Solving Three Dimensions Volterra Integral Equations (TDVIE) via a Neural Network

Nahdh S. M. Al-Saif¹, Ameen Sh. Ameen¹, Ghaith Fadhil Abbas²

¹Department of Applied Mathematics, College of Science, Anbar University, Rnady, Iraq
²Mathematics Department, College of Education for Pure Sciences, Tikrit University, Tikrit, Iraq

Article Info

Article history:
-Received: 20 / 12 / 2017
-Accepted: 8 / 3 / 2018
-Available online: / / 2018

Keywords: Three - dimensional Volterra Integral Equations, Artificial Neural Network, Linear Transfer Function, Levenberg-Marquardt Algorithm.

Corresponding Author:
Name: Nahdh S. M. Al-Saif
E-mail: nn_ss_m68@yahoo.com
Tel:

1. Introduction

In engineering and science problems, multidi mensional integral and differential equations proved to be an important tool for modeling and solving such problems [1-2].

There are many numerical methods to solv e such equations especially two dimensional integral equations, [3-12]. Three - dimensional integral equations can be solved by using some of these methods. For example, method of Degenerate Kernel Method used to solve three-dimensional nonlinear Volterra integral equations [13], in [14-15] differential transform method was used to solve nonlinear TDVIE, and Shifted Chebyshev Polynomialsmethod used for solving TDVIE[16].

In this study, we describe another numerical method to solve TDVIE by designing a feed forward neural network. Therefore, we consider the following TDVIE:

\[ u(x,y,z) = f(x,y,z) + \int_0^t \int_0^x \int_0^y K(x,y,z,r,s,t) u(r,s,t) drdsdt \] (1)

where \((x,y,z) \in D = [0,X] \times [0,Y] \times [0,Z]\), \(u(x,y,z)\) is the unknown function to be found, \(K(x,y,z,r,s,t)\) and \(f(x,y,z)\) are given functions defined, respectively on \(D\).

2. Artificial Neural Network (ANN)

An (ANN) formed from many artificial neurons(nodes equivalent to neurons of a human brain) that are joint together dependent on particular network -architecture. The goal of the neural network is to transform the inputs into significative outputs.

In another words (ANN) is an interconnected system of nodes by weighted arrows (equivalent to synapses between neurons). The outcome of (ANN) altered by changing of the arrow’s weights. The result of the network for the data that fed to the input layer displayed by the output layer. The input nodes (represent the independent variables)that used for predicting the dependent variable (i.e. the output neurons).

In [17], (ANN) characterized by:
1- "Its pattern of connections between the neurons (called its architecture)".
2- "Methods of determining the weights on the connections (called its training or learning, algorithm)".
3- "Its activation–function".

2.1 Neural Network Structure:

The structure or topology of an artificial neuron network means the way of regulation of neuronal computational cell in the network. Particularly how the information transmitted though the network Figure 1 and how the nodes are connected. The architecture can be classified in terms of three aspects
(Number of levels or layers, Connection pattern and Information flow).

\[ J = \begin{bmatrix}
\frac{\partial e_{11}}{\partial w_1} & \frac{\partial e_{11}}{\partial w_2} & \ldots & \frac{\partial e_{11}}{\partial w_N} \\
\frac{\partial e_{12}}{\partial w_1} & \frac{\partial e_{12}}{\partial w_2} & \ldots & \frac{\partial e_{12}}{\partial w_N} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial e_{1P}}{\partial w_1} & \frac{\partial e_{1P}}{\partial w_2} & \ldots & \frac{\partial e_{1P}}{\partial w_N}
\end{bmatrix}
\]

where \( I \) is the identity unit matrix, \( \mu \) is the learning parameter and \( \mathbf{j} \) is the Jacobian matrix of order \( M \times N \) at each iteration step.

2.2 Linear Transfer Function (purelin)

The output of a linear transfer function is equal to its input:

\[ a = n \]

as illustrated in figure 2.

2.3 Levenberg Marquardt Algorithm Training

Training neural network is basically a nonlinear least squares problem, so can be solved by using a several nonlinear least squares algorithms. One of them is (LMA). We consider (LMA) as a combination of the Gauss – Newton method and steepest descent. For (LMA), the performance index to be optimized as:

\[ F(w) = \sum_{p=1}^{P} \left[ \sum_{k=1}^{K} (d_{kp} - o_{kp})^2 \right] \]

Where \( \mathbf{w} = [w_1, w_2, \ldots, w_N]^T \) consists of all weights of the network, \( o_{kp} \) is the actual value of the \( k \)th output pattern and \( d_{kp} \) is the desired value of the \( k \)th output pattern, \( N \) is the number of the weights, \( P \) is the number of output units, and \( K \) is the number of the network output.

