ATINER's Conference Paper Proceedings Series
MAT2020-0189
Athens, 15 July 2020

Approximate Solution for Integro Partial Differential Equation via Neural Network

Nahdh S. M. Al-Saif
Mahmood A. Shamran
Ronak Bagelany
Saad N. Al-Azzawi

Athens Institute for Education and Research
8 Valaoritou Street, Kolonaki, 10683 Athens, Greece

ATINER's Conference Paper Proceedings Series
MAT2020-0189
Athens, 15 July 2020
ISSN: 2529-167X

Nahdh S. M. Al-Saif, Lecturer, Department of Applied Mathematics, College of Science, University of Anbar, Iraq
Mahmood A. Shamran, Lecturer, Department of Mathematics, College of Science for Women, University of Baghdad, Iraq
Ronak Bagelany, Assistant Lecturer, Department of Mathematics, College of Science, University of Kirkuk, Iraq
Saad N. Al-Azzawi, Professor, Department of Mathematics, College of Science for Women, University of Baghdad, Iraq

Approximate Solution for Integro Partial Differential Equation via Neural Network

ABSTRACT

The aim of this paper is to solve integro partial differential equations (IPDE$_s$) using artificial neural network through designing multi-layer feed forward Neural Network. A multi-layers design in the proposed method consists of a hidden layer having five hidden units with tanh (tansig) Transfer function used as each unit and one output unit with linear (purelin) transfer function in this design using Levenberg-Marquardt algorithm training. Moreover, examples on partial integro-differential equations carried out to demonstrate the efficiency and accuracy of the introduced technique.

Keywords: artificial neural network, integro partial differential equation, linear transfer function, Levenberg-Marquardt algorithm

Introduction

Finding the exact solutions of functional equations has attracted consideration of mathematician's interest in current years. Many issues in theoretical physics and different sciences lead to (IPDE). The solutions of this form of equations are regularly very complicated. For this cause in many cases, it is required to attain the approximate solutions. For instance, some integro-differential equations solution by techniques such as Jacobi polynomials [1], variation iteration [2], homotopy perturbation [3], Taylor [4], Legendre [5], Taylor-Lucas [6, 7], Laguerre Polynomial [8,9], Dickson polynomial [10-11], Chelyshkov collection [12], Bessel method [13], Bernoulli [14], and also, B-Splines [15], backward substitution [16]. Technique of this approach uses an answer in the shape of an array that includes the fee of the answer at a chosen group of point, different use groundwork characteristic to represent the answer in analytic shape and seriously change the original problem commonly to a system of algebraic equation.

These days there is a better approach of computing named Artificial Intelligence, which through distinctive strategies, is competent of managing the imprecisions and instabilities that show up when attempting to unravel issues related to the actual world presenting correct answer and are of simple implement-tation. One of these strategies is recognized as Artificial Neural Networks (ANN).

The goal is to learn about a new numerical approach primarily based on a neural network to resolve (IPDE) of the shape

$$\sum_{i=1}^{2}\frac{\partial^i u}{dx^i}+\sum_{i=1}^{2}\frac{\partial^i u}{dy^i}=g(x,y)-\int_0^x\int_0^y k(x,y)u(s,t))dsdt \qquad (1)$$
$$u(x,0)=f_1(x),\ u_x(x,0)=f_2(x)\ ,u(0,y)=g_1(y),u_y(0,y)=g_2(y)$$

where, $u(x,y)$ is the unknown characteristic to be found, $k(x,y)$ is kernel and the characteristic $g(x,y)$ given smooth function.

This paper is prepared as follows: the subsequent part defines and describes the shape of neural network; in part “Levenberg-Marquardt-Algorithm-Training (trainlm) [18]”, the Levenberg Algorithm derivation; description of important end result in part “Main Result”; in area “Illustrative Example” is reported the numerical.

