Algorithms for Solving Volterra Integral Equations Using Non-Polynomial Spline Function

Issa Hamza Lafta
Education College for Pure Sciences, Wasit University, Iraq
essaha1101@uowasit.edu.iq

Ahmed Shihab Al-Atabi
Directorate General of Wasit Education
drhmed99@gmail.com

Abstract— This paper discusses some algorithms for solving Volterra's integral equations of the second degree using non-boundary spline functions, including (linear and cubic), as the linear spline function presents some laws and algorithms of its own, as well as the cubic spline function has its laws and algorithm. They are applied and combined in a language program MATLAB R2022a to find the numerical solution for them and the difference between them through the error percentage that appears for each of them. The preview is done through the drawing, as shown in the examples below.

Keywords— Non-Polynomial Linear Spline Function, Cubic Non – Polynomial Spline Function, Solution of Second-Kind Linear VIEs, Using Linear Non-Polynomial Spline Function, Using Cubic Non-Polynomial Spline Function.

1.Introduction

History of spline functions. The extrapolation method used to acquire the numerical solution gave [1, 2] weakly anomalous nuclei in 1997. VIE. Frontier presented new spline functions to solve the Volterra equations. Haq 2009's [3] limit value and start numerical value solutions Applying spline learning. [Answer] in 2011. Cubic spline function-based boundary value problem solution. In 2012,[4] Hosseinpour proposed Spline Unlimited to solve integral differential equations. The integral differential Volterra equation [Majeed, S] [5]has a numerical solution for 2014. The second method—a numerical answer—uses non-boundary spline functions [Harbi, S., Murad, and S. Majeed]. Any second-order linear Volterra integral equation utilizes third-degree non-parametric spline functions in 2015[6]. [Other spline function method] Taqi, A., and Nissan, B. developed frontier versions of the nonlinear Volterra integral equations in 2016.[7]

2. I delved deeper into using not polynomial equations, Linear and Cubic.

A. *Non-Polynomial Linear Spline Function[3, 7]*

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

$$L_i(t) = \alpha_i \cos h\,(t - t_i) + \beta_i \sin h\,(t - t_i) + \sigma_i(t - t_i) + \delta_i \qquad (1)$$

s.t $i = 0, 1, \ldots, m$ the variables $\alpha_i, \beta_i, \sigma_i, and\ \delta_i$ Need to be determined. We divide the equation above three times for t to find the values of $\alpha, \beta, \sigma, and\ \delta$. The result is:

$$L_i'(t) = -h\alpha_i \sin h\,(t - t_i) + h\beta_i \cos h\,(t - t_i) + \sigma_i \qquad (2)$$
$$L_i''(t) = -h^2\alpha_i \cos h\,(t - t_i) - h^2\beta_i \sin h\,(t - t_i) \qquad (3)$$
$$L_i'''(t) = h^3\alpha_i \sin h\,(t - t_i) - h^3\beta_i \cos h\,(t - t_i) \qquad (4)$$

Consequently, changing t to t_i The connection and produces:

$$L_i(t_i) = \alpha_i + \delta_i \qquad (5)$$
$$L_i'(t_i) = h\beta_i + \sigma_i \qquad (6)$$
$$L_i''(t_i) = -h^2\alpha_i \qquad (7)$$
$$L_i'''(t_i) = -h^3\beta_i \qquad (8)$$

The values of $\alpha_i, \beta_i, \sigma_i, and\ \delta_i$ Are determined as follows from the equations above:

$$\alpha_i = -\frac{1}{h^2} L_i''(t_i) \qquad (9)$$
$$\beta_i = -\frac{1}{h^3} L_i'''(t_i) \qquad (10)$$
$$\sigma_i = L_i'(t_i) + h\beta_i \qquad (11)$$
$$\delta_i = L_i(t_i) + \alpha_i \qquad for\ \ i = 0, 1, \ldots, m \qquad (12)$$