Equation (2) can be written as follows:

\[ F(w) = E^T E \]

Where \( E = [e_{11}, \ldots, e_{k1}, e_{k2}, \ldots, e_{kp}]^T \), \( e_{kp} \) is the training error at output \( k \) when applying pattern \( p \) and define as \( e_{kp} = d_{kp} - o_{kp} \)

\( p = 1, \ldots, P, E \) is the cumulative error vector (for all pattern). From equation (3) the weights are calculated using the following equation

\[ w_{N+1} = w_n - (J^T J + \mu I)^{-1} J^T E \]

and the Jacobian matrix is defined as

\[ J = \begin{bmatrix}
\int_0^t \int_0^t K(x, y, z, r, s, t) u(r, s, t) \, dr \, ds \, dt \\
\int_0^t \int_0^t K(x, y, z, r, s, t) u(r, s, t) \, dr \, ds \, dt \\
\vdots \\
\int_0^t \int_0^t K(x, y, z, r, s, t) u(r, s, t) \, dr \, ds \, dt
\end{bmatrix} 
\]

where \( I \) is the identity unit matrix, \( \mu \) is the learning parameter and \( \mathbf{j} \) is the Jacobian matrix of order \( M \times N \) at each iteration step. At each iteration the \( \mu \) parameter automatically adjusted in order to secure convergence, the calculation of the Jacobian matrix \( J \) and the inverse of \( J^T J \) square matrix of order \( N \times N \).

3. Description of Method

In the current section, we will demonstrate condensing our approach to be used the approximation solution of the TDVIE.

\[ u(x, y, z) = f(x, y, z) + \int_0^t \int_0^t K(x, y, z, r, s, t) u(r, s, t) \, dr \, ds \, dt \]

where \( (x, y, z) \in D = \text{three dimension} = [0, X] \times [0, Y] \times [0, Z] \), and \( u(x, y, z) \) is unknown function to be found. If \( u_t(x, y, z, p) \) is a trial solution with adjustable parameters \( p \), the discretized from:

\[ \text{Min} \sum_{x,y,z} f(x, y, z, p) + \int_0^t \int_0^t K(x, y, z, r, s, t) u(r, s, t, p) \, dr \, ds \, dt \]

where \( x, y, z \) is variables such that \((x, y, z) \in D = \text{three dimension} = [0, X] \times [0, Y] \times [0, Z] \).

In our proposed approach, the trial solution \( u_t \) corresponds FFNN and the parameters, employ biases and weights of the neural-architecture, the form for the trial-function \( u_t(x, y, z) \) is \( u_t(x, y, z, p) = G(x, y, z, N(x, y, z, p)) \) where \( N(x, y, z, p) \) is a single output FFNN with parameter \( p \) and \( n \) input unit feed with the input vectors \( x, y, z \) The term \( G \) is constructed, since \( u(x, y, z) \) satisfy them. This term can be formed by using a (ANN) whose biases and weights are adjusted in order to deal with the minimization problem. The Minimized have form

\[ E(p) = \left[ y_t - \sum_{i=1}^{n} (f(x_t, y, z, i)) + \int_0^t \int_0^t K(x, y, z, r, s, t) \int_0^t \int_0^t u_t(r, s, t) \, dr \, ds \, dt \right]^2 \]
4. Applications to three dimensions volterra integral equation

To demonstrate the efficiency of the proposed method (ANNM), we consider the following examples and to test the accuracy of solution using mean square error MSE. All programming written in the MatLab to computed the results.

**Example 4.1**

Consider the (TDVIE)

\[ u(x, y, z) = f(x, y, z) - \int_0^x \int_0^y \int_0^z u(r, s, t) \, dr \, ds \, dt \]

where \((x, y, z) \in [0,1] \times [0,1] \times [0,1]\)

And \( f(x, y, z) = x + y + z + \frac{x^2yz + xyz^2 + x^2y^2z^2}{2} \)

has analytic function

\[ u(x, y, z) = x + y + z \]

by applying suggested method Table (1) shows the exact, neural result, error, and men square error. Table (2) gives the weight, bias, Epoch, and time performance of the network.

