac{h^2(1-cos(\theta/2))}{2k^2}S_{i-1/2}=y_{i+1/2}-2y_{i-\frac{1}{2}}+y_{i-\frac{3}{2}}-2\left(\frac{1}{k^2}-\frac{h}{8\left(1-\cos\left(\frac{\theta}{2}\right)\right.}-\frac{h^2}{2}\right)M_{i-\frac{1}{2}}+\left(\frac{1}{k^2}-\frac{h^2}{8\left(1-\cos\left(\frac{\theta}{2}\right)\right)}\right)\left(M_{i+\frac{1}{2}}+M_{i-\frac{1}{2}}\right) \quad (4.27)+$$

When k→0, formula derived becomes special case of equation (4.26) that is,

$$h^2\left(M_{i+1/2}-2M_{i-1/2}+M_{i-3/2}\right)=\frac{h^4}{8}\left(S_{i+1/2}+6S_{i-1/2}+S_{i-3/2}\right) \quad (4.28)$$

Eliminating S_i's from equation (4.27) and (4.28) we get

$$\begin{pmatrix} Y_{i+3/2}+4Y_{i+1/2} \\ -10\,Y_{i-1/2}+4Y_{i-3/2} \\ +Y_{i-5/2} \end{pmatrix}=\left(\frac{h^2}{16sin^2(\theta/4)}-\frac{1}{k^2}\right)\left(M_{i-5/2}+M_{i+3/2}\right)+\frac{4+3h^2k^2-2(4+h^2k^2)\cos(\theta/2)+4\cos(\theta)}{4k^2sin^2(\theta/4)}\left(M_{i-3/2}+M_{i+1/2}\right)+\frac{8-19h^2k^2+24(-1+h^2k^2)\cos(\theta/2)+16\cos(\theta)}{8k^2sin^2(\theta/4)}\left(M_{i-1/2}\right) \quad (4.29)$$

Equation (4.29) in simpler form renders the following equation,

[17]:

$$Y_{i-5/2}+Y_{i+3/2}+4\left(Y_{i-3/2}+Y_{i+1/2}\right)-10Y_{i-1/2}=h^2\begin{Bmatrix} \alpha\left(M_{i-5/2}+M_{i+3/2}\right) \\ +\beta\left(M_{i-3/2}+M_{i+1/2}\right) \\ +\gamma M_{i-1\backslash 2} \end{Bmatrix}$$

(4.30)

that is:

$$Y_{i+4} + 4Y_{i+3} - 10Y_{i+2} + 4Y_{i+1} + Y_i = \alpha M_{i+4} + \beta M_{i+3} + \gamma M_{i+2} + \beta M_{i+1} + \alpha M_i$$

where $\alpha = \frac{1}{16\sin^2\left(\frac{\vartheta}{4}\right)} - \frac{1}{\vartheta^2}$, $\beta = \frac{4+3\vartheta^2-2\left(4+\vartheta^2\right)\cos\left(\frac{\vartheta}{2}\right)+4\cos(\vartheta)}{4\vartheta^2\sin^2\left(\frac{\vartheta}{4}\right)}$

and $\gamma = \frac{8+19\,\vartheta^2+24\left(-1+\vartheta^2\right)\cos\left(\frac{\vartheta}{2}\right)+16\cos(\vartheta)}{8\vartheta^2\sin^2\left(\frac{\vartheta}{4}\right)}$ $\quad i = 1,2,\ldots.,n$

Theorem (4.3): The quadratic non-polynomial spline function is convergent if it satisfies the following condition:

$$-1/2(8 - 2\alpha - 2\beta - \gamma) = 0$$

Proof: First from (4.30), we get:

$$\rho(r) = (r-1)^2\,(r^2 + 6r + 1)$$

that is, by definition (4.5) ,the quadratic non-polynomial spline function is zero stable,

Second, using definition (4.2) ,we have:

$$c_0 = 0\,, c_1 = 0\ , c_2 = 8 - 2\alpha - 2\beta - \gamma = 0\ , c_3 = -1/2(8 - 2\alpha - 2\beta - \gamma) = 0$$

So by definition (4.3), the quadratic non-polynomial spline function is consistent.

Therefore, using theorem (4.1) the method is convergent.

4.4Discussion

In this chapter, we discussed the stability and convergent analysis of linear and quadratic non-polynomial spline functions method. The results show that:

1. The non-polynomial spline functions are stable method.
2. The linear non-polynomial spline function is a zero-stable and consistent method.
3. The quadratic non-polynomial spline function is a zero-stable and consistent method.
4. The linear non-polynomial spline function is convergent.
5. The quadratic non-polynomial spline function is convergent.

.

(5.1) Conclusions:

The numerical treatment of the liner VIE's of the 2nd kind and VIE's with weakly singular kernel using non-polynomial spline functions were introduced. Moreover, a VIE's of the 1st kind with $k(x,x) \neq 0$ are reduced to VIE's of the 2nd kind and solving it using the same algorithms, examples were solved and good results are achieved.

A comparison is made between these methods depending on the absolute error, between the numerical and the exact solutions.

Tables (5.1) and (5.2) give the absolute error for solving test examples (1) and (8), respectively , using linear and quadratic non-polynomial spline functions and polynomial spline functions including 1st and 2nd order and the result obtain in [36].

Tables (5.1): Absolut error for test example (1)

	Using Polynomial spline		Using Non- Polynomial spline		Result obtain in [36]
$\|\|err\|\|_\infty$	1st order	2nd order	linear	quadratic	
	0.34147098480	**0.34147098480**	**4.440892098500e-16**	**4.440892098500e-16**	**2.050806e-012**

Table (5.2): Absolut error for test example (8)

	Using Polynomial spline		Using Non- Polynomial spline	
$\|\|err\|\|_\infty$	1st order	2nd order	linear	quadratic
	0.50000000000	0.50000000000	1.1100e-16	5.5510e-17

Table (5.3) gives the absolute error for solving test example (10) using linear and quadratic non-polynomial spline functions and polynomial spline functions including 1st and 2nd order and the result obtain in [10].

Table (5.3): Absolut error for test example (10)

	Using Polynomial spline		Using Non- Polynomial spline		Result obtain in [10]
	1st order	2nd order	linear	quadratic	
$\|\|err\|\|_\infty$	1.666666666e+00	5.32907051e-15	1.29132868605e-01	5.32907051820e-15	4.03e-1

The comparison between the solutions obtained by the method of: linear and quadratic non-polynomial and polynomial spline functions including 1st and 2nd order [see appendix (B)] shows that the approximate solutions of our problems by using non-polynomial spline functions are better in the sense of accuracy and applicability. These have been verified by the maximum absolute errors (max |err|) given in tables .A new approach convergence analysis of the presented method is discussed.

Figures (B.1) and (B.2) in appendix (B) show a comparison between the exact and numerical solution which was presented in test example (1) using linear and quadratic non-polynomial spline functions and the second using quadratic non-polynomial and polynomial spline function of 2nd order, respectively .

