LJAST
Libyan Journal of Applied Science and Technology
مجلة ليبيا للعلوم التطبيقية والتقنية

Volume 11 Issue 01
June 2023
ISSN 2958-6119

Study Exact Solutions And Analytical Of Linear Vie's Using Non-polynomial Spline Functions

Ahmed .M . A . Elmishri[1], Mohamed . M . B . Al fetori[2] , Fateh . A . M . Elwaer[3]

[1,2]General department - Higher institute For Science and Technology, Al garaboulli - Tripoli – Libya

Elmeshri.ahmed@yahoo.com[1] , Mohamed Al fetori907@Gmail.com[2]

[3]General department - Higher institute For Science and Technology, Al shumoukh - Tripoli – Libya

Fatehelwaer1@Gmail.com[3]

Abstract

In this paper, 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.

In section (2) linear VIE's of the second kind have been solved using linear and quadratic non-polynomial spline functions in subsection (2.1) and (2.2) respectively. In section (3), the solution of linear VIE's of the first kind with $k(x,x) \neq 0$ will be introduced. In section (4), we used linear and quadratic non-polynomial spline functions to solve VIE's with weak singular kernel in subsection (4.1) and (4.2) respectively. In section (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 (6), including a discussion for this.

Algorithm developed for solving those equations using MATLAB programming.

Keywords: exact solutions, numerical solutions, Non-polynomial, Spline function, analytical solutions, Volterra integral equation, weakly singular kernel.

-1- Introduction.

In 1997, Diogo and T-Lima presented an inductive solution method for finding the numerical solution to Volterra integral equation with weak singular kernel. In 2012, Hossinpour solved integral differential equations using non-polynomial spline function.

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 applications such as the population dynamic, spread of epidemics [8].

-1.1- Definition. [4]

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:

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

$$where \quad g(t) = (t-x)^k = \begin{cases} (t-x)^k & if\ x \le t \\ 0 & otherwise \end{cases} \quad (2)$$

By Newton form of the interpolating polynomial we have, ([4]).

$$f(x_i, \dots, x_n) = \sum_{i=0}^{n} \frac{f(x_i)}{W'_{n+1}(x_i)} \quad (3)$$

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

-1.2- Linear Non-Polynomial Spline Function.

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

$$p_i(t) = a_i cosk(t-t_i) + b_i sink(t-t_i) + c_i(t-t_i) + d_i \quad , i = 0,1, \dots, n \quad (4)$$

where are constants to be determined. In order to obtain the value of, we differentiate equation (4) three times with respect to t, then we get:

$$\left.\begin{aligned} p'_i(t) &= -k a_i sink(t-t_i) + k b_i cosk(t-t_i) + c_i \\ p''_i(t) &= -k^2 a_i cosk(t-t_i) - k^2 b_i sink(t-t_i) \\ p'''_i(t) &= k^3 a_i sink(t-t_i) - k^3 b_i cosk(t-t_i) \end{aligned}\right\} \quad (5)$$

Hence replace t by in the relation (4) and (5) yields:

$p_i(t_i) = a_i + d_i$
$p'_i(t_i) = k b_i + c_i$
$p''_i(t_i) = -k^2 a_i$
$p'''_i(t_i) = -k^3 b_i$

From the above equations, the values of are obtained as follows:

$$a_i = \frac{-1}{k^2} p''_i(t_i) \quad (6)$$

$$b_i = \frac{-1}{k^3} p'''_i(t_i) \quad (7)$$

$$c_i = p'_i(t_i) + k b_i \quad (8)$$

$$d_i = p_i(t_i) + a_i \quad (9)$$

$$for\ i = 0,1, \dots, n$$

-1.3- Quadratic Non –Polynomial Spline Function.

