ISSN 2347-1921

Numerical Solution of Fuzzy Differential Equations Based on Taylor Series by Using Fuzzy Neural Networks

Eman A. Hussian , Mazin H. Suhhiem
Dep.of Mathematics,College of Sciences,AL-Mustansiriyah University,Baghdad,Iraq
Dep. of statistics, College of Adm. and Econ., University of sumar, Alrefiey,Iraq.

ABSTRACT

In this paper a new method based on learning algorithm of Fuzzy neural network and Taylor series has been developed for obtaining numerical solution of fuzzy differential equations.A fuzzy trial solution of the fuzzy initial value problem is written as a sum of two parts.The first part satisfies the fuzzy initial condition,it contains Taylor series and involves no fuzzy adjustable parameters.The second part involves a feed-forward fuzzy neural network containing fuzzy adjustable parameters (the fuzzy weights).Hence by construction,the fuzzy initial condition is satisfied and the fuzzy network is trained to satisfy the fuzzy differential equation . In comparison with existing similar neural networks,the proposed method provides solutions with high accuracy.Finally , we illustrate our approach by two numerical examples .

Keywords: Fuzzy differential equation ; Fuzzy neural network ; Feed-forward neural network ; BFGS method ;Hyperbolic tangent function .

Council for Innovative Research
Peer Review Research Publishing System
Journal: JOURNAL OF ADVANCES IN MATHEMATICS
Vol .11, No.3
www.cirjam.com , editorjam@gmail.com

ISSN 2347-1921

1. INTRODUCTION

Nowadays, fuzzy differential equations (FDEs) 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 which satisfies fuzzy initial or boundary conditions. The theory of fuzzy differential equations was treated by Kaleva [16], Ouyang and Wu [32], Khanna[17], Nieto[28], Buckley and Feuring [9], Seikkala also recently there appeared the papers of Bede, Bede and Gal [8], Diamond [10,11], Georgiou, Nieto and et al. [14] ,Nieto and Rodriguez-Lopez [29].

In the following, we have mentioned some numerical solution which have proposed by other scientists. Abbasbandy and Allahviranloo have solved fuzzy differential equations by Runge-Kuta and Taylor methods [1,2]. Also , Allahviranloo and 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 proposed a new numerical method to find the approximate solution of FDEs , this method can result in improved numerical methods for solving FDEs. 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 work is a three-Layer feed- forward neural network where connection weights and biases are fuzzy numbers .

The proposed method based on Taylor series.In fact, we will multiplying the fuzzy initial condition by a suitable Taylor series provided that the fuzzy trial solution must satisfy the fuzzy initial / boundary conditions, Therefore ,many Taylor serieses with respect to many functions can be used such as $e^x, cosx, coshx, etc$ .In [12] ,for the first time, Ezadi,Parandin and et al. used usual neural network based on the semi-Taylor series (with respect to the function e^x) to solve the first order FDEs . Here , we will use the same concepts in [12] ,but we will use fuzzy neural network instead of usual neural network .We will describe this new method for the first (and second) order FDE,and one can use the same procedure to solve high order FDE and fuzzy partial differential equation .The accuracy of this method depend mainly on the Taylor series which we choose for the trial solution.Of course, this chosen is not unique,therefore,the accuracy is different from problem to other.In general,this modified method is effective for solving FDEs .

2. PRELIMINARIES

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

Definition 2.1 [28]: A fuzzy number u is completely determined by any pair u= $(\underline{u}, \bar{u})$ of functions $\underline{u}(r)$, $\bar{u}(r) : R \longrightarrow [0,1]$ satisfying the conditions:

(1) $\underline{u}(r)$ is a bounded, monotonic, increasing (non – decreasing) left continuous function for all $r \in (0,1]$ and right continuous for r=0.

(2) $\bar{u}(r)$ is a bounded, monotonic, decreasing (non – increasing) left continuous function for all $r \in (0,1]$ and right continuous for r=0.

(3) For all $r \in (0,1]$ we have $\underline{u}(r) \leq \bar{u}(r)$.

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 $Eqs.(1) \longrightarrow (3)$ is denoted by E^1 . For $r \in (0,1]$, we define

the r - cuts of fuzzy number u with $[u]_r = \{x \in R | u(x) \geq r\}$ 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 $\longrightarrow$ 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 [14]: let u , 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 ∈ [a,b].We say that f is H-differential (Hukuhara-differential) at t_0 , if there exists an element f′(t_0) ∈ 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

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

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 and fuzzy rules) by exploiting approximation from neural network . Combining fuzzy system with neural network. Both neural network and fuzzy system have some things in common [7] .