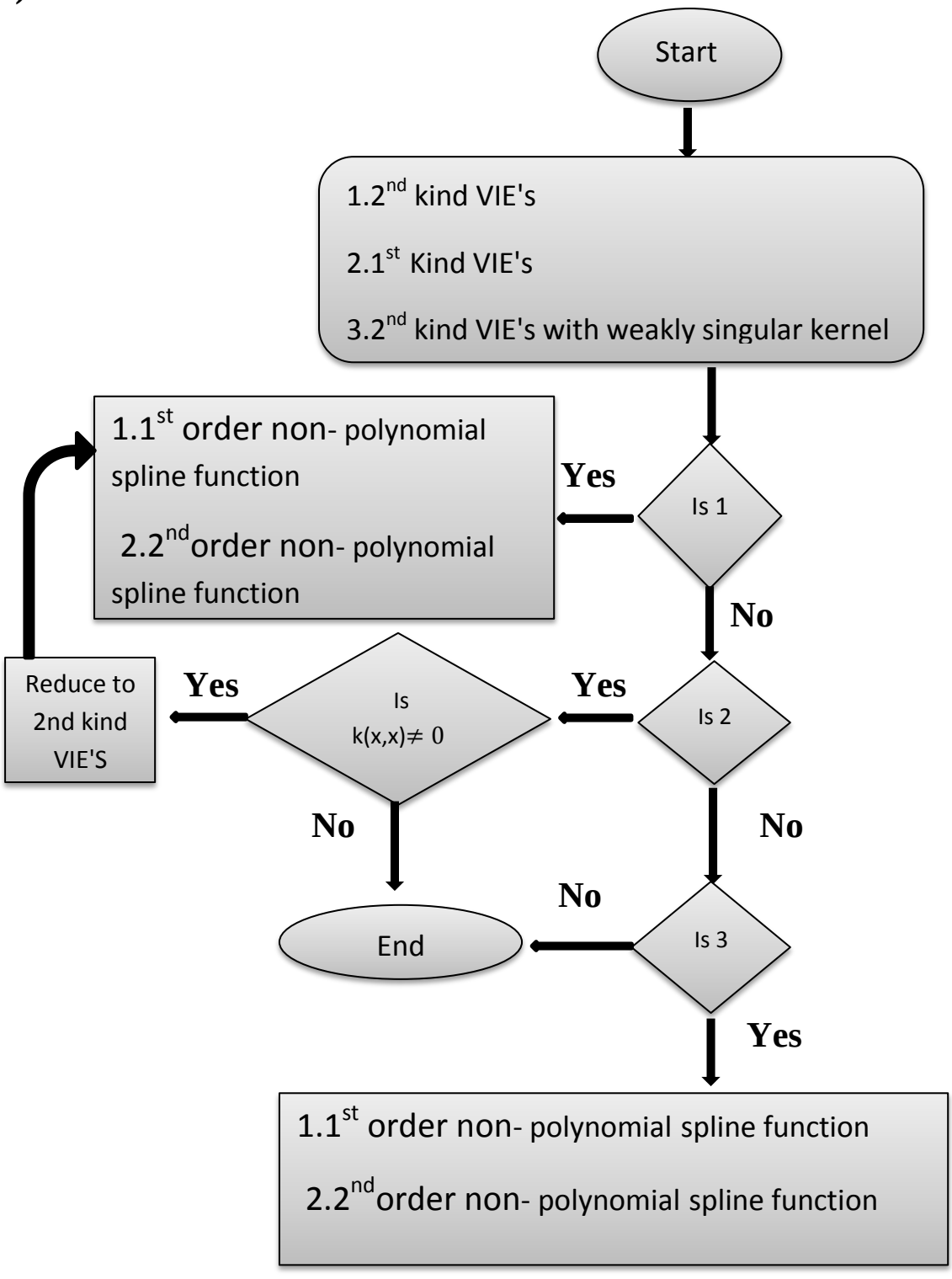
Figures (B.3) and (B.4) show a comparison between the exact and the numerical solution which was presented in test example (8) using linear and quadratic non-polynomial spline functions and the second using quadratic non-polynomial and polynomial spline function of 2nd order, respectively .

Figures (B.5) and (B.6) show a comparison between the exact and numerical solution which was presented in test example (10) using linear and quadratic non-polynomial spline functions and the second using quadratic non-polynomial and polynomial spline function of 2nd order, respectively .

From the above results tables and figures, the following conclusions are drawn:

- In general, methods which are used in this thesis, proved their effectiveness in solving linear VIE's of 2^{nd} kind and VIE's with weakly singular kernel numerically and finding an accurate results.
- The results that are obtained in our work show that quadratic non-polynomial spline function gives the best approximation to solve our problems.
- This new idea based on the use of the VIE and its derivatives. So it is necessary to mention that this approach can be used when $f(x)\ and\ k(x,t)$ are analytic.
- The proposed scheme is simple and computationally attractive and its accuracy is high and we can simply execute this method in a computer.

(5.2): Numerical Structure of our method:



(5.3)Recommendations:

Our recommendations for future work are

1. Using non-polynomial spline function to solve a system of linear VIE's.
2. Using non-polynomial spline function to solve non - linear VIE's of 2^{nd} kind.
3. Using another order of non-polynomial spline function to solve a linear VIE's of 2^{nd} kind and 1^{st} kind VIE's with weakly singular kernel.
4. Writing the package: which is a single program including programs for all cases of methods (non-polynomial spline functions).
5. Also, using the proposed method to find numerical solution for a linear Fredholm integral equations.
6. Using non-polynomial spline function to solve non - linear Fredholm integral equations.
7. Using non-polynomial spline function to solve a system of Fredholm integral equations.
8. Using non-polynomial spline function to solve Abel's equation.

Appendix (B): Figures

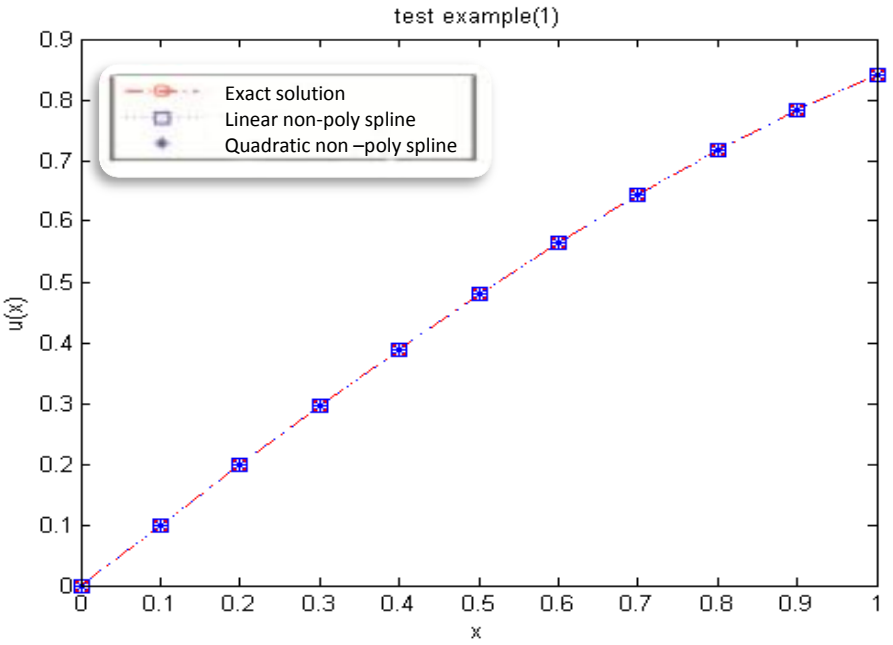


Fig (B.1): comparison between the exact and the approximate solution using linear and quadratic non-polynomial spline functions for test example (1)