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

$$Q_i(t) = a_i cosk(t-t_i) + b_i sink(t-t_i) + c_i(t-t_i) + d_i(t-t_i)^2 + e_i \quad (10)$$

$, i = 0,1, \dots, n$ 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 (10) four times with respect to t, and then we get the following equations:

$$\left.\begin{aligned} Q_i'(t) &= -ka_i sink(t-t_i) + kb_i cosk(t-t_i) + c_i + 2d_i(t-t_i) \\ Q_i''(t) &= -k^2 a_i cosk(t-t_i) - k^2 b_i sink(t-t_i) + 2d_i \\ Q_i'''(t) &= k^3 a_i sink(t-t_i) - k^3 b_i cosk(t-t_i) \\ Q_i''''(t) &= k^4 a_i cosk(t-t_i) + k^4 b_i sink(t-t_i) \end{aligned}\right\} \quad (11)$$

Hence replace t by in the relation (10) and (11) yields:

$Q_i(t_i) = a_i + e_i$
$Q_i'(t_i) = kb_i + c_i$
$Q_i''(t_i) = -k^2 a_i + 2d_i$
$Q_i'''(t_i) = -k^3 b_i$
$Q_i''''(t_i) = k^4 a_i$

We obtain the values of from the above relations as follows:

$$a_i = \frac{1}{k^4} Q_i''''(t_i) \quad (12)$$

$$b_i = \frac{-1}{k^3} Q_i'''(t_i) \quad (13)$$

$$c_i = Q_i'(t_i) - kb_i \quad (14)$$

$$d_i = \frac{1}{2}[Q_i''(t_i) + k^2 a_i] \quad (15)$$

$$e_i = Q_i(t_i) - a_i \quad (16)$$

$$for\ i = 0,1,\dots,n$$

-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:

$$u(x) = f(x) + \int_a^x k(x,t)u(t)dt \quad ,x \in [a,b] \quad (17)$$

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 (17), we need to differentiate equation (17) four times with respect to x, by using Libenze formula we realize:

$$u'(x) = f'(x) + \int_a^x \frac{\partial k(x,t)}{\partial x} u(t)dt + k(x,x)u(x) \quad (18)$$

$$u''(x) = f''(x) + \int_a^x \frac{\partial^2 k(x,t)}{\partial x^2} u(t)dt + \left(\frac{\partial k(x,t)}{\partial x}\right)_{t=x} u(x) + \frac{dk(x,x)}{dx} u(x) + k(x,x)u'(x) \quad (19)$$

$$u'''(x) = f'''(x) + \int_a^x \frac{\partial^3 k(x,t)}{\partial x} u(t)dt + (\frac{\partial^2 k(x,t)}{\partial x^2})_{t=x}u(x) + \frac{d}{dx}(\frac{\partial k(x,t)}{\partial x})_{t=x}u(x) + (\frac{\partial k(x,t)}{\partial x})_{t=x}u'(x) + \frac{d^2k(x,x)}{dx^2}u(x) + 2\frac{dk(x,x)}{dx}u'(x) + k(x,x)u''(x) \quad (20)$$

$$u^{(4)}(x) = f^{(4)}(x) + \int_a^x \frac{\partial^4 k(x,t)}{\partial x^4} u(t)dt + (\frac{\partial^3 k(x,t)}{\partial x^3})_{t=x}u(x) + \frac{d}{dx}(\frac{\partial^2 k(x,t)}{\partial x^2})_{t=x}u(x) + (\frac{\partial^2 k(x,t)}{\partial x^2})_{t=x}u'(x) + \frac{d^2}{dx^2}(\frac{\partial k(x,t)}{\partial x})_{t=x}u(x) + \frac{d}{dx}(\frac{\partial k(x,t)}{\partial x^2})_{t=x}u'(x) + \frac{d}{dx}(\frac{\partial k(x,t)}{\partial x})_{t=x}u'(x) + (\frac{\partial k(x,t)}{\partial x})_{t=x}u''(x) + \frac{d^3k(x,x)}{dx^3}u(x) + 3\frac{d^2k(x,x)}{dx^2}u'(x) + 3\frac{dk(x,x)}{dx}u''(x) + k(x,x)u'''(x) \quad (21)$$

