Journal of Computer Science and Applications.
ISSN 2231-1270 Volume 8, Number 1 (2016), pp. 1-20
© International Research Publication House
http://www.irphouse.com

Artificial Neural Networks with Nelder-Mead Optimization Method for Solving Nonlinear Integral Equations

M. Pashaie[1], M. Sadeghi[1] and A. Jafarian[1, *]
[1]*Department of Mathematics, Urmia Branch, Islamic Azad University, Urmia, Iran.*
[*] *Corresponding Author E-mail: Jafarian5594@yahoo.com*

Abstract

Nelder – Mead's unconstrained optimization method with multi-layer neural networks is implemented as a hybrid method for solving integral equations of different kinds. The proposed integral equation has been considered as an error function and Nelder-Mead method is used to minimize it. This method is capable of satisfying each type of integral equations. This algorithm by itself does have enough capability to be employed in finding solutions to each type of integral equations. Unsupervised learning is performed and the efficiency of the proposed method guarantees in each kind of Fredholm and Volterra equation. It may be time consuming in some rigorous problems, but high accuracy of convergence and approximation is preserved. We believe that large number of dimensions and variations does not create any problems in learning process. Numerical results obtained from Nelder Mead's method shows its potential capability compares to existing relative method.

Keywords: Integral equations, Multi-layer artificial neural networks, Nelder-Mead method, Hidden layer.

1. INTRODUCTION

Integral equations as one of the most beautiful topics are still applicable for many practitioners in all fields of science and engineering [26]. Some issues are expressed by integral equations such as population dynamics, birth and death of biological species that live together in the same habitat, fish populations and predators, road winding a or wire, rotary rod, dangling chain, and a wire which has a bean moving on it. Other issues in terms of ordinary differential equations or partial derivatives are determined with boundary conditions which can be converted to integral equations [26, 14]. Several numerical methods for solving integral equations exist including: Babylonian *et al*; [4] has suggested Haar function in finding the solution of integral equations. Karimi *et al*; [3] have solved Fredholm integral equations of the first kind using a least squares method. Adomian decomposition method, Homotopy perturbation method and Legendre wavelets are categorized in numerical methods which are performed in [1-5-6]. Dr. Babolian and Dr. Maleknejad *et al*; [15] have employed two-dimensional triangular functions to solve 2D Volterra integral-Fredholm equations. Dr. Maleknejad *et al*; [16] have suggested a direct method to solve 2D Volterra Hammerstein. Therefore, by using two-dimensional integral the given integral equation is converted to a non-linear equations system then the obtained solution is approximated. Maleknejad *et al*; [17] have implemented Bernstein polynomials to solve Volterra integral equations based on a new numerical hybrid method. Mosleh *et al*; [18] have solved fuzzy Volterra integral equations by an approximation of Bernstein polynomials. One tool that has been used in all branches of science is the science of artificial neural networks which is involved in solving integral equations. Jafarian *et al*; [20] were solved linear Fredholm integral equations by the neural network feed-back neural network approach. Golbabai *et al*; [8-9] have solved nonlinear integral equations system by RBF neural network with different learning rules. Jafarian *et al*; [19], have solved Abel integral equations by using an artificial neural network. In this paper, up and down functions as solutions in fuzzy function are approximated by a multi-layer neural network and back propagation learning algorithm is employed. In most of the studies, approximation is done by neural network which have created proper solution. In this paper, steepest descent optimization method is used. Asady *et al*; [21] have applied multi-layer artificial neural network and an optimization method to solve Fredholm integral equation of the second kind. Jafarian *et al*; [22] have considered local Bernstein polynomial to solve Fredholm linear equations.

In this paper, different types of integral equations are solved by a neural network with a hidden layer and Nelder-Mead optimization method. First, the given equation is converted to a function of energy (cost) then the solution is getting parallel to a multi-layer network. Finally, the network parameters (learning) are determined such that minimize the created cost function. The proposed method can be used even for different types of equations. Bidokhti *et al*; [27] have provided the solution to differential equations system with partial derivatives using neural networks and optimization techniques.

2. OPTIMIZATION NELDER MEAD'S SIMPLEX METHOD

Nelder - Mead method is a widely used algorithm in unconstrained optimization and it does not require calculation of derivatives. This algorithm works moderately well in applications of sciences and engineering. The function values at each vertex will be evaluated in each of iteration. It does not have to calculate derivatives to move along a function. However, through iterations, Nelder Mead's simplex method does not rely on the gradient or partial derivatives. The Nelder–Mead method is a commonly applied numerical method used to find the minimum or maximum of an error. It is a direct search method of optimization that works moderately well for stochastic problems. Each iteration in R^n is formed by $n+1$vertices such as

$X = \{X^0, X^1, \cdots, X^n\}$.simplex vertices are sorted in ascending order based on the values of the function f. In other words, $f(X^0)$ and $f(X^n)$ are the best vertex and the worst vertex, respectively. Depending on three operations such as reflection, expansion and contraction, iterations are done. [10]. in each iteration, the worst vertex X^n will be replaced by another vertex which has just been found on X^c, X^n.

$$X = X^c + \delta(X^c - X^n), \qquad \delta \in R$$

Where $X^c = \sum_{i=0}^{n-1} X^i / n$ is the centroids point of the best n vertices. The value of δ determines the type of repetition. For example, $\delta = 1, \delta = 2, \delta = \pm\frac{1}{2}$ represents reflection, expansion and contraction, respectively. (Fig. 1)

