DESIGN SUITABLE FEED FORWARD NEURAL NETWORK TO SOLVE TROESCH'S PROBLEM

Luma Naji Mohammed Tawfiq[1] and **Othman M. Salih**
Department of Mathematics, College of Education for Pure Science Ibn Al-Haitham, University of Baghdad, Baghdad, Iraq.
[1]Email: luma.n.m@ihcoedu.uobaghdad.edu.iq

ABSTRACT: *Purpose – The aim of this research is to find accurate solution for the Troesch's problem by using high performance technique based on parallel processing implementation.*
Design/methodology/approach – Feed forward neural network is designed to solve important type of differential equations that arises in many applied sciences and engineering applications. The suitable designed based on choosing suitable learning rate, transfer function, and training algorithm. The authors used back propagation with new implement of Levenberg - Marquardt training algorithm. Also, the authors depend new idea for choosing the weights. The effectiveness of the suggested design for the network is shown by using it for solving Troesch problem in many cases.
Findings – New idea for choosing the weights of the neural network, new implement of Levenberg - Marquardt training algorithm which assist to speeding the convergence and the implementation of the suggested design demonstrates the usefulness in finding exact solutions.
Practical implications – The authors have applied the suggested design of feed forward neural network to solve Troesch's problem which is an important equation in Magnetostatic Fields, Electrochemical, a Magnetic Field and others.
Originality/value – In this paper, main types of the differential equation have been solved by using parallel processing technique based on feed forward neural network.
The design has been accompanied with new implementation of training algorithm. This process constructs an efficient dealing to get the exact solutions of the non-linear problems which is easy to implement.
In addition to that, the accompanied regularization design with the used design proved its efficiency in handling many problems especially ill-posed problems.

KEYWORDS: Ordinary Differential Equation, Troesch's Problem, Feed Forward Neural Network, Back propagation Training Algorithms, Levenberg - Marquardt algorithm.

1. INTRODUCTION

Artificial Neural Networks (Ann) have achieved employment in many specialties: neurosciences, arithmetic, statistics, physical processes, information technology, and manufacturing. Greatest of the preceding workings in solving differential equations by using Ann is limited to the item of resolving the linear system of algebraic equations which product from the discretization of the domain. The reduce of the networks power function delivers the result to the system of equations [1]. Mohsen and Behnam [2] used modified Ann is called relax state to solve the Burger's equation in one dimensional partial differential equation. Hamid et al., [3] employed a numerical technique combines with Ann's for solving the Lane-Emden equations. Many researchers such [4-6] used multilayer perceptron and Radial basis function neural networks for solving differential equations of various types. Mall and Chakraverty [7] used Feed Forward Neural Network (FFNN) for solving ordinary differential equations (ODE) with regression based algorithm and discuss score with random initial weights for various number of neuron in hidden layers. Modjtaba et al., in [8] used Ann's has been expanded for solving the Stokes problem. In [9] Ladislav developed a new kind of Anns which is differential polynomial neural networks, it's implementation on principles, which are employed in the human brain learning. Yashtini and Malek [10] suggested a recurrent neural networks technique for solving a type of monotone variation inequalities problem. Vartziotis et al., [11] used Ann to solve system of ordinary and partial differential equations to supply health professionals with explanations for surgical planning and actual-time decision action. Ahmad Jafarian et al., [12] suggest combines the neural networks technique with the power series method to insert an effective iterative approach to product an approximate polynomial solution for particular kind of fractional Volterra integro-differential equation. Parand et al., [13] solved Lane-Emden equations by using an Unsupervised collective neural network. Sarkhosh et al., [14] Used the Chebyshev neural networks to get the numerical solution of specific modules of fractional integro-differential equation. Silvia and Robert [15] applied FFNN technique to presented algebraic approach for illustrate a multidimensional nonlinear function.

Nonlinear ordinary differential equations (ODEs) arise in an extensive variety of problems in applied science and engineering. Usually these non-linear problems are solved by using approximate analytical methods or numerical technique such as Homotopy perturbation method (HPM) [16], collocation method [17], Variational iteration method (VIM) [18], Adomian decomposition method (ADM) [19], Shooting method [20] etc.

The Troesch's problem is a type of ODE appear in an inspection of the restriction of a plasma column by corpuscular-radiation pressure [21], and furthermore in the model of gas porous electrodes [22, 23]. The problem was 1st modeled and resolved by Weibel [24]. Then resolved again by analytical closed type [25] or by using shooting method [26], and by using a Laplace remodel decomposition method [27].