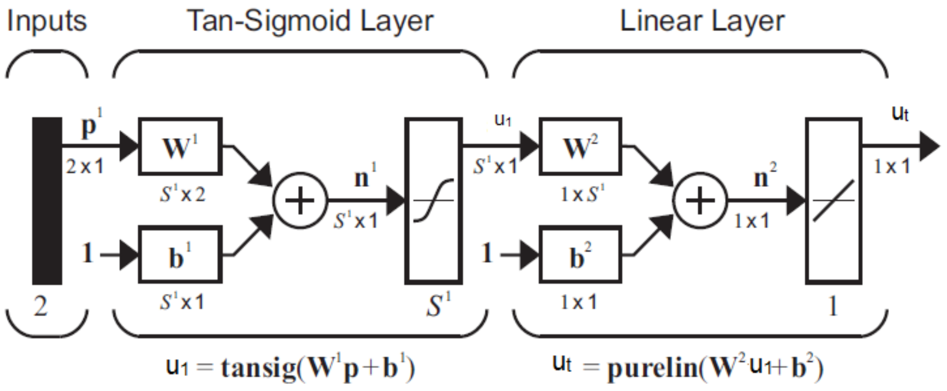
Artificial Neural Network (Neuron Model)

To begin with endeavors of building counterfeit neural systems (ANN) were spurred by the want to form models for common brains. The fundamental building piece of an (artificial) neural organization (ANN) is the neuron. A neuron may be a preparing unit which has a few (more often no more than one) inputs and as it were one yield. The shape or topology of neural network capability is the way of law of neuronal computational cell in the network. That is, how the nodes are linked and how the records are transmitted via the network. The structure can be classified in phrases of three elements (Number of stages or layers, Connection sample and Information flow).

In [17], (ANN) was characterized by:

1- Strategy of associations between the neurons (named its design).
2- Activation–function.
3- Methods of determining the weights (named its training or algorithm).

Figure 1. *Neural Network Structure*



Levenberg-Marquardt-Algorithm-Training (trainlm) [18]

Training neural network is basically a nonlinear squares problem; so it can be the solution by using a several nonlinear least squares algorithms. One of them is Levenberg-Marquardt-Algorithm (LMA). We can consider (LMA) a mixture of the Gauss–Newton technique and steepest descent.

In order to optimize LMA performance index, we assume that F(w) is a sum of squares function and we define it as,

$$F(\omega) = \sum_{p=1}^{p} \left[\sum_{k=1}^{k} \left(d_{kp} - o_{kp} \right)^2 \right] \qquad (2)$$

where $\omega = [\omega_1 w_2 \cdots \omega_N]^T$ includes network's weight, d_{kp} the desired value of the k^{th} output and the p^{th} design, o_{kp} the exact value of the k^{th} output and the p^{th} design, K the value of the network output, P the number of design and N the Num. of the weights, respectively. Equation (2) may be has form:

$$\boldsymbol{F(w) = E^T E} \qquad (3)$$

where $E = \left[e_{11} \ldots e_{k1}\, e_{12} \ldots e_{k2} \ldots e_{1p} \ldots e_{kp} \right]^T$, e_{kp} is the learning error at output k when applied design p and denoted by $e_{kp} = d_{kp} - o_{kp}$

$p = 1, \cdots, P$. E is the total error vector (for all design). From equation (3) the weights are calculated utilizing the taking after equation

$$w_{t+1} = w_t - (J_t^T J_t + \mu_t I)^{-1} J_t^T E_t \qquad (4)$$

where I identification unit matrix, μ is learning parameter and J Jacobin $(\partial e/\partial w)$ of m out- put error of the neural network with appreciate to n weights, respectively

Main Result

We illustrate how our method can be used to the approximate answer of the (IPDE) of the shape

$$\sum_{i=1}^{2} \frac{\partial^i u}{dx^i} + \sum_{i=1}^{2} \frac{\partial^i u}{dy^i} =$$
$$g(x,y) - \int_0^x \int_0^y k(x,y)u(s,t)\big)dsdt$$

With initial conditions

$$(x,0) = f_1(x),\ u_x(x,0) = f_2(x)\ , u(0,y) = g_1(y), u_y(0,y) = g_2(y)$$

x, y∊ I∊ R^2,define the domain and $u(x,y)$ is the answer to be computed. If $u_t\ (x,y,p)$ is trial answer with adjustable parameters p the problem is converted to discretize from:

$$Min \sum_{x_i, y_i \in I} g\ (x_i, y_i) - \int_0^{x_i} \int_0^{y_i} K(x_i, y_i)\ u_t(t,s,p)dtds \qquad (6)$$

