International Journal of Mathematical Archive-6(6), 2015, 84-94
Available online through www.ijma.info ISSN 2229 – 5046

NUMERICAL SOLUTION OF FUZZY DIFFERENTIAL EQUATIONS BY USING MODIFIED FUZZY NEURAL NETWORK

EMAN A. HUSSIAN*
Asst. Prof., Dep. of Mathematics. College of Sciences.
Al-Mustansiriyah University, Baghdad, Iraq.

MAZIN H. SUHHIEM
Asst. Lec., Dep. of statistics,
College of Adm. and Econ. University of sumar, Baghdad, Iraq.

(Received On: 31-05-15; Revised & Accepted On: 22-06-15)

ABSTRACT

In this paper, we introduce a hybrid approach based on modified fuzzy neural network and optimization teqnique to solve fuzzy differential equations. Using modified fuzzy 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 four numerical examples.

Keywords: *Fuzzy differential equation, Modified fuzzy neural network, Feedforward neural network, BFGS method, Hyperbolic tangent function.*

1. INTRODUCTION

Nowadays, fuzzy differential equations are 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 which satisfies fuzzy initial or boundary conditions. The theory of fuzzy differential equations was treated by Buckley and Feuring [9], Kaleva [16,17], Nieto [28], Ouyang and Wu [32], Seikkala also recently there appeared the papers of Bede, Bede and Gal [8], Diamond [10,11], Georgion *et al*., [14] Nieto and Rodriguez-Lopez [29]. 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-Kuta and Taylor methods [1, 2]. Also, Allahviranloo *et al*. solved differential equations by predictor- corrector and transformation methods [3, 4, 5]. Ghazanfari and Shakerami developed Runge-Kuta like formula of order 4 for solving fuzzy differential equations [13]. Nystrom method has been introduced for solving fuzzy differential equations [18].

In 1990 Lee and Kang [19] used parallel processor computers to solve a first order differential equations with Hopfield neural network models. Meade, Fernandes and Malek [22, 27] solved linear and nonlinear ordinary differential equations using feed-forward neural network architecture and B_1-splines. Recently, fuzzy neural networks have been successfully used for solving fuzzy polynomial equations and systems of fuzzy polynomial equations [6, 7], approximate fuzzy coefficients of fuzzy regression models [21, 25, 26], approximate solution of fuzzy linear system and fully fuzzy linear systems [31]. In Year 2012 Mosleh and Otadi [23] used fuzzy neural network to solve a first order fuzzy neural network, system of fuzzy differential equations [20] and second order fuzzy differential equation [24].

In this work we propose a new solution method for the approximated solution of FDEs, this hybrid method can result in improved numerical methods for solving fuzzy differential equations. In this proposed method, fuzzy neural network model (FNNM)is applied as universal approximator. We use fuzzy trial function; this fuzzy trial function is a combination of two terms. A first term is responsible for the fuzzy condition while the second term contains the fuzzy neural network adjustable parameters to be calculated. The main aim of this paper is to illustrate how fuzzy connection weights are adjusted in the learning of fuzzy neural networks. Our fuzzy neural network in this paper is a three-Layer feed- forward neural network where connection weights and biases are fuzzy numbers.

*Corresponding Author: Dr. Eman A. Hussian**

2. PRELIMINARIES

In this section the basic notations used in fuzzy calculus are introduced.

Definition 2.1: [**23**] A fuzzy number u is completely determined by any pair u= $(\underline{u}, \bar{u})$ of functions $\underline{u}$ (r), $\bar{u}$ (r) : R ⟶ [0,1] satisfying the conditions:

(1) $\underline{u}$ (r) is a bounded, monotonic, increasing (non – decreasing) left continuous function for all r ∈ (0,1] and right continuous for r=0.
(2) $\bar{u}$ (r) is a bounded, monotonic, decreasing (non – increasing) left continuous function for all r ∈ (0,1] and right continuous for r=0.
(3) For all r ∈ (0,1] we have $\underline{u}$ (r) ≤ $\bar{u}$ (r).

For every u =$(\underline{u}, \bar{u})$, v = $(\underline{v}, \bar{v})$ and $k > 0$ we define addition and multiplication as follows:

1. $\underline{(u + v)}$ (r) = $\underline{u}$ (r) + $\underline{v}$ (r) (1)
2. $\overline{(u + v)}$ (r) = $\bar{u}$ (r) + $\bar{v}$ (r) (2)
3. $\underline{(k\,u)}$ (r) = K $\underline{u}$ (r), $\overline{(k\,u)}$ (r) = K $\bar{u}$ (r) (3)

The collection of all fuzzy numbers with addition and multiplication as defined by $Eqs.$ (1) ⟶ (3) is denoted by E^1.

For r ∈ (0,1], we define the r - cuts of fuzzy number u with $[u]_r$ ={x ∈ R|u (x) ≥ r} and for r =0, the support of u is defined as $[u]_0$ =$\overline{\{x \in R|u(x) > 0\}}$

Definition 2.2 [**28**]**:** The function f: R ⟶ E^1 is called a fuzzy function. Now if, for an arbitrary fixed $t_1 \in R$ and $\epsilon > 0$ there exist a $\delta > 0$ such that: $|t - t_1| < \delta \Longrightarrow d[f(t), f(t_1)] < \epsilon$

Then f is said to be continuous function.

Definition 2.3 [**12**]**:** letu, v ∈ E^1. If there exist w ∈ E^1 such that u = v+w then w is called the H-difference (Hukuhara-difference) of u, v and it is denoted by w= u Θ v.

In this paper the Θ sign stands always for H-difference, and let us remark that u Θ v ≠ u + (-1) v.

Definition 2.4 [**14**]**:** Let f : [a, b] → E^1 and $t_0 \in$ [a,b].We say that f is H-differential (Hukuhara-differential) at t_0, if there exists an element f $(t_0) \in E^1$ such that for all h> 0 sufficiently small, ∃ f(t_0 +h) Θ f(t_0), f(t_0) Θ f(t_0 - h) and the limits

$$\lim_{h\to 0} \frac{f(t_0+h) \ominus f(t_0)}{h} = \lim_{h\to 0} \frac{f(t_0) \ominus f(t_0-h)}{h} = f'(t_0). \quad (4)$$

3. FUZZY NEURAL NETWORK

A fuzzy neural network or neuro -fuzzy system is a learning machine that finds the parameters of a fuzzy system (i.e. fuzzy sets, fuzzy rules) by exploiting approximation from neural network [23]. Combining fuzzy system with neural network. Both neural network and fuzzy system have some things in common. Artificial neural networks are an exciting form of the artificial intelligence which mimic the learning process of the human brain in order to extract patterns from historical data. Simple perceptrons need a teacher to tell the network what the desired output should by. These are supervised networks. In an unsupervised net, the network adapts purely in response to its input.