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 [15] .

4. OPERATIONS OF FUZZY NUMBERS

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 [21]:

$$(1)\ \mu_{A+B}(z) = \text{Max}\{\mu_A(x) \wedge \mu_B(y) \mid z = x + y\} \quad (5)$$

$$(2)\ \mu_{AB}(z) = \text{Max}\{\mu_A(x) \wedge \mu_B(y) \mid z = x\,y\} \quad (6)$$

$$(3)\ \mu_{f(net)}(z) = \text{Max}\{\mu_{net}(x) \mid z = f(x)\} \quad (7)$$

Where A,B and net are fuzzy number , $\mu(*)$ denotes the membership function of each fuzzy number, ∧ 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) .

The r-level set of a fuzzy number A is defined as: $[A]_r = \{ x \in R \mid \mu_A(x) \geq r \}$, $0 < r \leq 1$ (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 , from interval arithmetic , the above operations of fuzzy number are written for r-level set as follows:

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

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

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

5. INPUT-OUTPUT RELATIONS OF EACH UNIT

Let us consider a fuzzy three – layer feed – forward neural network with n input units , m hidden units and s output units . Target vector , connection weights and biases are fuzzy numbers and input vector is real number. For convenience in this discussion , FNNM with an input layer, a single hidden layer, and an output layer in Fig. (1) 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 [20] .

ISSN 2347-1921

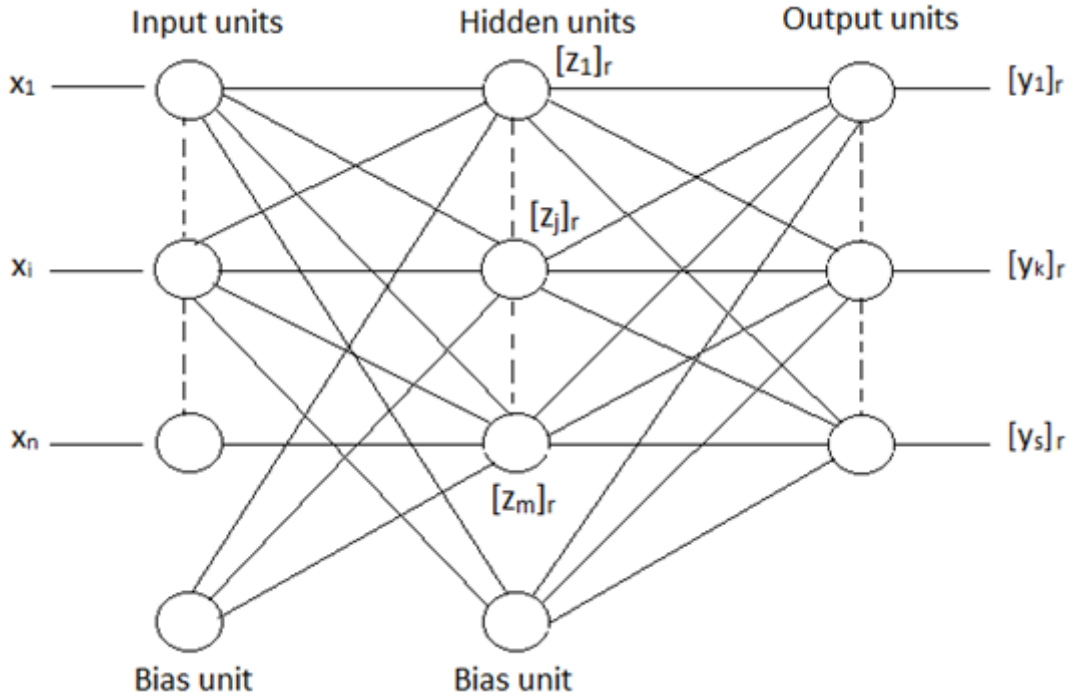

Fig 1 :Three-layer feed-forward Fuzzy neural network .

The architecture of the model shows how FNNM transforms the n inputs $(x_1, x_2, \dots, x_i, \dots, x_n)$ into the s fuzzy outputs $([y_1]_r, [y_2]_r, \dots [y_k]_r, \dots [y_s]_r)$ throughout the m hidden fuzzy neurons $([z_1]_r, [z_2]_r, \dots [z_j]_r, \dots [z_m]_r)$, where the cycles represent the neurons in each layer. Let $[b_j]_r$ be the fuzzy bias for the fuzzy neuron $[z_j]_r$, $[c_k]_r$ be the fuzzy bias for the fuzzy neuron $[y_k]_r$, $[w_{ji}]_r$ be the fuzzy weight connecting crisp neuron x_i to fuzzy neuron $[z_j]_r$, and $[w_{kj}]_r$ be the fuzzy weight connecting fuzzy neuron $[z_j]_r$ to fuzzy neuron $[y_k]_r$.