B. *Cubic Non – Polynomial Spline Function [3, 7]*

The Cubic Non-Polynomial Spline Function has the following form:

$$C_i(t) = \alpha_i \cos h\,(t - t_i) + \beta_i \sin h\,(t - t_i) + \sigma_i(t - t_i) + \delta_i(t - t_i)^2 + \eta_i(t - t_i)^3 + \gamma_i \qquad (13)$$

The variables $\alpha_i, \beta_i, \sigma_i, \delta_i, \eta_i, and\ \gamma_i$ Need to be determined. We differentiate the equation above three times concerning t to determine the values of $\alpha_i, \beta_i, \sigma_i, \delta_i, \eta_i, and\ \gamma_i$, and we then obtain the following equations:

$$C_i'(t) = -h\alpha_i \sin h\,(t - t_i) + h\beta_i \cos h\,(t - t_i) + \sigma_i + 2\delta_i(t - t_i) + 3\eta_i(t - t_i)^3 \qquad (14)$$
$$C_i''(t) = -h^2\alpha_i \cos h\,(t - t_i) - h^2\beta_i \sin h\,(t - t_i) + 2\delta_i + 6\eta_i(t - t_i)^2 \qquad (15)$$
$$C_i'''(t) = h^3\alpha_i \sin h\,(t - t_i) - h^3\beta_i \cos h\,(t - t_i) + 12\eta_i(t - t_i) \qquad (16)$$
$$C_i^4(t) = h^4\alpha_i \cos h\,(t - t_i) + h^4\beta_i \sin h\,(t - t_i) + 12\eta_i \qquad (17)$$
$$C_i^5(t) = -h^5\alpha_i \sin h\,(t - t_i) + h^5\beta_i \cos h\,(t - t_i) \qquad (18)$$

Consequently, changing $t\ to\ t_i$ In relation and results in:

$$C_i(t_i) = \alpha_i + \gamma_i \qquad (19)$$
$$C_i'(t_i) = h\beta_i + \sigma_i \qquad (20)$$
$$C_i''(t_i) = -h^2\alpha_i + 2\delta_i \qquad (21)$$
$$C_i'''(t_i) = -h^3\beta_i + 6\eta_i \qquad (22)$$
$$C_i^4(t_i) = h^4\alpha_i \qquad (23)$$
$$C_i^5(t_i) = h^5\beta_i \qquad (24)$$

From the relation above, we may deduce the values of $\alpha_i, \beta_i, \sigma_i, \delta_i, \eta_i, and\ \gamma_i$ As follows:

$$\alpha_i = \frac{1}{h^4} C_i^4(t_i) \qquad (25)$$

$$\beta_i = \frac{1}{h^5} C_i^5(t_i) \tag{26}$$
$$\sigma_i = C_i'(t_i) - h\beta_i \tag{27}$$
$$\delta_i = \frac{1}{2}[C_i''(t_i) + \ h^2 a_i] \tag{28}$$
$$\eta_i = \frac{1}{6}[C_i'''(t_i) + h^3\beta_i] \tag{29}$$
$$\gamma_i = C_i(t_i) - \alpha_i \qquad for\ i = 0,1, \dots ,\ m \tag{30}$$

3.The Volterra integral equation of the second kind was used with the non-polynomial spline function of all kinds (linear and cubic).

Math and science use integral equations. Numerical solutions to test problems with known precise answers validate the method. Many equations cannot be solved analytically. Numerical approaches should fix these issues. Numerical methods require faster processors and codes. Population dynamics and disease transmission are Volterra integral equation applications. Many research address the Volterra integral equation. This chapter used linear and quadratic non-polynomial spline functions to solve VIEs numerically. First-time algorithm proposals are crucial to this effort's success. Discuss the MATLAB R2022a algorithm for solving such problems.[1, 8, 9]

Solution of Second-Kind Linear VIEs.[10, 11]

In this section, we use the linear and quadratic non-polynomial spline functions to find the numerical solution of the linear VIEs of the second sort, which have the following form.

