View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by International Institute for Science, Technology and Education (IISTE): E-Journals

Mathematical Theory and Modeling
ISSN 2224-5804 (Paper) ISSN 2225-0522 (Online)
Vol.5, No.6, 2015
www.iiste.org

Modified Artificial Neural Networks For Solving Fuzzy Differential Equations

Eman A. Hussian[1] *Mazin H. Suhhiem*[2*]

1. Dep. of Mathematics, College of Sciences, AL-Mustansiriyah University, Baghdad,Iraq.

2. Dep. of statistics, College of Adm. and Econ., University of sumar, Alrefiey,Iraq.

Abstract

In this paper, we introduce a novel approach based on modified neural networks to solve fuzzy differential equations. Using modified neural network makes that training points should be selected over an open interval without training the network in the range of first and end points. Therefore, the calculating volume involving computational error is reduced. In fact, the training points depending on the distance selected for training neural network are converted to similar points in the open interval by using a new approach, then the network is trained in these similar areas. In comparison with existing similar neural networks proposed model provides solutions with high accuracy. The proposed method is illustrated by three numerical examples.

Keywords: Fuzzy differential equation, Modified neural network,

Feed-forward neural network, BFGS Teqnique, Hyperbolic tangent function.

1.Introduction

Nowadays, Fuzzy differential equations is a popular topic studied by many researchers since it is utilized widely for the purpose of modeling problems in science and engineering. Most of the practical problems require the solution of a fuzzy differential equation (FDE) which satisfies fuzzy initial conditions(FIC) or fuzzy boundary conditions(FBC).

The theory of the fuzzy differential equations was first formulated by Kaleva(1987) and Seikkala(1987). Kaleva had formulated FDEs in terms of Hukuhara derivative (H-derivative). Buckley and feuring (2000) have given a very general formulation of a first-order fuzzy initial value problem. They first find crisp solution, make it fuzzy then check if it satisfies the FDE .

In the following, we have mentioned some Numerical solutions which have proposed by other scientists. Abbasbandy and Allahviranloo have solved fuzzy differential equations by Runge-Kutta and Taylor methods (Abbasbandy & Allahviranloo 2002). Also, Allahviranloo and Ahmady solved fuzzy differential equations by using predictor-corrector and transformation methods (Allahviranloo,Ahmady & Ahmady 2007 , Allahviranloo,Ahmady & Ahmady 2008). Ghazanfari and Shakerami developed Runge-Kutta like formula of order 4 for solving fuzzy differential equations(Ghazanfari & Shakerami 2011), and other researchers (Ahmadi & Kiani 2011 , Corveleyn & Vandewalle 2011, Duraisamy & Usha 2010, Guo,Shang&Lu 2013, Jayakumar,Maheskumar& Kanagarajan 2012, Orouji,Parandin,Abasabadi & Hosseinpour 2014, Parimala,Rajarajeswari & Nirmala2014,Rostami,Kianpour & Bashardoust 2011, Saikia 2011, Tapaswini & Chakraverty 2014, Tapaswini & Chakraverty 2014). Effati and pakdaman (2010) used artificial neural network for solving fuzzy differential equations, they used for the first time the usual artificial neural network (UANN) to approximate fuzzy initial value problems. Mosleh and Otadi (2012) used UANN for solving fuzzy Fredholm integro-differential equations. Ezadi, parandin and Ghomashi (2013) used UANN based on semi-Taylor to solve first order FDEs.

In this paper, we will use modified artificial neural network (MANN) method to find an approximated solution of FDEs. This method can result in improved numerical methods for solving fuzzy initial/boundary conditions. We will illustrate this method by solving three examples.

2. Artificial Neural Network

An Artificial neural network (ANN) is a simplified mathematical model of the human brain, it can be implemented by both electric elements and computer software. It is a parallel distributed processor with large

numbers of connections, it is an information processing system that has certain performance characters in common with biological neural networks. ANN have been developed as generalizations of mathematical models of human cognition or neural biology, based on the assumptions:

1. lnformation processing occurs at many simple elements called neurons that is fundamental the operation of ANN's.

2. Signals are passed between neurons over connection links.

3. Each connection link has an associated weight which, in a typical neural net, multiplies the signal transmitted.

4. Each neuron applies an activation function (usually nonlinear) to its net input (sum of weighted input signals) to determine its output signal.

There are two main connection formulas (types):feedback(recurrent) and feed-forward connections.Feedback is one type of connection where the output of one layer routes back to the input of a previous layer , or to the same layer.Feed-forward neural network(FFNN) does not have a connection back from the output to the input neurons(Fig.1).There are many different training algorithms, but the most often used training algorithm is the back propagation(BP)rule.ANN is trained to map a set of input data by iterative adjusment of the weights.Information from inputs is feedforward through the network to optimize the wieghts between neurons. Optimization of the wieghts is made by backward propagation of the error during training phase.The ANN reads the input and output values in the training data set and changes the value of the wieghted links to reduce the difference between the predicted and target(observed)values.The error in prediction is minimized across many training cycles(iteration or epoch) until network reaches specified level of accuracy.A complete round of forward backward passes and wieght adjusments using all input output pairs in the data set is called an epoch or iteration. In order to perform a supervised training we need a way of evaluating the ANN output error between the actual and the expected outputs .A popular measure is the mean squared error (MSE) or root mean squared error(RMSE) (Tawfiq 2004, Tawfiq & Al-Abrahemee 2014).In this paper we will use BFGS Quasi-Newton to update the weights and biases(Ezadi,Parandin & Ghomashi 2013).

3. Illustration of The Method

3.1. First Order Fuzzy Differential Equation

A fuzzy differential equation of the first order is in the form:

$$y'(x) = f(x, y(x)) \quad , \text{with} \quad x \in [a, b] \tag{1}$$

and the fuzzy initial condition $y(a) = y_0$, where y is a fuzzy function of x , $f(x, y(x))$ is a fuzzy function of the crisp variable x and the fuzzy variable y , y' is the fuzzy derivative of y and y_0 is a fuzzy number. It is clear that the fuzzy function $f(x, y)$ is the mapping $f : R \times E^1 \longrightarrow E^1$.

Now it is possible to replace (1) by the following equivalent system(Effati & Pakdaman 2010):

$$\begin{aligned} \underline{y}'(x) &= \underline{f}(x, y) = F\left(x, \underline{y}, \overline{y}\right) \quad , \underline{y}(a) = \underline{y}_0 \\ \overline{y}'(x) &= \overline{f}(x, y) = G\left(x, \underline{y}, \overline{y}\right) \quad , \overline{y}(a) = \overline{y}_0 \end{aligned} \tag{2}$$

$$\begin{aligned} F\left(x, \underline{y}, \overline{y}\right) &= \text{Min}\left\{f(x, u) : u \in \left[\underline{y}, \overline{y}\right]\right\} \\ G\left(x, \underline{y}, \overline{y}\right) &= \text{Max}\left\{f(x, u) : u \in \left[\underline{y}, \overline{y}\right]\right\} \end{aligned} \tag{3}$$

The parametric form of system (2) is given by:

$$\begin{aligned} \underline{y}'(x, r) &= F\left[x, \underline{y}(x, r), \overline{y}(x, r)\right] \quad , \underline{y}(a, r) = \underline{y}_0(r) \\ \overline{y}'(x, r) &= G\left[x, \underline{y}(x, r), \overline{y}(x, r)\right] \quad , \overline{y}(a, r) = \overline{y}_0(r) \end{aligned} \tag{4}$$

Where $x \in [a, b]$ and $r \in [0, 1]$. Now with a discretization of the interval $[a, b]$, a set of points x_i, $i = 1,2,3, \ldots g$ are obtained. Thus for an arbitrary $x_i \in [a, b]$, the system (4) can be rewritten as:

$$\underline{y}'(x_i, r) - F\left[x_i, \underline{y}(x_i, r), \overline{y}(x_i, r)\right] = 0$$

$$\overline{y}'(x_i, r) - G\left[x_i, \underline{y}(x_i, r), \overline{y}(x_i, r)\right] = 0 \qquad (5)$$

With the initial conditions:

$$\underline{y}(a, r) = \underline{y}_0(r), \ \overline{y}(a, r) = \overline{y}_0(r), \ r \in [0, 1].$$

In this subsection (and then in this paper), the function approximation capabilities of feed-forward neural networks is used by expressing the trial solutions for the system (4) as the sum of two terms (see eq. 7). The first term satisfies the initial conditions (boundary conditions) and contains no adjustable parameters. The second term involves a feed-forward neural network to be trained so as to satisfy the differential equations. Since it is known that a multilayer perceptron with one hidden layer can approximate any function to arbitrary accuracy[9], the multilayer perceptron is used as the type of the network architecture.

If $\underline{y}_t(x, r, \underline{p})$ is a trial solution for the first equation in system (4) and $\overline{y}_t(x, r, \overline{p})$ is a trial solution for the second equation in system (4) where $\underline{p}$ and $\overline{p}$ are adjustable parameters.