Such network can learn to pick out structure in their input. Fig.1 shows typical three-layered perceptron. multi-layered perceptrons with more than three layers, use more hidden layers. Multi-layered perceptrons correspond the input units to the output units by a specific nonlinear function. From kolmogorov existence theorem we know that a three-layered perceptron with n(2n+1) nodes can compute any continuous function of n variables [15] .

4. OPERATION OF FUZZY NUMBERS AND ACTIVATION FUNCTION

we briefly on mention fuzzy numbers operation defined by the extension principle. since input vector of feed-forward neural network is fuzzy in this paper, the following addition, multiplication and nonlinear mapping of fuzzy number are necessary for defining our fuzzy neural network [23]:

1. $\mu_{A+B}(z) = \text{Max}\{\mu_A(x) \wedge \mu_B(y) \mid z = x + y\}$ (5)
2. $\mu_{AB}(z) = \text{Max}\{\mu_A(x) \wedge \mu_B(y) \mid z = x\,y\}$ (6)
3. $\mu_{f(net)}(z) = \text{Max}\{\mu_{net}(x) \mid z = f(x)\}$ (7)

Where A, B and net are fuzzy number, $\mu(*)$ denotes the membership function of each fuzzy number, $\wedge$ is the Minimum operator and $f(.)$ is a continuous activation function (such as Hyperbolic tangent function) inside the hidden neurons. the above operations of fuzzy numbers are numerically performed on level sets (i.e. r-cuts).

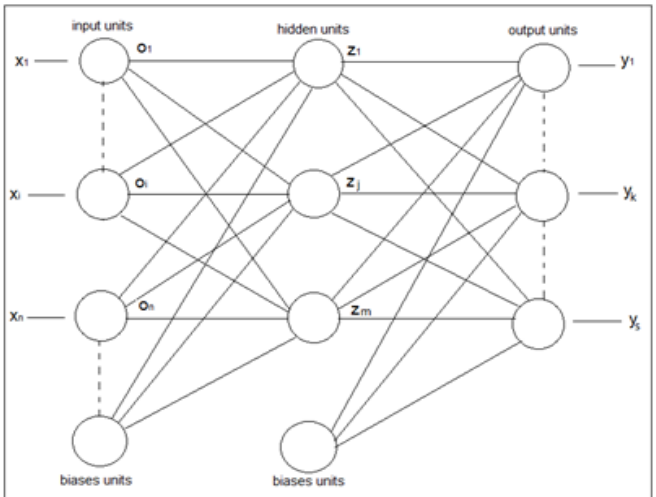

Figure 1 : Multiple layer feed - forward FNNM

The r-level set of a fuzzy number A is defined as:

$$[A]_r = \{ x \in R \mid \mu_A(x) \geq r\}, 0 < r \leq 1 \quad (8)$$

Since level sets of fuzzy numbers become closed intervals we denote $[A]_r$ as: $[A]_r = \left[[A]_r^L, [A]_r^U \right]$

Where $[A]_r^L$ and $[A]_r^U$ are the lower limit and the upper limit of the r-level set $[A]_r$ respectively [30], from interval arithmetic, the above operations of fuzzy number are written for r-level set as follows:

$$[A]_r + [B]_r = \left[[A]_r^L + [B]_r^L,\ [A]_r^U + [B]_r^U \right] \quad (9)$$

$$[A]_r\,[B]_r = \begin{bmatrix} \text{Min}\left\{ [A]_r^L.[B]_r^L, [A]_r^L.[B]_r^U, [A]_r^U.[B]_r^L, [A]_r^U.[B]_r^U \right\}, \\ \text{Max}\left\{ [A]_r^L.[B]_r^L, [A]_r^L.[B]_r^U, [A]_r^U.[B]_r^L, [A]_r^U.[B]_r^U \right\} \end{bmatrix} \quad (10)$$

$$f([\text{net}]_r) = f\left(\left[[\text{net}]_r^L, [\text{net}]_r^U \right] \right) = \left[f\left([\text{net}]_r^L \right), f\left([\text{net}]_r^U \right) \right] \quad (11)$$

Remark 4. 1: In the case of $0 \leq [B]_r^L \leq [B]_r^U$

Eq(10)can be simplified as: $[A]_r\,[B]_r = \begin{bmatrix} \text{Min}\left\{ [A]_r^L.[B]_r^L, [A]_r^L.[B]_r^U \right\}, \\ \text{Max}\left\{ [A]_r^U.[B]_r^L, [A]_r^U.[B]_r^U \right\} \end{bmatrix}$

5. NUMERICAL SOLUTION OF FUZZY DIFFERENTIAL EQUATIONS BASED ON THE POLYNOMIAL $H(x) = \epsilon(x^2 + x + 1)$ BY USING FUZZY NEURAL NETWORK

In this section we will discuss how we can use the fuzzy neural networks based on the polynomial $H(x) = \epsilon(x^2 + x + 1)$, $\epsilon \in (0,1)$, to solve the fuzzy differential equations. Our fuzzy neural network is a three-layered feed forward neural network where the connections weights, biases and targets are given as fuzzy numbers and inputs are given as real numbers. for convenience in this discussion, FNNM with an input layer, a single hidden layer, and an output Layer is represented as a basic structural architecture. Here the dimension of FNNM is denoted by the number of neurons in each layer, that is n × m × s, where n, m and s are the number of the neurons in the input layer, the hidden layer and the output layer, respectively(see Fig.(1)) The architecture of the model shows how FNNM transform the n inputs $(x_1, x_2, \dots, x_i, x_{i+1}, \dots, x_n)$ into the s outputs $(y_1, y_2, \dots, y_k, y_{k+1}, \dots, y_s)$ throughout the m hidden neurons $(Z_1, Z_2, \dots, Z_j, Z_{j+1}, \dots, Z_m)$ where the cycles represent the neurons in each Layer. Let B_j be the bias for the neurons Z_j, and let the C_k be the bias for the neurons y_k, and let W_{ji} be the Weights connecting the neurons x_i to the neurons and let W_{kj} be the weights connecting the neurons Z_j to the neurons y_k. When an n-dimensional input vector$(x_1, x_2, \dots, x_i, x_{i+1}, \dots, x_n)$ is presented to our fuzzy neural network, its Input-output relations can be written as:

Input Units:

$o_i = \mathrm{H}(x_i) = \epsilon\ (x_i^2 + x_i + 1),\ i = 1, \dots, n,\ \epsilon \in (0,1)$ (12)

Hidden units:

$Z_j = f(net_j),\ j = 1,2,\dots,\ \mathrm{m}$ (13)

$net_j = \sum_{i=1}^{n} o_i\, W_{ji} + B_j$ (14)

Output units:

$y_k = f\ (net_k),\ \mathrm{k} = 1,2,\dots,\ \mathrm{s}$ (15)

$net_k = \sum_{j=1}^{\mathrm{m}} W_{kj}\,.\,Z_j + C_k$ (16)

Where connection weights, biases and target are fuzzy numbers and the inputs are real numbers. The input- output relations in Eqs (12) - (16) is defined by the extension principle. The fuzzy output from each unit in Eqs (12) - (16) is numerically calculated for real inputs and level sets of fuzzy weights and fuzzy biases. The input-output relations of our fuzzy neural network can be written for r-level sets:

Input Units:

$o_i = \mathrm{H}(x_i) = \epsilon\ (x_i^2 + x_i + 1),\ i = 1, \dots, n,\ \epsilon \in (0,1)$ (17)

Hidden Unit:

$[Z_j]_r = f([net_j]_r),\ j = 1,2,\dots,\ \mathrm{m}$ (18)

$[net_j]_r = \sum_{i=1}^{\mathrm{n}} o_i\ [W_{ji}]_r + [B_j]_r$ (19)

Output Units:

$[y_k]_r = f([net_k]_r\ ),\ \mathrm{k} = 1,2,\dots,\ s$ (20)

$[net_k]_r = \sum_{j=1}^{\mathrm{m}} [W_{kj}]_r\ [Z_j]_r + [C_k]_r$ (21)

From Eqs (17) - (21), we can see that the r-level sets of the fuzzy outputs y_k´s are calculated from those of the fuzzy weights, fuzzy biases and crisp inputs. From the operations of fuzzy numbers, the above relations are rewritten as follows when the inputs x_i´s are non – negative, i.e., $x_i \geq 0$:

Input Units:

$o_i = \mathrm{H}(x_i) = \epsilon\ (x_i^2 + x_i + 1),\ i = 1, \dots, n,\ \epsilon \in (0,1)$ (22)

Hidden Unit:

$[Z_j]_r = [[Z_j]_r^L\ ,\ [Z_j]_r^U] = [\ f([net_j]_r^L\ ),\ f([net_j]_r^U\ )\ ]$ (23)

$[net_j]_r^L = \sum_{i=1}^{\mathrm{n}} o_i\ [W_{ji}]_r^L + [B_j]_r^L$ (24)

$[net_j]_r^U = \sum_{i=1}^{\mathrm{n}} o_i\ [W_{ji}]_r^U + [B_j]_r^U$ (25)

Output Units:

$[y_k]_r = [[y_k]_r^L\ ,\ [y_k]_r^U] = [\ f([net_k]_r^L\ ),\ f([net_k]_r^U\ )\ ]$ (26)

$[net_k]_r^L = \sum_{j \in a}\ [W_{kj}]_r^L\ [Z_j]_r^L + \sum_{j \in b}\ [W_{kj}]_r^L\ [Z_j]_r^U + [C_k]_r^L$ (27)

$[net_k]_r^U = \sum_{j \in c}\ [W_{kj}]_r^U\ [Z_j]_r^U + \sum_{j \in d}\ [W_{kj}]_r^U\ [Z_j]_r^L + [C_k]_r^U$ (28)

For $[Z_j]_r^U \geq [Z_j]_r^L \geq 0$, where $a = \{\ \mathrm{j} : [\mathrm{W_{kj}}]_\mathrm{r}^\mathrm{L} \geq 0\}$, $\mathrm{b} = \{\ \mathrm{j} : [\mathrm{W_{kj}}]_\mathrm{r}^\mathrm{L} < 0\}$,
$\mathrm{c} = \{\ \mathrm{j} : [\mathrm{W_{kj}}]_\mathrm{r}^\mathrm{U} \geq 0\}$, $\mathrm{d} = \{\ \mathrm{j} : [\mathrm{W_{kj}}]_\mathrm{r}^\mathrm{U} < 0\}$, $a \cup b = \{1, \dots, m\}$ and $\mathrm{c} \cup d = \{1, \dots, m\}$.

One drawback of the fully fuzzy neural network with fuzzy connection Weights is long Computation time. Another drawback is that the learning algorithm is complicated. For reducing the complexity of the learning algorithm, we propose a partially fuzzy neural network (PFNN) architecture, where connection weights to output units are fuzzy numbers while connection weights and biases to hidden units are real numbers [20,23,24].In this paper, the (PFNN) is of dimension (1 × m × 1) (see Fig.(2)).

For every entry x, the input to the hidden neurons is

$net_j = \epsilon\ (x^2 + x + 1)\ W_j + B_j\ ,\ j = 1,2,\dots\dots,\ \mathrm{m},\ \epsilon \in (0,1)$ (29)

where W_j is a weight parameter from input layer to the j th unit in the hidden layer, B_j is an j th bias for the unit in the hidden layer. The output, in the hidden neurons is

$Z_j = s(net_j), j = 1, 2,\ldots\ldots,m$ (30)

where s(.) is the hyperbolic tangent activation function, and the output N(x, p) of the network is

$N= \sum_{j=1}^{m} v_j Z_j$ (31)

Where v_j is a weight parameter from the j th unit in the hidden layer to the output layer. From Eqs (22)-(28), we can be rewritten for r-level sets of the Eqs (29)-(31). For reducing the complexity of the learning algorithm, the input x usually assumed as non-negative in fuzzy neural network. :

Input Unit:

$o= H(x) = \epsilon (x^2 + x+1), \epsilon \in (0,1)$ (32)

Hidden Unit:

$Z_j = s(net_j), j = 1,2,\ldots,m$ (33)

$net_j = o\, W_j + B_j$ (34)

Output Unit:

$[N]_r = [[N]_r^L, [N]_r^U][\sum_{j=1}^{m}[v_j]_r^L Z_j, \sum_{j=1}^{m}[v_j]_r^U Z_j]$ (35)