$$\mu(y) = \phi(y) + \int_{\alpha}^{y} k(y,t)\,\mu(t)dt \qquad s.t\ y \in [\alpha, \beta] \tag{31}$$

Where $\mu(y)$ is an unknown function while $k(y,t)$ and $\phi(y)$ are known parts that are continuous in $C_4\ in\ [\alpha, \beta]$. The Libenze formula reveals that we must differentiate the equation four times about y to solve it.

$$\mu'(y) = \phi'(y) + \int_{\alpha}^{y} \frac{\partial k(y,t)}{\partial y}\mu(t)dt\ k(y,y)\mu(y) \tag{32}$$

$$\mu''(y) = \phi''(y) + \int_{\alpha}^{y} \frac{\partial^2 k(y,t)}{\partial y^2}\mu(t)dt + \left(\frac{\partial k(y,t)}{\partial y}\right)_{t=y}\mu(y) + \left(\frac{dk(y,y)}{dy}\right)\mu(y) + k(y,y)\mu'(x) \tag{33}$$

$$\mu'''(y) = \phi'''(y) + \int_{\alpha}^{y} \frac{\partial^3 k(y,t)}{\partial y^3}\mu(t)dt + \left(\frac{\partial^2 k(y,t)}{\partial y^2}\right)_{t=y}\mu(y) + \frac{d}{dy}\left(\frac{\partial k(y,t)}{\partial y}\right)_{t=y}\mu(y) + \left(\frac{\partial k(y,t)}{\partial y}\right)_{t=y}\mu'(y) + \left(\frac{d^2 k(y,y)}{dy^2}\right)\mu(y) + 2\left(\frac{dk(y,y)}{dy}\right)\mu'(y) + k(y,y)\mu''(y) \quad (34)$$

$$\mu^4(y) = \phi^4(y) + \int_{\alpha}^{y} \frac{\partial^4 k(y,t)}{\partial y^4}\mu(t)dt + \left(\frac{\partial^3 k(y,t)}{\partial y^3}\right)_{t=y}\mu(y) + \frac{d}{dy}\left(\frac{\partial^2 k(y,t)}{\partial y^2}\right)_{t=y}\mu(y) + \left(\frac{\partial^2 k(y,t)}{\partial y^2}\right)_{t=y}\mu'(y) + \frac{d^2}{dy^2}\left(\frac{\partial k(y,t)}{\partial y}\right)_{t=y}\mu(y) + \frac{d}{dy}\left(\frac{\partial k(y,t)}{\partial y^2}\right)_{t=y}\mu'(y) + \left(\frac{\partial k(y,t)}{\partial y}\right)_{t=y}\mu''(y) + \left(\frac{d^3 k(y,y)}{dy^3}\right)\mu(y) + 3\left(\frac{d^2 k(y,y)}{dy^2}\right)\mu'(y) + 3\left(\frac{dk(y,y)}{dy}\right)\mu''(y) + k(y,y)\mu'''(y) \quad (35)$$

To finish our method for resolving VIEs. When we enter $y = \alpha$ in Equations through, we obtain:

$$\mu_0 = \mu(\alpha) = \phi(\alpha) \quad (36)$$

$$\mu_0' = \mu'(\alpha) = \phi'(\alpha) + k(\alpha,\alpha)\mu(\alpha) \quad (37)$$

$$\mu_0'' = \mu''(\alpha) = \phi''(\alpha) + \left(\left(\frac{\partial k(y,t)}{\partial y}\right)_{t=y}\right)_{y=\alpha}\mu(\alpha)\left(\frac{\partial k(y,t)}{\partial y}\right)_{t=y}\left(\frac{dk(y,y)}{dy}\right)\mu(\alpha) + k(\alpha,\alpha)\mu'(\alpha) \quad (38)$$