The Troesch's problem has been widely studied and different methods have been suggested to get the solution. Such methods are Homotopy analysis method (HAM), HPM, ADM, B-Spline Collocation, Shooting method and Reproducing Kernel method (RKM) for more details see [20], [26-31].

In this paper, the authors design suitable FFNN with new implementation of training algorithm to obtain the approximate solution of the Troesch's problem with boundary condition, and then to illustrate the efficiency and accuracy of suggested design some example are solved with comparisons between the results and exact solution are introduced

2. ARTIFICIAL NEURAL NETWORKS

A neural network is an interconnected assemblage of simplified process elements is said to be neurons, whose functionality is inaccurately based on the human neurons. This neuron is arranging in the network as a layer connected between them by the inter component connection say weights, where its initial value can be choosing randomly. Then adjust by suitable training algorithm.

The first layer says input layer, then hidden layers and last layer say output layer. Each neuron in the hidden or output layers has transfer function or activation function which is bounded monotonically increasing, differentiable function, that is sigmoid function. The most popular transfer function in hidden layer is tanh. transfer function and linear (pure lin.) transfer function in the output layer [32, 33].

There are two main connection procedures: forward connections and feedback connections (recurrent), The Feedback neural network (FBNN) have connection from neurons of output layer feeding backwards to the input nodes of former layers, or to the similar layer, while the FFNN is linking forwards starting from input layer to hidden then output layer [34]. Here, we suggest FFNN for solving the problem. A general structure of Ann's is illustrated in Figure (1).

The training of Ann's can be classified into three distinct types. These are supervised, unsupervised, and reinforcement learning [35]. Here, supervised training type has been presented, based on back propagation training algorithm. The back-propagation rule is a generality of the least mean square rule that adjusts network weights to reduce the mean square error among the preferred and real outputs of the network. That is, FFNN trained to minimize the error up to an acceptable accuracy.

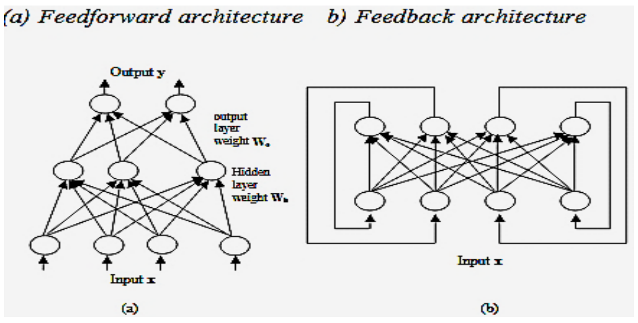

Figure (1): Structure of Ann's

3. ARCHITECTURES OF SUGGESTED DESIGN

The authors suggest fully connected three layers FFNN consist of five hidden neurons in the hidden layer with sigmoid transfer function such tansig and linear sigmoid for the output neuron (linsig.).

Let us consider the following represent the 2nd order differential equation:

$F(x, y(x), y'(x), y''(x)) = 0, x \in D$ (1)

where $D \subset R$, denotes the domain, and $y(x)$ is the solution to be computed. Here, G is the function which defines the structure of the problem. For solving a discretized domain D over finite set of points is considered. Thus, the problem changed into the system of equations as follows:

$F(x_i, y(x_i), y'(x_i), y''(x_i)) = 0, \quad x_i \in \bar{D}$ (2)

Let $y_t(x, w)$ denote the trail solution and then the problem can be formulated as:

$F(x_i, y_t(x_i, w), y_t'(x_i, w), y_t''(x_i, w)) = 0, x_i \in \bar{D}$ (3)

The error function with respect to each input data is written as:

$$\min_w \sum_{x_i \in \bar{D}} \left(F\left(x_i, y_t(x_i, w), y_t'(x_i, w), y_t''(x_i, w)\right)\right)^2 \quad (4)$$

The trail solution $y_t\,(X)$ of the problem by suggested FFNN can be computation from the following formula:

$$y_t(x,w) = \left[\frac{\omega A - \delta B}{\omega - \delta}\right] + \left[\frac{(B-A)}{\omega - \delta}\right]x + [(x-\delta)(x-\omega)]\bar{N}(x,w) \quad (5)$$

where; $\bar{N}(x, w)$ is the results of the suggested FFNN.