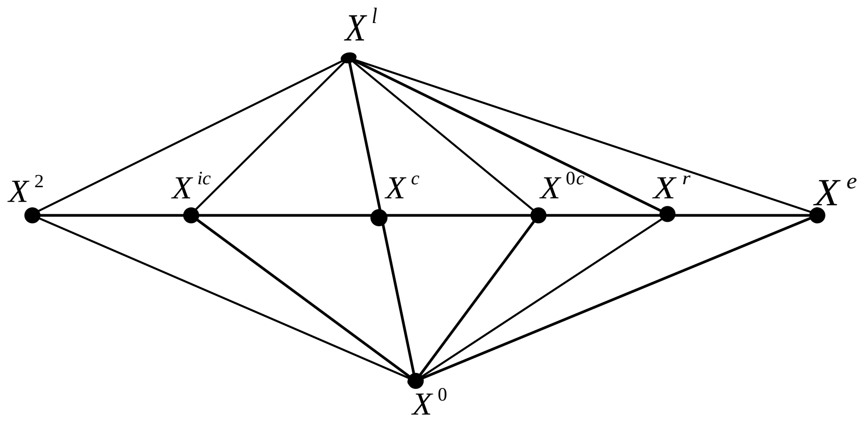

Fig.1. Reflection, expansion and contraction in a simplex with 4 vertices

Nelder Mead simplex method (SIM) may do shrink to form a new simplex. By Shrinking, all the vertices in X move forward to the best head vertex.

2.1 Overview of Nelder Mead's algorithm

The main steps of Nelder Mead's simplex method can be summarized as following:

- **Step 1:** Initialization: choose an initial simplex of vertices $X_0 = \{X_0^0, X_0^1, \cdots, X_0^n\}$. Evaluate f at the points in X0. Choose constants: $0 < v^s < 1, -1 < \delta^{ic} < 0 < \delta^{0c} < \delta^r < \delta^e$. for $k = 0,1,2,\cdots$ $X = X_k$
- **Step 2:** sorting: sort the $n+1$ vertices of $X = \{X^0, X^1, \cdots, X^n\}$ so that $f^0 = f(X^0) \le f' = f(X') \le \cdots \le f^n = f(X^n)$.

- **Step 3:** Reflection: Reflect the worst vertex X^n over the centroids $X^c = \sum_{i=0}^{n-1} X^i / n$ of the remaining n vertices: $X^r = X^c + \delta^r (X^c - X^n)$. Evaluate $f^r = f(X^r)$. If $f^0 \le f^r \le f^{n-1}$, then replace X^n by the reflected point X^r and terminate the iteration: $X_{k+1} = \{X^0, X^1, \cdots, X^{n-1}, X^r\}$.
- **Step 4:** expansion: If $f^r < f^0$, then calculate the extended points
  $X^e = X^c + \delta^e (X^c - X^n)$,
  and evaluate $f^e = f(X^e)$. If $f^e \le f^r$, replace y^n by the expansion point X^e and terminate the iteration:
  $X_{k+1} = \{X^0, X^1, \cdots, X^{n-1}, X^r\}$.
- **Step 5:** contraction: If $f^r \ge f^{n-1}$, then a contraction is performed between the best of X^r and X^n.
  Outside contraction: If $f^r < f^n$, perform an outside contraction $X^{0c} = X^c + \delta^{0c} (X^c - X^n)$
  and evaluate $f^{0c} = f(X^{0c})$. If $f^{0c} \le f^r$, then replace X^n by the outside contraction point X_k^{0c} and terminate the iteration: $X_{k+1} = \{X^0, X^1, \cdots, X^{n-1}, X^{0c}\}$. Otherwise, perform a shrink.
  Inside contraction: If $f^r \ge f^n$, perform an inside contraction $X^{ic} = X^c + \delta^{ic} (X^c - X^n)$
  Evaluate $f^{ic} = f(X^{ic})$. If $f^{ic} = f^r$, then replace X^n by the inside contraction point y_k^{ic} and terminate the iteration: $X_{k+1} = \{X^0, X^1, \cdots, X^{n-1}, X^{ic}\}$.Otherwise, perform a shrink.

- **Step 6:** Shrinking: Evaluation f at the n-points $X^0 + v^s (X^i - X^o)$, $i = 1, 2, \cdots, n$, and replace $X^1, \cdots, X^n$ by these points, terminating the iteration:
  $X_{k+1} = \{X^0 + v^s (X^i - X^o), i = 0, 1, \cdots, n\}$.

The most general properties of the Nelder-Mead algorithm and convergence have been mentioned in [7].

3. CONVERTING INTEGRAL EQUATION TO THE OPTIMIZATION PROBLEM

In this section, taking into account the given integral equation and determining the desired interval to solve it, it will be converted to unconstrained optimization problem. If the solution to integral equation is on interval $[c,d]$, then by selecting points from $[c,d]$ an educational complex is provided and Then a multi-layer network that is an approximator of integral equation , is trained. The integral equation is converted to a cost function so it will be unsupervised learning which is performed using Nelder-Mead.

3.1 Conversion Integral equation to optimization problem
Nonlinear Fredholm integral equation is considered:

$$\lambda g\left(u(x)\right)-\int_{\Omega}K(x,y)h\left(u(y)\right)dy=f(x),\qquad \Omega=[a,b] \tag{1}$$

Where $f\in L^2\left[a,b\right)$, $K\in L^2\left[a,b\right)^2$ are known functions, $u(x)$ is unknown and it is the solution to the equation, functions g and h are derivational on Ω. If $\lambda=0$, it will be considered the equation of the first kind. Later we will select Volterra case for (1). We will approximate the unknown function $u(x)$ by a multi-layer neural network with input x and output $u(x)$. Assume that $N(x,w,b)$ is the approximation of $u(x)$ which is a multi-layer network with weights w and bias b. we aim to solve equation (1) on interval $[c,d]$. Obviously, if $N(x,w,b)$ is the approximation of $u(x)$, for each $x_i\in[c,d]$ we have $N(x_i,w,b)$ which should be satisfied in (1). Therefore, by placing $N(x_i,w,b)$ in equation (1), then

$$\lambda g\left(N(x,w,b)\right)-\int_{\Omega}K(x,y)h\left(N(y,w,b)\right)dy=f(x). \tag{2}$$