Within the our proposed approach, the trial answer u_t employs a FFNN and the parameters p compare to the bias and weight of the neural structure, have shape

$$u_{ti}(x_i, y_i, p) = purline(w^2 * tansig\ (w^1 * ([x_i\ ;\ y_i]) + b^1) + b^2$$

where x_i, y_i input variable and i=1,2,…,N , N number of input , w^1, w^2 weight of hidden , output layer corresponding , b^1, b^2 biases of hidden and output layer. $u_t(x,y,p)$ Is the output of a FFN with two enters unite for x, y and weights W. the error volume to be minimizes given by:

$$E(p) = \left\{ \sum_{i=1}^{2} \frac{\partial^i u_t}{dx^i} + \sum_{i=1}^{2} \frac{\partial^i u_t}{dy^i} - g(x_i, y_i) + \int_0^{x_i} \int_0^{y_i} k\big(x_i, y_i)u_t(s,t)\big)\, ds dt \right\}^2$$

Illustrative Example

To demonstrate the effectiveness of the suggested method (ANNM), we think about the following examples and we take a look at the accuracy of solutions using mean square error MSE. All programing was written in the Mat Lab to compute the results.

Example 1

Consider the integro partial differential equation

$$u_{xx} + u_{yy} + u_x + u_y = f(x,y) + \int_0^x \int_0^y u(t,s) dt\, ds$$

where $f(x,y) = 2x - 2y + 1 - \frac{x^3 y}{3} + \frac{xy^3}{3} - \frac{x^2 y}{2} + 4xy$

and

$$u(x,0) = x^2 + x - 4\ ,\ \ u_x(x,0) = 2x + 1,\ \ u(0,y) = -y^2 - 4,\ \ u_y(0,y) = -2y$$

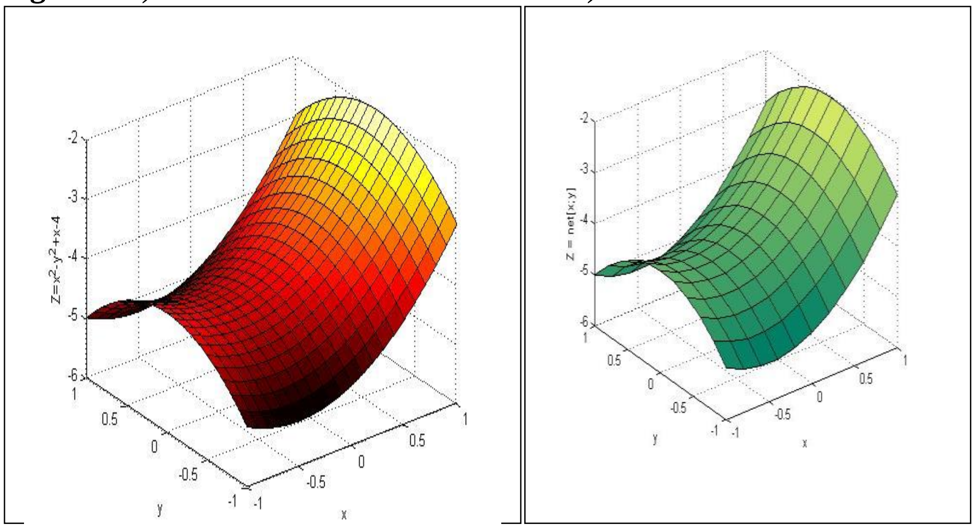
Its solution is $u(x,y) = x^2 - y^2 + x - 4$

by applying suggested method Table 1 shows the exact, neural result, error, and men square error.

Table 1. *Exact, Neural and Accuracy of Solution Example (1)*

X↓	Y	Exact $u_a(x,y)$	Trainlm $u_t(x,y)$	Error = $\lvert u_a - u_t \rvert$
-1	-1	-5.0000000000000e+000	-4.9999990633737e+000	9.3662627165259e-007
-0.8	-0.8	-4.8000000000000e+000	-4.7999967917592e+000	3.2082407797063e-006
-0.6	-0.6	-4.6000000000000e+000	-4.5999985965971e+000	1.4034029245380e-006
-0.4	-0.4	-4.4000000000000e+000	-4.4000005422921e+000	5.4229205126433e-007
-0.2	-0.2	-4.2000000000000e+000	-4.2000012502414e+000	1.2502413957449e-006
0.0	0.0	-4.0000000000000e+000	-4.0000008544533e+000	8.5445327346889e-007
0.2	0.2	-3.8000000000000e+000	-3.8000002146470e+000	2.1464702992802e-007
0.4	0.4	-3.6000000000000e+000	-3.6000002454182e+000	2.4541821730395e-007
0.6	0.6	-3.4000000000000e+000	-3.4000012143223e+000	1.2143223222516e-006
0.8	0.8	-3.2000000000000e+000	-3.2000018728234e+000	1.8728234487675e-006
1	1	-3.0000000000000e+000	-2.9999983735407e+000	1.6264592845872e-006
MSE			**1.624801901002671e-011**	