Now, rewrite equation (5) to concerting the problem

$$Error(w) = \sum_{k=1}^{m}\left[\left(\frac{d^2 y_t(x_k,w)}{dx^2}\right) - F\left(x_k, y_t(x_k,w), \left(\frac{dy_t(x_k,w)}{dx}\right)\right)\right]^2, \quad \delta \le x_k \le \omega \quad (6)$$

while:

$$y_t' = \left(\frac{B-A}{\omega-\delta}\right) + (x-\delta) + (x-\omega)\bar{N} + (x-\delta)(x-\omega)\bar{N}' \quad (7)$$

So,

$$y_t'' = 2\bar{N} + 2(x-\delta) + (x-\omega)\bar{N}' + (x-\delta)(x-\omega)\bar{N}'' \quad (8)$$

For checking the performance of the suggested designs, the calculating of the absolute error will be considered:

absolute error = $|y_t(x) - y_a(x)|$ (9)

4. BUILD TRAIN OF SUGGESTED DESIGN

The suggested FFNN initialized with randomly values of the parameters especially, the weights then adjust by suitable training algorithm. Here the Levenberg - Marquardt training algorithm (LM) and its MATLAB code is (trainlm) will be used with new implementation, the classical LM training update is as follow:

$w_{k+1} = w_k - \eta\,(J^T J + \mu I)^{-1} J_k\, E(w_k)$ (10)

where, η is the learning rate and choosing randomly in most of networks. Here, the authors suggest rule for estimate this value. The rule based on take value in weight space for each iteration of the weight update equation will be controlling the distance between w_{k+1} and w_k.

The training process starts with randomly initialize weights on all the connections in the FFNN. In step 2, distributing inputs feed forwarding. Therefore, each neuron in the hidden layer computes the summing for the inner product of its weights with the inputs. Take suitable transfer function for summing that value. Then its output is sent to the output layer as an input. So, the neurons in the output layer, sums its weighted inputs. Then its transfer function (purelin.) computes its output which is also the output of overall network. Hence, the error can be calculated. If the error is less than a specific value, the training stops and the weights are sent. Otherwise, the training proceeds to the next step. According to the equation (10) the correction laws of the weights between the hidden layer and the output layer are calculated.

The sensitivity term of the output is sent back to the hidden layer. Each hidden unit gets its input from the output layer, and then the sensitivity is computed for updating the weights in the hidden layer. Because there are 5 nodes in the hidden layer, the sensitivity term is a vector. Again, the correction laws (equation (10)) for the weights are calculated. Return to step 2, for next iteration and so on.

There are many types of costs associated with using LM training algorithm such derivatives, computation and storage. For this limitation, LM can fail to converge, or it can converge to a point that is not a minimum. To overcoming these disadvantages, we suggest the following improvement.

The improvement based on choosing the weights to guarantee that the Jacobian J is positive definite, to get this, we need to satisfy the following:

Let E(x) be an n-vector of functions of $x = (x_1, \dots, x_n)^T$. Assume that the Jacobian of E is Lipschitz continuous on an open convex set S with constant L. Then for any x, y∈ S,

$$\|E(y) - E(x) - \nabla E(x)^T (y - x)\| \le \frac{L}{2} \|y - x\|^2 \qquad (11)$$

Now, we illustrate the basic idea for new implementation for the Levenberg - Marquardt training algorithm, where the given data are divided into three sets of data: training, testing and validation set. Firstly, store pattern pairs (training data) so that when n-dimensional vector X from training set is presented as input, the FFNN recalls m-dimensional vector Y (output vector), but when Y is presented as input, the FFNN recalls X.

To develop the FFNN, we need to create a correlation matrix for each pattern pair we want to store. The correlation matrix is the matrix product of the input vector X, and the transpose of the output vector Y^T. The FFNN weight matrix is the sum of all correlation matrices, that is,

$$W = \sum_{i=1}^{n} X_i Y_i^T \qquad (12)$$

where n is the number of pattern pairs to be stored in the training set.

Now, to guarantee the convergence of new implementation for LM algorithm, we must satisfy the following condition (descent condition):

$$E(w_{k+1}) < E(w_k), \qquad (13)$$

This is possible if the search direction is a descent direction, that is, if E(w) = g(w), where g is the gradient of performance function.

We note that, the necessary condition for a minimizer the gradient of performance function at the optimal weights w∗ are equal to zero, i.e., g(w∗) = 0, which guarantee the convergence.