Now for $x_i\in[c,d]$ we attempt to satisfy (2), so it will be defined:

$$\xi_i=\lambda g\left(N(x_i,w,b)\right)-\int_{\Omega}K(x_i,y)h\left(N(y,w,b)\right)dy, \tag{3}$$

And

$$\varepsilon_i=\xi_i-f(x_i). \tag{4}$$

Note that for each $x_i\in[c,d]$ value of ε_i in (2), error is in x_i. To achieve the global error in $[c,d]$, the error sum of squares is stated as follows**:**

$$E\left(x_i,N\right)=\sum_{i\in I}\varepsilon_i^2, \tag{5}$$

Where I is the set of indexes for the points x_i on interval $[c,d]$. Points x_i are not equally divided on $[c,d]$ and it may contain certain points, such as the local or Gaussian points. Function $E\left(x_i,N\right)$ in (5) is called cost function which is a multivariate function based on neural network parameters (weights and bias). If E is close enough to zero, it has got its least value. Then the approximation of $N\left(x,w,b\right)$

would be appropriate to estimate$u(x)$. The goal is to find the neural network parameters as:

$$E = \underset{i}{Min}\, E\left(x_i, N\right) \tag{6}$$

To find E, an unconstrained minimum problem will step in. Nelder-Mead method is used to find the optimal point in the optimization problem (6). Network training using optimization method is unsupervised. It is necessary to provide enough x_i from $\left[c, d\right]$ to meet arbitrary learning accuracy. If there are more ofx_i, it will require long running time. Sometimes it may take up to 30 minutes. Thus, computer program is set such that at the start there is few training data but at end of the epochs training data will be added more to minimize the error. Finally by the least number of training data, learning process will be accomplished.

2.3 Conversion system of integral equations to the optimization problem and multilayer neural networks

In this section, it will be shown that the integral equations can also be converted into an unconstrained optimization problem. Then the nature of the neural network which is designed to be used in this paper will be expressed.
Consider the following nonlinear system of integral equations,

$$y(x) = \int_0^x K\left(x, t, y(t)\right)dt = g(x), \tag{7}$$

Where

$$y(x) = \left(y_1(x), \cdots, y_n x\right)^T,$$

$$g(x) = \left(g_1(x), \cdots, g_n x\right)^T,$$

$$K\left(x, t, y(t)\right) = \left(K_1\left(x, t, y(t)\right), \cdots, K_n\left(x, t, y(t)\right)\right)^T.$$

To get a solution to (7), assume that**:**

$$y(x) \approx N\left(w, x\right). \tag{8}$$

Where N is a multi-layer neural network with input parameters *w* and *x.* $\left(1 - n' - n\right)$ is the applied network, n' is the number of hidden layer neurons, n is the number of output. The structure of this neural network could be seen in Figure 2.

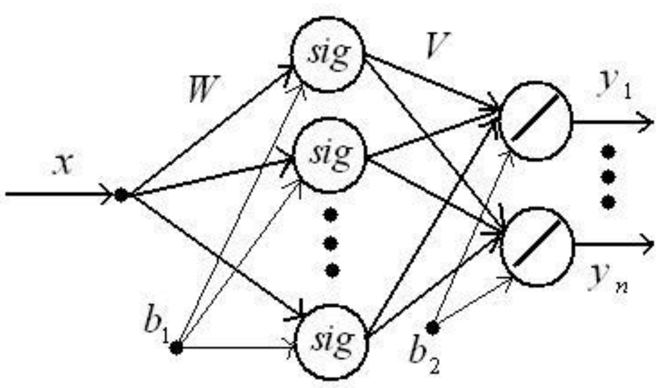

Fig.2. MLP network for solving integral equations.

The mathematical equations of network $N(w,x)$ that approximate the solution of integral equation is formed as follows**:**

$$y_i(x) \approx \sum_{j=1}^{n} V_{ij}\sigma\left(\sum_{p=1}^{n'}\sum_{j=1}^{n} w_{pj}x_j + b_p\right),$$

The transfer function is selected sigmoid function and network has n input and n output. The matrix form of this equation is as follows:

$$y(x) \approx V \cdot \sigma(wx+b).$$

These two equation are stated for Figure 3 and the last layer includes no bias. Because $y(0)=0$ and if $y(0)\neq 0$ such bias can also be added to the last layer, and if not, the learning process does not fail. Therefore, to ensure keeping $y(0)=0$, the bias output layer has been removed.

If the network N has only one input, then:

$$N\left(w, x_1, x_2, \cdots, x_n\right) = \sum_{i=1}^{H} V_i S\left[\left(\sum_{j=1}^{n} w_{ij}x_j\right) + b_i\right], \tag{9}$$

Where H is the number of hidden layer neurons. It can also have more inputs and more outputs. The Accuracy of approximation depends on the number of neurons in the hidden layer not the number of layers.

It has been proven that a multi-layer network with a hidden layer having enough neurons has more ability to approximate any continuous function with any accuracy [11]. The following theorems ensure the convergence of networks with a hidden layer.

Theorem 1: For every continuous function $f:[0,1]^n \to \mathbb{R}$ we have functions $y:\mathbb{R}\to\mathbb{R}$ and

$(i=0,\cdots,2n) S_i = [0,1]\to[0,1]$

So

$$f(x_1,\cdots,x_n) - \sum_{i=0}^{n} g\left[\sum_{i=1}^{n} V_i S_i(x_j)\right]. \tag{10}$$

Where $j=1,\cdots,n$, $0\le N_j \le 1$.

Proof: see [12]

Theorem 2: For every continuous function $f:[0,1]^n \to R$, there is bounded function $S:R\to R$, natural number H and b_i, N_i, w_{ij} $(i=1,\cdots,H, j=1,\cdots,n)$ constants. For each given $\xi>0$:

$$\left| f(x_1,\cdots,x_n) - \sum_{i=1}^{H} V_i S\left[\left(\sum_{j=1}^{n} w_{ij}x_j\right)+b_i\right]\right| < \xi. \tag{11}$$

Proof: it is proven by applying Ston - Weirestrass theorem and Cybenko theorem [13].