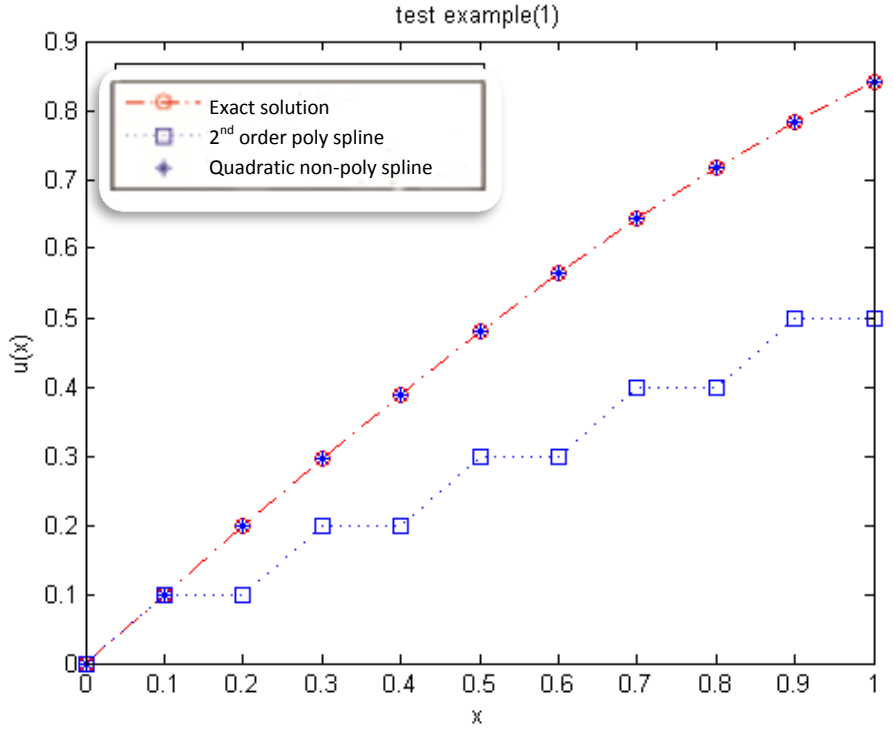


Fig (B.2): comparison between the exact and the approximate solution using 2nd order polynomial and quadratic non-polynomial spline functions for test example (1)

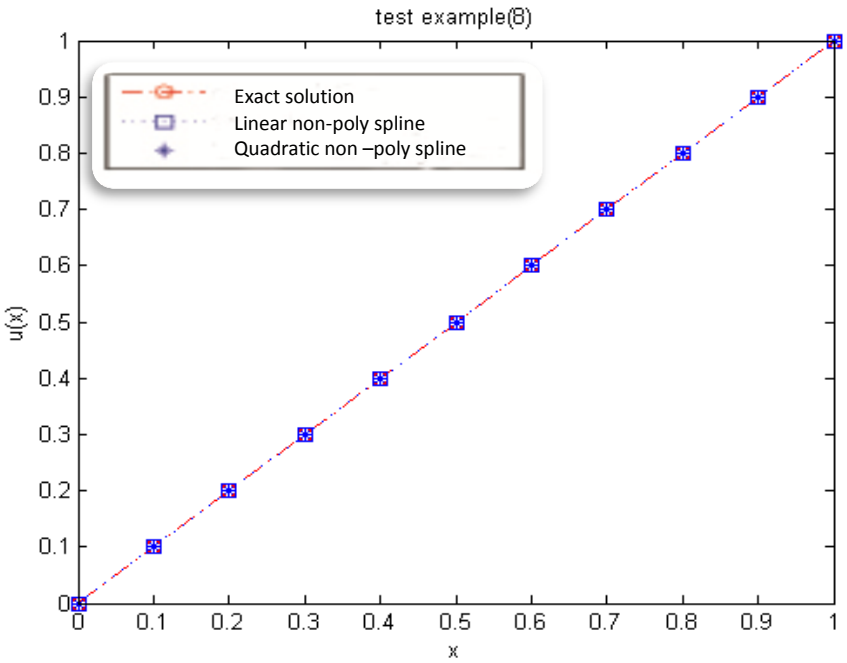


Fig (B.3): comparison between the exact and the approximate solution using linear and quadratic non-polynomial spline functions for test example (8)

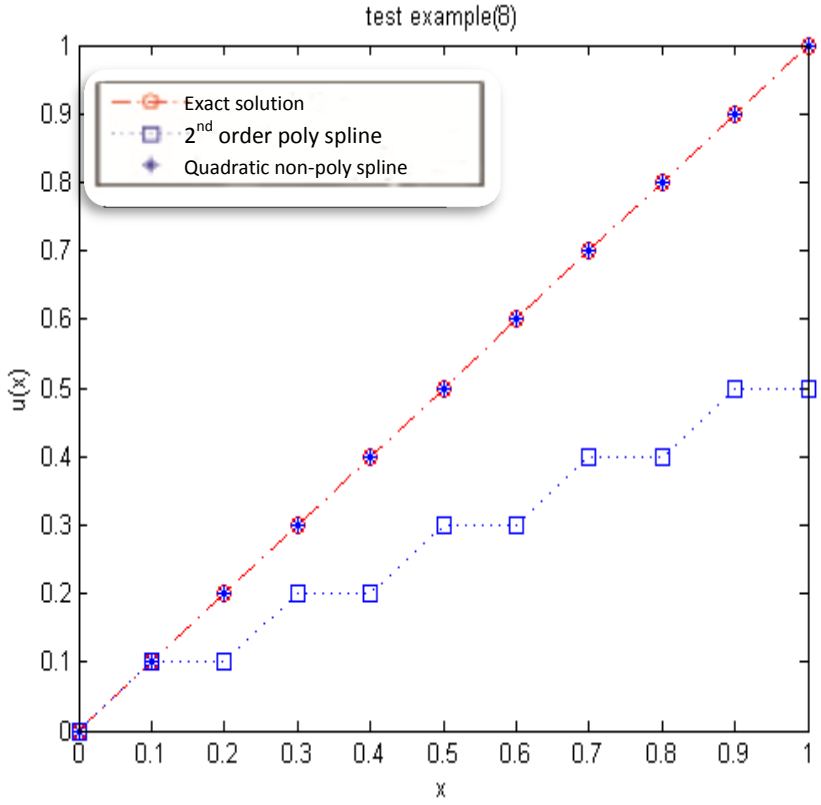


Fig (B.4): comparison between the exact and the approximate solution using 2nd order polynomial and quadratic non-polynomial spline functions for test example (8)