5.SOLVING TROESCH'S PROBLEM BY SUGGESTED NETWORK

Troesch's problem arises of the confinement of a plasma column by radiation pressure. In this section, the suggested design of FFNN will be implemented to solve Troesch's problem. Consider the following nonlinear Troesch's problem:

$y'' = \gamma \sinh(\gamma y); \quad 0 \le x \le 1$

subject to Dirichlet boundary conditions: y(0) = 0 , y(1) = 1

The analytic solution was considered in [22], [23], [36], by using other methods as follow:

$$y(x) = \left(\frac{2}{\gamma}\right) sinh^{-1}\left\{\left(\frac{y'(0)}{2}\right) sc\left(\gamma x \middle| 1 - \frac{1}{4(y'(0))^2}\right)\right\}$$

Such that:

$$y'(0) = 2(\sqrt{1-m})$$

where m is that the resolution of this equation:

$$\frac{sinh(\frac{\gamma}{2})}{\sqrt{1-m}} = sc(\gamma|m)$$

and;

$$sc(\gamma|m) = \sin(\emptyset)/\cos(\emptyset)$$

Such that $\emptyset, \gamma$ and m are associated over the integral:

$$\gamma = \int_0^{\emptyset} \frac{1}{\sqrt{1 - msin^2\theta}} d\theta$$

According to the equation (5), the suggested FFNN present solution for the problem as:

$y_t(x) = x + x(x-1).\bar{N}$

The results of suggested FFNN with LM train algorithm, its new implementation and its new implementation with suggested rule of η are given at different point in the domain in Table (1), where $\gamma = 0.5$ and the accuracy of the results are given in Table (2). Similarly, the results of the problem when γ = 1, 10 can gives in Table (3) and (5) respectively and the accuracy of that results can give in Table (4) and (6) respectively. To assess the performance of suggested training algorithm in comparison with others in: time of training, number of iterations (epoch), performance of training, and mean square error of regularization (Mse.reg.) for each value of γ will be evaluated as in Tables (7-9).

Table 1: Results of FFNN for Troesch Problem, when (γ= 0.5)

	analytic solution	results of FFNN		
x	$y_a(x)$	New Train(LM)& new η	New Train (LM)	Train (LM)
0	0	-2.220446049250313e-16	1.221245327087672e-13	8.030546116977177e-09
0.1	0.095176902000000	0.095175139198288	0.095173245757378	0.095176868643766
0.2	0.190633869100000	0.190633869100000	0.190633869099879	0.190633913685827
0.3	0.286653403000000	0.286653403000000	0.286653402999235	0.286653402997715
0.4	0.383522928800000	0.383522547631396	0.383520422227102	0.383522893845726

0.5	0.481537385400000	0.481537385400000	0.481533866102942	0.481537379265672
0.6	0.581001974900000	0.581002381646378	0.581000087157298	0.581002010366724
0.7	0.682235132600000	0.682235132600000	0.682235132599731	0.682235144861293
0.8	0.785571786700000	0.785571786700000	0.785571786701434	0.785571734361145
0.9	0.891366987500000	0.891369265944251	0.891366987500929	0.891367021147331
1	1	1	0.999999999999724	0.999999992469936

Table 2: Accuracy of the results for Troesch's Problem, when γ = 0.5

Error = | $y_t(x)$ - $y_a(x)$ |, where y_a is analytic solution & y_t given by		
New Train (LM) & new η	**New Train (LM)**	**Train (LM)**
2.220446049250313e-16	1.221245327087672e-13	8.030546116977177e-09
1.762801712310025e-06	3.656242621594141e-06	3.335623403877275e-08
2.775557561562891e-17	1.214306433183765e-13	4.458582694710778e-08
1.110223024625157e-16	7.646105970593453e-13	2.284505917771185e-12
3.811686036248041e-07	2.506572898186565e-06	3.495427386424055e-08
1.110223024625157e-16	3.519297058218740e-06	6.134327934503858e-09
4.067463783563596e-07	1.887742702044726e-06	3.546672344700852e-08
1.110223024625157e-16	2.693401057740630e-13	1.226129342501992e-08
2.220446049250313e-16	1.434186103210777e-12	5.233885469468902e-08
2.278444250602973e-06	9.289236047038685e-13	3.364733103250472e-08
0	2.764455331316640e-13	7.530064127792002e-09