Indeed, $\underline{y}_t(x, r, \underline{p})$ and $\overline{y}_t(x, r, \overline{p})$ are approximation of $\underline{y}(x, r)$ and $\overline{y}(x, r)$ respectively, then a discretized version of the system (4) can be converted to the following optimization problem (Effati & Pakdaman 2010,Ezadi,Parandin& Ghomashi 2013, Mosleh & Otadi 2012):

$$\min_{\vec{p}} \sum_{i=1}^{g} \left(\begin{array}{l} \left(\underline{y}'_t(x_i, r, \underline{p}) - F\left[x_i, \underline{y}_t(x_i, r, \underline{p}), \overline{y}_t(x_i, r, \overline{p})\right]\right)^2 \\ + \left(\overline{y}'_t(x_i, r, \overline{p}) - G\left[x_i, \underline{y}_t(x_i, r, \underline{p}), \overline{y}_t(x_i, r, \overline{p})\right]\right)^2 \end{array} \right) \qquad (6)$$

$\left(\text{Here } \vec{p} = (\underline{p}, \overline{p}) \text{contains all adjustable parameters}\right)$ subject to the initial conditions: $\underline{y}_t(a, r, \underline{p}) = \underline{y}_0(r)$, $\overline{y}_t(a, r, \overline{p}) = \overline{y}_0(r)$.

Each trial solution $\underline{y}_t(x, r, \underline{p})$ and $\overline{y}_t(x, r, \overline{p})$ employs one feed-forward neural network (see Fig.1) for which the corresponding networks are denoted by $\underline{N}(x, r, \underline{p})$ and $\overline{N}(x, r, \overline{p})$ with adjustable parameters $\underline{p}$ and $\overline{p}$ respectively. The trial solutions $\underline{y}_t$ and $\overline{y}_t$ should satisfy the initial conditions, and the networks must be trained to satisfy the differential equations. Thus $\underline{y}_t$ and $\overline{y}_t$ can be chosen as follows:

$$\underline{y}_t(x, r, \underline{p}) = \underline{y}(a, r) + (x - a)\, \underline{N}(x, r, \underline{p})$$

$$\overline{y}_t(x, r, \overline{p}) = \overline{y}(a, r) + (x - a)\, \overline{N}(x, r, \overline{p}) \qquad (7)$$

Where $\underline{N}(x, r, \underline{p})$ and $\overline{N}(x, r, \overline{p})$ are single-output feed-forward neural network with adjustable parameters $\underline{p}$ and $\overline{p}$ respectively. Here x and r are the network inputs (Fig.1). It is easy to see that in (7), $\underline{y}_t$ and $\overline{y}_t$ satisfy the initial conditions. Thus the error function that must be minimized over all adjustable neural network parameters will be:

$$E = \sum_i \left(\begin{array}{l} \left[\frac{\partial\, \underline{y}_t(x_i, r, \underline{p})}{\partial x} - F\left(x_i, \underline{y}_t(x_i, r, \underline{p})\right)\right]^2 + \\ \left[\frac{\partial\, \overline{y}_t(x_i, r, \overline{p})}{\partial x} - G\left(x_i, \overline{y}_t(x_i, r, \overline{p})\right)\right]^2 \end{array} \right) \qquad (8)$$

Where x_i ś are points in $[a, b]$.

3.2. Second Order Fuzzy Differential Equation

Now, we consider the second order fuzzy differential equation:

$$y''(x) = f(x, y(x), y'(x)), \quad x \in [a, b] \qquad (9)$$

with the fuzzy conditions: $y(a) = A_1$, $y'(a) = A_2$, such that the functions: $y : R \longrightarrow E^1$ and $f : R \times E^1 \times E^1 \longrightarrow E^1$

where y is a function with fuzzy derivative y′, also A_1 and A_2 are fuzzy numbers in E^1 with r-level sets: $[A_1]_r = [\underline{A}_1, \overline{A}_1]$ and $[A_2]_r = [\underline{A}_2, \overline{A}_2]$.

The same procedure in the subsection (3.1) can be applied on (9) , thus the trial solutions $\underline{y}_t$ and $\overline{y}_t$ can be chosen as follows:

$$\underline{y}_t(x, r, \underline{p}) = \underline{A}_1 + \underline{A}_2 (x - a) + (x - a)^2 \underline{N}(x, r, \underline{p})$$

$$\overline{y}_t(x, r, \overline{p}) = \overline{A}_1 + \overline{A}_2 (x - a) + (x - a)^2 \overline{N}(x, r, p) \qquad (10)$$

For the two point fuzzy boundary conditions:

$y(a) = A$ and $y(b) = B$, where A and B are fuzzy numbers in E^1 with r-level sets: $[A]_r = [\underline{A}, \overline{A}]$ and $[B]_r = [\underline{B}, \overline{B}]$.

The trial solutions $\underline{y}_t$ and $\overline{y}_t$ can be chosen as follows:

$$\underline{y}_t(x, r, \underline{p}) = \underline{A}\,\frac{b - x}{b - a} + \underline{B}\,\frac{x - a}{b - a} + (x - a)(x - b)\,\underline{N}(x, r, \underline{p})$$

$$\overline{y}_t(x, r, \overline{p}) = \overline{A}\,\frac{b - x}{b - a} + \overline{B}\,\frac{x - a}{b - a} + (x - a)(x - b)\,\overline{N}(x, r, \overline{p}) \qquad (11)$$

In the above two cases of the second-order fuzzy differential equation, the corresponding error function that must be minimized over all adjustable neural network parameters (weights and biases) will be:

$$E = \sum_i \left(\left[\frac{\partial^2 \underline{y}_t(x_i, r, \underline{p})}{\partial x^2} - F\left(x_i, \underline{y}_t(x_i, r, \underline{p}), \frac{\partial \underline{y}_t(x_i, r, \underline{p})}{\partial x}\right)\right]^2 + \left[\frac{\partial^2 \overline{y}_t(x_i, r, \overline{p})}{\partial x^2} - G\left(x_i, \overline{y}_t(x_i, r, \overline{p}), \frac{\partial \overline{y}_t(x_i, r, \overline{p})}{\partial x}\right)\right]^2 \right) \qquad (12)$$

Where x_i ś are points in [a , b] .

● For the higher (third and more) order fuzzy differential equation we apply the same procedure in the subsections (3.1) and (3.2) to find the trial solutions $\underline{y}_t$ and $\overline{y}_t$ and the error function E that must be minimized.

For solving fuzzy differential equations which described in the subsections (3.1) and (3.2), we will use two artificial neural networks, each network is of dimension $2 \times m \times 1$: two input units x and r, one hidden layer with m units and one linear output unit.

For every entries x and r the input neurons makes no changes in its input, so the inputs to the hidden neurons are:

$$\underline{net}_j = x\,\underline{w}_{j1} + \underline{w}_{j2}\,r + \underline{b}_j \ , \ \overline{net}_j = x\,\overline{w}_{j1} + \overline{w}_{j2}\,r + \overline{b}_j \ , j=1,2,\ldots,m \qquad (13)$$

$\underline{w}_{j1}$ and $\underline{w}_{j2}$ are the weight parameters from the input layer to the *j*th unit in the hidden layer in the first network in the Fig. (1) , $\overline{w}_{j1}$ and $\overline{w}_{j2}$ are the weight parameters from the input layer to the *j*th unit in the hidden layer in the second network in the Fig. (1) , $\underline{b}_j$ and $\overline{b}_j$ are the *j*th weight biases for the *j*th units in the hidden layers in the first and second network in Fig (1) respectively. The outputs in the hidden neurons are:

$$\underline{z}_j = s\left(\underline{net}_j\right) = s\left(x\,\underline{w}_{j1} + r\,\underline{w}_{j2} + \underline{b}_j\right)$$

$$\overline{z}_j = s\left(\overline{net}_j\right) = s\left(x\,\overline{w}_{j1} + r\,\overline{w}_{j2} + \overline{b}_j\right) \qquad (14)$$

The output neurons makes no changes in its inputs, so the inputs to the output neurons are equal to outputs:

$$\underline{N}(x , r, \underline{p}) = \sum_{j=1}^{m} \underline{v}_j \, \underline{z}_j = \sum_{j=1}^{m} \underline{v}_j \, s\left(x \, \underline{w}_{j1} + r \, \underline{w}_{j2} + \underline{b}_j\right)$$

$$\overline{N}(x , r, \overline{p}) = \sum_{j=1}^{m} \overline{v}_j \, \overline{z}_j = \sum_{i=1}^{m} \overline{v}_j \, s\left(x \, \overline{w}_{j1} + r \, \overline{w}_{j2} + \overline{b}_j\right) \qquad (15)$$

Where $\underline{v}_j$ and $\overline{v}_j$ are the weight parameters from the jth units in the hidden layers to the output layer in first and second network in Fig. (1).