To complete our procedure for solving VIE's. we substitute $x = a$ in equations (17) - (21), then we get:

$$u_0 = u(a) = f(a) \quad (22)$$

$$u_0' = u'(a) = f'(a) + k(a,a)u(a) \quad (23)$$

$$u_0'' = u''(a) = f''(a) + [(\frac{\partial k(x,t)}{\partial x})_{t=x}]_{x=a}u(a) + (\frac{\partial k(x,t)}{\partial x})_{x=a}\frac{dk(x,x)}{dx}u(a) + k(a,a)u'(a) \quad (24)$$

$$u_0''' = u'''(a) = f'''(a) + [(\frac{\partial^2 k(x,t)}{\partial x^2})_{t=x}]_{x=a}u(a) + [\frac{d}{dx}(\frac{\partial k(x,t)}{\partial x})_{t=x}]_{x=a}u(a) + [(\frac{\partial k(x,t)}{\partial x})_{t=x}]_{x=a}u'(a) + [\frac{d^2k(x,x)}{dx^2}]_{x=a}u(a) + 2[\frac{dk(x,x)}{dx}]_{x=a}u'(a) + k(a,a)u''(a) \quad (25)$$

$$u_0^{(4)} = u^{(4)}(a) = f^{(4)}(a) + [(\frac{\partial^3 k(x,t)}{\partial x^3})_{t=x}]_{x=a}u(a) + [\frac{d}{dx}(\frac{\partial^2 k(x,t)}{\partial x^2})_{t=x}]_{x=a}u(a) + (\frac{\partial^2 k(x,t)}{\partial x^2})_{t=x}u'(a) + [\frac{d^2}{dx^2}(\frac{\partial k(x,t)}{\partial x})_{t=x}]_{x=a}u(a) + 2[\frac{d}{dx}(\frac{\partial k(x,t)}{\partial x})_{t=x}]_{x=a}u'(a) + [(\frac{\partial k(x,t)}{\partial x})_{t=x}]_{x=a}u''(a) + [\frac{d^3k(x,x)}{dx^3}]_{x=a}u(a) + 3[\frac{d^2k(x,x)}{dx^2}]_{x=a}u'(a) + 3[\frac{dk(x,x)}{dx}]_{x=a}u''(a) + k(a,a)u'''(a) \quad (26)$$

Now, we try to solve equation (17) using linear and quadratic nonpolynomial spline functions

-2.1- Using Linear Non-Polynomial Spline Function.

We approximate the solution of linear VIE's of the second kind (17) by using linear non-polynomial spline function (3). We introduce a method of solution in algorithm (VIE2NPS1):

The Algorithm (VIE2NPS1).

Step 1: Set $h = \frac{b-a}{n}$, $t_i = t_0 + ih, i = 0, \dots, n$ ($where\ t_0 = a, t_n = b$) $and\ u_0 = f(a)$.
Step 2: Evaluate $a_0, b_0, c_0\ and\ d_0$ and by substituting (22) -(25) in equations (6) - (9).
Step 3: Calculate $P_0(t)$ using step2 and equation (3).
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 (6) -(9) 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 (3).
Step 8: Approximate $u_{i+1} = P_i(t_{i+1})$.