Table 3: Results of FFNN for Troesch Problem, when (γ= 1)

	analytic solution	Results of FFNN $y_t(x)$		
x	ya(x)	New Train (LM) & new η	New Train (LM)	Train (LM)
0	0	-1.110223024625157e-16	3.723621411211298e-11	2.145591815327919e-09
0.1	0.081796996600000	0.081796996600000	0.081796996689098	0.081796985130997
0.2	0.164530870900000	0.164561561223236	0.164530870922984	0.164530904607417
0.3	0.249167360800000	0.249191414088678	0.249166899526918	0.249167313014770
0.4	0.336732209200000	0.336732209200000	0.336731772513958	0.336732237994166
0.5	0.428347161000000	0.428338569286573	0.428347161042117	0.428347179161660
0.6	0.525274029600000	0.525274029600000	0.525274029627832	0.525273980428399
0.7	0.628971143400000	0.628971143400000	0.628970590160913	0.628971187696739
0.8	0.741168378200000	0.741153563056388	0.741168378209879	0.741168356745979
0.9	0.863970020600000	0.863970020600000	0.863973133295373	0.863970026460652
1	1	1	1.000000000008383	0.999999999353309

Table 4: Accuracy of the results for Troesch's Problem, when γ = 1

Error = | $y_t(x)$ - $y_a(x)$ |, where y_a is analytic solution &y_t given by		
New Train (LM) & new η	New Train (LM)	Train(LM)
0	9.681144774731365e-14	9.949124594621495e-06
8.189792160412379e-17	3.637846758150350e-05	1.811043796381836e-05
6.476880911145344e-05	5.849265299548434e-14	8.716584308026689e-06
7.663632431025946e-05	3.874076623736067e-13	7.650600952155981e-07
5.448357568935505e-05	2.494911829215019e-13	1.903633290691520e-06
3.295974604355934e-17	2.736848680126060e-05	7.262222605167873e-07
9.367506770274758e-17	1.688759363077490e-04	1.774033088382254e-07
2.795214898675541e-04	5.047955415286159e-04	4.662853937686950e-08
6.938893903907228e-18	3.501574030728705e-13	1.897660294875037e-08
8.326672684688674e-17	7.757405828812125e-13	9.057067845708033e-09
2.220446049250313e-16	1.220801237877822e-12	2.423705347531779e-09

Table 5: Results of FFNN for Troesch Problem, when (γ= 10)

	analytic solution	results of FFNN $y_t(x)$		
x	$y_a(x)$	New Train (LM)& new η	New Train (LM)	Train (LM)
0	0	0	-9.681144774731365e-14	9.949124594621495e-06
0.1	7.630000000000000e-05	7.629999999991810e-05	3.992153241849650e-05	5.818956203618164e-05
0.2	1.299000000000000e-04	1.946688091114535e-04	1.298999999415074e-04	1.386165843080267e-04
0.3	3.589000000000000e-04	4.355363243102595e-04	3.589000003874077e-04	3.596650600952156e-04
0.4	9.779000000000000e-04	0.001032383575689	9.778999997505090e-04	9.759963667093086e-04
0.5	0.002659000000000	0.002659000000000	0.002686368486801	0.002659726222261
0.6	0.007228900000000	0.007228900000000	0.007397775936308	0.007228722596691
0.7	0.019664000000000	0.019943521489868	0.020168795541529	0.019664046628539
0.8	0.053730300000000	0.053730300000000	0.053730299999650	0.053730281023397
0.9	0.152114000000000	0.152114000000000	0.152113999999224	0.152114009057068
1	1	1.000000000000000	0.999999999998779	0.999999997576295

Table 6: Accuracy of the results for Troesch's Problem, when γ = 10

Error = | $y_t(x)$ - $y_a(x)$ |, where y_a is analytic solution &y_t given by		
New Train (LM) & new η	New Train (LM)	Train(LM)
1.110223024625157e-16	3.723621411211298e-11	2.145591815327919e-09
5.551115123125783e-17	8.909833981718407e-11	1.146900258097716e-08
3.069032323585463e-05	2.298425338942423e-11	3.370741694097568e-08
2.405328867841061e-05	4.612730821473843e-07	4.778522963433396e-08
1.110223024625157e-16	4.366860422155838e-07	2.879416560741532e-08
8.591713426819858e-06	4.211658799491147e-11	1.816165967616357e-08
0	2.783218100432805e-11	4.917160156825418e-08
0	5.532390874307680e-07	4.429673927663913e-08
1.481514361179048e-05	9.878764473114643e-12	2.145402089315240e-08
0	3.112695372564645e-06	5.860652185774029e-09
0	8.382627925129782e-12	6.466911450786483e-10

Table 7: The details of training of FFNN, when γ = 0.5

Training rule	**Performance**	**epochs**	**Times**	**Mse.reg.**
New Train (LM) & new η	3.36e-33	3117	0:00:23	7.4593e-009
New Train(LM)	3.49e-25	1671	0:00:19	2.3352e-008
Train(LM)	5.08e-10	2500	0:00:19	4.1541e-011