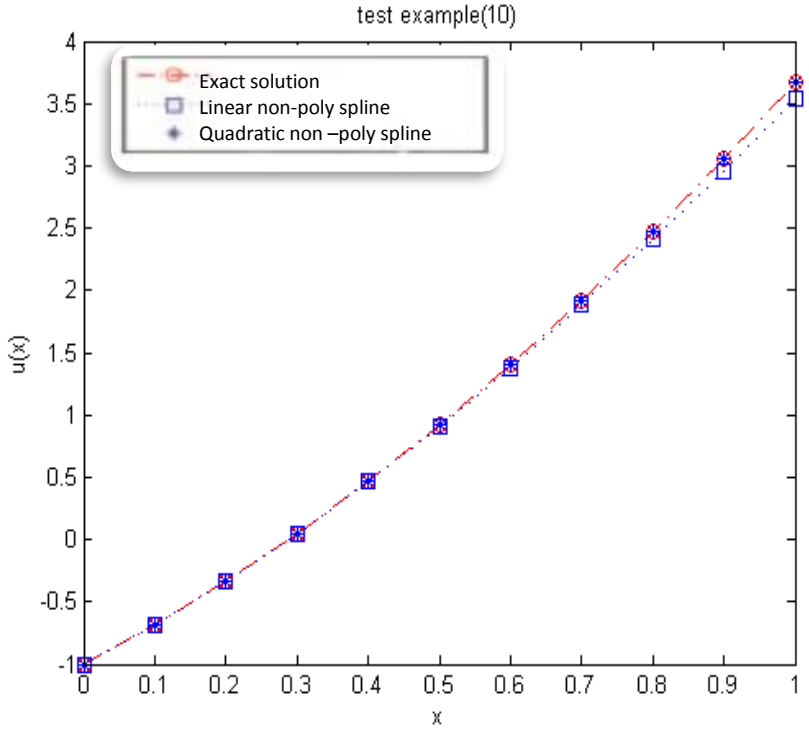


Fig (B.5): comparison between the exact and the approximate solution using linear and quadratic non-polynomial spline functions for test example (10)

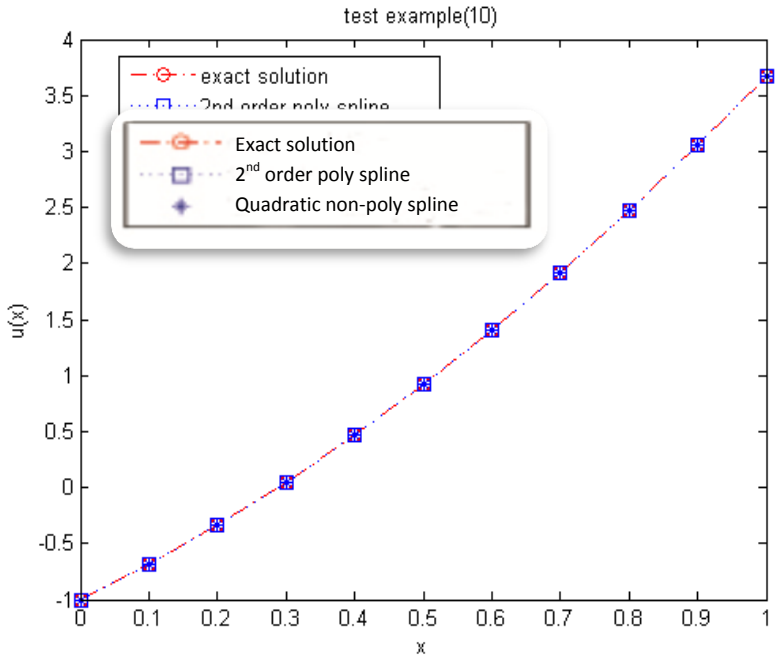


Fig (B.6): comparison between the exact and the approximate solution using 2nd order polynomial and quadratic non-polynomial spline functions of for test example (10)

Program 1: linear non-polynomial spline function for solving VIE's of the second kind:

```
function [u,err]=volnonpolyspline1st(ker,f,ex,a,b,n)
syms x t s
h=(b-a)/n;
u(1)=subs(f,a);
if isempty(diff(f,1))==1
   z1=0;
else
   z1=diff(f,1);
end
du(1)=subs(z1,a)+subs(ker,{x,t},{a,a})*u(1);
if isempty(diff(f,2))==1
   z2=0;
else
   z2=diff(f,2);
end
if isempty(diff(subs(ker,{t},{x}),'x'))==1
   z3=0;
else
   z3=diff(subs(ker,{t},{x}),'x');
end
if isempty(diff(ker,'x'))==1
zz3=0;
else
zz3=diff(ker,'x');
end
d2u(1)=subs(z2,a)+subs(zz3,{x,t},{a,a})*u(1)+subs(z3,a)*u(1)+subs(ker,{x,t},{a
,a})*du(1);
if isempty(subs(diff(subs(diff(ker,'x'),{t},{x})),{x},{a}))==1
z=0
else
   z= subs(diff(subs(diff(ker,'x'),{t},{x})),{x},{a})
end
if isempty(diff(f,3))==1
   z4=0;
else
   z4=diff(f,3);
end
if isempty(diff(ker,2,'x'))==1
   z5=0;
else
   z5=diff(ker,2,'x');
```

```
end
if isempty(subs(diff(subs(ker,{x,t},{x,x}),2),{x},{a}))==1
   z6=0;
else
   z6=subs(diff(subs(ker,{x,t},{x,x}),2),{x},{a});
end
if isempty(subs(diff(subs(ker,{x,t},{x,x})),{x},{a}))==1
   z7=0;
else
   z7=subs(diff(subs(ker,{x,t},{x,x})),{x},{a});
end
d3u(1)=subs(z4,a)+subs(subs(z5,{x,t},{x,x}),{x},{a})*u(1)+z*u(1)+subs(subs(dif
f(ker,'x'),{x,t},{x,x}),{x},{a})*du(1)+z6*u(1)+2*z7*du(1)+subs(ker,{x,t},{a,a})*
d2u(1);
a(1)=-d2u(1); b(1)=-d3u(1);
c(1)= du(1)+d3u(1); d(1)=u(1)+d2u(1);
for i=1:n
   u(i+1)=a(i)*cos(h)+b(i)*sin(h)+h*c(i)+d(i);
   du(i)=-a(i)*sin(h)+b(i)*cos(h)+c(i);
   d2u(i)=-a(i)*cos(h)-b(i)*sin(h);
   d3u(i)=a(i)*sin(h)-b(i)*cos(h);
   a(i+1)=-d2u(i);b(i+1)=-d3u(i);
   c(i+1)=du(i)+d3u(i); d(i+1)=u(i+1)+d2u(i);
end
err=abs(u-subs(ex,0:h:1));
```