-2.2- Using Quadratic Non-Polynomial Spline Function:
In order to approximate the solution of linear VIE's of second kind (17) by using quadratic non-polynomial spline function (10). We present a method of solution in algorithm (VIE2NPS2):

The Algorithm (VIE2NPS2):

Step 1: Set $h = \frac{b-a}{n}$, $t_i = t_0 + ih, i = 0, \dots, n$ ($where\ t_0 = a, t_n = b$) $and\ u_0 = f(a)$.
Step 2: Evaluate $a_0, b_0, c_0, d_0\ and\ e_0$ and by substituting (22) -(26) in equations (12) - (16).
Step 3: Calculate $P_0(t)$ using step2 and equation (10).
Step 4: Approximate $u_1 = P_0(t_1)$.
Step 5: For $i = 1\ to\ n-1$ do the following steps:
Step6: Evaluate $a_i, b_i, c_i, d_i\ and\ e_i$ by using equations (12) -(16) 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 (10).
Step 8: Approximate $u_{i+1} = P_i(t_{i+1})$.

-3- Solution 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:

$$g(x) = \int_a^x r(x,t)u(t)dt, \quad x \in [a,b] \tag{27}$$

where $r(x,t)$ and $g(x)$ are known functions and continuous in $C^4[a,b]$, but $u(x)$ is unknown function. When $r(x,t) \neq 0$, we differential equations (27) one time with respect to x.Therefore, we get conversion to the second kind, *i.e.*;

$$u(x) = \frac{1}{r(x,x)}\left[g'(x) - \int_a^x \frac{\partial r(x,t)}{\partial x}u(t)dt\right] \tag{28}$$

then we use the non-polynomial spline function and algorithms (VIE2NPS1) and (VIE2NPS2) to solve (28).

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

-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:

$$u(x) - \int_0^x \frac{t^{\mu-1}}{x^{\mu}} u(t)dt = f(x),\ \ x \in [0,T] \tag{29}$$

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 (29), we multiply both sides of (29) by x^{μ} yields to:

$$x^{\mu}u(x) - \int_0^t t^{\mu-1}\, u(t)dt = f(x)x^{\mu} \tag{30}$$

Hence differentiation (30) with respect to x, we get:

$$x^{\mu}u'(x) + \mu x^{\mu-1}u(x) - \frac{1}{x^{1-\mu}}u(x) = \mu x^{\mu-1}f(x) + f'(x)x^{\mu} \tag{31}$$

And multiplication both sides of (31) by $x^{1-\mu}$ yields,

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

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 $u_0 = \frac{\mu}{\mu-1} f(0)$.

Hence equation (29) may be converted into the following form [2]:

$$xu'(x) + (\mu - 1)u(x) = \mu f(x) + xf'(x) \tag{32}$$

With

$$u_0 = \frac{\mu}{\mu - 1} f(0) \tag{33}$$

Hence differentiate equation (32) four times with respect to x, we get:

$$\left.\begin{aligned} xu''(x) + \mu u'(x) &= (\mu+1)f'(x) + xf''(x) \\ xu'''(x) + (\mu+1)u''(x) &= (\mu+2)f''(x) + xf'''(x) \\ xu^{(4)}(x) + (\mu+2)u'''(x) &= (\mu+3)f'''(x) + xf^{(4)}(x) \\ xu^{(5)}(x) + (\mu+3)u^{(4)}(x) &= (\mu+4)f^{(4)}(x) + xf^{(5)}(x) \end{aligned}\right\} \tag{34}$$

Hence replace $x\ by\ a$ in the relation, yields:

$$\left.\begin{aligned} u_0' &= \frac{\mu+1}{\mu} f'(a) \\ u_0'' &= \frac{\mu+2}{\mu+1} f''(a) \\ u_0''' &= \frac{\mu+3}{\mu+2} f'''(a) \\ u_0^{(4)} &= \frac{\mu+4}{\mu+3} f^{(4)}(a) \end{aligned}\right\} \qquad (35)$$

-4.1- Using Linear Non-Polynomial Spline Function.