3.3. Fuzzy Partial Differential Equation

We treat here two-dimensional problems only. However, it is straightforward to extend the method to more dimensions. For example, consider the two-dimensional fuzzy Poisson equation:

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = f(x , y) , \; x , y \in [a , b] \qquad (16)$$

With the Dirichlet fuzzy boundary conditions (for $x , y \in [0,1]$):

$$U(0, y) = f_0(y) , \; U(1, y) = f_1(y) , \; U(x , 0) = g_0(x) \text{ and } U(x , 1) = g_1(x).$$

Where: $f(x , y)$, $f_0(y)$, $f_1(y)$, $g_0(x)$ and $g_1(x)$ are fuzzy numbers or fuzzy functions with r-level sets (parametric form) :

$$[f_0(y)]_r = \left[\underline{f}_0(y) , \; \overline{f}_0(y)\right] , \; [f_1(y)]_r = \left[\underline{f}_1(y) , \; \overline{f}_1(y)\right]$$

$$[g_0(x)]_r = \left[\underline{g}_0(x) , \; \overline{g}_0(x)\right] , \; [g_1(x)]_r = \left[\underline{g}_1(x) , \; \overline{g}_1(x)\right]$$

In this subsection , we apply the same procedure in the subsections (3.1) and (3.2) to find the trial solutions $\underline{U}_t$ and $\overline{U}_t$ and the error functions. Thus the trial solutions can be chosen as follows:

$$\underline{U}_t(x , y, r, \underline{p}) = \underline{A}(x , y) + x \, y \, (1 - x) \, (1 - y) \, \underline{N}(x , y, r, \underline{p})$$

$$\overline{U}_t(x , y, r, \overline{p}) = \overline{A}(x , y) + x \, y \, (1 - x) \, (1 - y) \, \overline{N}(x , y, r, \overline{p}) \qquad (17)$$

Where $\underline{A}(x , y)$ and $\overline{A}(x , y)$ are chosen so as to satisfy the boundary conditions, namely:

$$\underline{A}(x , y) = (1 - x) \, \underline{f}_0(y) + x \, \underline{f}_1(y) + (1 - y)$$

$$\left[\underline{g}_0(x) - \left[(1 - x) \, \underline{g}_0(0) + x \, \underline{g}_0(1)\right]\right] + y \left[\underline{g}_1(x) - \left[(1 - x) \, \underline{g}_1(0) + x \, \underline{g}_1(1)\right]\right]$$

$$\overline{A}(x , y) = (1 - x) \, \overline{f}_0(y) + x \, \overline{f}_1(y) + (1 - y)$$

$$\left[\overline{g}_0(x) - \left[(1 - x) \, \overline{g}_0(0) + x \, \overline{g}_0(1)\right]\right] + y \left[\overline{g}_1(x) - \left[(1 - x) \, \overline{g}_1(0) + x \, \overline{g}_1(1)\right]\right] \qquad (18)$$

The corresponding error function that must be minimized over all adjustable neural network parameters will be:

$$E = \sum_i \left(\left[\frac{\partial^2 \underline{U}_t(x_i , y_i, r, \underline{p})}{\partial x^2} + \frac{\partial^2 \underline{U}_t(x_i , y_i, r, \underline{p})}{\partial y^2} - F(x_i , y_i)\right]^2 + \left[\frac{\partial^2 \overline{U}_t(x_i , y_i, r, \overline{p})}{\partial x^2} + \frac{\partial^2 \overline{U}_t(x_i , y_i, r, \overline{p})}{\partial y^2} - G(x_i , y_i)\right]^2 \right) \qquad (19)$$

Where (x_i , y_i) are points in the domain $[0 ,1] \times [0 ,1]$.

For solving fuzzy partial differential equations which described in this subsection, we will use two artificial neural network, each network is of dimension $3 \times m \times 1$: three input units, one hidden layer with m units and one linear output unit.

For every entries x , y and r the input neurons makes no changes in its inputs, so the inputs to the hidden neurons are:

$$\underline{net}_j = x\,\underline{w}_{j1} + y\,\underline{w}_{j2} + r\,\underline{w}_{j3} + \underline{b}_j$$

$$\overline{net}_j = x\,\overline{w}_{j1} + y\,\overline{w}_{j2} + r\,\overline{w}_{j3} + \overline{b}_j \qquad (20)$$

Where $j = 1,2,\dots\dots\dots m$, $\underline{w}_{j1}$, $\underline{w}_{j2}$ and $\underline{w}_{j3}$ are the weight parameters from the input layer to the *j*th unit in the hidden layer in the first network in the Fig. (2) , $\overline{w}_{j1}$, $\overline{w}_{j2}$ and $\overline{w}_{j3}$ are the weight parameters from the input layer to the *j*th unit in the hidden layer in the second network in the Fig. (2) , $\underline{b}_j$ and $\overline{b}_j$ are the *j*th weight biases for the *j*th units in the hidden layers in the first and second network in Fig (2) respectively. The outputs in the hidden neurons are:

$$\underline{z}_j = s\left(\underline{net}_j\right) = s\left(x\,\underline{w}_{j1} + y\,\underline{w}_{j2} + r\,\underline{w}_{j3} + \underline{b}_j\right)$$

$$\overline{z}_j = s\left(\overline{net}_j\right) = s\left(x\,\overline{w}_{j1} + y\,\overline{w}_{j2} + r\,\overline{w}_{j3} + \overline{b}_j\right) \qquad (21)$$

The output neurons makes no changes in its inputs, so the inputs to the output neurons are equal to outputs:

$$\underline{N}(x , y, r, \underline{p}) = \sum_{j=1}^{m} \underline{v}_j\,\underline{z}_j = \sum_{j=1}^{m} \underline{v}_j\, s\left(x\,\underline{w}_{j1} + y\,\underline{w}_{j2} + r\,\underline{w}_{j3} + \underline{b}_j\right)$$

$$\overline{N}(x , y, r, \overline{p}) = \sum_{j=1}^{m} \overline{v}_j\,\overline{z}_j = \sum_{j=1}^{m} \overline{v}_j\, s\left(x\,\overline{w}_{j1} + y\,\overline{w}_{j2} + r\,\overline{w}_{j3} + \overline{b}_j\right) \qquad (22)$$

Where $\underline{v}_j$ and $\overline{v}_j$ are the weight parameters from the *j*th unit in the hidden layer to the output layer in the first and second network in Fig.(2)

4. Proposed Method

In this section we will introduce a novel method to modify the artificial neural network ANN . This new method based on replaced every x in the input vector (training set) $\vec{x} = (x_1 , x_2 , \dots\dots , x_n)$, $x_j \in [a, b]$ by a polynomial of degree one. Ezadi and parandin(2013) used the function: $Q(x) = \epsilon\,(x + 1)$ with $\epsilon \in (0,1)$

Then the input vector will be: $(Q(x_1) , Q(x_2) , \dots\dots\ Q(x_n))$, $Q(x_j) \in (a , b)$ In this paper, we named this proposed method modified artificial neural network (MANN). Using modified artificial neural network makes that training points should be selected over the open interval (a , b) without training the neural network in the range of first and end points. Therefore, the calculating volume involving computational error is reduced. In fact, the training points depending on the distance $[a , b]$ selected for training neural network are converted to similar points in the open interval (a , b) by using a new approach, then the network is trained in these similar areas.

5. Numerical Examples

In this section we report on the solution of a number of model problems. In all cases we used a multilayer perceptron having one hidden layer with ten hidden units and one linear output unit. The activation function of each hidden unit is hyperbolic tangent function $s(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$. For each test problem, the exact analytical solutions $\underline{y}_a(x , r)$ and $\overline{y}_a(x , r)$ were known in advance. Therefore, we test the accuracy of the obtained solutions by computing the deviation:

$$\overline{e}(x , r) = \left|\overline{y}_a(x , r) - \overline{y}_t(x , r)\right| , \underline{e}(x , r) = \left|\underline{y}_a(x , r) - \underline{y}_t(x , r)\right| \qquad (23)$$

Example (1): Consider the following fuzzy initial value problem:

$y' = -y + x + 1$, with $x \in [0 , 1]$

$y(0) = [0.96 + 0.04r , 1.01 - 0.01r]$, where $r \in [0 , 1]$.

The analytical solutions for this problem are:

$\underline{y}_a(x,r) = x + (0.96 + 0.04r)\, e^{-x}$, $\overline{y}_a(x,r) = x + (1.01 + 0.01r)\, e^{-x}$

By using (7), the trial solutions for this problem are:

$\underline{y}_t(x,r) = (0.96 + 0.04r) + x\,\underline{N}(x, r, \underline{p})$, $\overline{y}_t(x,r) = (1.01 + 0.01r) + x\,\overline{N}(x, r, \overline{p})$.