Program 2: quadratic non-polynomial spline function for solving VIE's of the second kind:

```
function [u,err]=volnonpolyspline2nd(ker,f,ex,a,b,n)
syms x t s
h=(b-a)/n;
u(1)=subs(f,a);
if isempty(diff(f,1))==1
   z1=0;
else
   z1=diff(f,1);
end
du(1)=subs(z1,a)+subs(ker,{x,t},{a,a})*u(1);
if isempty(diff(f,2))==1
   z2=0;
else
   z2=diff(f,2);
```

```
end
if isempty(diff(subs(ker,{t},{x}),'x'))==1
   z3=0;
else
   z3=diff(subs(ker,{t},{x}),'x');
end
if isempty(diff(ker,'x'))==1
zz3=0;
else
zz3=diff(ker,'x');
end
d2u(1)=subs(z2,a)+subs(zz3,{x,t},{a,a})*u(1)+subs(z3,a)*u(1)+subs(ker,{x,t},{a
,a})*du(1);
if isempty(subs(diff(subs(diff(ker,'x'),{t},{x})),{x},{a}))==1
z=0 ;
else
   z= subs(diff(subs(diff(ker,'x'),{t},{x})),{x},{a});
end
if isempty(diff(f,3))==1
   z4=0;
else
   z4=diff(f,3);
end
if isempty(diff(ker,2,'x'))==1
   z5=0;
else
   z5=diff(ker,2,'x');
end
if isempty(subs(diff(subs(ker,{x,t},{x,x}),2),{x},{a}))==1
   z6=0;
else
   z6=subs(diff(subs(ker,{x,t},{x,x}),2),{x},{a});
end
if isempty(subs(diff(subs(ker,{x,t},{x,x})),{x},{a}))==1
   z7=0;
else
   z7=subs(diff(subs(ker,{x,t},{x,x})),{x},{a});
end
d3u(1)=subs(z4,a)+subs(subs(z5,{x,t},{x,x}),{x},{a})*u(1)+z*u(1)+subs(subs(dif
f(ker,'x'),{x,t},{x,x}),{x},{a})*du(1)+z6*u(1)+2*z7*du(1)+subs(ker,{x,t},{a,a})*
d2u(1);
if isempty(diff(f,4))==1
   z8 =0;
else
```

```
    z8=diff(f,4);
end
   if isempty(diff(ker,3,'x'))==1
    z9=0;
else
    z9= diff(ker,3,'x');
   end
    if isempty(subs(diff(ker,2,'x'),{t},{x}))==1
    z10=0;
    else
z10= subs(diff(ker,2,'x'),{t},{x});
    end
   if isempty (subs(diff(z10),{x},{a}))==1
    z1010=0;
   else
    z1010=subs(diff(z10),{x},{a});
   end
    if isempty(diff(ker,2,'x'))==1
   z11=0;
        else
            z11=diff(ker,2,'x');
    end
        if isempty (diff(ker,'x'))==1
z12=0;
    else
    z12=diff(ker,'x');
        end
        if isempty(subs(z12,{t},{x}))==1
            c=0;
        else
            c=subs(z12,{t},{x});
        end
     if isempty (sym(c))==1
       z1212=0;
     else
        z1212= sym(c);
     end
        if isempty(subs(diff(ker,'x'),{t},{x}))==1

    z13=0;
else
   z13= subs(diff(ker,'x'),{t},{x});
        end
      if isempty (subs(diff(z13),{x},{a}))==1
```

```
        z1313=0;
    else
        z1313=subs(diff(z13),{x},{a});
    end
if isempty(subs(diff(subs(ker,{x,t},{x,x}),3),{x},{a}))==1
z14=0;
else
   z14=subs(diff(subs(ker,{x,t},{x,x}),3),{x},{a});
end
if isempty(subs(diff(subs(ker,{x,t},{x,x}),2),{x},{a}))==1
z15=0;
  else
     z15=subs(diff(subs(ker,{x,t},{x,x}),2),{x},{a});
end
if isempty(subs(diff(subs(ker,{x,t},{x,x})),{x},{a}))==1
z16=0;
else
   z16=subs(diff(subs(ker,{x,t},{x,x})),{x},{a});
end

d4u(1)=subs(z8,a)+subs(subs(z9,{x,t},{x,x}),{x},{a})*u(1)+z1010*u(1)+subs(su
bs(z11,{x,t},{x,x}),{x},{a})*du(1)+subs(diff(z1212,'x',2),{x},{a})*u(1)+2*z1313*
du(1)+subs(subs(diff(ker,'x'),{x,t},{x,x}),{x},{a})*d2u(1)+z14*u(1)+3*z15*u(1)
+2*z16*d2u(1)+subs(ker,{x,t},{a,a})*d3u(1);
a(1)=d4u(1);   b(1)=-d3u(1);
c(1)= du(1)+d3u(1); d(1)=(1/2)*(d2u(1)+d4u(1));
e(1)=u(1)-d4u(1);
for i=1:n
   u(i)=a(i)*cos(h)+b(i)*sin(h)+h*c(i)+d(i)*h^2+e(i);
   du(i)=-a(i)*sin(h)+b(i)*cos(h)+c(i)+2*d(i)*h;
   d2u(i)=-a(i)*cos(h)-b(i)*sin(h)+2*d(i);
   d3u(i)=a(i)*sin(h)-b(i)*cos(h);
   d4u(i)=a(i)*cos(h)+b(i)*sin(h);
 a(i+1)=d4u(i);b(i+1)=-d3u(i);
  c(i+1)=du(i)+d3u(i); d(i+1)=(1/2)*(d2u(i)+d4u(i));
   e(i+1)=u(i)-d4u(i);
end
err=abs(u-subs(ex,h:h:1));
```

Program 3: first order polynomial spline function for solving VIE's of the second kind:

```
function [u,err]=volpoly(ker,f,ex,a,b,n)
syms x t s
h=(b-a)/n;
u(1)=subs(f,a);
if isempty(diff(f,1))==1
   z1=0;
else
   z1=diff(f,1);
end
du(1)=subs(z1,a)+subs(ker,{x,t},{a,a})*u(1);
a(1)=du(1); b(1)=u(1);
for i=1:n
   u(i+1)=a(i)*h+b(i);
   du(i)=a(i);
 a(i+1)=du(i);b(i+1)=u(i);
end
err=abs(u-subs(ex,0:h:1));
```

Program 4: second order polynomial spline function for solving VIE's of the second kind:

```
function [u,err]=volpoly2(ker,f,ex,a,b,n)
syms x t s
h=(b-a)/n;
u(1)=subs(f,a);
if isempty(diff(f,1))==1
   z1=0;
else
   z1=diff(f,1);
end
du(1)=subs(z1,a)+subs(ker,{x,t},{a,a})*u(1);
if isempty(diff(f,2))==1
   z2=0;
else
   z2=diff(f,2);
end
if isempty(diff(subs(ker,{t},{x}),'x'))==1
   z3=0;
else
   z3=diff(subs(ker,{t},{x}),'x');
```

```
end
if isempty(diff(ker,'x'))==1
zz3=0;
else
zz3=diff(ker,'x');
end
d2u(1)=subs(z2,a)+subs(zz3,{x,t},{a,a})*u(1)+subs(z3,a)*u(1)+subs(ker,{x,t},{a
,a})*du(1);
a(1)=(1/2)*d2u(1); b(1)=du(1);
c(1)= u(1);
for i=1:n
   u(i+1)=a(i)*(h^2)+b(i)*h+c(i);
   du(i)=2*a(i)*h+b(i);
    d2u(i)=2*a(i);
 a(i+1)=(1/2)*d2u(i);b(i+1)=du(i);
 c(i+1)= u(i);
end
err=abs(u-subs(ex,0:h:1));
```

Program 5: linear non-polynomial spline function for solving VIE's of the second kind with weakly singular kernel:

```
unction [u,err]=nonpolyavolteraa(f,ex,a,b,n,m)
syms x t s
h=(b-a)/n;x=a:h:b;
u(1)=(m/(m-1))*subs(f,a);
du(1)=(m+1)/m*subs(diff(f,1),0)
d2u(1)=((m+2)/(m+1))*subs(diff(f,2),0)
d3u(1)=((m+3)/(m+2))*subs(diff(f,3),0)
a(1)=-d2u(1); b(1)=-d3u(1)
c(1)= du(1)+d3u(1); d(1)=u(1)+d2u(1)
for i=1:n
   u(i+1)=a(i)*cos(h)+b(i)*sin(h)+h*c(i)+d(i);
   du(i)=-a(i)*sin(h)+b(i)*cos(h)+c(i);
   d2u(i)=-a(i)*cos(h)-b(i)*sin(h);
   d3u(i)=a(i)*sin(h)-b(i)*cos(h);
   a(i+1)=-d2u(i);b(i+1)=-d3u(i);
   c(i+1)=du(i)+d3u(i); d(i+1)=u(i+1)+d2u(i);
end
for i=1:n
  err(i)=abs(u(i)-subs(ex,x(i)));
end
```