| Table (1): exact, neural and Accuracy of solution example (4.1) |
|-----------------|-----------------|-----------------|
| \(X\) | \(Y\) | \(Z\) | \(\text{Exact } u_a(x, y, z)\) | \(\text{Trainlm } u_a(x, y, z)\) | \(\text{Error } = |u_a - u_a|\) |
| 0.1 | 0.1 | 0.1 | 3.0000e-001 | 2.9990e-001 | 9.6546e-005 |
| 0.01 | 0.1 | 0.1 | 2.1000e-001 | 2.0999e-001 | 7.4557e-006 |
| 0.01 | 0.01 | 0.1 | 1.2000e-001 | 1.2000e-001 | 2.2235e-007 |
| 0.01 | 0.01 | 0.01 | 3.0000e-002 | 3.0000e-002 | 9.6546e-009 |
| 0.001 | 0.01 | 0.01 | 2.1000e-002 | 2.1000e-002 | 7.4557e-010 |
| 0.001 | 0.001 | 0.01 | 1.2000e-002 | 1.2000e-002 | 2.2235e-011 |
| 0.001 | 0.001 | 0.001 | 3.0000e-003 | 3.0000e-003 | 9.6546e-013 |

**Table (2): weight, bias, Epoch, time and performance of the network**

| Net. | train | net |Epoch| time | performance |
|------|------|-----|-----|------|------------|
| LW[1,1] | 0.0477 | 0.6665 | 0.8819 | 0.8555 | 0.00 |
| LW[1,2] | 0.9831 | 0.1781 | 0.6692 | 0.6448 | 0.00 |
| B[1] | 0.3015 | 0.1280 | 0.1904 | 0.3763 | 0.00 |
| Epoch | time | performance |
| 9 | 0.00.02 | 1.03e-33 |

**Example 4.2**

Consider the (TDVIE).

\[ u(x, y, z) = f(x, y, z) - 24x^2y \int_0^x \int_0^y \int_0^z u(r, s, t) \, dr \, ds \, dt \]

where \((x, y, z) \in [0,1] \times [0,1] \times [0,1]\).

And \( f(x, y, z) = x^2y + xyz + yz^2 \)

which has an analytic solution

\[ u(x, y, z) = x^2y + xyz + yz^2 \]

by applying suggested method Table (3) shows the exact, neural result, error, and men square error. Table (4) gives the weight, bias, Epoch, time and performance of the network.

| Table (3): exact, neural and Accuracy of solution example (4.2) |
|-----------------|-----------------|-----------------|
| \(X\) | \(Y\) | \(Z\) | \(\text{Exact } u_a(x, y, z)\) | \(\text{Trainlm } u_a(x, y, z)\) | \(\text{Error } = |u_a - u_a|\) |
| 0.1 | 0.1 | 0.1 | 3.0000e-003 | 2.9601e-003 | 3.9873e-005 |
| 0.01 | 0.1 | 0.1 | 1.1100e-003 | 1.1100e-003 | 3.8101e-008 |
| 0.01 | 0.01 | 0.1 | 1.1100e-004 | 1.1100e-004 | 3.6387e-010 |
| 0.01 | 0.01 | 0.01 | 3.0000e-006 | 3.0000e-006 | 3.4274e-011 |
| 0.001 | 0.01 | 0.01 | 1.1100e-006 | 1.1100e-006 | 3.4096e-014 |
| 0.001 | 0.001 | 0.01 | 1.1100e-007 | 1.1100e-007 | 3.3924e-016 |
| 0.001 | 0.001 | 0.001 | 3.0000e-009 | 3.0000e-009 | 3.3713e-017 |

**Table (4): weight, bias, Epoch, time and performance of the network**

| Net. | train | net |Epoch| time | performance |
|------|------|-----|-----|------|------------|
| LW[1,1] | 0.4401 | 0.9577 | 0.2548 | 0.0067 | 0.4609 |
| LW[1,2] | 0.5271 | 0.2407 | 0.2240 | 0.6022 | 0.7702 |
| B[1] | 0.4547 | 0.6761 | 0.6678 | 0.3868 | 0.3225 |
| Epoch | time | performance |
| 10 | 0.00.01 | 9.67e-10 |
Example 4.3:
Let the following (TDVIE)\([15]\):
\[ u(x, y, z) = f(x, y, z) + \int_{0}^{s} \int_{0}^{t} u(r, s, t) \, dr \, dt \]
where \((x, y, z) \in [0,1] \times [0,1] \times [0,1] \),
and \( f(x, y, z) = e^{x+y} + e^{x+z} + e^{y+z} - e^x - e^y - e^z + 1 \).

has an analytic solution
\[ u(x, y, z) = e^{x+y+z} \]
by applying the suggested method Table (5) shows the exact, neural result, error, and mean square error. Table (6) gives the weight, bias, Epoch, and time performance of the network.