Network outputs $N(w,x)$ are solution to integral equations system. In case of having integral equations system, network owns n output, and in case of having the integral equations network owns an output. Substituting network $N(w,x)$ in (7), a cost function is provided:

$$\xi_{i,j} = N_j(x_i) - \int_0^{x_i} K\left(x_i,t,N_j(t)\right)dt - g(x_i), \tag{12}$$

In which the N_j (outputs) are network components that approximates y_i. The sum of squared errors $\xi_{i,j}$ is cost function which will be minimized.

$$E = \underset{w}{Min}\, E\left(\xi_{i,j}\right) = \sum_{i=1}^{m}\sum_{j=1}^{n} \xi_{i,j}^2, \tag{13}$$

Where m is the number of learning samples. Nelder-Mead method will be used to find the minimum $E\left(\xi_{i,j}\right)$.

Remark the existing definite integral in the equation need to be approximated in each iteration of the optimization process. Numerical trapezoidal methods, Simpson and

Romberg method provide acceptable solution to Fredholm integral equation or Volterra equation of the second kind, but due to high sensitivity of the first kind equations, the 10 - point Gaussian approximation method is used to solve them.

3. ALGORITHM PROCEDURE

To solve the equation using the method of converting equations to optimization problem takes a series of steps:

- Step 1: identifying the interval $[c,d]$ then solving the integral equation. ξ is determined for the accuracy approximation.
- Step 2: Determining the points x_i of $[c,d]$ as the training data. (Training data percent may be selected for verification.)
- Step 3: Designing of neural network. If it is an equation system, the number of outputs and unknown functions will be equal.
- Step 4: creating the cost function which needs to be minimized and determining an appropriate numerical method to approximate definite integrals. If it is in a bad state, Romberg method or 10-point Gauss method is recommended to be employed.
- Step 5: Implementing Nelder-Mead optimization method to minimize E. training will be continued until the amount of the cost function is getting lower than ξ.

It is clear that after the successful implementation of the algorithm and finding the optimal parameters of the network N, the solution will be given as a neural network. Determining each $x \in [c,d]$, solution to integral equation will be achieved. 10-point Gauss method can be applied to enhance the speed and accuracy in the integral approximation method used 10-point Gauss. For example, to approximate

$$\int_0^1 f(t)dt \simeq \sum_{i=1}^{n} w_i f(t_i)$$

Parameters are (table 1).

Table 1. Nods and weights of the Gauss-Legendre quadrature for n=10

i	t_i	w_i
1	0.0130467357414141399610179	0.03333567215434406879678440
2	0.0674683166555077446339516	0.07472567457529029657288817
3	0.1602952158504877968828363	0.10954318125799102199776746
4	0.2833023029353764046003670	0.13463335965499817754561346
5	0.4255628305091843945575870	0.14776211235737643508694649
6	0.5744371694908156054424130	0.14776211235737643508694649
7	0.7166976970646235953996330	0.13463335965499817754561346
8	0.8397047841495122031171637	0.10954318125799102199776746
9	0.9325316833444922553660483	0.07472567457529029657288817
10	0.9869532642585858600389820	0.03333567215434406879678440

For other interval transmission can be used. Comparing to Simpson and Romberg methods, this form requires less running time. It can be said that 8 to 10 neuron will be good enough to appropriate sine function with a relatively high frequency, but if learning is not efficient, number of neurons can be increased. Another way to select the appropriate number of hidden layer neuron is using growth and pruning method [23]. This method starts with one neuron, then by decreasing learning error it will increase, and when the error is large, the number of neurons decreases to the point that in a number of well-balanced, learning process ends.

In this method, to achieve the desired accuracy, the following will be satisfied:

a. Choosing initial weights and biases which are always small random numbers.
b. The right choice for the number of repetitions in Nelder-Mead method.
c. The number of epochs is required to achieve the desired accuracy. This number can be determined by trial and error or a defined scale.
d. Nelder Mead method parameters for the majority of applications are: Expansion $V = 2$, $\beta = 0.5$ (contraction) , $\alpha = 1$ (reflection)
e. The number of intervals $\left[c,d\right]$ to provide the test set. Thereby increasing the amount of these points' results in increasing running time. In ordinary matters, the number of points is between 10 and 20.

4. NUMERICAL RESULTS

In this section, several examples are illustrated to point out the effectiveness of the method. As mentioned in previous sections, this method can be used to solve different kinds of integral equations. As a result, examples of different kinds have been selected.

Example 1. Fredholm integral equation is given:

$$u(x) = \frac{1}{2}\sin(x) + \int_0^{\frac{\pi}{2}} \sin(x)\cos(t)u(t)dt \ , \tag{14}$$

The Exact solution of function is $u(x) = \sin x$.

After the converting (14) to an unconstrained optimization problem and solving the problem by using Nelder-Mead method, obtained results are given in table 2.

Table 2: Obtained results of optimization error u (x) using neural network

$x = \frac{\pi r}{20}$	$\lvert u(x) - \hat{u}(x) \rvert$
r = 0	1.74147×10^{-4}
r = 1	8.96395×10^{-5}
r = 2	9.34409×10^{-5}
r = 3	5.73489×10^{-5}

r = 4	1.53469×10^{-4}
r = 5	1.56712×10^{-4}
r = 6	1.27430×10^{-4}
r = 7	3.15813×10^{-4}
r = 8	5.41612×10^{-6}
r = 9	4.77419×10^{-4}
r = 10	1.01914×10^{-3}

Properties of method in the process of learning and optimization are as follows:

a. The number of hidden layer neurons: 4
b. The number of points for interval $\left[0, \frac{\pi}{2}\right]$ to create training data: 100
c. Numerical method for integral approximation: Romberg with $n = 5$
d. The number of irritation of Nelder – Mead method: 8000
e. If trapezoidal or Simpson method is used instead of Romberg method, we will get less accurate approximation.