When an n – dimensional input vector $(x_1, x_2, \dots, x_i, \dots, x_n)$ is presented to our fuzzy neural network , its input – output relation can be written as follows , where $F : R^n \longrightarrow E^s$:

Input units :

$$o_i = x_i \quad , \quad i = 1,2,3, \dots n \tag{12}$$

Hidden units :

$$z_j = F\left(net_j\right) \quad , \quad j = 1,2,3, \dots,m, \tag{13}$$

$$net_j = \sum_{i=1}^{n} o_i \, w_{ji} + b_j \tag{14}$$

Output units :

$$y_k = F\left(net_k\right) \quad , \quad k = 1,2,3, \dots, s, \tag{15}$$

$$net_k = \sum_{j=1}^{m} w_{kj} \, z_j + c_k \tag{16}$$

The architecture of our fuzzy neural network is shown in Fig.(1) , where connection weights , biases , and targets are fuzzy numbers and inputs are real numbers. The input – output relation in Eqs. $(12 – 16)$ is defined by the extension principle .

6. CALCULATION OF FUZZY OUTPUT

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 the r – level sets [23] :

Input units :

$$o_i = x_i \quad , \quad i = 1,2,3, \dots n \tag{17}$$

Hidden units :

$$[z_j]_r = F\left([net_j]_r\right) \quad , j = 1,2,3, \dots,m, \tag{18}$$

$$[net_j]_r = \sum_{i=1}^{n} o_i \, [w_{ji}]_r + [b_j]_r \tag{19}$$

Output units :

$$[y_k]_r = F\left([net_k]_r\right) \quad , k = 1,2,3, \dots, s, \tag{20}$$

ISSN 2347-1921

$$[\text{net}_k]_r = \sum_{j=1}^{m} [w_{kj}]_r \, [z_j]_r + [c_k]_r \ . \quad (21)$$

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 the 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 :

$$o_i = x_i \quad (22)$$

Hidden units :

$$[z_j]_r = F\left([\text{net}_j]_r\right) = \left[[z_j]_r^L \, , \ [z_j]_r^U\right] = . \left[F\left([\text{net}_j]_r^L\right) , F\left([\text{net}_j]_r^U\right)\right] \quad (23)$$

where

$$[\text{net}_j]_r^L = \sum_{i=1}^{n} o_i \, [w_{ji}]_r^L + [b_j]_r^L \quad (24)$$

$$[\text{net}_j]_r^U = \sum_{i=1}^{n} o_i \, [w_{ji}]_r^U + [b_j]_r^U \quad (25)$$

Output units :

$$[y_k]_r = F\left([\text{net}_k]_r\right) = [[y_k]_r^L \, , \ [y_k]_r^U] = . \left[F([\text{net}_k]_r^L) , F([\text{net}_k]_r^U)\right] \quad (26)$$

$$[\text{net}_k]_r^L = \sum_{j \in a} [w_{kj}]_r^L \ [z_j]_r^L + \sum_{j \in b} [w_{kj}]_r^L \ [z_j]_r^U + [c_k]_r^L \quad (27)$$

$$[\text{net}_k]_r^U = \sum_{j \in c} [w_{kj}]_r^U \ [z_j]_r^U + \sum_{j \in d} [w_{kj}]_r^U \ [z_j]_r^L + [c_k]_r^U \quad (28)$$

For $[z_j]_r^U \geq [z_j]_r^L \geq 0$, where

$a = \left\{j : [w_{kj}]_r^L \geq 0\right\}$, $b = \left\{j : [w_{kj}]_r^L < 0\right\}$, $c = \left\{j : [w_{kj}]_r^U \geq 0\right\}$, $d = \left\{j : [w_{kj}]_r^U < 0\right\}$ and

$a \cup b = \{1,2,3, \ldots, m\}$ and $c \cup d = \{1,2,3, \ldots, m\}$.

7.FUZZY NEURAL NETWORK APPROACH FOR SOLVING FDEs

To solve any fuzzy ordinary differential equation (i.e., first order FDE , second order FDE ,etc.) we consider a three – layered fuzzy neural network model (FNNM) with one unit entry x , one hidden layer consisting of m activation functions and one unit output $N(x, p)$. The activation function for the hidden units of our fuzzy neural network is hyperbolic tangent function.Here, the dimension of FNNM is (1 x m x 1).