In order to approximate the solution of linear VIE's of the second kind with weakly singular kernel (29) by using linear non-polynomial spline function (3). We present a method of solution in algorithm **(**VIE2WSKNPS1):

The Algorithm: (VIE2WSKNPS1):

Step 1: Set $h = \frac{b-a}{n}$, $t_i = t_0 + ih, i = 0,1,\dots,n$ ($where\ t_0 = a, t_n = b$) $and\ u_0 = \frac{\mu}{\mu-1} f(a)$.
Step 2: Evaluate $a_0, b_0, c_0\ and\ d_0$ and by substituting (33) -(35) in equations (6) - (9).
Step 3: Calculate $P_0(t)$ using step2 and equation (3).
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 (6) -(9) 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 equation (3).
Step 8: Approximate $u_{i+1} = P_i(t_{i+1})$.

-4.2- Using 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 (10). In the following algorithm (VIE2WSKNPS2):

The Algorithm (VIE2WSKNPS2):

Step 1: Set $h = \frac{b-a}{n}$, $t_i = t_0 + ih, i = 0,1,\dots,n$ ($where\ t_0 = a, t_n = b$) $and\ u_0 = \frac{\mu}{\mu-1} f(a)$.
Step 2: Evaluate $a_0, b_0, c_0, d_0\ and\ e_0$ and by substituting (33) -(35) in equations (12) - (16).
Step 3: Calculate $P_0(t)$ using step2 and equation (10).
Step 4: Approximate $u_1 = P_0(t_1)$.
Step 5: For $i = 1\ to\ n-1$ do the following steps:

Step6: Evaluate $a_i, b_i, c_i, d_i \; and \; e_i$ substituting in equations (12) -(16) and replacing $u(t)$ and its derivatives by $P_i(t)$ and its derivative's.
$u(t_i), u'(t_i), u''(t_i), u'''(t_i) and \; u^{(4)}(t_i) by \; P_i(t_i), P_i'(t_i), P_i''(t_i), P_i'''(t_i) and \; P_i^{(4)}(t_i)$.
Step 7: Calculate $P_i(t)$ using step 6 and equation (10).
Step 8: Approximate $u_{i+1} = P_i(t_{i+1})$.

-5- Numerical Examples.

Example (1): Consider the VIE of the second kind [5].

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

With exact solution $\emptyset(x) = sinx$.
Table (5.1) presents a comparison between the exact and numerical solution using linear and quadratic nonpolynomial spline functions, where $P_i(x)$ denotes the approximate solution using non-polynomial spline function, with h=0.1.

Table 5.1: 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 (5.2) present a comparison between the error in our methods and other method in [5], where Error =|exact value –numerical value| and $\|err\|_\infty = max|Error|$.

Table 5.2: comparison between the error with reference [5].

x	Error in linear	Error in quadratic	Error obtains in [5]
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	3.33066907387547e-16	3.33066907387547e-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 (5.3) present a comparison between error obtain using linear and quadratic non-polynomial spline functions and polynomial spline function including (1st order and 2nd order: Algorithm (**VIE2PS1**) and Algorithm (**VIE2PS1**)), with h=0.1.

Table 5.3: Comparison between error obtain using polynomial and nonpolynomial 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	0.000166583353	0.000166583353
0.2	0	0	0.098669330795	0.098669330795
0.3	5.55111512312e-17	5.55111512312e-17	0.095520206661	0.095520206661
0.4	1.110223024625e-16	1.110223024625e-16	0.189418342308	0.189418342308
0.5	1.110223024625e-16	1.110223024625e-16	0.179425538604	0.179425538604
0.6	1.110223024625e-16	1.110223024625e-16	0.264642473395	0.264642473395
0.7	1.110223024625e-16	1.110223024625e-16	0.244217687237	0.244217687237
0.8	2.224460492503e-16	2.224460492503e-16	0.317356090899	0.317356090899
0.9	3.330669073875e-16	3.330669073875e-16	0.283326909627	0.283326909627
1	4.440892098500e-16	4.440892098500e-16	0.341470984807	0.341470984807
$\|\|err\|\|_\infty$	**4.440892098500e-16**	**4.440892098500e-16**	**0.341470984807**	**0.341470984807**

Example (2): Consider the VIE of the second kind [7].