Example 2. Consider the nonlinear Fredholm integral equations system, [2]

$$\begin{cases} f_1(x) = x - \dfrac{5}{18} + \displaystyle\int_0^1 \dfrac{1}{3}\left(f_1(t) + f_2(t)\right)dt \\ f_2(x) = x^2 - \dfrac{2}{9} + \displaystyle\int_0^1 \dfrac{1}{3}\left(f_1^2(t) + f_2(t)\right)dt \end{cases} \tag{15}$$

Exact solutions include: $f_1(x) = x$ and $f_2(x) = x^2$. This example is solved by Adomian decomposition method in [2] and the maximum error for $x \in [0,1]$ in the functions $f_1(x)$ and $f_2(x)$ are 0.0230 and 0.0443, respectively.

In the method of converting to the optimization problem, after implementation of the Nelder-Mead method and successful learning, calculation error of functions f_1 ‘ f_2 shown in table (3). This table reveal the approximate value of the function f and $\hat{f}$.

Network specifications include:

- The number of neurons: 10
- The number of interval $[0,1]$: 40
- Numerical integration method: Gaussian 10- point

Table 3: Error of functions f_1 ، f_2 using neural network in example 2

x	$\left\|f_1(x)-\hat{f}_1(x)\right\|$	$\left\|f_2(x)-\hat{f}_2(x)\right\|$
0.0	1.13760×10^{-4}	2.52494×10^{-3}
0.1	1.81569×10^{-4}	9.75329×10^{-4}
0.2	1.06722×10^{-4}	1.41206×10^{-3}
0.3	3.70786×10^{-4}	4.93032×10^{-4}
0.4	3.80033×10^{-4}	5.70629×10^{-4}
0.5	1.69787×10^{-4}	1.05137×10^{-3}
0.6	8.78871×10^{-5}	6.90843×10^{-4}
0.7	2.08392×10^{-4}	3.03296×10^{-4}
0.8	9.73456×10^{-5}	1.24563×10^{-3}
0.9	1.78070×10^{-4}	9.51582×10^{-4}
1	3.53236×10^{-4}	2.282100×10^{-3}

Steps to reduce the error sum of squares, shown in the Figure 3.

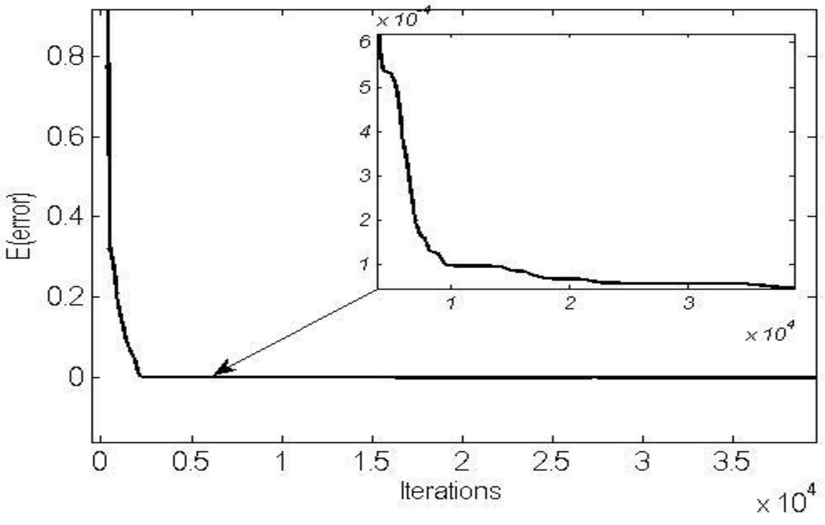

Fig. 3. Reduction of the calculated error in Nelder-Mead method in example 2

It can be seen in all iterations, error is reduced and the learning process is not swingy using Nelder-Mead method, because error has not increased, during the iteration.

The greatest amount of error for the estimation of f1 belongs to Adomian decomposition method [2] of 0.0230. While the error of multi-layer neural network provides 3.80033×10^{-4}. This result guaranties that estimation of f1 on $[0,1]$ by multi-layer neural network is very successful over Adomian decomposition method [2].

The highest amount of errors for estimation of f2 using Adomian decomposition method is 0.0443, while for the same amount of error using multi-layer network is: 2.282100×10^{-3}.

Neural network method superior to Adomian decomposition method in estimating f2.

Example 3. Consider the following Fredholm integral equation of the first kind [3]:

$$\int_0^1 \left(t^2 + S^2\right)^{\frac{1}{2}} u(s)ds = \frac{(1+t^2)^{\frac{3}{2}} - t^3}{3}, \tag{16}$$

$u(x) = x$ is the exact solution. This example is solved in [3] by using the least squares method. In this paper, Gauss 8 - points is employed to approximate the infinite integrals. Here after creating cost function and implementation of the Nelder-Mead method a novel approximation to $u(x)$ is achieved. The results are displayed in table (4):

Table 4: Obtained errors using neural network and comparing them with Gauss 10-points in example 3

t	$Lu^*(t) - u_4(t)$	$Lu^*(t) - u^G(t)$	Error of Neural Network
0.98014492824876 81	1.94×10^{-4}	4.28×10^{-2}	1.96×10^{-3}
0.4082826787521751	4.77×10^{-5}	6.59×10^{-3}	2.30×10^{-4}
0.1016667612931866	7.94×10^{-4}	1.79×10^{-3}	6.20×10^{-4}
0.2372337950418355	3.93×10^{-4}	2.98×10^{-3}	5.84×10^{-4}
0.5917173212478249	1.69×10^{-4}	1.59×10^{-2}	6.35×10^{-4}
0.8983332387068134	7.96×10^{-5}	4.52×10^{-2}	3.26×10^{-4}
0.7627662049581645	8.45×10^{-5}	3.04×10^{-2}	8.34×10^{-4}
0.0198550717512319	8.47×10^{-4}	1.96×10^{-3}	7.27×10^{-4}

In this example, Gauss 10-points method is employed to approximate infinite integrals.