The ANN trained using a grid of ten equidistant points in [0 , 1]. The error function that must be minimized for this problem is in the form:

$$E = \sum_{i=1}^{11} \left(\begin{bmatrix} x_i \sum_{j=1}^{10} \underline{v}_j\, \underline{w}_{j1} s'\left(x_i\, \underline{w}_{j1} + r\, \underline{w}_{j2} + \underline{b}_j\right) + (1 + x_i) \\ \sum_{j=1}^{10} \underline{v}_j\, s\left(x_i\, \underline{w}_{j1} + r\, \underline{w}_{j2} + \underline{b}_j\right) - x_i + 0.04r - 0.04 \end{bmatrix}^2 + \begin{bmatrix} x_i \sum_{j=1}^{10} \overline{v}_j\, \overline{w}_{j1} s'\left(x_i\, \overline{w}_{j1} + r\, \overline{w}_{j2} + \overline{b}_j\right) + (1 + x_i) \\ \sum_{j=1}^{10} \overline{v}_j\, s\left(x_i\, \overline{w}_{j1} + r\, \overline{w}_{j2} + \overline{b}_j\right) - x_i - 0.01r + 0.01 \end{bmatrix}^2 \right) \quad (24)$$

Then one can use (24) to update the weights and biases with respect to usual artificial neural network (UANN).

● For modified artificial neural network (MANN), we can get:

$$E = \sum_{i=1}^{11} \left(\begin{bmatrix} x_i \sum_{j=1}^{10} \underline{v}_j\, \underline{w}_{j1} s'\left(Q(x_i)\, \underline{w}_{j1} + r\, \underline{w}_{j2} + \underline{b}_j\right) + (1 + x_i) \\ \sum_{j=1}^{10} \underline{v}_j\, s\left(Q(x_i)\, \underline{w}_{j1} + r\, \underline{w}_{j2} + \underline{b}_j\right) - x_i + 0.04r - 0.04 \end{bmatrix}^2 + \begin{bmatrix} x_i \sum_{j=1}^{10} \overline{v}_j\, \overline{w}_{j1} s'\left(Q(x_i)\, \overline{w}_{j1} + r\, \overline{w}_{j2} + \overline{b}_j\right) + (1 + x_i) \\ \sum_{j=1}^{10} \overline{v}_j\, s\left(Q(x_i)\, \overline{w}_{j1} + r\, \overline{w}_{j2} + \overline{b}_j\right) - x_i - 0.01r + 0.01 \end{bmatrix}^2 \right) \quad (25)$$

Then we use (25) to update the weights and biases with respect to modified artificial neural network. Note that for this eqample, Since $x \in [0 , 1]$ then $Q(x) \in (0 , 1)$ and since $Q(x) = \epsilon (x + 1)$ with $\epsilon \in (0 , 1)$, then we must choose $\epsilon < 0.5$. For $\epsilon = 0.3$, the training set will be:

x : 0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

Q(x): 0.3 0.33 0.36 0.39 0.42 0.45 0.48 0.51 0.54 0.57 0.60

Analytical and trial solutions for this problem can be found in table (1) and Fig. (3).

Example (2) : Consider the non-linear problem:

$y''(x) = -\left(y'(x)\right)^2$. with $x \in [0 , 2]$

$y(0) = [r , 2 - r]$, $y(2) = [1 + r , 3 - r]$ and $r \in [0 , 1]$.The analytical solutions for this problem are:

$\underline{y}_a(x,r) = \ln\left(x + \frac{2}{e-1}\right) + r - \ln\frac{2}{e-1}$

$\overline{y}_a(x,r) = \ln\left(x + \frac{2}{e-1}\right) + 2 - r - \ln\frac{2}{e-1}$.

The trial solutions are:

$\underline{y}_t(x,r) = r\,\frac{2-x}{2} + (1 + r)\,\frac{x}{2} + x\,(x - 2)\,\underline{N}(x, r, \underline{p})$

$\overline{y}_t(x,r) = (2 - r)\,\frac{2-x}{2} + (3 - r)\,\frac{x}{2} + x\,(x - 2)\,\overline{N}(x, r, \overline{p})$.

The ANN trained using a grid of ten equidistant points in [0,2].The error function that must be minimized is in the form: $E = E_1 + E_2$, where:

$$E_1=\sum_{i=1}^{11}\left[\begin{array}{c}(x_i^2-2x_i)\sum_{j=1}^{10}\underline{v}_j\ \underline{w}_{j1}^2 s''\left(Q(x_i)\,\underline{w}_{j1}+r\,\underline{w}_{j2}+\underline{b}_j\right)\\ +(2x_i-2)\sum_{j=1}^{10}\underline{v}_j\underline{w}_{j1}\,s'\left(Q(x_i)\,\underline{w}_{j1}+r\,\underline{w}_{j2}+\underline{b}_j\right)+\\ 2\sum_{j=1}^{10}\underline{v}_j\,s\left(Q(x_i)\,\underline{w}_{j1}+r\,\underline{w}_{j2}+\underline{b}_j\right)+\\ \left(0.5+(x_i^2-2x_i)\ \sum_{j=1}^{10}\underline{v}_j\underline{w}_{j1}\,s'\left(Q(x_i)\,\underline{w}_{j1}+r\,\underline{w}_{j2}+\underline{b}_j\right)\right.\\ \left.+(2x_i-2)\ \sum_{j=1}^{10}\underline{v}_j\,s\left(Q(x_i)\,\underline{w}_{j1}+r\,\underline{w}_{j2}+\underline{b}_j\right)\right)^2\end{array}\right]^2$$

$$E_2=\sum_{i=1}^{11}\left[\begin{array}{c}(x_i^2-2x_i)\sum_{j=1}^{10}\overline{v}_j\,\overline{w}_{j1}^2 s''\left(Q(x_i)\,\overline{w}_{j1}+r\,\overline{w}_{j2}+\overline{b}_j\right)+\\ (2x_i-2)\sum_{j=1}^{10}\overline{v}_j\overline{w}_{j1}\,s'\left(Q(x_i)\,\overline{w}_{j1}+r\,\overline{w}_{j2}+\overline{b}_j\right)+\\ 2\sum_{j=1}^{10}\overline{v}_j\,s\left(Q(x_i)\,\overline{w}_{j1}+r\,\overline{w}_{j2}+\overline{b}_j\right)\\ \left(0.5+(x_i^2-2x_i)\ \sum_{j=1}^{10}\overline{v}_j\overline{w}_{j1}\,s'\left(Q(x_i)\,\overline{w}_{j1}+r\,\overline{w}_{j2}+\overline{b}_j\right)\right.\\ \left.+(2x_i-2)\ \sum_{j=1}^{10}\overline{v}_j\,s\left(Q(x_i)\,\overline{w}_{j1}+r\,\overline{w}_{j2}+\overline{b}_j\right)\right)^2\end{array}\right]^2$$

For ϵ = 0.6,analytical and trial solution for this problem can be found in table (2) and Fig.(4).

Example (3) : Consider the fuzzy Poisson equation:

$\frac{\partial^2 \tilde{U}}{\partial x^2}(x,y)+\frac{\partial^2 \tilde{U}}{\partial y^2}(x,y)=\tilde{K}\,x\,e^y$, with $x,y\in[0,1]$

Where $\tilde{K}[r]=\left[\underline{K}(r),\overline{K}(r)\right]=\left[0.75+0.25r\,,\,1.25-0.25r\right]$, with the fuzzy boundary conditions:

$\tilde{U}(0,y)=0$, $\tilde{U}(1,y)=\tilde{K}\,e^y$, $\tilde{U}(x,0)=\tilde{K}\,x$ and $\tilde{U}(x,1)=e\,\tilde{K}\,x$.

The analytical solutions for this problem are:

$\underline{U}_a(x,y,r)=(0.75+0.25r)\,x\,e^y$, $\overline{U}_a(x,y,r)=(1.25-0.25r)\,x\,e^y$.

The trial solutions are :

$\underline{U}_t(x,y,r,\underline{p})=(0.75+0.25r)\,x\,e^y+x\,y\,(1-x)\,(1-y)\,\underline{N}(x,y,r,\underline{p})$

$\overline{U}_t(x,y,r,\overline{p})=(1.25-0.25r)\,x\,e^y+x\,y\,(1-x)\,(1-y)\,\overline{N}(x,y,r,\overline{p})$.