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

With exact solution $y(x) = cosx$.
Tables (5.4) present a comparison between the exact and numerical solution of linear and quadratic nonpolynomial spline functions where $P_i(x)$ denote the approximate solution using non-polynomial spline function, with h=0.1.

Table 5.4: 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 (5.5) present a comparison between the error in our methods and other method in [7], where error =|exact value –numerical value| and $\|err\|_\infty = max|Error|$.

Table 5.5: comparison between the error with reference [7].

x	Error in linear	Error in quadratic	Error obtains in [7]
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
$\|\|err\|\|_\infty$	**444089209850e-16**	**444089209850e-16**	**1.142e-011**

Table (5.6) present a comparison between error obtain using linear and quadratic non-polynomial spline functions and polynomial spline function including (1st order and 2nd order: Algorithm (**VIE2PS1**) and Algorithm (**VIE2PS1**)), with h=0.1.

Table 5.6: Comparison between error obtain using polynomial and nonpolynomial 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	0.004995834721	0.000004165278
0.2	0.1110223024e-16	0.1110223024e-16	0.019933422158	0.004933422158
0.3	0.1110223024e-16	0.1110223024e-16	0.044663510874	0.014663510874
0.4	0.2220446049e-16	0.2220446049e-16	0.078939005997	0.028939005997
0.5	0.2220446049e-16	0.2220446049e-16	0.122417438109	0.047417438109
0.6	0.2220446049e-16	0.2220446049e-16	0.174664385093	0.069664385090
0.7	0.3330669073e-16	0.3330669073e-16	0.235157812715	0.095157812715
0.8	0.3330669073e-16	0.3330669073e-16	0.303293290652	0.123293290652
0.9	0.3330669073e-16	0.3330669073e-16	0.378390031729	0.153390031729
1	444089209850e-16	444089209850e-16	0.459697694131	0.184697694131
$\|\|err\|\|_\infty$	**444089209850e-16**	**444089209850e-16**	**0.459697694131**	**0.184697694131**

Example (3): Consider the VIE of the second kind [6].

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

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

Tables (5.7) 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 5.7: Exact and numerical solution of test example (3).

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 (5.8) present a comparison between the error in our methods and other method in [6], where error =|exact value –numerical value| and $\|err\|_\infty = max|Error|$.

Table 5.8: comparison between the error with reference [6].

x	Error in linear	Error in quadratic	Error obtains in [6]
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	-
$\|err\|_\infty$	**4.440892098500626e-16**	**4.440892098500626e-16**	**3.6208210e-04**

Table (5.9) present a comparison between error obtain using nonpolynomial spline function including linear and quadratic polynomial spline function including (1st order and 2nd order: Algorithm (**VIE2PS1**) and Algorithm (**VIE2PS1**)]) with h=0.1.

Table 5.9: Comparison between error obtain using polynomial and nonpolynomial 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	0.0001665833531
0.2	0	0	0.0986693307950	0.0986693307950
0.3	0	0	0.0955202066613	0.0955202066613

0.4	2.2204460492e-16	2.2204460492e-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	2646424733950
0.7	1.1102230246e-16	1.1102230246e-16	0.2442176872376	0.2442176872376
0.8	2.2204460492e-16	2.2204460492e-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**

Example (4): Consider the VIE of the first kind [3].

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

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 (5.10) 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 5.10: Exact and numerical solution of test example (4).

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 (5.11) present a comparison between the error in our methods and other method in [3], where error =|exact value –numerical value| and $\|err\|_\infty = max|Error|$.

Table 5.11: comparison between the error with reference [3].

x	Error in linear	Error in quadratic	Error obtains in [3]
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	-
$\|\|err\|\|_\infty$	**0**	**0**	**-**

Table (5.12) present a comparison between error obtain using linear and quadratic non-polynomial and polynomial spline function including (1st order and 2nd order: Algorithm (**VIE2PS1**) and Algorithm (**VIE2PS1**)) with h=0.1.