Table 8: The details of training of FFNN, when γ = 1

Training rule	**Performance**	**epochs**	**Times**	**Mse.reg.**
New Train (LM) & new η	1.95e-32	1715	0:00:12	7.0441e-013
New Train (LM)	5.26e-25	2946	0:00:34	2.9127e-012
Train(LM)	9.76e-15	10341	0:01:23	7.9862e-016

Table 9: The details of training of FFNN, when γ = 10

Training rule	**Performance**	**epochs**	**Times**	**Mse.reg.**
New Train (LM) & new η	3.96e-33	131	0:00:01	1.4840e-010
New Train (LM	4.22e-26	3839	0:00:43	8.5078e-013
Train(LM)	9.59e-15	11738	0:01:32	7.8462e-016

There are several previous studies have shown the importance to solve this equation such: The Adomian decomposition method (ADM) [19], the variational iteration method (VIM) [18], the modified homotopy perturbation method (MHPM) [30], a combination of the ADM with the reproducing kernel method (RKM) [28], and others. Table (10), represents a comparison between those solutions of Troesch's Problem when γ = 10.

It is worth mentioning that the main advantages of the suggested design of FFNN in comparison to the other methods are the fact that there is no need to construct a homotopy and solve the corresponding equations in HPM. No need for the large evaluating the Lagrange multiplier like the VIM. Without any restricted assumption for nonlinear terms like finding Adomian polynomials to deal with the nonlinear terms in ADM. Moreover, the FFNN provided good accuracy even for a few number of neurons and layers. Moreover, the suggested FFNN provided good accuracy even for γ = 0.5 and 1. Also, Figures (2-4) illustrate the efficiency of implemented suggested network wherein demonstrate its results in validation, training, and testing where γ = 0.5, 1 and 10 respectively.

Table 10: Comparison between the different methods for solving Troesch's problem, γ = 10

x	ADM-RKM	VIM	ADM	MHPM
0.1	0.0000576	0.1186109866	667081.1874	17.61750
0.2	0.0001902	0.4461962517	1333955.1189	33.69333
0.3	0.0005676	3.8003366781	1999860.1189	46.78583
0.4	0.0016654	79.89147273	2661970.7366	55.65333
0.5	0.0048331	1880.3539472	3310585.4201	59.35417
0.6	0.0137488	41642.365193	3914127.8659	57.34667
0.7	0.0374013	878764.64189	4374578.5342	49.58917
0.8	0.0936540	18064027.967	4406724.4178	36.64000
0.9	0.2189270	366613074.02	3290268.6374	19.75750

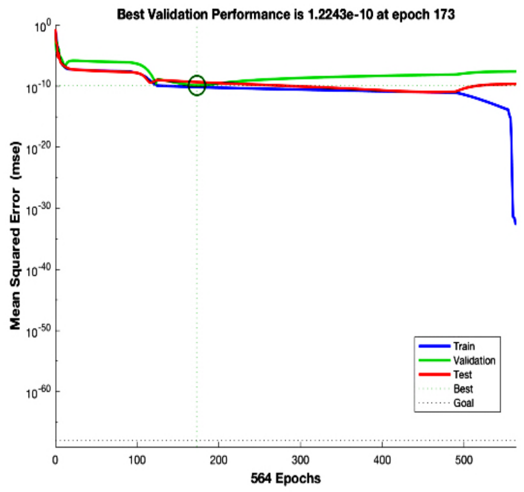

Figure 2: Accuracy of suggested network in validation, training, and testing where γ = 0.5

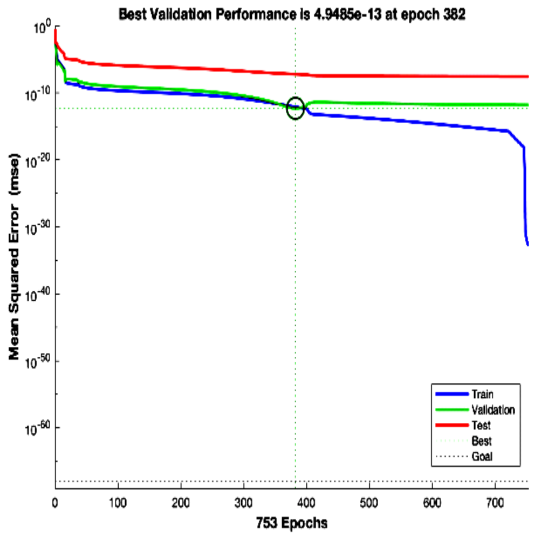

Figure 3: Accuracy of suggested network in validation, training, and testing where γ = 1