Program 6: quadratic non-polynomial spline function for solving VIE's of the second kind with weakly singular kernel:

```
function [u,err]=nonpolyavolteraa2(f,ex,a,b,n,m)
syms x t s
h=(b-a)/n;x=a:h:b;
u(1)=(m/(m-1))*subs(f,a);
du0=(m+1)/m*subs(diff(f,1),0);
d2u0=((m+2)/(m+1))*subs(diff(f,2),0);
d3u0=((m+3)/(m+2))*subs(diff(f,3),0);
d4u0=((m+4)/(m+3))*subs(diff(f,4),0);
a(1)=d4u0; b(1)=-d3u0;
c(1)= du0+d3u0; d(1)=(1/2)*(d2u0+d4u0);
e(1)=u(1)-d4u0;
for i=1:n
   u(i+1)=a(i)*cos(h)+b(i)*sin(h)+h*c(i)+d(i)*h^2+e(i);
   du(i+1)=-a(i)*sin(h)+b(i)*cos(h)+c(i)+2*d(i)*h;
   d2u(i+1)=-a(i)*cos(h)-b(i)*sin(h)+2*d(i);
   d3u(i+1)=a(i)*sin(h)-b(i)*cos(h);
   d4u(i+1)=a(i)*cos(h)+b(i)*sin(h);
 a(i+1)=d4u(i+1);b(i+1)=-d3u(i+1);
  c(i+1)=du(i+1)+d3u(i+1); d(i+1)=(1/2)*(d2u(i+1)+d4u(i+1));
   e(i+1)=u(i+1)-d4u(i+1);
end
for i=1:n
  err(i)=abs(u(i)-subs(ex,x(i)));
end
```

Program 7: first order polynomial spline function for solving VIE's of the second kind with weakly singular kernel:

```
function [u,err,u0]=singular(f,m,b,n,ex)
h=b/n;x=h:h:b;
df=diff(f,1);
u0=(m/(m-1))*subs(f,0);
du0=(m+1)/m*subs(df,0);
a(1)=du0; b(1)=u0;
for i=1:n
   u(i)=a(i)*h+b(i);
   du(i)=a(i);
 a(i+1)=du(i);b(i+1)=u(i);
end
for i=1:n
  err(i)=abs(u(i)-subs(ex,x(i)));
end
```

Program 8: second order polynomial spline function for solving VIE's of the second kind with weakly singular kernel:

```
function [u,err,u0]=sigular2(f,m,b,n,ex)
h=b/n;x=h:h:b;
df=diff(f,1);d2f=diff(f,2);
u0=(m/(m-1))*subs(f,0);
du0=(m+1)/m*subs(df,0);
d2u0=((m+2)/(m+1))*subs(d2f,0);
a(1)=(1/2)*d2u0; b(1)=du0;
c(1)= u0;
for i=1:n
   u(i)=a(i)*(h^2)+b(i)*h+c(i);
   du(i)=2*a(i)*h+b(i);
    d2u(i)=2*a(i);
 a(i+1)=(1/2)*d2u(i);b(i+1)=du(i);
 c(i+1)= u(i);
end
for i=1:n
  err(i)=abs(u(i)-subs(ex,x(i)));
end
```

Appendix (A): Polynomial Spline Functions

In this appendix, we use polynomial spline functions which contain first order and second order to approximate a solution of linear VIE's and VIE's with weakly singular kernel, as follows:

A.1 Polynomial Spline Function:

Consider the partition $\Delta = \{t_0, t_1, t_2, \dots, t_n\}$ of $[a,b] \subset \mathbb{R}$. Let S(Δ) denote the set of piecewise polynomials on subinterval $I_i = [t_i, t_{i+1}]$ of partition Δ. Let *u(t)* be the exact solution, Each polynomial spline of n order P_i *(t)* has the form*:*

$$P_i(t) = a_i(t - t_i) + b_i(t - t_i)^2 + \cdots + y_i(t - t_i)^{n-1} + z_i \qquad (A.1)$$

Where $a_i, b_i, \dots$ and z_i constant.

A.1.1First Order Polynomial Spline Function

The form of first order polynomial spline is:

$$P_i(t) = a_i(t - t_i) + b_i \qquad i = 0, .., n \qquad (A.2)$$

Where a_i, and b_i are constant. In order to obtain the value of $a_i$and b_i, we differentiate equation (A.2) one time with respect to t ,we get:

$$p_i'(t) = a_i \qquad (A.3)$$

Hence replace t by t_i *in the realtion*$(A.2)$*and* $(A.3)$ yields:

$$P_i(t_i) = b_i$$

$$p_i'(t_i) = a_i$$

From the equations above, we obtain the values of a_i and b_i as follows:

$$a_i = p_i'(t_i) \qquad (A.4)$$

$b_i = P_i(t_i)$ (A.5)

$for\ i = 0,..,n$

A.1.2 Second Order Polynomial Spline Function

The form of second order polynomial spline function is:

$$P_i(t) = a_i(t - t_i)^2 + b_i(t - t_i) + c_i \quad \text{(A.6)}$$

where $a_i, b_i and c_i$ are constant of the polynomial functions .We consider the following relations:

$P_i(t_i) = c_i = u(t_i)$

$P'_i(t_i) = b_i = u'(t_i)$

$P''_i(t_i) = 2a_i = u''(t_i)$

We can obtain the values of a_i, b_i and c_i as follows:

$a_i = \frac{1}{2}u''(t_i)$ (A.7)

$b_i = u'(t_i)$ (A.8)

$c_i = u(t_i)$ (A.9)

$for\ i = 0,..,n$

A.2 Solution of Linear VIE's of the Second Kind:

In this section, we use 1^{st}order and 2^{nd}order polynomial spline function to find the numerical solution of second kind linear VIE'S, which has a form:

$$u(x) = f(x) + \int_a^x k(x,t)u(t)dt \quad x \in [a,b] \quad \text{(A.10)}$$

Where $k(x,t)$and $f(x)$ are known functions and continues in [a,b] , but $u(t)$ is unknown function. For solving the equation (A.10), we have to

differentiate the equation (A.10) two times with respect to x , by using Libenze formula we realize:

$$u'(x) = f'(x) + \int_a^x \frac{\partial k(x,t)}{\partial x}\, u(t)dt + k(x,x)\ \ u(x) \qquad (A.11)$$

$$u''(x) = f''(x) + \int_a^x \frac{\partial^2 k(x,t)}{\partial x^2}\, u(t)dt\ + \left(\frac{\partial k(x,t)}{\partial x}\right)_{t=x}\ \ u(x) + \frac{dk(x,x)}{dx}\, u(x) + +k(x,x)u'(x) \qquad (A.12)$$

To complete our ways for solution VIE's. we put x=a in equations(A.10)-(A.12) , then we get :

$$u_0 = u(a) = f(a) \qquad (A.13)$$

$$u_0' = u'(a) = f'(a) + k(a,a)u(a) \qquad (A.14)$$

$$u_0'' = u''(a) = f''(a) + \left(\left(\frac{\partial k(x,t)}{\partial x}\right)_{t=x}\right)_{x=a} \mathrm{u(a)} + \left(\frac{dk(x,x)}{dx}\right)_{x=a}\ u(a)$$

$$+k(a,a)u'(a) \qquad (A.15)$$

A.2.1 Using 1^{st}Order Polynomial Spline Function:

We approximate the solutions of second kind linear VIE's by using 1^{st}order polynomial spline .We introduce a method of solution in algorithm (VIE2PS1):

The Algorithm (VIE2PS1):

Step 1: set h = (b-a) /n, $t_i = t_0$ +ih , i=0,…,n,(where t_0 =a , t_n =b) and $u_0 = f(a)$.