The error function that must be minimized is : $E=E_1+E_2$,where:

$$E_1=\sum_{i=1}^{11}\left[\begin{array}{c}(y_i-y_i^2)\left(\begin{array}{c}(x_i-x_i^2)\ \sum_{j=1}^{10}\underline{v}_j\ \underline{w}_{j1}^2 s''\left(Q(x_i)\,\underline{w}_{j1}+Q(y_i)\,\underline{w}_{j2}+r\,\underline{w}_{j3}+\underline{b}_j\right)\\ +(2-4x_i)\sum_{j=1}^{10}\underline{v}_j\underline{w}_{j1}\,s'\left(Q(x_i)\,\underline{w}_{j1}+Q(y_i)\,\underline{w}_{j2}+r\,\underline{w}_{j3}+\underline{b}_j\right)\\ -2\sum_{j=1}^{10}\underline{v}_j\,s\left(Q(x_i)\,\underline{w}_{j1}+Q(y_i)\,\underline{w}_{j2}+r\,\underline{w}_{j3}+\underline{b}_j\right)\end{array}\right)\\ +\\ (x_i-x_i^2)\left(\begin{array}{c}(y_i-y_i^2)\ \sum_{j=1}^{10}\underline{v}_j\ \underline{w}_{j1}^2 s''\left(Q(x_i)\,\underline{w}_{j1}+Q(y_i)\,\underline{w}_{j2}+r\,\underline{w}_{j3}+\underline{b}_j\right)\\ +(2-4y_i)\sum_{j=1}^{10}\underline{v}_j\underline{w}_{j2}\,s'\left(Q(x_i)\,\underline{w}_{j1}+Q(y_i)\,\underline{w}_{j2}+r\,\underline{w}_{j3}+\underline{b}_j\right)\\ -2\sum_{j=1}^{10}\underline{v}_j\,s\left(Q(x_i)\,\underline{w}_{j1}+Q(y_i)\,\underline{w}_{j2}+r\,\underline{w}_{j3}+\underline{b}_j\right)\end{array}\right)\end{array}\right]^2$$

$E_{2=}$

$$\sum_{i=1}^{11}\left[\begin{array}{c}(y_i-y_i^2)\left(\begin{array}{c}(x_i-x_i^2)\sum_{j=1}^{10}v_j\,w_{j1}^2\,s''\left(Q(x_i)\,w_{j1}+Q(y_i)\,w_{j2}+r\,w_{j3}+b_j\right)\\+(2-4x_i)\sum_{j=1}^{10}v_j\,w_{j1}\,s'\left(Q(x_i)\,w_{j1}+Q(y_i)\,w_{j2}+r\,w_{j3}+b_j\right)\\-2\sum_{j=1}^{10}v_j\,s\left(Q(x_i)\,w_{j1}+Q(y_i)\,w_{j2}+r\,w_{j3}+b_j\right)\end{array}\right)\\+\\(x_i-x_i^2)\left(\begin{array}{c}(y_i-y_i^2)\sum_{j=1}^{10}v_j\,w_{j1}^2\,s''\left(Q(x_i)\,w_{j1}+Q(y_i)\,w_{j2}+r\,w_{j3}+b_j\right)\\+(2-4y_i)\sum_{j=1}^{10}v_j\,w_{j2}\,s'\left(Q(x_i)\,w_{j1}+Q(y_i)\,w_{j2}+r\,w_{j3}+b_j\right)\\-2\sum_{j=1}^{10}v_j\,s\left(Q(x_i)\,w_{j1}+Q(y_i)\,w_{j2}+r\,w_{j3}+b_j\right)\end{array}\right)\\\cdot\end{array}\right]^2$$

For ϵ = 0.2, analytical and trial solutions for this problem can be found in table (3), Fig. (5) and Fig. (6).

6. Conclusion

In this paper, we presented a hybrid approach based on modified artificial neural networks for solving fuzzy differential equations. We demonstrate, for the first time, the ability of modified artificial neural networks to approximate the solutions of FDEs . From the numerical examples in this work , it is clear that the modified artificial neural network gives best results and better accuracy in comparison with usual artificial neural network . We can conclude that the method we propused can handle effectively all types of the fuzzy differential equations and provide accurate approximate solution throughout the whole domain and not only at the training set. Therefore, one can use the interpolation techniques (such as curve fitting method) to find the approximate solution at points between the training points or at points outside the training set. The main reason for using modified artificial neural networks was their applicability in function approximation. Further research is in progress to apply and extend this method to solve three-dimensional fuzzy partial differential equations FPDEs and fuzzy integral equations .

References

Abbasbandy, S. & Allahviranloo,T.(2002), "Numerical Solution of Fuzzy Differential Equations by Runge-Kutta Method", J. Sci. Teacher Training University, 1(3).

Abbasbandy, S. & Allahviranloo,T.(2002) , "Numerical Solution of Fuzzy Differential Equations by Taylor Method", Journal of Computational Methods in Applied Mathematics, 2, 113-124.

Ahmadi,M.B. & Kiani,N.A. (2011) , "Solving Fuzzy Partial Differential Equation by Differential Transformation Method" , Journal of Applied Mathematics,27,1-16.

Allahviranloo,T. , Ahmady,N. & Ahmady,E.(2007), "Numerical Solution of Fuzzy Differential Equations by predictor- corrector Method", Information Sciences, 177 , 1633-1647.

Allahviranloo,T. , Ahmady,T.E. & Ahmady,N. (2008),"Nth- Order Fuzzy Linear Differential Equations" , Information Sciences, 178, 1309-1324.

Buckley,J.J. & Feuring ,T. (2000),"Fuzzy Differential Equations", Fuzzy Sets and Systems, 110 , 69 - 77.

Corveleyn,S. & Vandewalle,S.(2011) , "Numerical Solution of Fuzzy Elliptic Partial Differential Equations by a Polynomial Galerkin Approximation",Katholieke Universiteit Leuven,Department of Computer Science,Report TW 585,1-22.

Duraisamy,C. & Usha ,B.(2010), "Another Approach to Solution of Fuzzy Differential Equations", Applied Mathematics Sciences,4(16),777-790.

Effati,S. & Pakdaman,M.(2010), "Articial Neural Network Approach for Solving Fuzzy Differential Equations", Information Sciences , 180, 1434 -1457.

Ezadi,S. & Parandin,N.(2013),"An Application of Neural Networks to Solve Ordinary Differential Equations",International Journal of Mathematical Modelling & Computations,3(3),245-252.

Ezadi,S. , Parandin,N. & Ghomashi,A.(2013), "Numerical Solution of Fuzzy Differential Equations Based on Semi-Taylor by Using Neural Network", Journal of Basic and Applied Scientific Research , 477 - 482.

Ghazanfari,B. & Shakerami,A.(2011), "Numerical Solution of Fuzzy Differential Equations Extended Runge – Kutta- Like Formulae of Order 4", Fuzzy Sets and Systems, 189, 74 – 91.

Guo,X. , Shang,D. & Lu,X.(2013), "Fuzzy Approximate Solutions of Second Order Fuzzy Linear Boundary Value Problems,Springer Open Journal,212,1-17.

Jayakumar,T. ,Maheskumar,D. & Kanagarajan,K.(2012), "Numerical Solution of Fuzzy Differential Equation by Runge-Kutta Method of Order Five",Applied Mathematical Sciences,6(60),2989-3002.

Kaleva,O.(1987),"Fuzzy Differential Equations", Fuzzy Sets and Sydtems, 24 , 301 – 317.

Mosleh,M. & Otadi,M.(2012)," Fuzzy Fredholm Integro-Differential Equations with Artificial Neural Network",International Scientific Publications and Consulting Services,20(12),1-13.
Orouji,B.,Parandin,N. ,Abasabadi,L. & Hosseinpour,A.(2014)," An Implicit Method For Solving Fuzzy Partial Differential Equation with Nonlocal Boundary Conditions " , American Journal of Engineering Research(AJER),3(6),15-19.
Parimala,V.,Rajarajeswari,P. & Nirmala,V.(2014)," A Second Order Runge-Kutta Method to Solve Fuzzy Differential Equations with Fuzzy Initial Condition",International Journal of Sciences and Research,3(3),428-431.
Rostami,M.Kianpour,M. & Bashardoust,E.(2011), "A Numerical Algorithm for Solving Nonlinear Fuzzy Differential Equations",The Journal of Mathematics and Computer Science,2(4),667-671.
Seikkala,S.(1987)," On The Fuzzy Initial Value Problem", Fuzzy Sets and Systems, 24, 319 - 330.
Saikia ,R.K.(2011)," Fuzzy Numerical Solution of Poisson Equation Using Fuzzy Data",International Journal of Engineerring and Technology(IJEST),3(12),8450-8456.
Tawfiq,L.N.M.(2004),"On Design and Training of Artificial Neural Networks for Solving Differential Equations",PH.D Thesis,52-65.
Tawfiq,L.N.M. &Al-Abrahemee,K.M.M.(2014),"Design Neural Network to Solve Singular Perturbation Problems", Applied & Computational Mathematics,3.
Tapaswini,S. & Chakraverty,S.(2014)," New Midpoint-Based Approach for the Solution of N-th Order Interval Differential Equations",Department of Mathematics",National Institute of Technology Rourkela,Odisha-769 008,India,25-43.
Tapaswini,S. &Chakraverty,S.(2014),"New Analytical Method for Solving N-th Order Fuzzy Differential Equations",Annals of Fuzzy Mathematics and Informatics,8(2),231-244.