$$\mu_0''' = \mu'''(\alpha) = \phi'''(\alpha) + \left(\left(\frac{\partial^2 k(y,t)}{\partial y^2}\right)_{t=y}\right)_{y=\alpha}\mu(\alpha) + \left(\frac{d}{dy}\left(\frac{\partial k(y,t)}{\partial y}\right)_{t=y}\right)_{y=\alpha}\mu(\alpha) + \left(\left(\frac{\partial k(y,t)}{\partial y}\right)_{t=y}\right)_{y=\alpha}\mu'(\alpha) + \left(\frac{d^2 k(y,y)}{dy^2}\right)_{y=\alpha}\mu(\alpha) + 2\left(\frac{d^2 k(y,y)}{dy^2}\right)_{y=\alpha}\mu'(\alpha) + k(\alpha,\alpha)\mu''(\alpha) \quad (39)$$

$$\mu_0^4 = \mu^4(\alpha) = \phi^4(\alpha) + \left(\left(\frac{\partial^3 k(y,t)}{\partial y^3}\right)_{t=y}\right)_{y=\alpha}\mu(\alpha) + \left(\frac{d}{dy}\left(\frac{\partial^2 k(y,t)}{\partial y^2}\right)_{t=y}\right)_{y=\alpha}\mu(\alpha) + \left(\frac{\partial^2 k(y,t)}{\partial y^2}\right)_{t=y}\mu'(\alpha) + \left(\frac{d^2}{dy^2}\left(\frac{\partial k(y,t)}{\partial y}\right)_{t=y}\right)_{y=\alpha}\mu(\alpha) + 2\left(\frac{d}{dy}\left(\frac{\partial k(y,t)}{\partial y}\right)_{t=y}\right)_{y=\alpha}\mu'(\alpha) +$$

$$\left(\left(\frac{\partial k(y,t)}{\partial y}\right)_{t=y}\right)_{y=\alpha}\mu''(\alpha)+\left(\frac{d^3k(y,y)}{dy^3}\right)_{y=\alpha}\mu(\alpha)+3\left(\frac{d^2k(y,y)}{dy^2}\right)_{y=\alpha}\mu'(\alpha)+$$
$$3\left(\frac{dk(y,y)}{dy}\right)_{y=\alpha}\mu''(\alpha)+k(\alpha,\alpha)\mu'''(\alpha) \qquad (40)$$

Now, we attempt to solve the equation using non-polynomial spline functions with a linear and quadratic trend.

A. Using Linear Non-Polynomial Spline Function[7, 12]

We estimate the linear VIEs of the second equation using a linear non-polynomial spline function sort (31), (1). We present an algorithmic solution technique (VIE2NPS1):

The Algorithm (VIE2NPS1)

Step1: Set $\omega=\frac{\beta-\alpha}{m}$, $t_i=t_0+i\omega \ \ni i=0,1,\dots,m\ (t_0=\alpha,t_m=\beta)$ and $\mu_0=\phi(\alpha)$

Step2: Evaluate $\alpha_0,\beta_0,\sigma_0\ and\ \delta_0$ by substituting $(36)-(39)$ in equations $(9)-(12)$.

Step 3: Calculate $L_0(t)$ using step 2 and equation (1).

Step 4: Approximate $\mu_1=l_0(t_1)$.

Step 5: For $(i=1,\dots,m-1)$ do the following steps.

Step 6: Evaluate $\alpha_i,\beta_i,\sigma_i\ and\ \delta_i$ by using equations $(9)-(12)$ and replacing. $\mu(t_i),\mu'(t_i),\mu''(t_i) and\ \mu'''(t_i)$ by $l_i(t_i),l_i'(t_i),l_i''(t_i) and\ l_i'''(t_i)$.

Step 7: Calculate $L_i(t)$ using step 6 and equation (1).

Step 8: Approximate $\mu_{i+1}=L_i(t_{i+1})$.