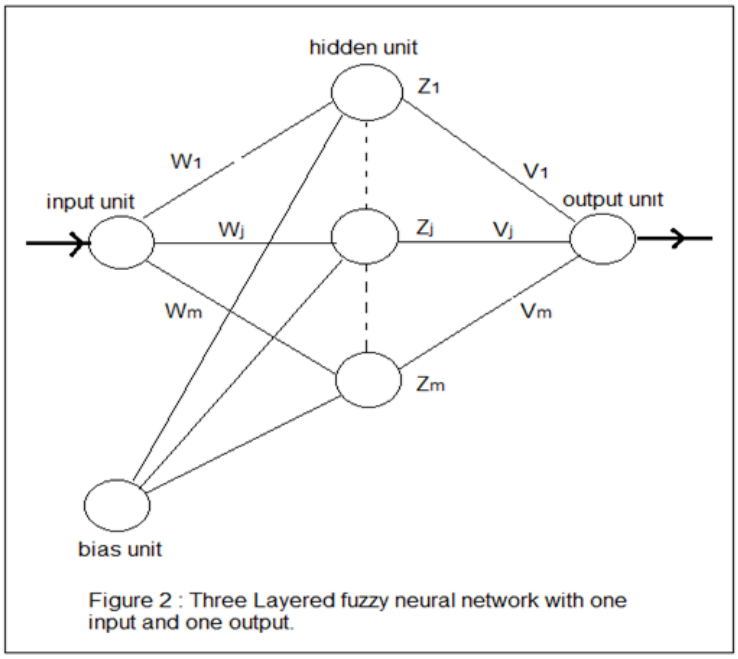

Figure 2 : Three Layered fuzzy neural network with one input and one output.

5.1. Solution of First Order Fuzzy Differential Equation

Let us consider the FDE:

$y' = f(x, y), \quad x \in [a, b]$ and $y(a) = A$ (36)

Where y is a fuzzy function of x, f(x, y)is a fuzzy function of the crisp variable x and the fuzzy variable y, and y' is the fuzzy derivative of y,and A is a fuzzy number in E^1 with r-level set:

$[A]_r = [[A]_r^L, [A]_r^U], \; r \in (0, 1]$.

the related trial function will be in the form :

$y_t(x, p) = A + (x - a) N(x, p)$, (37)

this solution by intention satisfies the initial condition in Eq. (36).

The error function that must be minimized for problem (36) has the form [34, 38, 39]:

$E = \sum_{i=1}^{g} (E_{ir}^L + E_{ir}^U)$,

Where :

$E_{ir}^L = ([\frac{dy_t(x_i,p)}{dx}]_r^L - [f(x_i, y_t(x_i, p)]_r^L)^2$ (38)

$E_{ir}^U = ([\frac{dy_t(x_i,p)}{dx}]_r^U - [f(x_i, y_t(x_i, p)]_r^U)^2$ (39)

and $\{x_i\}_{i=1}^{g}$ are discrete points belonging to the interval $[a,b]$, E_{ir}^{L} and E_{ir}^{U} can be viewed as the squared errors for the $lower\ and\ the\ Upper\ limits\ of\ the\ r-level\ sets.$

Now differentiating the trial function $y_t(x, \mathrm{p})$ in Eq (37), we obtain:

$$\frac{\partial\,[y_t(x,\mathrm{p})]_r^L}{\partial x} = [N(x,p)]_r^L + (x\text{-}a)\,\frac{\partial [N(x,p)]_r^L}{\partial x} \quad (40)$$

$$\frac{\partial\,[y_t(x,\mathrm{p})]_r^U}{\partial x} = [N(x,p)]_r^U + (x\text{-}a)\,\frac{\partial [N(x,p)]_r^U}{\partial x} \quad (41)$$

where

$$[N(x,p)]_r^L = \sum_{j=1}^{m}[v_j]_r^L\ s(H(x)W_j + B_j) \quad (42)$$

$$[N(x,p)]_r^U = \sum_{j=1}^{m}[v_j]_r^U\ s(H(x)W_j + B_j) \quad (43)$$

$$\frac{\partial [N(x,p)]_r^L}{\partial x} = \sum_{j=1}^{m}\epsilon(2x_i + 1)W_j[v_j]_r^L\ s'(H(x)W_j + B_j) \quad (44)$$

$$\frac{\partial [N(x,p)]_r^U}{\partial x} = \sum_{j=1}^{m}\epsilon(2x_i + 1)W_j[v_j]_r^U\ s'(H(x)W_j + B_j) \quad (45)$$

5.2. Solution of Second Order Fuzzy Differential Equation

Now, we consider the second order fuzzy differential equation:

$$y'' = \mathrm{f}(x,y,y'),\ x \in [a,b] \quad (46)$$

$$y(a) = A, y'(a) = B.$$

such that the functions:

$y: [a,b] \to E^1$ and $\mathrm{f}: [a,b] \times E^1 \times E^1 \to E^1$

where y is a function with fuzzy derivative y', also A and B are fuzzy numbers in E^1 with r-level sets :

$[A]_r = [[A]_r^L, [A]_r^U]$, $[B]_r = [[B]_r^L, [B]_r^U]$.

The related trial function will be in the form:

$$y_t(x, \mathrm{p}) = \mathrm{A} + \mathrm{B}(x-a) + (x-a)^2\,\mathrm{N(x, p)}, \quad (47)$$

This solution by intention satisfies the conditions in Eq. (46).

Also, the error function that must be minimized for problem (46) is

$$\mathrm{E} = \sum_{i=1}^{g}(E_{ir}^L + E_{ir}^U),$$

where

$$E_{ir}^L = ([y''_t(x_i,p)]_r^L - \mathrm{f}(x_i, y_t(x_i,p), y'_t(x_i,p))]_r^L)^2 \quad (48)$$

$$E_{ir}^U = ([y''_t(x_i,p)]_r^U - \mathrm{f}(x_i, y_t(x_i,p), y'_t(x_i,p))]_r^U)^2 \quad (49)$$

and $\{x_i\}_{i=1}^{g}$ are discrete points belonging to the interval $[a,b]$.