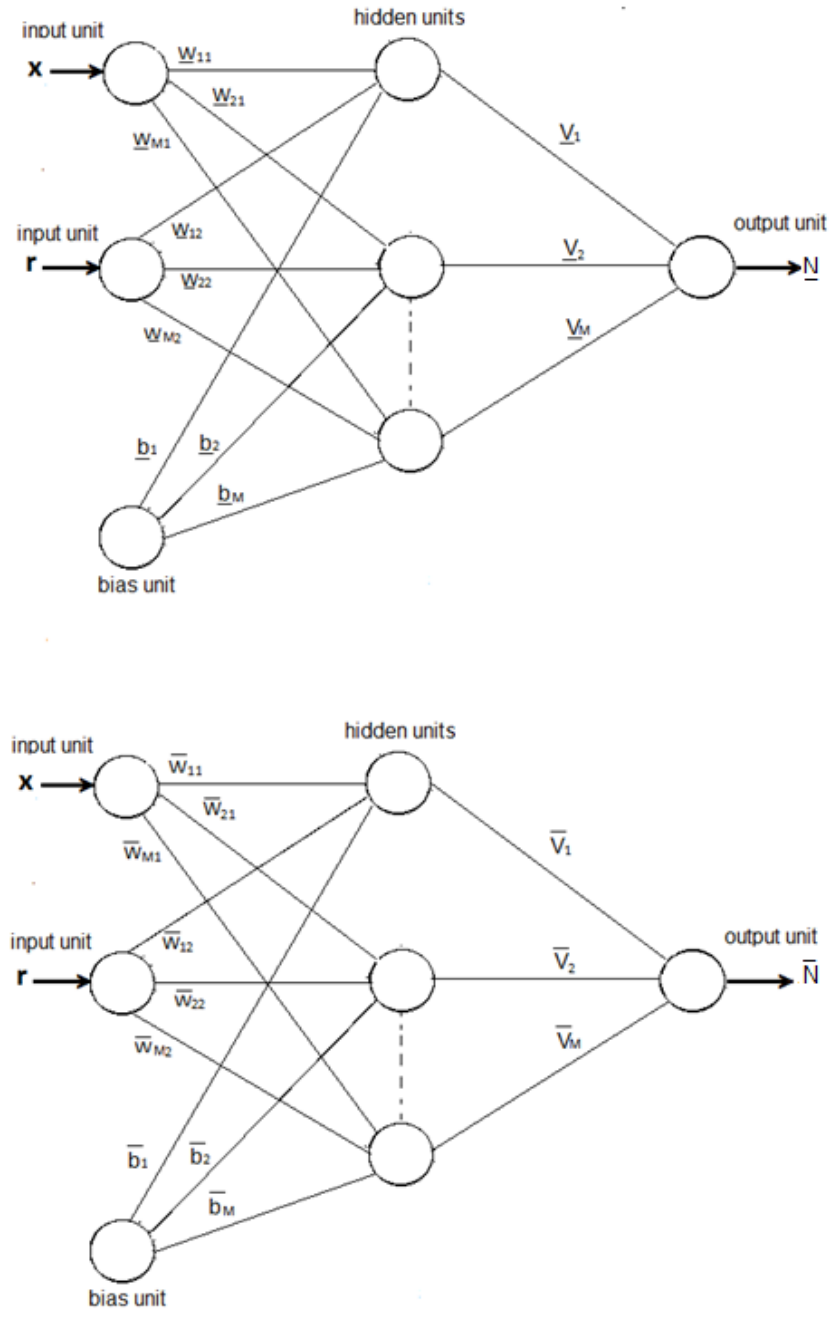

Figure 1. (2 x m x 1) Feed-forword neural networks.

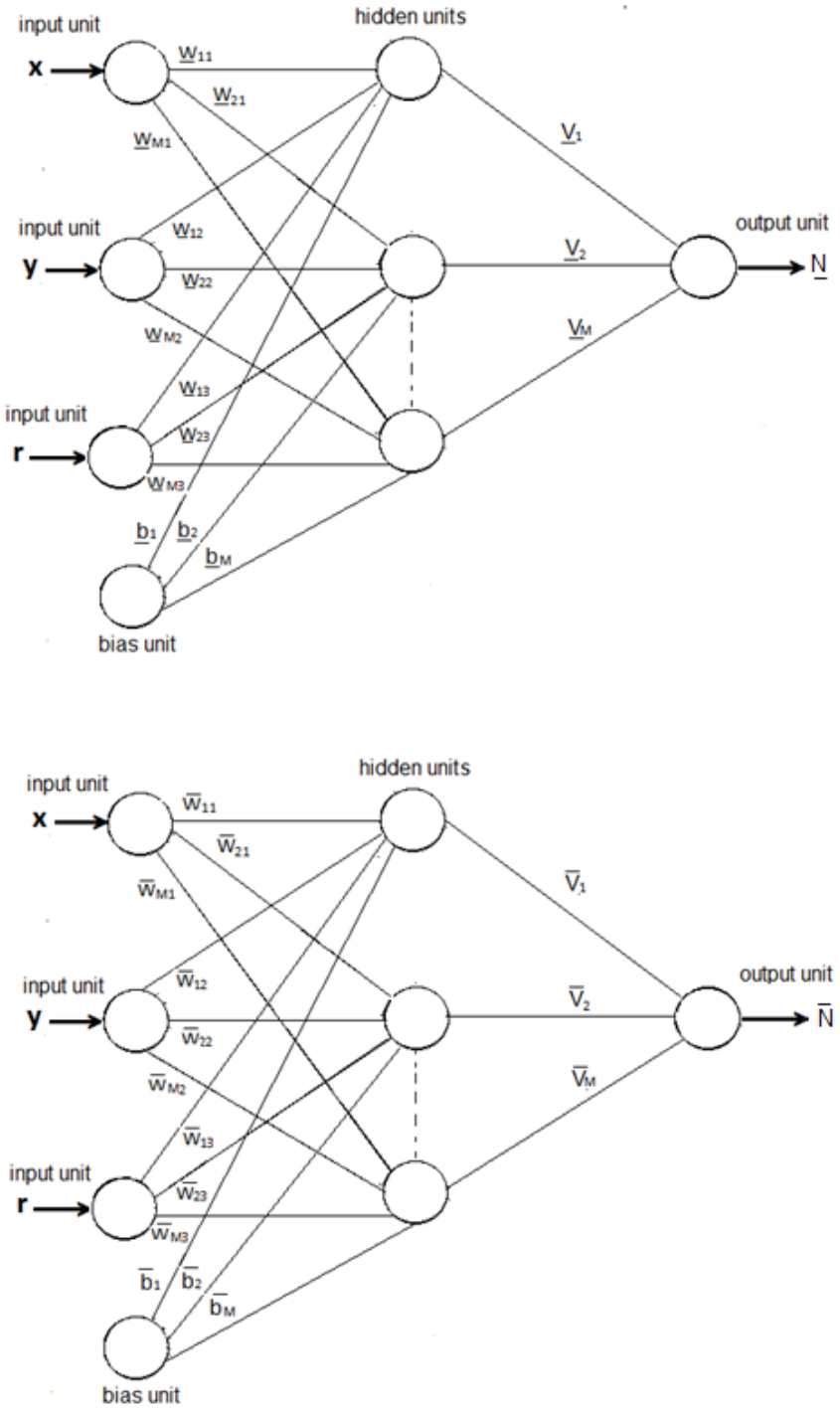

Figure 2. (3 x m x 1) Feed-forward neural networks.