Properties of algorithm are as follows:

a. The number of hidden layer neurons: 4
b. The number of points on $[0,1]$ to create training data: 10
c. Numerical integrating method: Gauss-Legendre 10-points
d. The number of iterative in Nelder - Mead method: 2500

The amount of calculation error in the Gaussian 8-point for a number of different epochs in Nelder-Mead method along with exact and approximate figures are shown in the following figures.

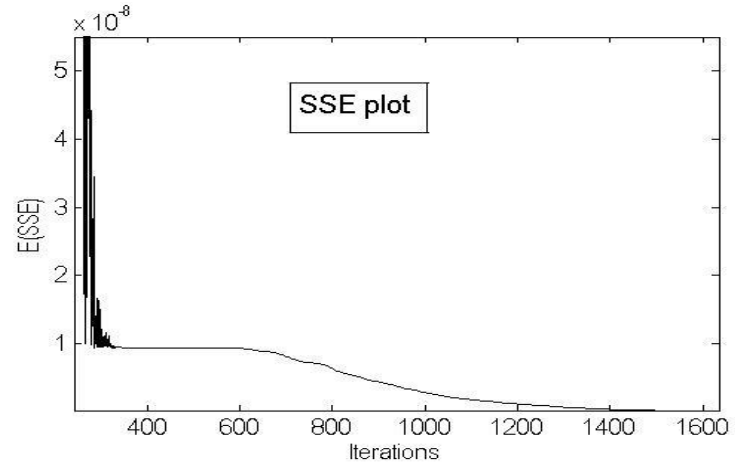

Fig. 4. Graph of sum of squares error in different iterations

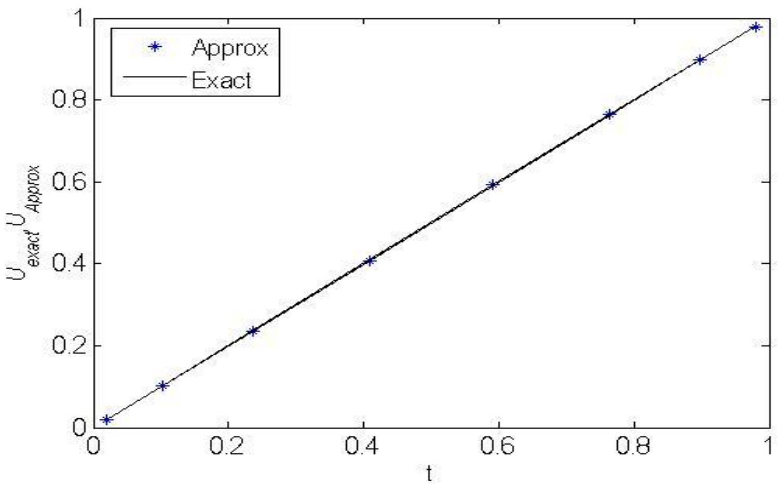

Fig. 5. Approximate and exact solution of function u(x) after neural network learning in example 3

According to the table 4, in the Gaussian 8 -point, obtained approximation by this method in third point, is better than approximation of $u_4(t)$ which is provided by using method in [3] while in other points, is either the same or very small. We note that the results obtained by proposed method have better result than approximation $u^G(t)$ in [3] which guarantees this method more efficient for solving integral equations. The sum of squares error after ending to learning process is

$$E = 3.1274 \times 10^{-12}$$

Example 4. Consider the following Fredholm integral equation of the first kind [3]:

$$\int_0^1 e^{ts} u(s) ds = \frac{e^{t+1} - 1}{t+1}, \tag{17}$$

Where $u^*(t) = e^t$ is the exact solution. After the implementation of the proposed method, the error sum of squares is 3.93454856 × 10-14. In this instance, a numerical method- Gauss-Legendre 10-point- is used to approximate infinite integrals. Here there are *3198* iterative in Nelder-Mead method and *3* neurons in the hidden layer.

Table 5. Error of approximate and exact solution of function u(x) in some points [0, 1] in example 4

x	$\lvert u(x) - u^*(x) \rvert$
0.0	4.36075×10^{-3}
0.1	2.62983×10^{-5}
0.2	3.39746×10^{-3}
0.3	3.03088×10^{-3}
0.4	5.50632×10^{-4}
0.5	4.47659×10^{-3}
0.6	4.94309×10^{-3}
0.7	3.53404×10^{-4}
0.8	8.10120×10^{-3}
0.9	5.15755×10^{-3}
1	1.76206×10^{-2}

Given the maximum error in [3] and the table 5, it can be said that, in some points on $[0,1]$ the proposed approximation method seems to be efficient as method in [3] equal. And error in other points is negligible. It is clear that integral equation of the first kind gives the appropriate results in Table 5. In fact, the greatest error in the method in [3] is 8.33 × 10-3. Based on the above table it is obvious that, multi-layer neural network performance is more efficient than of the method in [3]. Note that a designed multi-layer network with 3 neurons in hidden layer is achieved in Table 6. Error curve and how to reduce it during the iterations in illustrated in Figure 6.

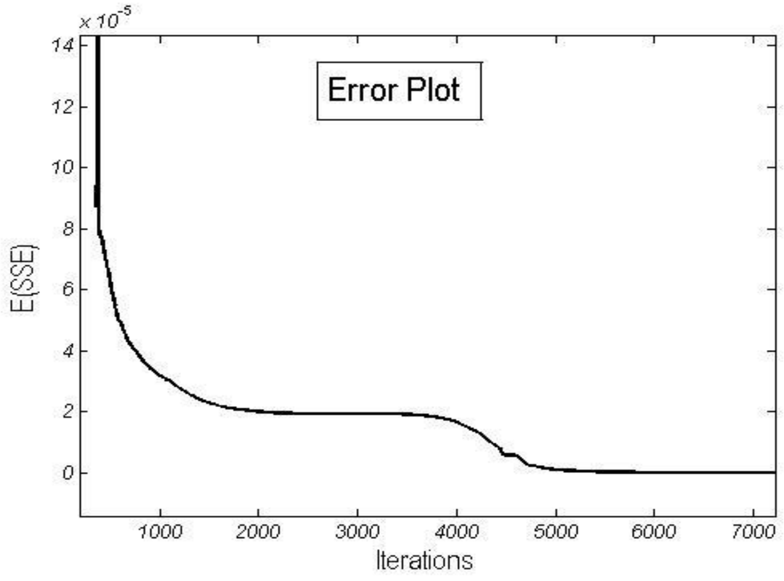

Fig. 6. Steps of reducing the error using neural network in example 4.