For every entry x the input neuron makes no changes in its input, so the input to the hidden neurons is:

$$\text{net}_j = x \, w_j + b_j \quad , j = 1,2,3, \ldots,m, \quad (29)$$

Where w_j is a weight parameter from input layer to the jth unit in the hidden layer, b_j is an jth bias for the jth unit in the hidden layer.

The output , in the hidden neurons is :

$$z_j = s\left(\text{net}_j\right) \quad , j = 1,2,3, \ldots,m, \quad (30)$$

Where s is the hyperbolic tangent activation function . The output neuron make no change in its input , so the input to the output neuron is equal to output :

$N = v_1 \, z_1 + v_2 \, z_2 + v_3 \, z_3 + \ldots + v_j \, z_j + \ldots + v_m \, z_m = \sum_{j=1}^{m} v_j \, z_j$ (31) Where v_j is a weight parameter from jth unit in the hidden layer to the output layer.

From Eqs. (22 – 28) , we can see that the r – level sets of the Eqs. (29 – 31) are calculated from those of the fuzzy weights , fuzzy biases and crisp inputs (Fig.2). For our fuzzy neural network , we can derive the learning algorithm without assuming that the input x is non – negative. For reducing the complexity of the learning algorithm , input x usually assumed as non – negative in the fuzzy neural network, i.e., $x \geq 0$ [20] :

Input unit :

$$o = x \ , \quad (32)$$

ISSN 2347-1921

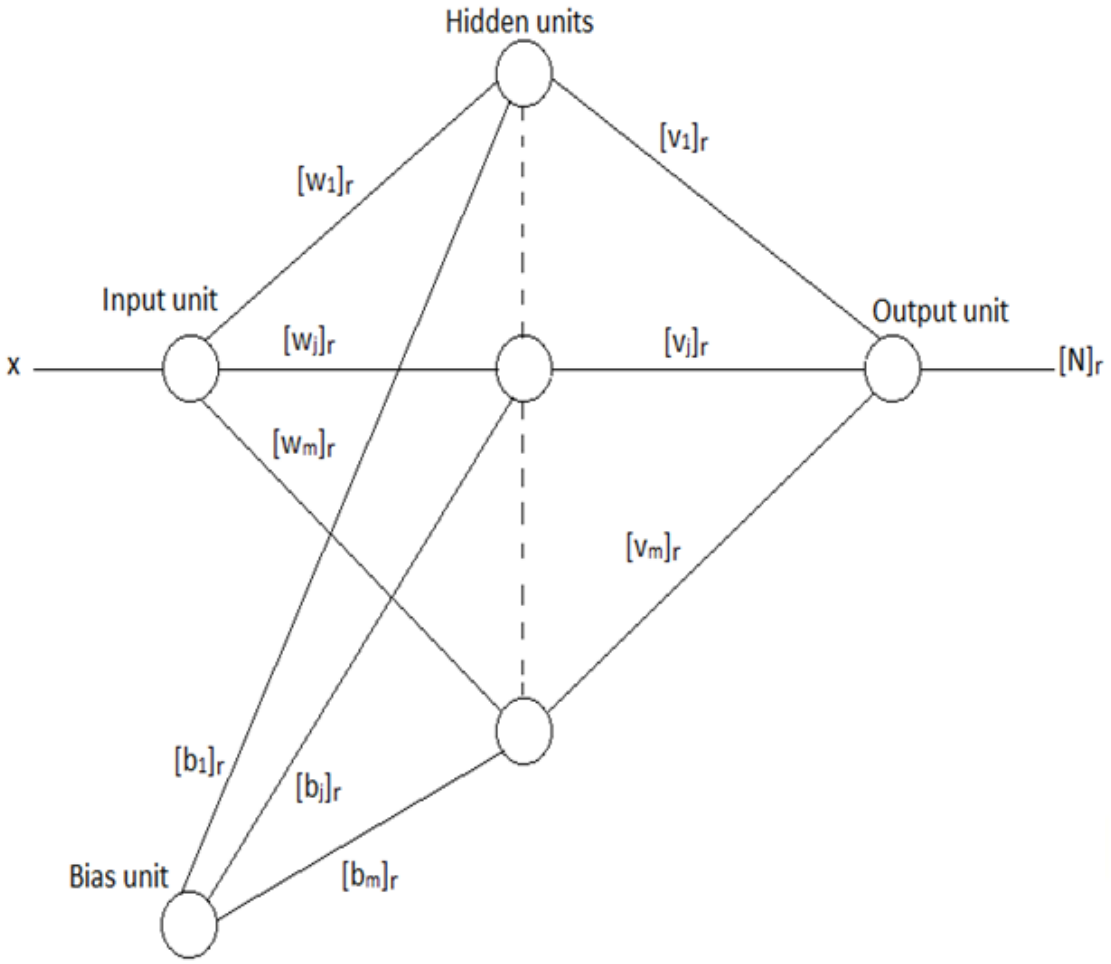

Fig 2 : (1 x m x 1) Feed-forward fuzzy neural network .