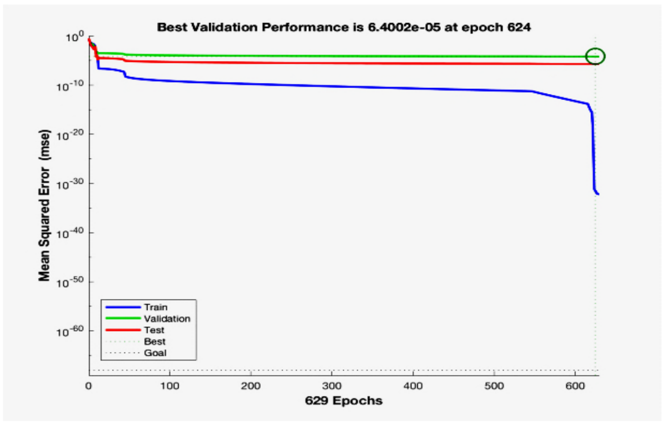

Figure 4: Accuracy of suggested network in validation, training, and testing where γ = 10

6. CONCLUSION

In this work, a feed forward neural network (FFNN) with new implementation of training algorithm is successfully applied to get an accurate solution for Troesch's Problem. This technique appears to be very promising to get the solution. its characterized by applying without any restrictive assumptions for nonlinear terms, discretization or perturbation techniques. The numerical results showed that the FFNN accurate, reliable and better than other methods such HPM, ADM and VIM.

REFERENCES

[1] Hoda, S. A. and Nagla, H. A., "Neural network methods for mixed boundary value problems," *International Journal of Nonlinear Science*, **11**(3): 312-316(2011)

[2] Hayati, M. and Karami, B., "Feedforward neural network for solving partial differential equations," *Journal of Applied Sciences*, **7**(19): 2812-2817(2007)

[3] Jalab, H. A., Ibrahim, R. W., Murad, S. A., Melhum, A. I. and Hadid, S. B., "Numerical solution of Lane-Emden equation using neural network," *In AIP Conference Proceedings*, **1482**(1): 414-418 AIP (2012, September)

[4] Kumar, M. and Yadav, N., "Multilayer perceptrons and radial basis function neural network methods for the solution of differential equations: a survey," *Computers & Mathematics with Applications*, **62**(10): 3796-3811(2011)

[5] Tawfiq, L. N. M., "On Training of Artificial Neural Networks," *Al-Fatih journal*, **1**(23): 130-139(2005)

[6] Tawfiq, L. N. M., "Design and training artificial neural networks for solving differential equations," [Ph. D. thesis]. *University of Baghdad, College of Education-Ibn-Al-Haitham, Baghdad, Iraq*(2004)

[7] Mall, S. and Chakraverty, S., "Comparison of artificial neural network architecture in solving ordinary differential equations," *Advances in Artificial Neural Systems*, **2013**: 12(2013)

[8] Baymani, M., Kerayechian, A. and Effati, S., "Artificial neural networks approach for solving stokes problem," *Applied Mathematics*, **1**(04): 288(2010)

[9] Zjavka, L., "Construction and adjustment of differential polynomial neural network," *Journal of Engineering and Computer Innovations*, **2**(3): 40-50(2011)

[10] Yashtini, M. and Malek, A., "Solving complement-arity and variational inequalities problems using neural networks," *Applied Mathematics and Comput-ation*, **190**(1): 216-230(2007)

[11] Vartziotis, F., Fotiadis, D. I., Likas, A. and Lagaris, I. E., "A portable decision making tool for health professionals based on neural networks," *Health Informatics Journal*, **9**(4): 273-282(2003)

[12] Jafarian, A., Rostami, F., Golmankhaneh, A. K. and Baleanu, D., "Using ANNs approach for solving fractional order Volterra integro-differential equations," *International Journal of Computational Intelligence Systems*, **10**(1): 470-480(2017)

[13] Parand, K., Roozbahani, Z. and Bayat Babolghani, F., "Solving nonlinear Lane-Emden type equations with unsupervised combined artificial neural networks," *International Journal of Industrial Mathematics*, **5**(4): 355-366(2013)