### Table (5): Exact, neural and Accuracy of solution example (4.3)

| X_1 | Y | Z | Exact u(x,y,z) | Trainmu_(x,y,z) | Error = \|u_T - u_n\| |
|-----|---|---|----------------|-----------------|------------------|
| 0.1 | 0.1 | 0.1 | 1.3499e+000   | 1.3507e+000    | 1.8419e-004     |
| 0.01 | 0.1 | 0.1 | 1.2337e+000   | 1.2338e+000    | 8.5500e-005     |
| 0.01 | 0.01 | 0.1 | 1.1275e+000   | 1.1275e+000    | 8.3164e-006     |
| 0.01 | 0.01 | 0.01 | 1.0305e+000   | 1.0305e+000    | 1.6259e-007     |
| 0.001 | 0.01 | 0.01 | 1.0212e+000   | 1.0212e+000    | 7.8907e-008     |
| 0.001 | 0.001 | 0.01 | 1.0121e+000   | 1.0121e+000    | 7.8634e-009     |
| 0.001 | 0.001 | 0.001 | 1.0030e+000   | 1.0030e+000    | 1.6224e-010     |

### Table (6): weight, bias, Epoch, time and performance of the network

| weight and bias | Epoch, time and performance |
|----------------|-----------------------------|
| Net_IW[1,1]   | Net_LW[1,2]   | Net_B[1]   | Epoch | Time | performance  |
| 0.9138 | 0.1704 | 0.4022 | 0.3508 | 0.2992 |
| 0.7067 | 0.2578 | 0.6207 | 0.6855 | 0.4526 |
| 0.5578 | 0.3968 | 0.1544 | 0.2941 | 0.4226 |
| 0.3134 | 0.0740 | 0.3813 | 0.5306 | 0.3596 |
| 0.1662 | 0.6841 | 0.1611 | 0.8324 | 0.5583 |
| 0.6225 | 0.4024 | 0.7581 | 0.5975 | 0.7425 |
| 0.9879 | 0.9828 | 0.8711 | 0.3353 | 0.4243 |

To study accurate and efficient, we compared the suggested method (ANNM) [14], depend on absolute error.

### Table (7): Absolute error of (SCPM), (RDTM) and (ANNM)

| X_1 | Y | Z | Exact u(x,y,z) | SCP Method | RDT Method | ANN Method |
|-----|---|---|----------------|------------|------------|------------|
| 0.1 | 0.1 | 0.1 | 1.3499e+000   | 3.3089e-002 | 2.0875e-004 | 1.8419e-004 |
| 0.01 | 0.1 | 0.1 | 1.2337e+000   | 2.1664e-002 | 1.9079e-004 | 8.5500e-005 |
| 0.01 | 0.01 | 0.1 | 1.1275e+000   | 1.1933e-002 | 1.7437e-004 | 8.3164e-006 |
| 0.01 | 0.01 | 0.01 | 1.0305e+000   | 3.6683e-003 | 1.7045e-007 | 1.6259e-007 |
| 0.01 | 0.01 | 0.01 | 1.0121e+000   | 2.5502e-003 | 1.6893e-007 | 7.8907e-008 |
| 0.001 | 0.001 | 0.01 | 1.0030e+000   | 1.4508e-003 | 1.6741e-007 | 7.8634e-009 |
| 0.001 | 0.001 | 0.001 | 1.0030e+000   | 3.6986e-004 | 1.6704e-0010| 1.6224e-010 |

To study accurate and efficient, we compared the suggested method (ANNM) [14], depend on absolute error.

### Conclusion
Analytic solution of (TDVIE) are usually difficult, many cases require numerical solutions. In this paper, we introduced a new numerical method to solve TDVIE. The results indicate minimal mean square error and were compared with the solution for