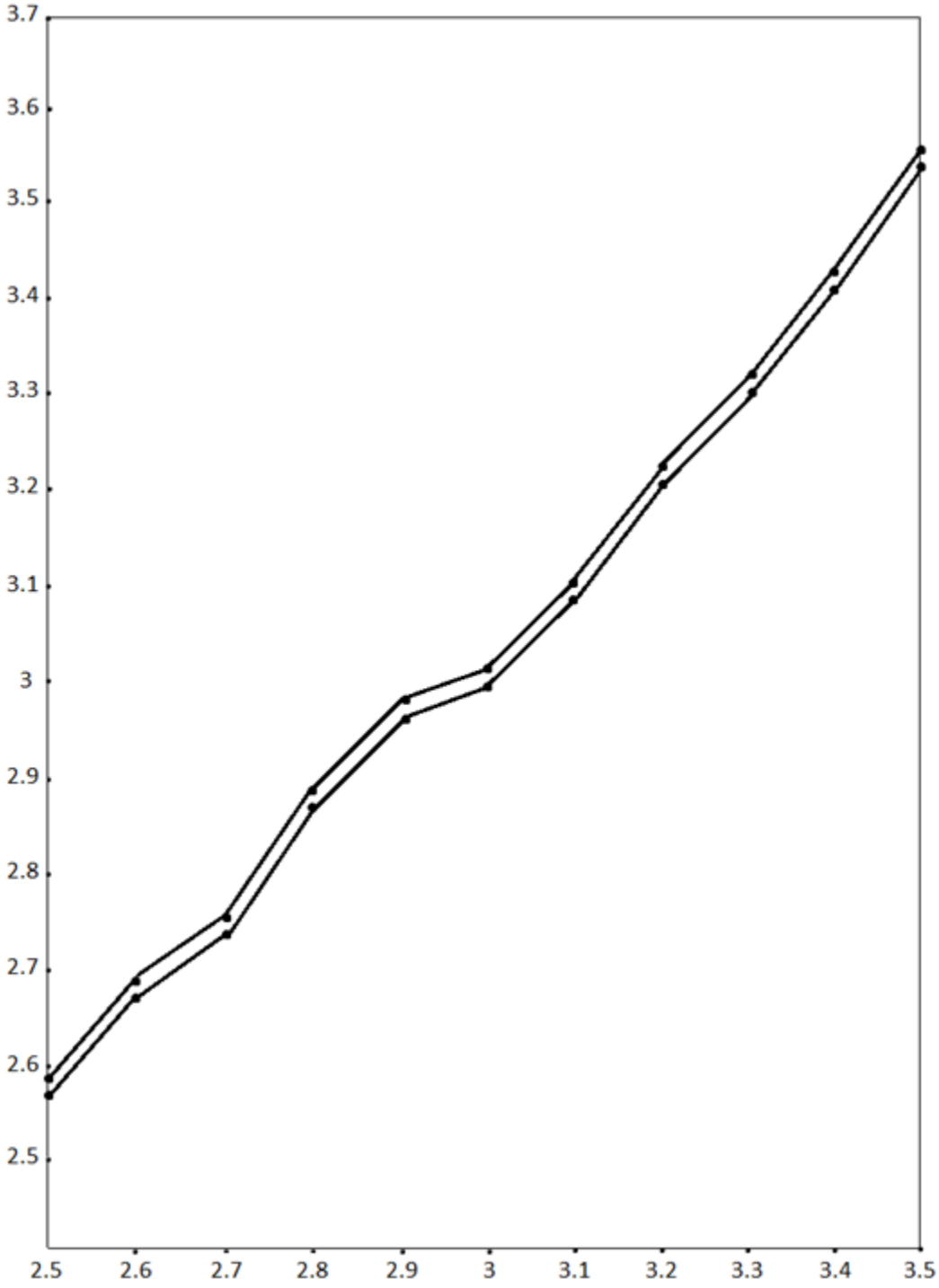

Figure 3. Trial solutions for example (1) , for r = 0.5.

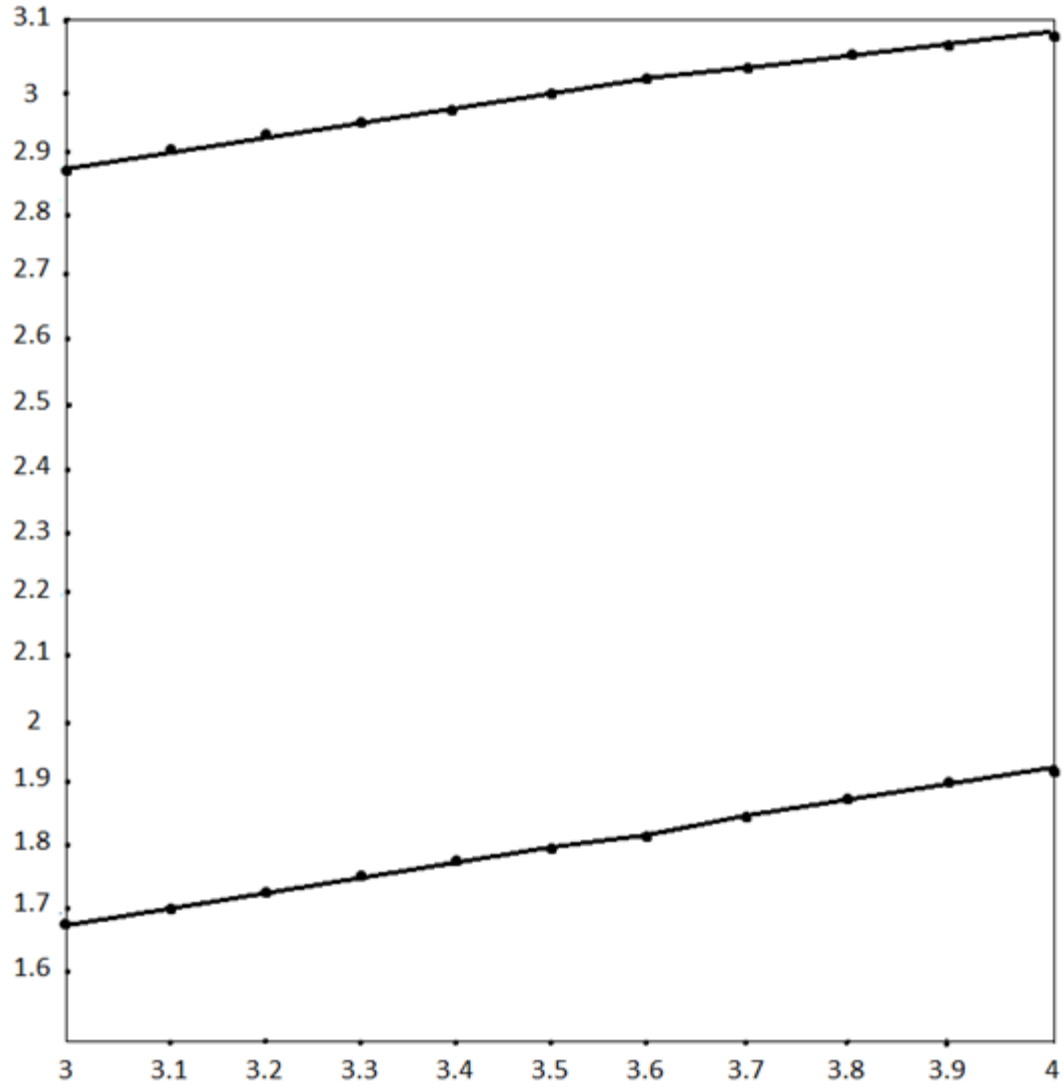

Figure 4. Trial solutions for example (2), for r = 0.4.

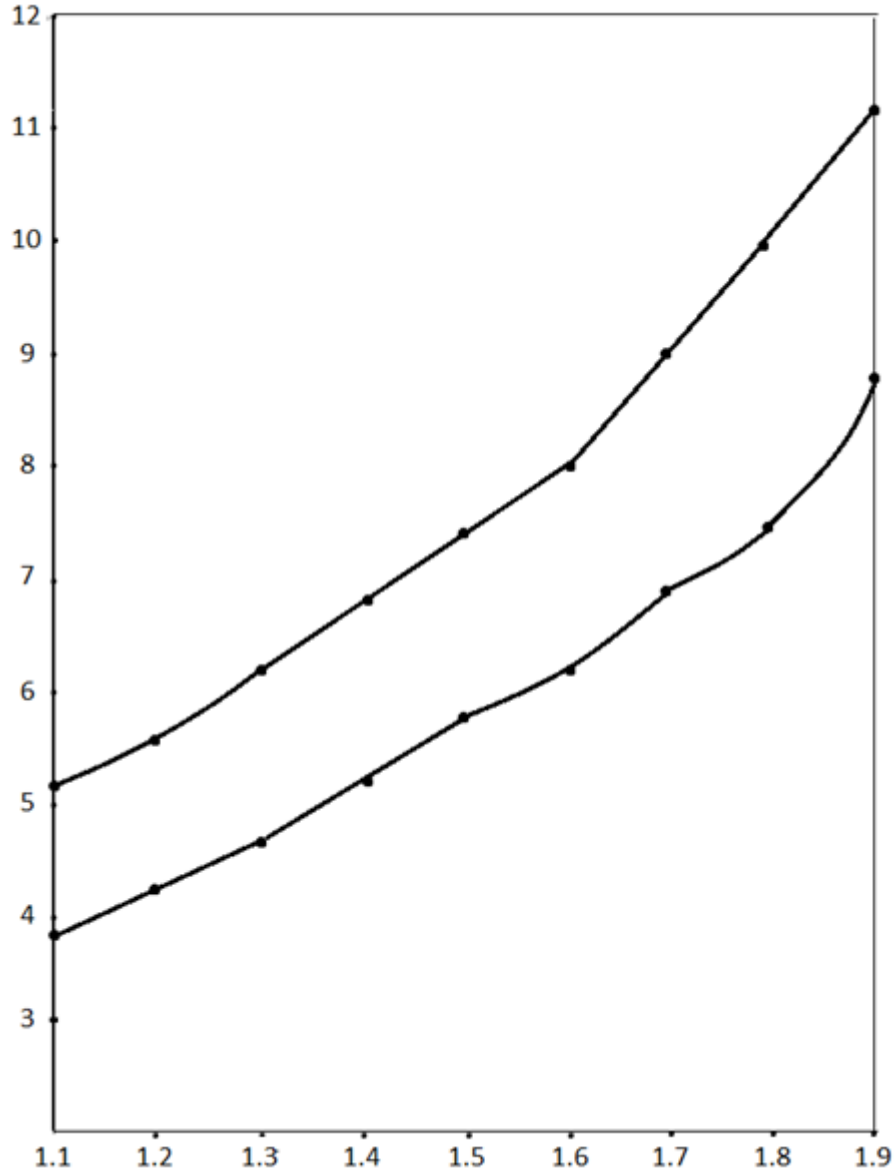

Figure 5. Trial solutions for example (3),r = 0.5, x =1.5 and y $\in$ [1.1,1.9].