Step 2: evaluate a_0 and b_0 by substituting (A.13)-(A.14) in equations (A.4)-(A.5).

Step 3: calculate $p_0(t)$using step2 and equation (A.2).

Step 4: approximant $u_1 = p_0(t_1)$

Step 5: for i=1 to n-1 do the following steps:

Step6: evaluate a_i, b_i, c_i and d_i by using equations (A.4)-(A.5) and replacing $u(t_i), u'(t_i)\ by p_i(t_i)\ , p_i'(t_i)$

Step 7: calculate $p_i(t)$ using step 6, and equation (A.2).

Step 8: approximate $u_{i+1} = p_i(t_{i+1})$

A.2.2 Using 2^{nd} Order Polynomial Spline Function:

In order to, approximate the solution of second kind linear VIE's by using 2^{nd} order polynomial spline function .We present a method of solution in algorithm (VIE2PS2):

The Algorithm (VIE2PS2):

Step 1: set h = (b-a) /n, $t_i = t_0$ +ih , i=0,…,n,(where t_0 =a , t_n =b) and $u_0 = f(a)$.

Step 2: evaluate $a_0, b_0$and c_0 by substituting (A.13)-(A.15) in equations (A.7)-(A.9).

Step 3: calculate $p_0(t)$using step2 and equation (A.6).

Step 4: approximant $u_1 = p_0(t_1)$

Step 5: for i=1 to n-1 do the following steps:

Step6: evaluate a_i, b_i, c_i, d_i and e_i by using equations (A.7)-(A.9)

and replacing $u(t_i), u'(t_i) and u''(t_i)\ by\ \ p_i(t_i)\ , p_i'(t_i) and, p_i''(t_i)$.

Step 7: calculate $p_i(t)$ using step 6, and equation (A.6).

Step 8: approximate $u_{i+1} = p_i(t_{i+1})$.

A.3 Solution of VIE' of the Second Kind with Weakly Singular kernel:

In this section, the 1^{st}order and 2^{nd} order polynomial spline function will be used to compute the numerical solution of second kind linear VIE'S with weakly singular kernel, which is:

$$u(t) - \int_0^t \frac{s^{\mu-1}}{t^{\mu}}\, u(s)ds = f(t)\,, \quad t \in [0,T] \qquad (A.16)$$

Where 0< μ<1 and f is known function, a function (16) can be converted in to the following equation form [13]:

$$t\,u'(t) + (\mu - 1)u(t) = \mu f(t) + t\,f'^{(t)} \qquad (A.17)$$

$$u_0 = \frac{\mu}{\mu - 1}\mathrm{f}(0) \qquad (A.18)$$

Hence with differentiate equation (A.17) two times with respect to t, we get

$$\left.\begin{aligned} &\mathrm{t\,u''(t) + \mu\,u'(t) = (\mu+1)f'^{(t)} + t\,f''^{(t)}} \\ &\mathrm{t\,u'''^{(t)} + (\mu+1)u''(t) = (\mu+2)f''^{(t)} + t\,f'''^{(t)}} \end{aligned}\right\} \qquad (A.19)$$

Hence replace t by a *in the realtion above* $(A.19)$ yields:

$$\mathrm{u'_0} = \frac{\mu+1}{\mu}\mathrm{f'(a)} \qquad (A.20)$$

$$\mathrm{u''_0} = \frac{\mu+2}{\mu+1}\mathrm{f''(a)} \qquad (A.21)$$

A.*3.1 First Order Polynomial Spline Function*

In order to approximate the solution of second kind linear VIE's with weakly singular kernel by using 1^{st} order polynomial spline function .We present a method of solution in algorithm **(**VIE2WSKPS1):

The Algorithm: (VIE2WSKPS1):

Step 1: Set h=(b-a)/n; $t_i = t_0 + ih, i = 0,1, \dots, n$, (where$t_0 = a, t_n =$ b)and $u_0 = \frac{\mu}{\mu\_1} f(a)$

Step 2: evaluate a_0 and b_0 by substituting (A.18) and (A.20) in equations (A.4)-(A.5).

Step 3: Calculate $P_0(t)$ using step 2 and equations (A.2).

Step 4: Approximate$u_1 \approx P_0(t_1)$.

Step 5: For i=1 to n-1 do the following steps:

Step6: Evaluatea_i, b_i, c_i , and $d_i$using equations(A.4)-(A.5).and replacing u(t) and his derivatives by $P_i(t)$ and his derivative's.

Step 7: Calculate $P_i(t)$ using step 6 and equations (A.2).

Step 8: approximate $u_{i+1} = p_i(t_{i+1})$

3.4.2 Second Order Polynomial Spline Function

We approximate the solution of second kind linear VIE'S with weakly singular kernel by using 2^{nd}order polynomial spline .In the following algorithm (VIE2WSKPS2):

The Algorithm (VIE2WSKPS2):

Step 1: Set h=(b-a)/n; $t_i = t_0 + ih, i = 0,1, \dots, n$, (where $t_0 = a, t_n = b$) and $u_0 = \frac{\mu}{\mu-1} f(a)$

Step 2: evaluate a_0, b_0 and c_0 by substituting (A.18),(A.20)and (A.21) in equations (A.7)-(A.9).

Step 3: calculate $p_0(t)$using step2 and equation (A.6).

Step 4: approximant $u_1 = p_0(t_1)$

Step 5: for i=1 to n-1 do the following steps:

Step6: evaluate a_i, b_i, c_i, d_i and e_i by using equations (A.7)-(A.9).

and replacing $u(t_i), u'(t_i) and u''(t_i)$ by $p_i(t_i)$, $p_i'(t_i) and, p_i''(t_i)$.

Step 7: calculate $p_i(t)$ using step 6, and equation (A.6).

Step 8: approximate $u_{i+1} = p_i(t_{i+1})$.

References

[1]Abdelwahid,F. (2010). Adomain Decomposition Method Appliedto Non-linearIntegral Equation. Alexandria Journal of Mathematics 1(1):11-18.

[2]Abd-AL-Hammeed,F.T (2002).Numerical Solution of Fredholm Integro-differential Equation Using Spline Functions ;Thesis M.SC, University of Technology ,Iraq.

[3]Babolian ,E.; Biazar ,J. ;and Vahidi, A.R.(2004).On the decomposition method for system linear equations and linear system of Volterra integral equations. Applied Mathematics and Computational.147: 19-27.

[4]Brunner,H.(1983).Non-Polynomial Spline Collection for Volterra Equation with Weakly Singular Kernels.SIAM Journal on Numerical Analysis ,20(6):1106-1119.

[5]Bizar, J.;and Eslami, M.(2011).Homotopy Pertubtion and Taylar series for VIE's of Second Kind. Middle East Journal of Scientific Research.7(4):604-609.

[6]Berengure , M.I.; Gamez , D.; Garralda-Guillem ,A.I.; Galan M.R.; and Perez ,M.C.S.(2009). Analytical Techniques for Numerical Solution of Linear VIE of second kind. Hinawi Publishing Corporation Abstract and Applied Analysis :1-11.

[7]Burden,R.L. ; and Faries ,J.D. (2010). Numerical Analysis .Ninth Edition.Brooks/cole, Cengage Learing.