Now differentiating the trial function $\mathrm{y_t}(x, \mathrm{p})$ in eq. (47), we obtain:

$$\frac{\partial [y_t(x,p)]_r^L}{\partial x} = [B]_r^L + 2(x-a)[N(x,p)]_r^L + (x-a)^2\,\frac{\partial [N(x,p)]_r^L}{\partial x} \quad (50)$$

$$\frac{\partial [y_t(x,p)]_r^U}{\partial x} = [B]_r^U + 2(x-a)[N(x,p)]_r^U + (x-a)^2\,\frac{\partial [N(x,p)]_r^U}{\partial x} \quad (51)$$

$$\frac{\partial^2 [y_t(x,p)]_r^L}{\partial x^2} = 2[N(x,p)]_r^L + 4(x-a)\,\frac{\partial [N(x,p)]_r^L}{\partial X} + (x-a)^2\,\frac{\partial^2 [N(x,p)]_r^L}{\partial x^2} \quad (52)$$

$$\frac{\partial^2 [y_t(x,p)]_r^U}{\partial x^2} = 2[N(x,p)]_r^U + 4(x-a)\,\frac{\partial [N(x,p)]_r^U}{\partial X} + (x-a)^2\,\frac{\partial^2 [N(x,p)]_r^U}{\partial x^2} \quad (53)$$

Where

$$[N(x,p)]_r^L = \sum_{j=1}^{m}[v_j]_r^L\ s(H(x)W_j + B_j) \quad (54)$$

$$[N(x,p)]_r^U = \sum_{j=1}^{m}[v_j]_r^U\ s(H(x)W_j + B_j) \quad (55)$$

$$\frac{\partial [N(x,p)]_r^L}{\partial x} = \sum_{j=1}^{m}\epsilon(2x+1)W_j[v_j]_r^L\ s'(H(x)W_j + B_j) \quad (56)$$

$$\frac{\partial [N(x,p)]_r^U}{\partial x} = \sum_{j=1}^{m}\epsilon(2x+1)W_j[v_j]_r^U\ s'(H(x)W_j + B_j) \quad (57)$$

$$\frac{\partial^2 [N(x,p)]_r^L}{\partial x^2} = \sum_{j=1}^{m}[\,\epsilon^2 W_j^2 (2x+1)^2 \left[v_j\right]_r^L s''(H(x)W_j + B_j) + 2\epsilon W_j [v_j]_r^U\ s'(H(x)W_j + B_j)\,] \quad (58)$$

$$\frac{\partial^2 [N(x,p)]_r^U}{\partial x^2} = \sum_{j=1}^{m}[\,\epsilon^2 W_j^2 (2x+1)^2 \left[v_j\right]_r^U s''(H(x)W_j + B_j) + 2\epsilon W_j [v_j]_r^U\ s'(H(x)W_j + B_j)\,] \quad (59)$$

To find a numerical solution for third (and more) order fuzzy differential equation and and fuzzy partial differential equation by using this method, we apply the same procedure in the subsections (5.1) and (5.2).

6. NUMERICAL EXAMPLES

To show the behavior and properties of the new method, four problem will be solved in this subsection. For each example, the accuracy of the method is illustrated by computing the deviations $\underline{E}(x, r)$ and $\overline{E}(x, r)$
Where $\underline{E}(x,r) = \left|\underline{y_t}(x,r) - \underline{y_a}(x,r)\right|$ *and* $\overline{E}(x,r) = \left|\overline{y_t}(x,r) - \overline{y_a}(x,r)\right|$.

And $y_a(x,r) = \left[\underline{y_a}(x,r), \overline{y_a}(x,r)\right]$ is the known exact solution and $y_t(x,r) = \left[\underline{y_t}(x,r), \overline{y_t}(x,r)\right]$ is the trial (approximated) solution. Note that, for all examples, a multilayer perceptron consisting of one hidden layer with 10 hidden units and one linear output unit is used.We will use BFGS quasi – Newton method to minimize the error function.

Example 1: Consider the following fuzzy initial value problem:
$y' = -y + x + 1,\ x \in [0,1]$
$[y(0)]_r = \left[0.96 + 0.04\,r, 1.01 - 0.01r\right]$, where $r \in [0,1]$.

The fuzzy exact solution is:
$y_a(x,r) = \left[x + (0.96 + 0.04\,r)e^{-x}, x + (1.01 - 0.01\,r)e^{-x}\right]$.

The fuzzy trial solution for this problem is:
$y_t(x,r) = [0.96 + 0.04r, 1.01 - 0.01r] + x\,[N(x,p)]_r$. Fig. (1) shows the exact solution and the approximated solution for $x = 0.1$ and $\epsilon = 0.6$. Numerical results for $x = 0.2$ and $r = 0.5$ can be found in table (1).

Note that for $x = 0.2$ and $r = 0.5$, the fuzzy exact solution is
$y_a(0.2,0.5) = [1.0023561, 1.0228244]$, and $H(x) = \epsilon\,(x^2 + x + 1) = 1.24\,\epsilon$, $\epsilon \in (0,1)$.

ϵ	$\underline{y_t}(0.2, 0.5)$	$\underline{E}(0.2, 0.5)$	$\overline{y_t}(0.2, 0.5)$	$\overline{E}(0.2, 0.5)$
0.1	1.02372368	0.02136754	1.04226973	0.01944532
0.2	1.01359991	0.01124377	1.03861400	0.01578959
0.3	1.00667758	0.00432144	1.02374696	0.00092255
0.4	1.00646736	0.00411122	1.03168062	0.00885621
0.5	1.00312854	0.00077240	1.02335920	0.00053479
0.6	1.00238529	0.00002915	1.02284285	0.00001844
0.7	1.01221038	0.00985424	1.03882331	0.01599890
0.8	1.02857729	0.02622115	1.06269903	0.03987462
0.9	1.03357745	0.03122131	1.09397723	0.07115282

Table-1: Comparison of the exact and approximated solution for example (1), for $x = 0.2$ and $r = 0.5$.

Example 2: Consider the following nonlinear FIVP:
$y'(x) = 3Ay^2,\ x \in [0,0.1]$
$[y(0)]_r = \left[0.5\sqrt{r}, 0.2\sqrt{1-r} + 0.5\right]$, where $r \in [0,1]$, and $A = \left[1+r, 3-r\right]$ is a fuzzy parameter.

For this problem, the fuzzy exact solution is:
$y_a(x,r) = \left[\frac{0.5\sqrt{r}}{1-3(1+r)(0.5\sqrt{r})x}, \frac{0.2\sqrt{1-r}+0.5}{1-3(3+r)(0.2\sqrt{1-r}+0.5)x}\right]$

while the fuzzy trial solution is:
$y_t(x,r) = [0.5\sqrt{r}, 0.2\sqrt{1-r} + 0.5] + x\,[N(x,p)]_r$

Fig. (2) shows the exact solution and the approximated solution for $x = 0.1$ and $\epsilon = 0.7$. Numerical results can be found in table (2). Note that for $r = 0.2$, the fuzzy exact solution is:
$y_a(x,0.2) = \left[\frac{0.22360680}{1-0.80498447\,x}, \frac{0.67888544}{1-5.70263768\,x}\right]$, $H(x) = 0.7(x^2 + x + 1)$.