B. Using Cubic Non-Polynomial Spline Function[7, 12]

Using a cubic non-polynomial spline function, the second kind of linear VIE's (31) approximation solution (13). We demonstrate an algorithmic solution method (VIE2NPS3):

The Algorithm (VIE2NPS3)

Step1: Set $\omega=\frac{\beta-\alpha}{m}$, $t_i=t_0+i\omega \ \ni i=0,1,\dots,m\ (t_0=\alpha,t_m=\beta)$ and $\mu_0=\phi(\alpha)$

Step 2: Evaluate $\alpha_0,\beta_0,\sigma_0,\delta_0,\eta_0\ and\ \gamma_0$ by substituting $(36)-(40)$ in equations $(25)-(30)$.

Step 3: Calculate $C_0(t)$ using step 2 and equation (13).

Step 4: Approximate $\mu_1=C_0(t_1)$.

Step 5: For $(i=1,\dots,m-1)$ do the following steps.

Step 6: Evaluate $\alpha_i,\beta_i,\sigma_i,\delta_i,\eta_i\ and\ \gamma_i$ by using equations $(25)-(30)$ and replacing. $\mu(t_i),\mu'(t_i),\mu''(t_i),\mu'''(t_i)\ ,\mu^4(t_i)\ and\ \mu^5(t_i)$ by $C_i(t_i),C_i'(t_i),C_i''(t_i),C_i'''(t_i),C_i^4(t_i)\ and\ C_i^5(t_i)$.

Step 7: Calculate $C_i(t)$ using step 6 and equation (13).

Step 8: Approximate $\mu_{i+1}=C_i(t_{i+1})$.

4.Examples using Numbers.

Text Example 1: Take into consideration the second kind of VIE from:

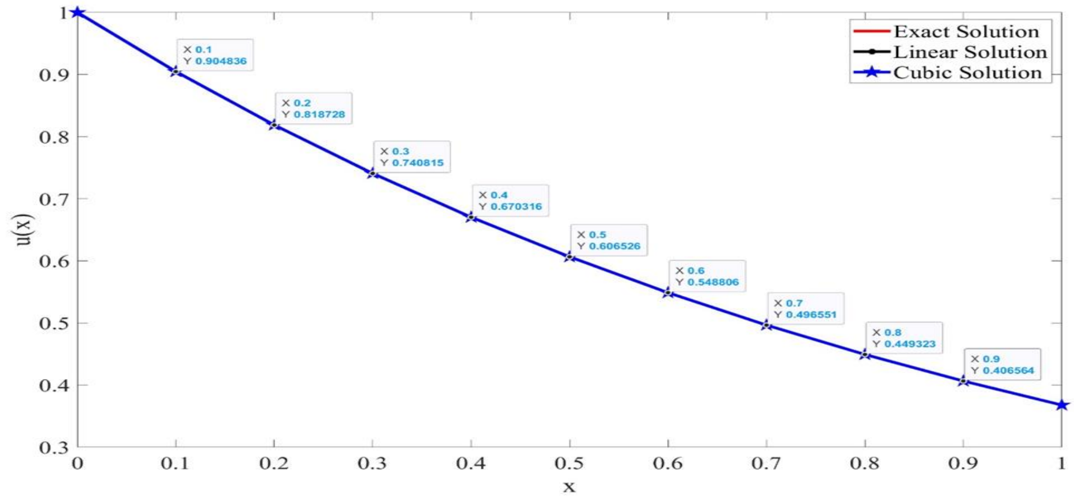
$$\mu(y)=y\cos y+\int_0^y t\,\mu(t)dt \qquad 0\le y\le 1$$

$\mu(y) = e^{-y}$ Has an exact solution. The precise and numerical solutions using linear and Cubic non-polynomial spline functions are compared in Table (1), where $Non\_P_i(y)$ stands for the approximation solution using a non-polynomial spline function, with $\omega = 0.1$.

y	Exact Solution	$Non\_P_i(y)$	
		Linear	Cubic
0	1.000000000000000	1.000000000000000	1.000000000000000
0.1	0.904837418035960	0.904828369707021	0.904835909974853
0.2	0.818730753077982	0.818714378626665	0.818728023980020
0.3	0.740818220681718	0.740795996468462	0.740814516599875
0.4	0.670320046035639	0.670293233770047	0.670315577250228
0.5	0.606530659712633	0.606500333937799	0.606525605311529
0.6	0.548811636094026	0.548778708383702	0.548806148005106
0.7	0.496585303791410	0.496550544036750	0.496579510329994
0.8	0.449328964117222	0.449293019237907	0.449322973104307
0.9	0.406569659740599	0.406533070117780	0.406563561241442
1	0.367879441171442	0.367842655066661	0.367873309898517