### References
1. Atkinson, K.E. (1997). The Numerical Solution of Integral Equations of the Second Kind. Cambridge University Press.
2. Chari M. V. K. and Saloni S. J. (2000); Numerical Methods in Electromagnetism. Academic Press.
3. Borhanifar A. and Sadri K. (2014); Numerical solution for system of two-dimensional integral equation by using Jacobi operational collocation method. Sohag J. Math. 1, No. 1, 15-26.
4. Hendi F. A. and Bakodah H.O. (2011); Discrete Adomian Decomposition Solution of Nonlinear Mixed Integral Equation. Journal of American Science,7 (12),1081-1084
5. Mirzaee F. and Hadadiyan E. (2012); Approximate solutions for mixed nonlinear Volterra-Fredholm type integral equations via modified block-pulse functions. Journal of the Association of Arab Universities for Basic and Applied Science12, 65-73.
6. Ziyaee F., Tari A. (2014); Regularization Method for Two- dimensional Fredholm Integral Equations of the First Kind. International Journal of Nonlinear Science. Vol.18 No.3, 189-194.
7. Saberi Nadjadi J., Reza NavidSamadi O. and Tohid E. I (2011); Numerical Solution of two- dimensional Volterra Integral Equations by Spectral Galerkin Method. Journal of Applied Mathematics and Bioinformatics. vol. 1, no. 2, 159-174.
8. Abdou M.A., Badr A.A. and Soliman M.B. (2011); On a method for solving a two- dimensional non-linear integral equation of the second kind. J. Comput. Appl. Math., 235, 35893598
9. Saeed R. K. and Berdawood K. A. (2016); Solving Two- dimensional Linear Volterra- Fredholm Integral Equations of Second Kind by Using Successive Approximation Method and method of Successive Substitutions. ZANCO Journal of Pure and Applied Sciences. 28 (2); 35-46.
10. Sadegh Behzadi Sh. (2012); The Use of Iterative Methods to Solve Two - Dimensional Nonlinear Volterra – FredholmIntegro - Differential Equations. Communication in Numerical Analysis. 1-20.
11. AVAZZADEH Z. and HEYDARI M. (2012); Chebyshev polynomials for solving two-dimensional linear and nonlinear integral equations of the second kind. Computational and applied Mathematics. Volume 31, No.1, 127-142.
12. Avazzadeh Z., Heydarib M. and Loghmania G. B. (2011); A Comparison between Solving Two-Dimensional Integral Equations by the Traditional Collocation Method and Radial Basis Function. Applied Mathematics Sciences, Vol. 5, no. 23, 1145-1152.
13. Basseem(2015); Degenerate Kernel Method for Three Dimension Nonlinear Integral Equations of the Second Kind. University Journal of Integral Equations 3,61-66.
14. Ziqan A., Armiti S. and I. Suwan(2016); Solving three- dimensional Volterra integral equation by the reduced differential transform method. International journal of Applied Mathematical Research. 5 (2) 103-106.
15. Bakhshi M. and Asghari-Larimi M. (2012); Three- dimensional differential transform method for solving three-dimensional Volterra integral equations. The Journal of Mathematics and Computer Science Vol. 4 No. 2 246-256.
16. Mohamed D. Sh.: Shifted Chebyshev polynomials for Solving Three- Dimensional Volterra Integral Equations of the second kind. https://www.researchgate.net/publication /308692522
17. Hristev R. M. . (1998). The ANN Book.GNU public license, Edition 1.

حل معادلة فولتير اكتمالية ذات البعد الثلاثي

ناهض سليم محمد1, امين شامان امين1, غيث فاضل عباس2

1 قسم الرياضيات التطبيقية , كلية العلوم , جامعة الانبار , رمادي , العراق
2 قسم الرياضيات, كلية التربية للعلوم الصرفة , جامعة تكريت , تكريت , العراق

الملخص

في الآونة الأخيرة ازداد اهتمام الباحثون في إيجاد طرق عددية جديدة لحل معادلة فولتير التكاملية ذات البعد الثلاثي (ANN). إن الهدف الرئيسي لهذا البحث هو تقديم طريقة عددية جديدة لحل هذا النوع من المعادلات باستخدام الشبكات العصبية الصناعية (ANN)، حيث تم تصميم شبكة عصبية ذات طبقة إذابة (FFNN) سريعة. يتألف هذا التصميم من طبقة متعددة حيث يحتوي على طبقة واحدة خفيفة تحتوي على سهولة ودقة تطبيقات الدالة الخطية (purelin). تم تدريب الشبكة باستخدام خوارزمية (traianlm) Marquardt.

وبيان دقة وكفاءة الطريقة المقترحة تم مقارنة نتائج الامثلة التوضيحية مع الحلول المصبوطة لهذه الامثلة اضافة إلى مقارنتها مع طريقة متعددة من خلال مقارنة بين مبرحتي التحليلي (RDTM) وطريقة التهويل التفاضلي المنخفض (SCPM) وحل معادلة فولتير اكتمالية ذات البعد الثلاثي. 

صغير جدا.