Table-2: Comparison of the exact and approximated solution for example (2), for r = 0.2 and ϵ =0.7.

x	$\underline{y_a}\,(x,0.2)$	$\underline{y_t}(x,0.2)$	$\underline{\mathrm{E}}\,(x,0.2)$	$\overline{y_a}(x,0.2)$	$\overline{y_t}(x,0.2)$	$\overline{\mathrm{E}}\,(x,0.2)$
0	0.2236068	0.2236068	0.0000000	0.6788854	0.6788854	0.0000000
0.01	0.2254214	0.2254873	0.0000659	0.7199411	0.7199199	0.0000212
0.02	0.2272657	0.2272736	0.0000079	0.7662820	0.7662744	0.0000076
0.03	0.2291404	0.2291883	0.0000479	0.8189991	0.8190363	0.0000372
0.04	0.2310463	0.2310662	0.0000199	0.8795055	0.8794966	0.0000089
0.05	0.2329842	0.2329901	0.0000059	0.9496653	0.9496574	0.0000079
0.06	0.2349549	0.2350173	0.0000624	1.0319890	1.0320011	0.0000121
0.07	0.2369592	0.2369680	0.0000088	1.1299402	1.1299366	0.0000036
0.08	0.2389980	0.2389134	0.0000846	1.2484354	1.2484338	0.0000016
0.09	0.2410721	0.2410563	0.0000158	1.3946951	1.3946728	0.0000223
0.10	0.2431826	0.2431805	0.0000021	1.5797724	1.5797694	0.0000030

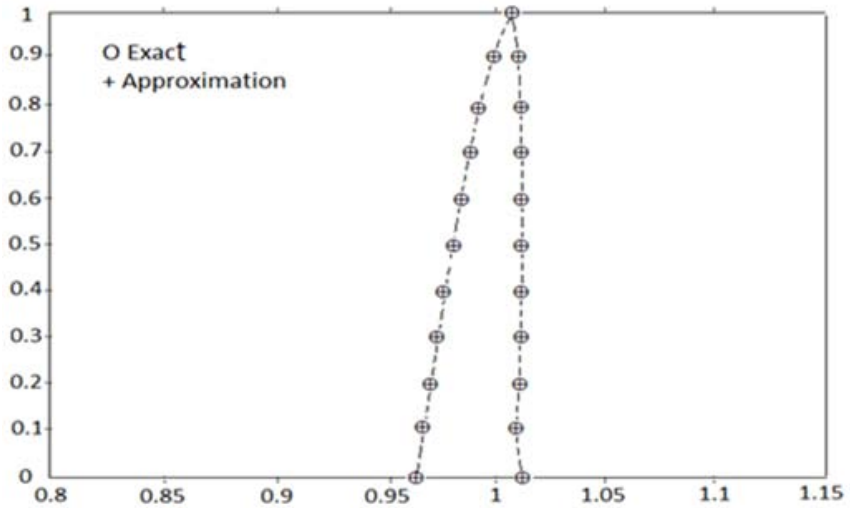

Fig. -1: Analytical and trial solutions for example (1), for x= 0.1and ϵ=0.6

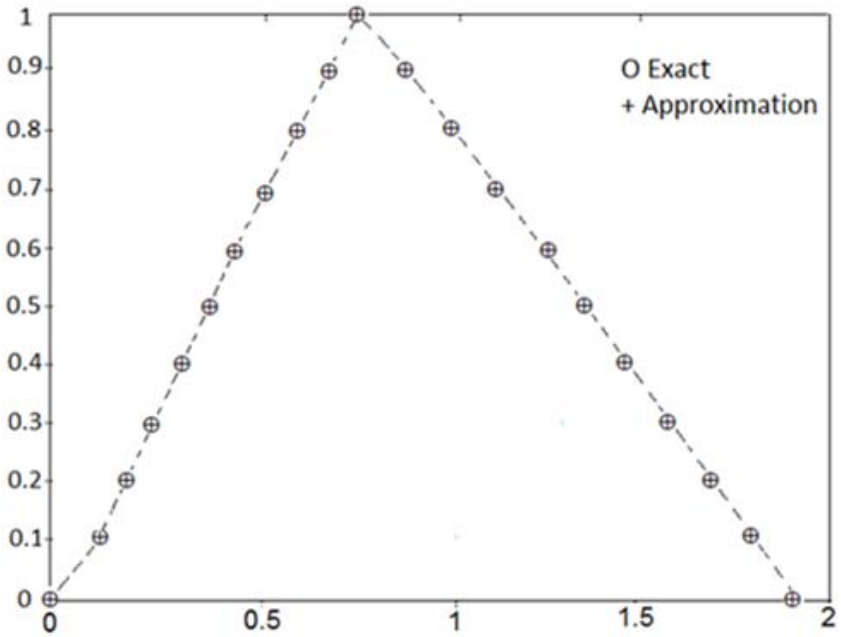

Fig.-2: Analytical and trial solutions for example (2) for x = 0.1 and ϵ = 0.7

Example 3: Consider the homogeneous second order FDE:

$y(\tilde{x}) - 4y(\tilde{x}) + 4y(x)$ =0, $x \in [0,1]$.

$[y(0)]_r = [2 + r\ ,4 - r]$, $[\,y(\tilde{0})]_r$ =[5 + r , 7 - r], r ϵ [0,1].

The fuzzy exact solution is:
$y_a(x,r) = [(2+r)\,e^{2x} + (1-r)xe^{2x}, (4-r)e^{2x} + (r-1)xe^{2x}]$.

The fuzzy trial solution for this problem is:
$y_t(x,r) = [2+r, 4-r] + [5+r, 7-r]x + x^2\,[N(x,p)]_r$.

Table (3) shows the exact solution and the trial solution for x =0.1 and r= 0.5. Numerical results for ϵ= 0.4 and r= 0.5 can be found in table (4)

Note that for x =0.1 and r=0.5, the fuzzy exact solution is:
y_a (0.1, 0.5) = [3.11457703, 4.21383952], $H(x)$ = 1.11ϵ *and* for ϵ=0.4 and r=0.5, the fuzzy exact solution is:
y_a (x, 0.5) = [2.5 e^{2x} + 0.5 xe^{2x}, 3.5 e^{2x} - 0.5 x e^{2x}], $H(x)$=0.4 (x^2+x+1).