Figure 4. a) *Exact Solution* b) *Neural Solution*



Example 2

Consider the (IPED)

$$u_{xx} + u_{yy} + u_x + u_y = f(x,y) + \int_0^x \int_0^y u(t,s)dt\,ds$$

where $f(x,y) = 3x^2 + 4x - 4y - 3y^2 - \frac{x^4 y}{4} + \frac{x^3 y}{3} - \frac{x y^3}{3} + \frac{x y^4}{4}$

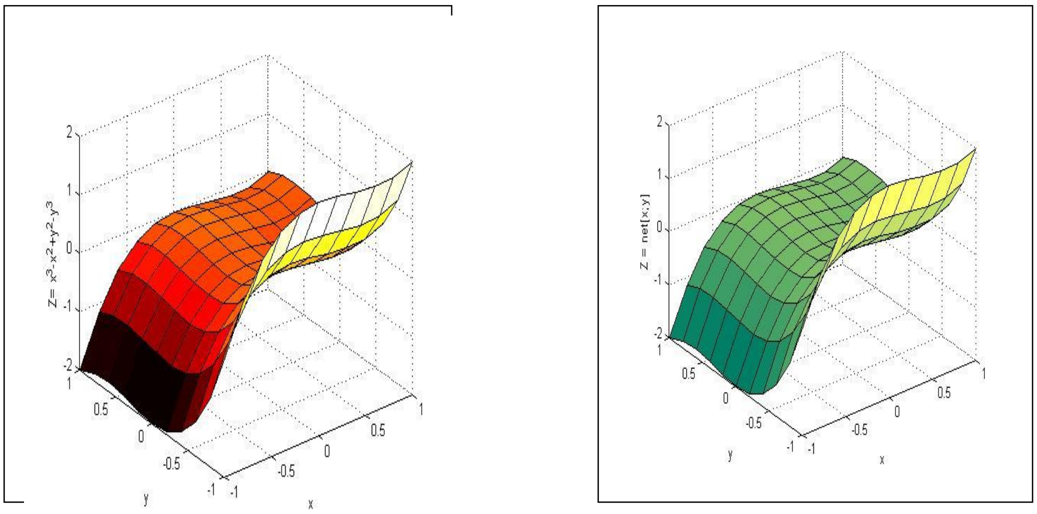
$u(x,0) = x^3 - x^2$, $u(0,y) = -y^3 + y^2$, $u_x(x,0) = 3x^2 - 2x$, $u_y(0,y) = 2y - 3y^2$

whose solution is $u(x,y) = x^4 - x^2 + y^2 - y^3$
by applying suggested method Table 2 shows the exact, neural result, error, and men square error.

Table 2. *Exact, Neural and Accuracy of Solution Example (2)*

X↓	Y	Exact $u_a(x,y)$	Trainlm $u_t(x,y)$	Error = $\lvert u_a - u_t \rvert$
-1	-1	0.0000000000000e+000	-6.7996736630160e-006	6.7996736630160e-006
-0.8	-0.8	0.0000000000000e+000	1.3898209798580e-006	1.3898209798597e-006
-0.6	-0.6	0.0000000000000e+000	4.7626288761649e-006	4.7626288761649e-006
-0.4	-0.4	0.0000000000000e+000	1.5584520696166e-006	1.5584520696027e-006
-0.2	-0.2	0.0000000000000e+000	-1.3898209798580e-006	1.3898209798597e-006
0.0	0.0	0.0000000000000e+000	-1.3825466493866e-006	1.3825466493866e-006
0.2	-1	1.9680000000000e+000	1.9680292288707e+000	2.9228870712217e-005
0.4	-0.8	1.0560000000000e+000	1.0560226797822e+000	2.2679782238022e-005
0.6	-0.6	4.3200000000000e-001	4.3197017676292e-001	2.9823237075899e-005
0.8	-0.4	9.6000000000000e-002	9.6015058994091e-002	1.5058994091030e-005
1	-0.2	4.8000000000000e-002	4.8020338870181e-002	2.0338870181005e-005
MSE			**5.652305785123967e-011**	