[14] Chaharborj, S. S., Chaharborj, S. S. and Mahmoudi, Y., "Study of fractional order integro-differential equations by using Chebyshev Neural Network," *Journal of Mathematics and Statistics*, **13**: 1-13(2017)

[15] Ferrari, S. and Stengel, R. F., "Smooth function approximation using neural networks," *IEEE Transactions on Neural Networks*, **16**(1): 24-38(2005)

[16] Mirmoradia, S. H., Hosseinpoura, I., Ghanbarpour, S. and Barari, A., "Application of an approximate analytical method to nonlinear Troesch's problem," *Applied Mathematical Sciences*, **3**(32): 1579-1585(2009)

[17] Salih, H., Tawfiq, L. N. M., Yahya, Z. R. and Zin, S. M., "Solving Modified Regularized Long Wave Equation Using Collocation Method," *Journal of Physics: Conference Series*, **1003**(1): 012062, IOP Publishing (2018)

[18] Momani, S., Abuasad, S. and Odibat, Z., "Variational iteration method for solving nonlinear boundary value problems," *Applied Mathematics and Computation*, **183**(2): 1351-1358(2006)

[19] Deeba, E., Khuri, S. A. and Xie, S., "An algorithm for solving boundary value problems," *Journal of Computational Physics*, **159**(2): 125-138(2000)

[20] Chang, S.H., "Numerical solution of Troesch's problem by simple shooting method," *Applied Mathematics and Computation*, **216**(11): 3303-3306(2010)

[21] Weibel, E.S., "On the confinement of a plasma by magnetostatic fields," *The Physics of Fluids*, **2**(1): 52-56(1959)

[22] Gidaspow, D. and Baker, B. S., "A model for discharge of storage batteries," *Journal of the Electrochemical Society*, **120**(8): 1005-1010(1973)

[23] Markin, V. S., Chernenko, A. A., Chizmadehev, Y. A. and Chirkov, Y. G., "Aspects of the theory of gas porous electrodes," *Fuel Cells: Their Electrochemi-cal Kinetics*, 22-33(1966)

[24] Tawfiq, L. N. M. and Jabber, A.K., "Steady State Radial Flow in Anisotropic and Homogenous in Confined

Aquifers," *Journal of Physics: Conference Series*, **1003**(012056):1-12, IOP Publishing (2018).

[25] Roberts, S. M. and Shipman, J. S., "On the closed form solution of Troesch's problem," *Journal of Computational Physics*, **21**(3): 291-304(1976)

[26] Troesch, B. A., "A simple approach to a sensitive two-point boundary value problem," *Journal of Computational Physics*, **21**(3): 279-290(1976)

[27] Khuri, S. A., "A numerical algorithm for solving Troesch's problem," *International Journal of Computer Mathematics*, **80**(4): 493-498(2003)

[28] Geng, F. and Cui, M., "A novel method for nonlinear two-point boundary value problems: combination of ADM and RKM," *Applied Mathematics and Computation*, **217**(9): 4676-4681(2011)

[29] Feng, X., Mei, L. and He, G., "An efficient algorithm for solving Troesch's problem," *Applied Mathematics and Computation*, **189**(1): 500-507(2007)

[30] Mohyud-Din, S. T., "Solution of Troesch's problem using He's polynomials," *Rev. Un. Mat. Argentina*, **52**: 143-148(2011)

[31] Khuri, S. A. and Sayfy, A., "Troesch's problem: A B-spline collocation approach," *Mathematical and Computer Modelling*, **54**(9-10): 1907-1918(2011)

[32] Oonsivilai, A. and Saichoomdee, S., "Distance transmission line protection based on radial basis function neural network," *World Academy of Science, Engineering and Technology*, **60**(2009)

[33] Tawfiq, L. N. and Oraibi, Y. A., "Fast Training Algorithms for Feed Forward Neural Networks," *Ibn Al-Haitham Journal for Pure and Applied Science*, **26**(1): 275-280(2017)

[34] Tawfiq, L. N. and Hussein, A. A., "Neural Network and Mathematical Modeling of a Spherical Body of Gas," *International Journal of Modern Computer Science and Engineering*, **2**(1): 1-11(2013)

[35] Abramowitz, M. and Stegun, I. A., "*Handbook of mathematical functions: with formulas, graphs, and mathematical tables,* **55**: 886, New York: Dover publications (1972)

[36] Tawfiq, L. N. M. and Hassan, M. A., "Estimate the Effect of Rainwaters in Contaminated Soil by Using Simulink Technique," In *Journal of Physics: Conference Series*, **1003**(012057):1-7, (2018)