Table 2: demonstrates a contrast between the flaw in our approach and alternative methods, where $Error = |exact\ value - numerical\ value|$ and $\|err\|_\infty = max|error|$.

y	Exact Solution	Error Non_$P_i(y)$ in	
		Linear	Cubic
0	1.000000000000000	0.000000000000000	0.000000000000000
0.1	0.904837418035960	0.000009048328939	0.000001508061107
0.2	0.818730753077982	0.000016374451316	0.000002729097962
0.3	0.740818220681718	0.000022224213256	0.000003704081843
0.4	0.670320046035639	0.000026812265593	0.000004468785411
0.5	0.606530659712633	0.000030325774835	0.000005054401104
0.6	0.548811636094026	0.000032927710324	0.000005488088920
0.7	0.496585303791410	0.000034759754660	0.000005793461416
0.8	0.449328964117222	0.000035944879315	0.000005991012915
0.9	0.406569659740599	0.000036589622819	0.000006098499157
1	0.367879441171442	0.000036786104781	0.000006131272925
$\|err\|_\infty = max\|error\|$		0.000036786104781	0.000006131272925



Text Example 2: Think about the VIE of the second category from:

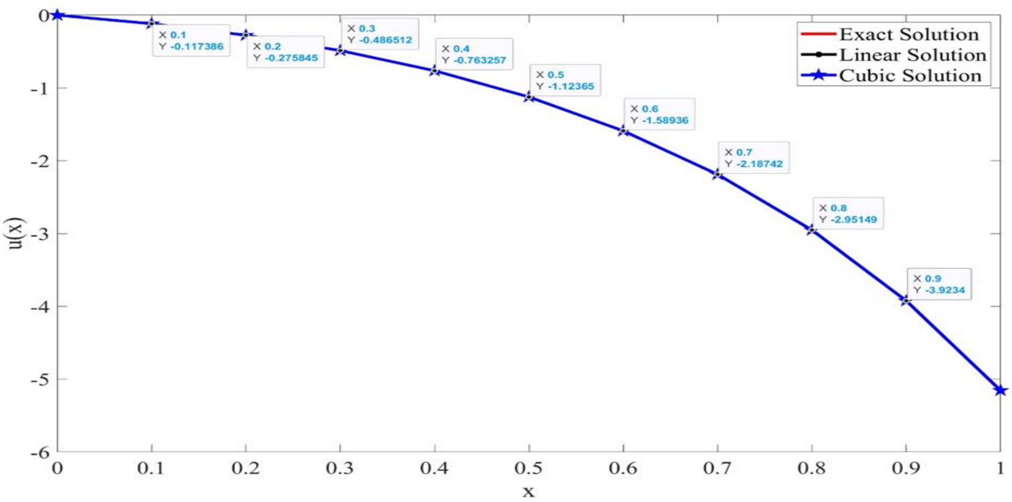
$$\mu(y) = 1 + y + \int_0^y (y - t)\, \mu(t) dt \qquad 0 \le y \le 1$$

$\mu(y) = 3^y(1 - e^y)$, precise solution. Comparing the exact and numerical solutions using linear and Cubic non-polynomial spline functions are shown in Table 3. $P_i(y)$ stands for the approximation using a non-polynomial spline function, and $\omega = 0.1$.