Figure 5. a) *Exact Solution* b) *Neural Solution*



Consider the integro partial differential equation

$$u_{xx} + u_{yy} + u_x + u_y = f(x,y) + \int_0^x \int_0^y u(t,s)dt\,ds$$

where $f(x,y) = -4x^3 - 12x^2 - 10x + 20xy - 8y^3 - 24y^2 + 12y - 18 - \frac{10x^3y^2}{6} + \frac{5x^3y}{3} + \frac{4xy^3}{3} + \frac{x^5y}{5} + \frac{2xy^5}{5}$

$$u(x,0) = -5x^2 - x^4\ , \qquad u(0,y) = -4y^2 - 2y^4\ ,$$

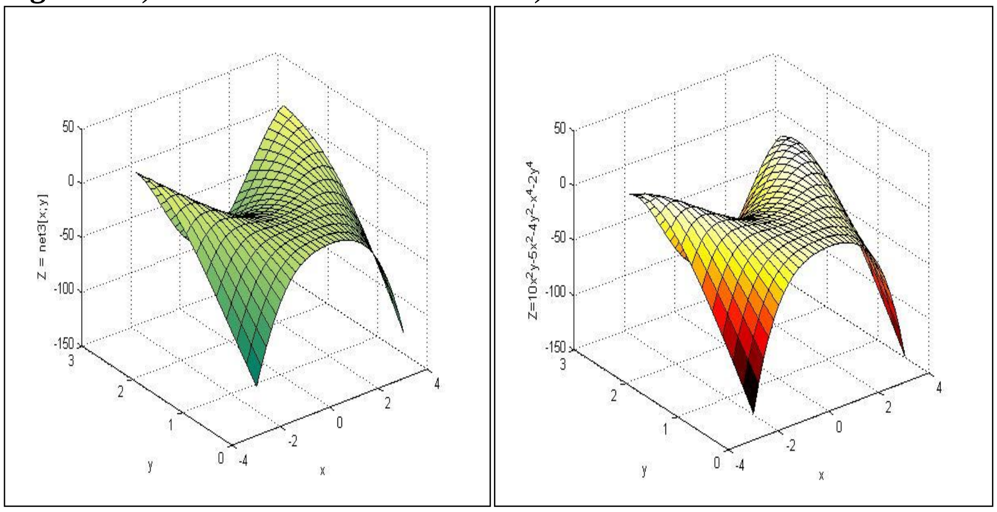
$$u_x(x,0) = -10x - 4x^3\ , \qquad u_y(0,y) = -8y - 8y^3$$

whose solution is $u(x,y) = 10x^2y - 5x^2 - 4y^2 - x^4 - 2y^4$
by applying suggested method Table 3 shows the exact, neural result, error, and men square error.

Table 3. *Exact, Neural and Accuracy of Solution Example (3)*

X↓	Y	Exact $u_a(x,y)$	Trainlm $u_t(x,y)$	Error = $\lvert u_a - u_t \rvert$
-1	-1	-2.2000000000000e+001	-2.1977824946347e+001	2.2175053652632e-002
-0.8	-0.8	-1.2108800000000e+001	-1.2112684164983e+001	3.8841649829635e-003
-0.6	-0.6	-5.7888000000000e+000	-5.7879891353203e+000	8.1086467965452e-004
-0.4	-0.4	-2.1568000000000e+000	-2.1514644933911e+000	5.3355066088674e-003
-0.2	-0.2	-4.4480000000000e-001	-4.4194407422071e-001	2.8559257792854e-003
0.0	0.0	0.0000000000000e+000	-2.3838789132924e-003	2.3838789132924e-003
0.2	0.2	-2.8480000000000e-001	-2.8935944882744e-001	4.5594488274398e-003
0.4	0.4	-8.7680000000000e-001	-8.7876025943319e-001	1.9602594331924e-003
0.6	0.6	-1.4688000000000e+000	-1.4673570619209e+000	1.4429380791070e-003
0.8	0.8	-1.8688000000000e+000	-1.8699869398412e+000	1.1869398411688e-003
1	1	-2.0000000000000e+000	-2.0022750055757e+000	2.2750055757020e-003
MSE			**2.1712e-004**	