Example 5. Consider the nonlinear integral equation [k1]:

$$u(x)+\int_0^1 e^{x-2y}\left(u(y)\right)^3 dy = e^{x+1} \qquad a \le x \le 1$$

Where $u(x)=e^x$ is the exact solution. Babylonian *et al*; [24] have solved this equation using Haar wavelet method. In this paper, this equation is solved by multi-layer network and unsupervised learning rule, Nelder-Mead method. In both methods, the error in some points on $[0,1]$, can be seen in the following table:

Table 6: Comparing the errors of the Haar wavelet method [24] and neural network Nelder-Mead method for example 5

x	$u_{exact} - u_{Haar-wavelet}$ [24]	neural-network , nelder-Mead
0.1	0.002046	4.8851×10^{-4}
0.2	0.003299	8.3051×10^{-4}
0.3	0.008693	9.6731×10^{-5}
0.4	0.016906	9.1702×10^{-4}
0.5	0.018681	1.4244×10^{-3}
0.6	0.011742	9.8000×10^{-4}
0.7	0.002927	3.098×10^{-4}
0.8	0.008084	1.6423×10^{-3}
0.9	0.021624	1.3996×10^{-3}

Once neural network is trained, the exact function and approximate figure can be seen in the figure 7.

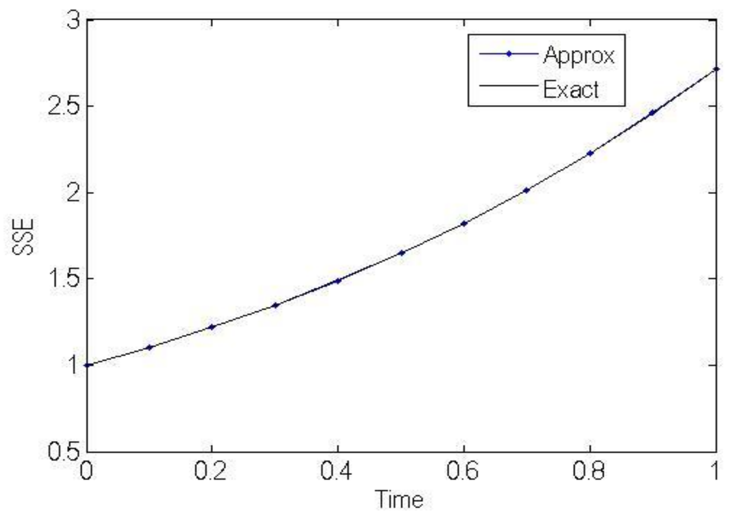

Fig.7. The exact and approximate graph $u(x) = e^x$ of neural network for example 5

Also steps to reducing the error for a period of time during iteration of Nelder- Mead, is seen in the figure 8:

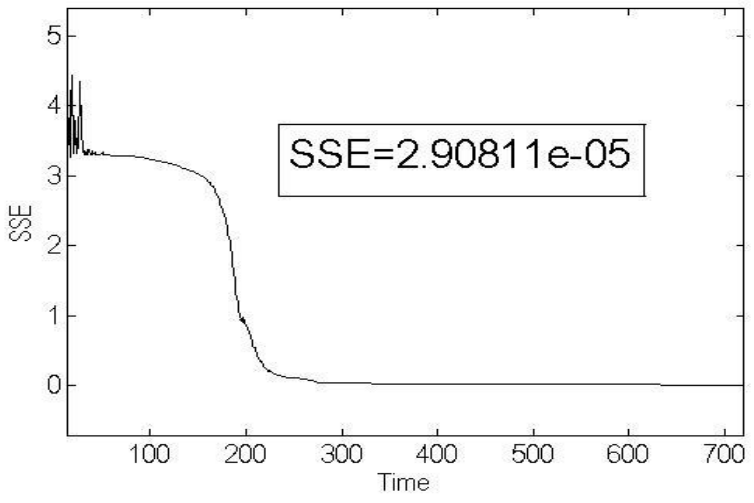

Fig.8. Amount of error during iterations of Nelder- Mead for example 5

Based on the results of this section as summarized in looking up Table we believe that the accuracy of multi-layer neural network to solve Example 5 is more efficient than Haar wavelet-method [24]. Haar wavelet method errors, on $[0,1]$ is about 0.01 or 0.001 while in multi-layered neural networks nit can be about 0.00001, This ability is due to high approximation of multi-layer networks, and on the other hand Nelder-Mead method is also able to reduce arbitrarily the error sum of squares. As previously mentioned, one of the important features of multi-layer neural networks is that it can be used in any type of integral equation.