[8]Burton,T.A (1983).Volterra Integral and Differential Equations, Acadermic press,Inc.

[9]Collins,P.j. (2006). Differential and Integral equations.Oxford University Press Inc. New York.

[10]Diogo,T.; Ford, N.j ; Lima, p. ;and Valtchev, S. (2006). Numerical method for a Volterra Integral Equation with Non-Smooth Solutions. Journul of Computational And Applied Mathematics:412-423.

[11]Diogo,T.; and Lima,P.(2008). Superconvergence of Collocation Methods for a class of weakly singular Volterra integral equations. Journal of Computational And Applied Mathematics 307-316.

[12]Eldanafof,T.S.;and Abdel Alaal, F .El. (2009). Non- polynomial Spline Method for the Solution of the dissipative Wave Equation. International Journal of Numerical method for heat and Fluid flow 19(8):950-959.

[13]G ,Ch. (2003) .Class of Bezier,Aided Geom, Design 20, 29-39.

[14]Ghoreishi , F.;and Hadizadeh , M. (2009). Numerical Computation of Tau approximation for the Volterra-Hammerstein integral equations. Springer Science Business Media 52:541-559.

[15]Geng ,F. ;and Shen ,F. (2010).Solving Integral Equation with Weakly Singular Kernel in the Reproducing Kernel Space.Islamici Azad University –Karaj Branch . 4(2):159-170.

[16]Hossinpour , A. (2012). The Solve of Integral Differential Equation by Non-Polynomial Spline Function and Quadrature Formula. International Conference on Applied Mathematics and Pharmaceutical science Jan.7-8:595-597.

[17] Haq ,F. i. (2009).Numerical Solution of Bounded Value Problem and Initial Value Problems Using Spline Function .ph.D ,thesis ,Ghulam Ishaq Institute of Engineering Science and Technology ,Pakistan.

[18]Jerri A j. (1985). Introduction to Integral Equation with Application.

Marcel Dekker,INC.

[19] Keffer , D. (1999). Advanced Analytical Techniques for the Solution of Single and multi-dimensional Integral equations. University of Tennessee.ChE 505,Augest:1-31.

[20] Lima ,P.;and Diago, T. (1997).An extrapolation method for a Volterra integral equation with weakly singular kenel. Applied Numerical Mathematics 131-148.

[21] Lambert, J.D. (1973).Computational Methods in Ordinary Differential Equations.University of Dundee,Scotland.Jhon Wiely and SonsLtd.

[22]Linze , P. (1969); Numerical Method for Volterra Integral Equations of First Kind .Courant Institute of Mathematical Sciences,N.Y.

[23]Lax,P.D.; and Richtmyer, R.D. (1956). Survey of the Stability of Linear Finite Difference Equations. Communications on Pure And Applied Mathematics,(IX):267-293.

[24]Mandal, B.N.; and chakrabarti ,A. (2011) . Applied Singular Integral Equations, science publishers,USA.

[25]Maleknejad ,K.; Hashmizadeh ,E.;and Ezzati , R. (2011).new approach to the numerical solution of VIE's by using Bernsteins approximation. Commum Non-linear Sci Numer simalat 16:647-655.

[26]Matinfar ,M.; Saeidy ,M.; and Vahidi , J. (2012). Application of Homotopy Analysis Method for solving systems of VIE.Advance in Applied Mathematics and Mechannics 4(1):36-45.

[27]Mustafa, M.M. (2004).Numerical solution for system of volterra integral Equations using spline function; ph.D. thesis,alMustansiriy University.

[28]Maleknejad ,K.; and Shahrezaee, M. (2004) .Using Runge- Kutta method for solution of the system of VIE.Applied Mathematics and Computational149:399-410.

[29]Masouia ,Z. (2012).Numerical expansion –iterative method for solving second kind Volterra and Fredholm integral equation.ISPACS:1-7.

[30]Okayama ,T.; Matsuo, T.;and Sugihara, M. (2011). Theoretical Analysis for Since-Nystrom method for VIE's.University of Tokyo:1-25

[31]Proshokouhi , M;and Ghanbari ,B. (2011). Variational Iteration Method for Solving Volterra and Fredholm Integral Equations of Second Kind. ICSRS Publication 2(1):143-148.

[32]Quaraderoni, A.; Sacco, R.; and Saleri, F.(2000).Numerical Mathematics.Springer-Verlag NewYork,Inc.

[33]Rice ,J.R. (1985). Numerical Method software and Anaylisis. SoftwareAnd analysis,Mcgra Hill.

[34]Rahman ,M. (2007).Integral Equation and their Applications .Dalhusie university,canda. WIT Press.

[35]Ramadan ,M. A.; EL-Danaf ,T.; and E.I.Abdaal ,F.(2007).Application of the Non- Polynomial Spline Approach to the Solution of the Burgers Equation.The Open Applied Mathematics Journal(1):15-20.

[36]Rahman , M.M, Hakim M. A., Hasan M. K., Alam M. K. and Nowsher, L., 2012, Numerical Solution of Volterra Integral Equations of Second Kind with the Help of Chebyshev Polynomials, Annals of Pure and Applied Mathematics, 1(2): 158-167.

[37]Rashidinia,J. ; and Zarebnia, M. (2008). New Approach for numerical solution of VIE's of the second kind .International Journal of Engineering Science,1(5-2):59-65

[38]Rashidinia,J.;mohammadi ,R. (2009).Non-Polynomial Spline Approximation of singularity perturbed boundary value problem ,TWMSJ. pure Apple.Math. 1(2.2): 336-251.

[39]Rashidinia J,.; Jalilian ,R.; and Farajeyan ,K. (2009).Non-Polynomial Spline Approach to the Solution of Fifth Order Boundary Problems. Journal of Information and Computing Science 4(4):256-274.

[40]Rashidinia , J.; Najafi , E.; and Arzhang , A. (2012). An iterative Scheme for numerical solution of VIE's using collection method and Chevpolynomial. Springer open journal 6:1-10.

[41]Suhmaker ,L.L. (2007). Spline Function Basic Therory; Third Edition, Cabride University Press.

[42]Siddiqi,S.S.; Akram ,G. ;and Kanwal ,A. (2011). Non-Polynomial Spline is used for the Numerical Solution of Fifth Order Singularity Perturbed Boundary Problems European Journal of Scientific Research 56 (3), :415-421.

[43]Tahmasbi, A. (2008).New Approach numerical solution of linear VIE's of the second kind.3 (32), 1607-1610.

[44]wazwaz ,A. (2011).Linear and Non Linear Integral Equation Method and Application, Higher Education Press ,Beijing.

[45]Wang ,W. (2006).A mechanical algorithm for solving VIE.Applied Mathematics and Computation147:1323-1341.

[46]Zarebnia, M.; Hoshyar ,M.;and Sedahti ,M. (2011).Non-Polynomial Spline Method for the Solution of Problem in Calculus of Variations. word Academy Engendering Technology (51):986-991.

[47]Zamani ,M. (2009).Three Simple Spline Method for Approximation on and Interpolation Data. Department of Technology and engineering ,Yasouj University, Iran.Contemporarag engineering Science 2:373-381.

[48] Approximation by Spline Function ,lecture 6. www.Ice.fi.teaching /s||4.1100/.

[49] Approximation Function; Math2601;www.hkmath.hku.hk,

[50] polynomial function.www.en.wikipedaia.org,