Figure 6. a) *Exact Solution* b) *Neural Solution*



Example 4

Consider the (IPDE)

$$u_{xx} + u_{yy} + u_x + u_y = f(x,y) + \int_0^x \int_0^y u(t,s)dt\,ds$$

where $f(x,y) = \cos(x) + \sin(x) + y + 1 - xy + ysin(x) - \frac{xy^3}{6}$

$$u(x,0) = 1 - \cos(x)\ ,\quad u(0,y) = 1 + 0.5 * y^2\ ,$$

$u_x(x,0) = \sin(x)\,,\ u_y(0,y) = y$

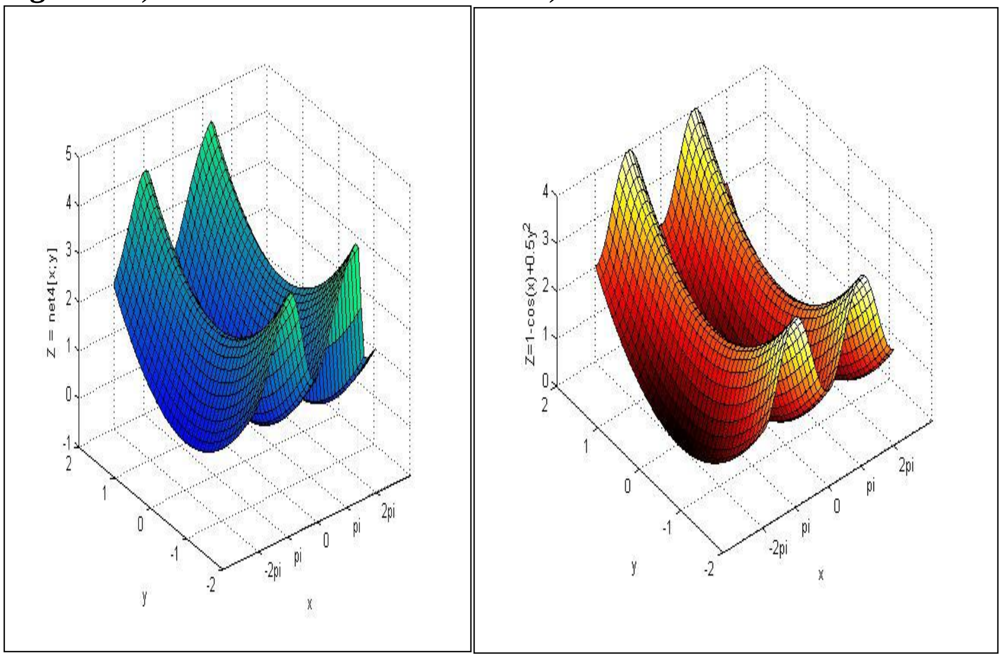
whose solution is $u(x,y) = 1 - \cos(x) + 0.5 * y^2$

by applying suggested method Table 4 shows the exact, neural result, error, and men square error.

Table 4. *Exact, Neural and Accuracy of Solution Example (4)*

X↓	Y	Exact $u_a(x,y)$	Trainlm $u_t(x,y)$	Error = $\lvert u_a - u_t \rvert$
-2π	-2	2.0000000000000e+000	1.9999986815427e+000	1.3184572857927e-006
-3π/2	-1.55	2.2012500000000e+000	2.2012510375950e+000	1.0375949788255e-006
-π	-1.10	2.6050000000000e+000	2.6049936589026e+000	6.3410974000533e-006
-π/2	-0.65	1.2112500000000e+000	1.2112498622264e+000	1.3777360297063e-007
0	-0.20	2.0000000000000e-002	2.0005337744973e-002	5.3377449731329e-006
π/2	0.25	1.0312500000000e+000	1.0312500444931e+000	4.4493079665031e-008
π	0.70	2.2450000000000e+000	2.2449926060133e+000	7.3939867100137e-006
3π/2	1.15	1.6612500000000e+000	1.6612549674107e+000	4.9674106916697e-006
2π	1.60	1.2800000000000e+000	1.2799969496180e+000	3.0503819574701e-006
2π	2	2.0000000000000e+000	1.9999986815427e+000	1.3184572857927e-006
MSE			**9.577414408279936e-011**	

Figure 7. a) *Exact Solution* b) *Neural Solution*



Conclusion

In this study, it is suggested a method based on the artificial neural network through design feed forward Neural Network. A design in our proposed method consists of a hidden layer with tanh (tansig). Transfer function and using Levenberg-Marquardt algorithm learning is used to fix some of nonlinear and linear (IPDE). The illustrative examples with the exceptional effects had been carried out to reveal the application of this method. The effects point out the proposed method that can be considered as the easy method and these are relevant to the numerical answer of this kind of equations. It is predicted that the neural network approach will be an effective tool for investigating approximate solutions and even analytic to nonlinear and linear practical equations depended on absolute error. For numerical functions the pc programmers have been written in Mat lap 10a.