Table 5.12: 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	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**

Example (5): Consider the VIE of second kind with Weakly Singular Kernel [2].

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

Where $f(x) = x + 1$ and $\mu = 0.5$, with $u(x) = \frac{\mu}{\mu-1} + \frac{\mu+1}{\mu}x$.

Tables (5.13) 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 non-polynomial spline functions, with h=0.1.

Table 5.13: 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	-0.700000000000000	-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 (5.14) present a comparison between the error in our methods and other method in [2] where error =|exact value –numerical value| and $\|err\|_\infty = max|Error|$.

Table 3.26: comparison between the error with reference [2].

x	Error in linear	Error in quadratic	Error obtains in [2]
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
$\|err\|_\infty$	**2.220446049250313e-15**	**2.220446049250313e-15**	-

Table (5.15) present a comparison between error obtain using nonpolynomial spline function including linear and polynomial spline function including (1st order and 2nd order: Algorithm (**VIE2PS1**) and Algorithm (**VIE2PS1**)), with h=0.1.

Table 5.15: Comparison between error obtain using polynomial and nonpolynomial 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	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**

Example (6): Consider the VIE of second with Weakly Singular Kernel [1].

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

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 (5.16) 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 5.16: Exact and numerical solution of test example (6).

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 (5.17) present a comparison between the error in our methods and other method in [1], where error =|exact value –numerical value| and $\|err\|_\infty = max|Error|$. with h=0.01.

Table 5.17: comparison between the error with reference [1].

x	Error in linear	Error in quadratic	Error obtains in [1]
$\|err\|_\infty$	**2.220446049250313e-15**	**2.220446049250313e-15**	-

Table (5.18) present a comparison between error obtain using nonpolynomial spline function including linear and polynomial spline function including (1st order and 2nd order: Algorithm (**VIE2PS1**) and Algorithm (**VIE2PS1**)]) with h=0.1.

Table 5.18: Comparison between error obtain using polynomial and nonpolynomial 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	1.666666666e-02	0
0.2	1.3884260085417e-05	5.551115120e-17	6.666666666e-02	5.551115120e-17
0.3	2.2192613747207e-04	2.220446049e-16	1.500000000e-01	2.220446049e-16
0.4	1.1216304186860e-03	5.551115120e-17	2.666666666e-01	5.551115120e-17
0.5	3.5366466762822e-03	0	4.166666666e-01	0
0.6	8.6085396345747e-03	0	5.999999999e-01	0
0.7	1.7785383032259e-02	4.440892090e-16	8.166666666e-01	4.440892090e-16
0.8	3.2807290948292e-02	8.881784190e-16	1.066666666e+00	8.881784190e-16
0.9	5.5689031157217e-02	8.881784190e-16	1.350000000e+00	8.881784190e-16
1	8.8699894235547e-02	4.440892090e-16	1.666666666e+00	4.440892090e-16
$\|err\|_\infty$	**8.8699894235547e-02**	**8.881784190e-16**	**1.666666666e+00**	**8.881784190e-16**

6. Conclusion.

In this paper, we have introduced numerical methods for approximating the solution of three types of integral equations: which 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). So, we get the following:

- 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.

References

[1] 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.

[2] 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.

[3] Linze, P. (1969); Numerical Method for Volterra Integral Equations of First Kind. Courant Institute of Mathematical Sciences, N.Y.

[4] Quaraderoni, A.; Sacco, R.; and Saleri, F.(2000). Numerical Mathematics.Springer-Verlag NewYork,Inc.

[5] 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.

[6] 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.

[7] Tahmasbi, A. (2008). New Approach numerical solution of linear VIE's of the second kind.3 (32), 1607-1610.

[8] Wazwaz, A. (2011). Linear and Non-Linear Integral Equation Method and Application, Higher Education Press, Beijing.