ϵ	$\underline{y_t}$(0.1 , 0.5)	$\underline{\mathrm{E}}$ (0.1 , 0.5)	$\overline{y_t}$(0.1 , 0.5)	$\overline{\mathrm{E}}$ (0.1 , 0.5)
0.1	3.0931182	0.0214588	4.2560272	0.0421877
0.2	3.1257025	0.0111255	4.1784181	0.0354215
0.3	3.1110849	0.0034922	4.2063000	0.0075395
0.4	3.1145594	0.0000176	4.2137520	0.0000875
0.5	3.1145965	0.0000195	4.2138825	0.0000430
0.6	3.1151645	0.0005875	4.2137174	0.0000655
0.7	3.1091315	0.0054455	4.2237174	0.0098779
0.8	3.0418927	0.0726844	4.1717159	0.0421237
0.9	3.0647202	0.0498568	4.2464383	0.0325987

Table-3: Comparison of the exact and trial solution for example (3), for x =0.1, r=0.5

x	$\underline{y_a}$ (x,0.5)	$\underline{y_t}$ (x,o.5)	$\underline{\mathrm{E}}$ (x,0.5)	$\overline{y_a}$ (x,0.5)	$\overline{y_t}$ (x,o.5)	$\overline{\mathrm{E}}$(x,o.5)
0	2.5000000	2.5000000	0.0000000	3.5000000	3.5000000	0.0000000
0.1	3.1145770	3.1145594	0.0000176	4.2138395	4.2137520	0.0000875
0.2	3.8787442	3.8788188	0.0000746	5.0722040	5.0722893	0.0000853
0.3	4.8286148	4.8285781	0.0000367	6.1040980	6.1041436	0.0000456
0.4	6.0089605	6.0089507	0.0000098	7.3442851	7.3442226	0.0000625
0.5	7.4752750	7.4752796	0.0000046	8.8344159	8.8344185	0.0000026
0.6	9.2963274	9.2963053	0.0000221	10.6243742	10.6244224	0.0000482
0.7	11.5573199	11.5573149	0.0000050	12.7738799	12.7738714	0.0000085
0.8	14.3637940	14.3637205	0.0000735	15.3544005	15.3544753	0.0000748
0.9	17.8464600	17.8464675	0.0000075	18.4514248	18.4514645	0.0000397
1	22.1671683	22.1671053	0.0000630	22.1671683	22.1671535	0.0000148

Table-4: Comparison of the exact and trial solution for example (3), for ϵ=0.4, r=0.5

Example (4): Consider the nonhomogeneous second order FDE:
$y'' + 4y = 2cosx,\ x \in [0,1]$.
$[y(0)]_r = [2r, 4-2r],\quad [y'(0)]_r = [-2+2r, 2-2r],\ r\,\epsilon\,[0,1]$.

The fuzzy exact solution is:
$y_a(x,r) = [(2r-\frac{2}{3})cos2x + (r-1)sin2x + \frac{2}{3}\cos x, (\frac{10}{3}-2r)\,cos2x + (1-r)sin2x + \frac{2}{3}\cos x]$.

The fuzzy trial solution for this problem is:
$y_t(x$, r) = [2r, 4-2r] + [-2 + 2r, 2-2r] $x + x^2\,[N(x\,,p)]_r$.

Table (5) shows the exact and the trial solution for x = 1 and r = 0.7.

Numerical results for ϵ = 0.8 and r = 0.7 can be found in table (6).

Note that for x = 1 and r= 0.7, the fuzzy exact solution is:
y_a (1, o.7) = [-0.21776204, -0.17155979], $H(x)$ = 3ϵ

And for ϵ = 0.8, r = 0.7, the fuzzy exact solution is
y_a (x , 0.7) = [o.73333 cos2x – 0.3 sin2x + $\frac{2}{3}$cos$x$, 1.93333 cos2x + 0.3 sin2x + $\frac{2}{3}$ cos$x$].

ϵ	$\underline{y_t}(1\ ,\ 0.7)$	$\underline{\mathrm{E}}\ (1\ ,\ 0.7)$	$\overline{y_t}(1,\ 0.7)$	$\overline{\mathrm{E}}\ (1\ ,\ 0.7)$
0.1	-0.3028857	0.0851237	-0.1216841	0.0498757
0.2	-0.2135259	0.0042361	-0.1456746	0.0258852
0.3	-0.2256072	0.0078451	-0.1723489	0.0007891
0.4	-0.2166399	0.0011221	-0.1719289	0.0003691
0.5	-0.2177031	0.0000590	-0.1716162	0.0000564
0.6	-0.2177965	0.0000344	-0.1715105	0.0000493
0.7	-0.2177688	0.0000068	-0.1715653	0.0000055
0.8	-0.2177586	0.0000034	-0.1715626	0.0000028
0.9	-0.2176746	0.0000875	-0.1720495	0.0004898

Table-5: Comparison of the exact and the trial solution for example (4) for x = 1 and r = 0.7.

x	$\underline{y_a}\ (x,0.7)$	$\underline{y_t}\ (x,0.7)$	$\underline{\mathrm{E}}\ (x,0.7)$	$\overline{y_a}(x,0.7)$	$\overline{y_t}(x,0.7)$	$\overline{\mathrm{E}}(x,0.7)$
0	1.4000000	1.4000000	0.0000000	2.6000000	2.6000000	0.0000000
0.1	1.3224508	1.3223751	0.0000757	2.6177966	2.6177966	0.0000643
0.2	1.2119969	1.2120015	0.0000046	2.5509211	2.5509281	0.0000070
0.3	1.0727444	1.0727385	0.0000059	1.9773386	1.9773348	0.0000038
0.4	0.9097521	0.9097470	0.0000051	2.1762138	2.1761849	0.0000289
0.5	0.7288354	0.7289057	0.0000703	1.8820808	1.8821797	0.0000989
0.6	0.5363410	0.5363339	0.0000072	1.5303938	1.5303961	0.0000023
0.7	0.3389024	0.3389635	0.0000611	1.1341329	1.1340851	0.0000478
0.8	0.1431861	0.1431835	0.0000026	0.7078908	0.7078820	0.0000088
0.9	-0.044363	-0.044339	0.0000240	0.2673036	0.2673062	0.0000026
1	-0.217762	-0.217759	0.0000030	-0.171559	-0.171563	0.0000040

Table-6: Comparison of the exact and the trial solution for example (4), for ϵ = 0.8 and r = 0.7.