5. CONCLUSIONS AND RECOMMENDATIONS

In this paper, our main focus was to present an emerging method based on the concept of neural networks for solving differential equations. Here in this study, we have started with converting the given integral equation to an unconstrained optimization problem in order to find solution with sufficient accuracy. We have also presented some basic concept of neural network and Nelder-Mead that is required for the study. Different neural network methods based on multi-layer perceptron, Nelder-Mead method, and finite element etc. are then presented for solving differential equations. It has been pointed out that the employment of proposed method owns many attractive features towards the problem compared to the other existing methods in the literature. The parameters (weights, centers and widths) of the approximate solution are adjusted by using an unconstrained optimization problem. Preparation of input data, robustness of methods and the high accuracy of the solutions made this method highly acceptable. The main advantage of the proposed approach is that once the network is trained, it allows evaluation of the solution at any desired number of points instantaneously with negligible error. Moreover, it can be used to solve the equations of various kinds. Once the type of differential equation is changed, the method is still convergence with different accuracy. More precisely, once the numerical examples are employed, Nelder-Mead method is converging to approximate solution for various problems. But sometimes to approach the accuracy of more than three decimal, we need to devote running time about 30 to 40 minutes. The proposed method satisfies in Volterra or Fredholm integral equation of the first and second kind. Note that in most integral equations, the network with 1-4 neurons in the hidden layer converges with acceptable accuracy to the solution. The following is a list of a few changes can rise the efficiently of the method:

1. Change the optimization method.
2. Change the definite integral approximation method.
3. Change the network architecture. For example, RBF network instead of a multi-layer network.
4. The use of computers with high-speed calculation to perform as much iteration.

REFERENCES

[1] J. Biazar, H. Ghazvini. He’s Homotopy perturbation method for solving systems of Volterra integral equations of the second kind. Chaos, Solitons and Fractals 39 (2). (2009) 770–777.

[2] E. Babolian, J. Biazar, A.R. Vahidi. The decomposition method applied to systems of Fredholm integral equations of the second kind. Applied Mathematics and Computation 148 (2004) 443–452

[3] S. Karimi, M. Jozi. A new iterative method for solving linear Fredholm integral equations using the least squares method. Applied Mathematics and Computation 250 (2015) 744–758

[4] E. Babolian, S. Bazm, P. Lima, Numerical Solution of nonlinear two – djmensional integral equations using rationalized Haar functions, Commun. Nonlinear Sci. Numer. Simulat. 16 (2011) 1164 – 1175.

[5] S. Abbasbandy. Homotopy perturbation method for quadratic Riccati differential equation and comparison with Adomian_s decomposition method. Applied Mathematics and Computation 172 (2006) 485–490.

[6] H. Jafari, H.Hosseinzadeh, S. Mohammadzadeh, Numerical Solution of system of linear integral equations by using legender warelets. Int. J. Open problems, Comput. Math. 5(2010). 63-71.

[7] A. Jafarian, S. Masoomy Nia, Jtiliting feed-back neural network approach for solving linear Fredholm integral equations system, Appl. Math. Mode. 2012, doi:http://dx.doi.org/10.1016/j.apm.

[8] A. Golbabai b, M. Mammadova, S. Seifollahi, Solving a system of nonlinear integral equations by an RBF network, Computers and Mathematics with Applications. 57 (2006). 1651-1658.

[9] A. Golbabai, S. Seifollahi. Numerical solution of the second kind integral equations using radial basis function networks. Applied Mathematics and Computation 174 (2006) 877–883.

[10] S.S. RAO, Engineering Optimization, Theory and practice, purdue University, West Lafayette, Hndiana, (1996).

[11] E.K. Blum, L.K. Li, Approximation theory and feed forward networks, Neural Network, 4(1991): 511-515.

[12] A.K. Kolmogorov, On the Representation of Continuous functions of several variables by superpositions of continuous functions of one variable and addition, DOKl, Akad, Nauk, SSSR, 114, (1957), 953-956.

[13] G. Cybenko, Approximation by superpositions of a sigmoidal function, Math, Control Signals Syst, 2(4) (1989), 304-314.

[14] A. Jerri, Introduction to Integral Equations with Applications, INC, John Wiley and Sons, (1999).

[15] E. Babolian, K. Maleknejad, M. Roodaki, H. Almasieh, Two-dimensional triangular functions and their applications to nonlinear 2D Volterra–Fredholm integral equations, Compu, Math. Appl. 60, 1711-1722, (2010).

[16] M. Maleknejad, S. Sohrabi, B. Baranji, Two dimensional PCBF s application to nonlinear Volterra integral equations, in: Proceedings of the world, (2009).

[17] K. Maleknejad, E. Hashemizadeh, R. Ezzati, A New approach to the numerical solution of Volterra integral equation by using Bernstein's approximation, Commun Nonlinear SCi Numer Simulat, (16) 647-655 (2011).

[18] M. Mosleh, M. Otadi, Solution of Fuzzy Volterra Integral Equations In a Bernstein Polynomial Basis, Journal of Advances in Information Technology, Vol. 4, No. 3, August (2013).

[19] A. Jafarian, S. Measoomy Nia, Artificial neural network approach to the fuzzy Abel integral equation problem. Journal of Intelligent and Fuzzy Systems 27, 83-91, (2014).

[20] A. Jafarian, S. Measoomy Nia, Using feed-Back Neural Network method for solving linear Fredholm integral equations of the second kind, Journal of Hyperstructures 2(1), (2013) 53-71.

[21] B. Asady, F. Hakimzadegan, R. Nazarlue, Utilizing artificial neural network approach for solving two-dimensuonal integral equation's, Math, Sci, (2014), 8-117.

[22] A. Jafarian, S. Measoomy Nia, A.K. Golmankhanh, D. Baleanu, Numerical Solution of linear integral equations system using the Bernstein collocation method, Advances in Difference Equations, (2013), 1-15.

[23] H. Han, Q. Chen, J. Qiao, Research on an online Self-Organizing radial basis function neural network. Neural Comput and Applications. 19(2010) 667-676.

[24] Babolian. E, Shahsavaran. A, Numerical Solution of nonlinear Fredholm integral equations of the second kind using Haar-wavelet, Journal of Computational and Applied mathematics 225(2009) 87-95.

[25] Atkinson, K.E., The Numerical Solution of Integral equations of the Second kind, Cambridge, Cambridge, University Press (1997).

[26] Wazwaz, A.M., A First course in integral equations, World Scientific, Signapore (1997).

[27] Bidokhti, Sh, Malek, A., Solving initial boundary Value problems for systems of partial differential equations using neural networks and optimization techniques, journal of the Franklin Institute 34, (2009), 898-913.