y	Exact Solution	$Non\_P_i(y)$	
		Linear	Cubic
0	0.000000000000000	0.000000000000000	0.000000000000000
0.1	-0.117383698898652	-0.117397323763397	-0.117385969681071
0.2	-0.275808265956336	-0.275844757906987	-0.275814347803731
0.3	-0.486439897193312	-0.486512237295359	-0.486451953465241
0.4	-0.763235980176865	-0.763362130114732	-0.763257004241696
0.5	-1.123618200803270	-1.123822718400300	-1.123652285248810
0.6	-1.589305303715770	-1.589621438502710	-1.589357989551650
0.7	-2.187343074400120	-2.187815478709110	-2.187421802825800
0.8	-2.951377916817380	-2.952066144866990	-2.951492612292560
0.9	-3.923231266352900	-3.924214280648210	-3.923395087514460
1	-5.154845485377140	-5.156227449069750	-5.155075790230570

Table 4: demonstrates a contrast between the flaw in our approach and alternative methods, where $Error\ = |exact\ value - numerical\ value|$ and $\|err\|_{\infty} = max|error|$.

y	Exact Solution	Error Non_$P_i(y)$ in	
		Linear	Cubic
0	0.000000000000000	0.000000000000000	0.000000000000000
0.1	-0.117383698898652	0.000013624864746	0.000002270782419
0.2	-0.275808265956336	0.000036491950652	0.000006081847395
0.3	-0.486439897193312	0.000072340102046	0.000012056271929
0.4	-0.763235980176865	0.000126149937867	0.000021024064831
0.5	-1.123618200803270	0.000204517597036	0.000034084445542
0.6	-1.589305303715770	0.000316134786933	0.000052685835882
0.7	-2.187343074400120	0.000472404308993	0.000078728425684
0.8	-2.951377916817380	0.000688228049607	0.000114695475176
0.9	-3.923231266352900	0.000983014295312	0.000163821161565
1	-5.154845485377140	0.001381963692618	0.000230304853433
$\|err\|_{\infty} = max\|error\|$		0.001381963692618	0.000230304853433



5.References.

1. Diogo, T., et al., *Numerical methods for a Volterra integral equation with non-smooth solutions.* Journal of computational and applied mathematics, 2006. **189**(1-2): p. 412-423.
2. Lima, P. and T. Diogo, *An extrapolation method for a Volterra integral equation with weakly singular kernel.* Applied Numerical Mathematics, 1997. **24**(2-3): p. 131-148.
3. El Danaf, T.S. and F.E.A. Alaal, *Non-polynomial spline method for the solution of the dissipative wave equation.* International Journal of Numerical Methods for Heat & Fluid Flow, 2009.

4. Hossinpour, A. *The Solve of Integral Differential Equation by Non-Polynomial Spline Function and Quadrature Formula*. in *International Conference on Applied Mathematics and Pharmaceutical science Jan*. 2012.
5. Majeed, S.N., *Solution of Second Kind Volterra Integro Equations Using linear Non-Polynomial Spline Function.*
6. Harbi, S.H., M.A. Murad, and S.N. Majeed, *A Solution of Second Kind Volterra Integral Equations Using Third Order Non-Polynomial Spline Function.* Baghdad Science Journal, 2015. **12**(2).
7. H Taqi, A. and B. F Jumaa, *Non-polynomial spline method for the solution of the system of two nonlinear Volterra integral equations.* Kirkuk University Journal-Scientific Studies, 2016. **11**(3): p. 15-25.
8. Biazar, J. and M. Eslami, *Homotopy perturbation and Taylor series for Volterra integral equations of the second kind.* Middle-East Journal of Scientific Research, 2011. **7**(4): p. 604-609.
9. Ghoreishi, F. and M. Hadizadeh, *Numerical computation of the Tau approximation for the Volterra-Hammerstein integral equations.* Numerical Algorithms, 2009. **52**(4): p. 541-559.
10. Rahman, M., *Integral equations and their applications*. 2007: WIT press.
11. Wazwaz, A.-M., *Linear and nonlinear integral equations*. Vol. 639. 2011: Springer.
12. Schumaker, L., *Spline functions: basic theory*. 2007: Cambridge University Press.

Article submitted 10 February 2023. Published as resubmitted by the authors 14 March 2023.