8. CONCLUSIONS

In this paper, we presented a hybrid approach based on modified fuzzy neural networks for solving fuzzy differential equations. We demonstrate, for the first time, the ability of modified fuzzy neural networks to approximate the solutions of FDEs. By comparing our results with the results obtained by using numerical methods, it can be observed that the proposed method yields more accurate approximations. Even better results may be possible if one uses more neurons or more training points. Moreover, after solving a FDE the solution is obtainable at any arbitrary point in the training interval (even between training points). The main reason for using modified fuzzy neural networks was their applicability in function approximation. Further research is in progress to apply and extend this method to solve fuzzy partial differential equations FPDEs.

REFERENCES

1. S. Abbasbandy and T. Allahviranloo, Numerical solution of fuzzy Differential equations by Runge-Kutta method, J. Sci. Teacher Training University, 1(3), 2002.
2. S. Abbasbandy and T. Allahviranloo, Numerical solution of fuzzy Differential equations by Taylor method, Journal of Computational Methods in Applied Mathematics, 2 (2002), 113-124.
3. T. Allahviranloo, N. Ahmady, and E. Ahmady, Numerical solution of fuzzy Differential equations by predictor- corrector method, Information Sciences, 177 (2007), 1633-1647.
4. T. Allahviranloo, E. Ahmady, and N. Ahmady, Nth- order fuzzy linear Differential equations , Information Sciences, 178 (2008), 1309-1324.
5. T. Allahviranloo,N. A. Kiani, and N. Motamedi, Solving fuzzy differential equations by differential transformation method, Information Sciences,179 (2009), 956-966.
6. S.Abbasbandy, M. Otadi, Numerical solution of fuzzy polynomials by fuzzy neural network, Appl. Math. Comput. 181 (2006) 1084 -1089.
7. S.Abbasbandy, M. Otadi, M. Mosleh, Numerical solution of a system of fuzzy polynomials by fuzzy neural network, Information sciences 178 (2008) 1948 – 1960.
8. B. Bede and S. G. Gal, Generalizations of the differentiability of fuzzy number- valued functions with applications to fuzzy differential equations, Fuzzy Sets and Systems, 151 (2005), 581 – 599.
9. J.J. Buckley and T. Feuring, Fuzzy differential equations, Fuzzy Sets and Systems, 110 (2000), 43 – 54.
10. P.Diamond, Stability and periodicity in fuzzy differential equations, IEEE Trans, Fuzzy Systems, 8 (2000), 583 – 590.

11. P. Diamond, Brief note on the variation of constants formula for fuzzy differential equations, Fuzzy Sets and Systems, 129 (2002), 65 – 71.
12. S. Ezadi, N. Parandin, A. Ghomashi, Numerical solution of fuzzy differential equations based on semi-Taylor by using neural network, Journal of basic and applied scientific research (2013), 477-482.
13. B. Ghazanfari and A. Shakerami, Numerical solution of fuzzy differential equations extended Runge – Kutta-like formulae of order 4, Fuzzy Sets and Systems, 189 (2011), 74 – 91.
14. D.N.Georgiou, J.J.Nieto, and R. Rodriguez-Lopez, Initial value problems for higher-order fuzzy differential equations, Nonlinear Anal.,63 (2005),587-600.
15. K. Hornick, M. Stinchcombe, Multi-layer feed forward networks are universal approximators, Neural Networks 2 (1989) 359 - 366.
16. O. Kaleva, Fuzzy differential equations, Fuzzy Sets and Sydtems, 24 (1987), 301 – 317.
17. T.Khanna, Foundations of neural networks, Addison-Wesly, Reading, MA,(1990).
18. A. Kastan and K. Ivaz, Numerical solution of fuzzy differential equations by Nystrom method. Chaos, Solitons and Fractals, 41(2009), 859 – 868.
19. H. Lee and I. S. Kang, Neural algorithms for solving differential equations, Journal of Computational Physics, 91 (1990), 110-131.
20. M.Mosleh, Fuzzy neural network for solving a system of fuzzy differential equations, Applied Soft Computing, 13(2013), 3597-3607.
21. M.Mosleh, T. Allahviranloo, and M.Otadi, Evaluation of fully fuzzy regression models by fuzzy neural network, Neural Comput and Applications,21 (2012), 105 – 112.
22. A. J. Meade and A.A. Fernandez, The numerical solution of linear ordinary differential equations by feed-forward neural network, Mathematical and Computer Modelling, 19(12) (1994), 1 – 25.
23. M. Mosleh, M. Otadi, Simulation and evaluation of fuzzy differential equation by fuzzy neural network, Applied soft computing 12 (2012) 2817 - 2827.
24. M.Mosleh,M.Otadi, Solving the second order fuzzy differential equations by fuzzy neural network, Journal of Mathematical Extension 81 (2014), 11 – 27.
25. M.Mosleh,M.Otadi, and S.Abbasbandy, Evaluation of fuzzy regression models by fuzzy neural network, Journal of Computational and Applied Mathematics,234 (2010), 825 – 834.
26. M.Mosleh,M.Otadi, and S.Abbasbandy, Fuzzy polynomial regression with fuzzy neural network, Applied Mathematical Modeling,35 (2011), 5400 – 5412.
27. A.Malek and R. Shekari Beidokhti, Numerical solution for high order differential equations using a hybrid neural network-Optimization method, Appl. Math. Comput. 183 (2006), 260 – 271.
28. J .J. Nieto, The Cauchy problem for continuous fuzzy differential equations, Fuzzy Sets and Systems, 102 (1999), 259 – 262.
29. J.J. Nieto and R.Rodriguez-Lopez, Bounded solutions for fuzzy differential and integral equations, Chaos, Solitons and Fractals, 27 (2006), 1376 – 1386.
30. M. Otadi, M. Mosleh, S. Abbasbandy, solving fuzzy linear system by neural network and applications in Economics, Journal of mathematics extension (2011) 47 - 66.
31. M. Otadi. M.Mosleh, and S. Abbasbandy, Numerical solution of fully fuzzy linear systems by fuzzy neural network, Soft Computing, 15 (2011), 1513 – 1522.
32. H. Ouyang and Y. Wu, On fuzzy differential equations, Fuzzy Sets and Systems, 32 (1989), 321 – 325.

Source of support: Nil, Conflict of interest: None Declared