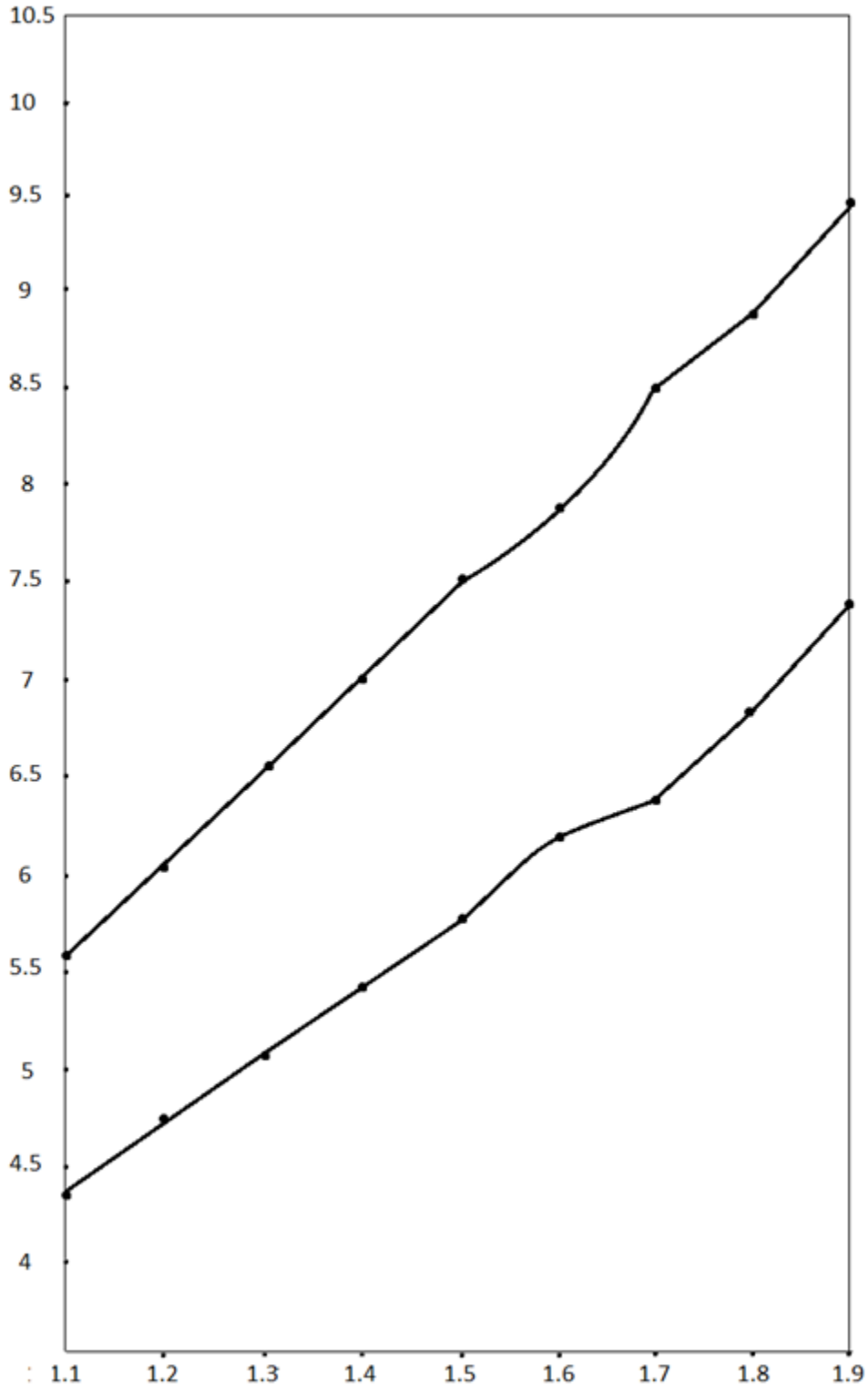

Figure 6. Trial solutions for example (3),r = 0.5, y =1.5 and x ∈ [1.1,1.9].

Table 1. Analytical and trial solutions for example (1) , for r = 0.5.

x	$\underline{y}_a(x,r)$	$\underline{y}_t(x,r)$	$\underline{e}(x,r)$	$\overline{y}_a(x,r)$	$\overline{y}_t(x,r)$	$\overline{e}(x,r)$
0	0.980000000	0.980000000	0.000000000	1.005000000	1.005000000	0.000000000
0.1	0.986740669	0.986741141	0.000000472	1.009361605	1.009362248	0.000000643
0.2	1.002356138	1.002356525	0.000000387	1.022824407	1.022824322	0.000000085
0.3	1.026001856	1.026002061	0.000000205	1.044522312	1.044523023	0.000000711
0.4	1.056913645	1.056913016	0.000000629	1.073671646	1.073671240	0.000000406
0.5	1.094400047	1.094399412	0.000000635	1.109563313	1.109563409	0.000000096
0.6	1.137835403	1.137835497	0.000000094	1.151555694	1.151555790	0.000000114
0.7	1.186653598	1.186653646	0.000000048	1.199068230	1.199068592	0.000000362
0.8	1.240342385	1.240342311	0.000000074	1.251575609	1.251574896	0.000000713
0.9	1.298438267	1.298438992	0.000000725	1.308602508	1.308603048	0.000000054
1	1.360521852	1.360521344	0.000000508	1.369718838	1.369719666	0.000000828

Table 2. Analytical and trial solutions for example (2) , for r = 0.4.

x	$\underline{y}_a(x,r)$	$\underline{y}_t(x,r)$	$\underline{e}(x,r)$	$\overline{y}_a(x,r)$	$\overline{y}_t(x,r)$	$\overline{e}(x,r)$
0	0.400000000	0.400000000	0.000000000	1.600000000	1.600000000	0.000000000
0.2	0.558565078	0.558565701	0.000000623	1.758565078	1.758565803	0.000000725
0.4	0.695394529	0.695395179	0.000000650	1.895394529	1.895395153	0.000000624
0.6	0.815735221	0.815734544	0.000000677	2.015735222	2.015735745	0.000000523
0.8	0.923137163	0.923137093	0.000000070	2.123137164	2.123136742	0.000000422
1	1.020114507	1.020114434	0.000000073	2.220114514	2.220114514	0.000000007
1.2	1.108513067	1.108512992	0.000000075	2.308513067	2.308513062	0.000000005
1.4	1.189728044	1.189728122	0.000000078	2.389728044	2.389728059	0.000000015
1.6	1.264839725	1.264839733	0.000000008	2.464839725	2.464839842	0.000000117
1.8	1.334701664	1.334701750	0.000000086	2.534701664	2.534701778	0.000000114
2	1.400000000	1.400000000	0.000000000	2.600000000	2.600000000	0.000000000

Table 3. Analytical and trial solutions for example(3),for r=0.5.

x	y	$\underline{U}_a(x,y,r)$	$\underline{U}_t(x,y,r)$	$\underline{e}(x,y,r)$	$\overline{U}_a(x,y,r)$	$\overline{U}_t(x,y,r)$	$\overline{e}(x,y,r)$
0	0	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000	0.000000000
0.1	0.1	0.096702455	0.096706465	0.000004010	0.124331728	0.124332900	0.000001172
0.2	0.2	0.213745482	0.213749391	0.000003909	0.274815620	0.274816765	0.000001145
0.3	0.3	0.354337937	0.354341016	0.000003079	0.455577347	0.455576229	0.000001118
0.4	0.4	0.522138644	0.522136568	0.000002076	0.671321113	0.671320022	0.000001091
0.5	0.5	0.721315555	0.721314482	0.000001073	0.927405714	0.927406777	0.000001063
0.6	0.6	0.956612370	0.956611317	0.000001053	1.229930190	1.229931226	0.000001036
0.7	0.7	1.233423533	1.33422518	0.000001015	1.585830257	1.585831239	0.000000982
0.8	0.8	1.557878650	1.557879607	0.000000957	2.002986836	2.002987791	0.000000955
0.9	0.9	1.936937450	1.936938355	0.000000905	2.490348150	2.490347223	0.000000927
1	1	2.378496600	2.378496600	0.000000000	3.058067057	3.058067057	0.000000000

The IISTE is a pioneer in the Open-Access hosting service and academic event management. The aim of the firm is Accelerating Global Knowledge Sharing.

More information about the firm can be found on the homepage:
http://www.iiste.org

CALL FOR JOURNAL PAPERS

There are more than 30 peer-reviewed academic journals hosted under the hosting platform.

Prospective authors of journals can find the submission instruction on the following page: http://www.iiste.org/journals/ All the journals articles are available online to the readers all over the world without financial, legal, or technical barriers other than those inseparable from gaining access to the internet itself. Paper version of the journals is also available upon request of readers and authors.

MORE RESOURCES

Book publication information: http://www.iiste.org/book/

Academic conference: http://www.iiste.org/conference/upcoming-conferences-call-for-paper/

IISTE Knowledge Sharing Partners

EBSCO, Index Copernicus, Ulrich's Periodicals Directory, JournalTOCS, PKP Open Archives Harvester, Bielefeld Academic Search Engine, Elektronische Zeitschriftenbibliothek EZB, Open J-Gate, OCLC WorldCat, Universe Digtial Library , NewJour, Google Scholar