References

[1] A. Borhanifar , K. Sadri, A generalized operational method for solving integro–partial differential equations based on Jacobi polynomials, Journal of mathematics and statistics, vol. (45) (2), pp. 311–335, 2016.

[2] Y. Chu, F. You, J. M. Wassick, Integrated planning and scheduling under production uncertainties: Bi-level model formulation and hybrid solution method, Computers and Chemical Engineering, vol.72, pp. 255–272, 2015.

[3] F. Shakeri, M. Dehghan , Solution of an integro-differential equation arising in oscillating magnetic field using He's homotopy perturbation method, Prog. Electromagnet Res. PIER, vol.78, pp. 361–376, 2008.

[4] N. Savaşaneril, M. Sezer, Solution of High-Order Linear Fredholm Integro-Differential Equations with Piecewise Intervals, Num. *Meth Partial Diffe. Equ*, vol.27, pp. 1327–1339, 2011.

[5] W. Tan, B. Khoshnevis, Integration of process planning and scheduling review, Journal of Intelligent Manufacturing, vol. 11(1), pp. 51–63, 2000.

[6] N. Savaşaneril, M. Sezer, Hybrid Taylor–Lucas Collocation Method for Numerical Solution of High-Order Pantograph Type Delay Differential Equations with Variables Delays, Appl. Math. Inf. Sci., vol.11(6), pp. 1795–1801, 2017.

[7] Ö.K. Kürkçü, E. Aslan, M. Sezer, A numerical method for solving some model problems arising in science and convergence analysis based on residual function," Appl. Numer. Math., vol 121, pp. 134–148, 2017.

[8] N. Savaşaneril and M. Sezer, Laguerre Polynomial Solution of High-Order Linear Fredholm Integro-Differential Equations, New Trends in Math. Sci., vol. 4(2), pp. 273–284, 2016.

[9] B. Gürbüz, M. Sezer, Laguerre collocation method for solving Fredholm-integro-differential equations with functional arguments, J. Appl. Math., vol. 2014, Article ID: 682398, pp.12 2014.

[10] Ö.K Kürkçü, E. Aslan, M. Sezer, A numerical approach with error estimation to solve general integro-differential–difference equations using Dickson polynomials, Appl. Math. Comput. vol. 276, pp. 324–339, 2016.

[11] Ö.K. Kürkçü, E. Aslan, M. Sezer, A novel collocation method based on residual error analysis for solving integro-differential equations using hybrid Dickson and Taylor polynomials, Sains Malays., vol. 46, pp. 335–347, 2017.

[12] C. Oğuz, M. Sezer, Chelyshkov collocation method for a class of mixed functional integro-differential equations, Appl. Math. Comput., vol. 259, pp. 943–954, 2015.

[13] N. Şahin, M. Sezer, A Bessel polynomial approach for solving general linear Fredholm integro-differential equations, Int. J. Comput. Math. vol.88 (14), pp. 3093–3111, 2011.

[14] K. Erdem, M. Sezer, A Bernoulli polynomial approach with residual correction for solving mixed linear Fredholm integro differential-difference equations, *J.* Difference Equ. Appl., vol.19, no.10, pp. 1619–1631, 2013.

[15] H. Gherjalar, H. Mohammodikia ,Numerical solution of functional integral and integro-differential equations by using B-Splines, Appl. Math., vol. 3, pp. 1940–1944, 2012.

[16] S. Reutskiy, The backward substitution method for multipoint problems with linear Volterra - Fredholm integro-differential equations of the neutral type, J. Comput. Appl. Math., vol. 296, pp. 724–738, 2016.

[17] R. Hristev, The ANN Book, GNU public license, Edition, (1998).

[18] B. wilamowski, S. Iplikei, an Algorithm for fast convergence in training neural network, IEEE. (2001): 1778–1782.