Hidden units :

$$[z_j]_r = \left[[z_j]_r^L,\ [z_j]_r^U\right] = \left[s\left([net_j]_r^L\right), s\left([net_j]_r^U\right)\right] \quad (33)$$

Where $[net_j]_r^L = o\,[w_j]_r^L + [b_j]_r^L$ and $[net_j]_r^U = o\,[w_j]_r^U + [b_j]_r^U$

Output unit : $[N]_r = [[N]_r^L,\ [N]_r^U]$, where

$$[N]_r^L = \sum_{j\in a}[v_j]_r^L\,[z_j]_r^L + \sum_{j\in b}[v_j]_r^L\,[z_j]_r^U \quad (34)$$

$$[N]_r^U = \sum_{j\in c}[v_j]_r^U\,[z_j]_r^U + \sum_{j\in d}[v_j]_r^U\,[z_j]_r^L \quad (35)$$

For $[z_j]_r^U \geq [z_j]_r^L \geq 0$, where : $a = \{j : [v_j]_r^L \geq 0\}$, $b = \{j : [v_j]_r^L < 0\}$ $c = \{j : [v_j]_r^U \geq 0\}$,

$d = \{j : [v_j]_r^U < 0\}$ and $a \cup b = \{1,2,\dots m\}$ and $c \cup d = \{1,2,\dots m\}$.

For illustration the solution steps, we will consider the first order fuzzy differential equation [23] :

$$\frac{d\,y\,(x)}{dx} = F\,(x, y) \quad , \ x \in [a, b] \ , y\,(a) = A \quad (36)$$

Where A is a fuzzy number in E^1 with r – level sets :

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

The fuzzy trial solution for this problem is :

$$[y_t(x, p)]_r = [A]_r + (x - a)\,[N(x, p)]_r \quad (37)$$

This fuzzy solution by intention satisfies the fuzzy initial condition in (36)

ISSN 2347-1921

The error function that must be minimized for the problem (36) is in the form : E = $\sum_{i=1}^{g}\left(E_{ir}^{L} + E_{ir}^{U}\right)$ (38)

$$E_{ir}^{L} = \left[\left[\frac{d\, y_t\,(x_i\,,p)}{dx}\right]_r^L - \left[F\left(x_i\,, y_t\,(x_i\,,p)\right)\right]_r^L \right]^2 \quad (39)$$

$$E_{ir}^{U} = \left[\left[\frac{d\, y_t\,(x_i\,,p)}{dx}\right]_r^U - \left[F\left(x_i\,, y_t\,(x_i\,,p)\right)\right]_r^U \right]^2 \quad (40)$$

Where $\{x_i\}_{i=1}^{g}$ are discrete points belonging to the interval $[a\,,b]$ (training set) and in the cost function (38) , E_r^L and E_r^U can be viewed as the squared errors for the lower and upper limits of the r – level sets . It is easy to express the first derivative of $[N(x\,,p)]_r$ in terms of the derivative of the hyperbolic tangent , i.e.,

$$\frac{\partial\,[N]_r^L}{\partial x} = \sum_a [v_j]_r^L \frac{\partial\,[z_j]_r^L}{\partial\,[net_j]_r^L} \frac{\partial\,[net_j]_r^L}{\partial x} + \sum_b [v_j]_r^L \frac{\partial\,[z_j]_r^U}{\partial\,[net_j]_r^U} \frac{\partial\,[net_j]_r^U}{\partial x} \quad (41)$$

$$\frac{\partial\,[N]_r^U}{\partial x} = \sum_c [v_j]_r^U \frac{\partial\,[z_j]_r^U}{\partial\,[net_j]_r^U} \frac{\partial\,[net_j]_r^U}{\partial x} + \sum_d [v_j]_r^U \frac{\partial\,[z_j]_r^L}{\partial\,[net_j]_r^L} \frac{\partial\,[net_j]_r^L}{\partial x} \quad (42)$$

Where $a = \left\{j : [v_j]_r^L \geq 0\right\}$, $b = \left\{j : [v_j]_r^L < 0\right\}$, $c = \left\{j : [v_j]_r^U \geq 0\right\}$, $d = \left\{j : [v_j]_r^U < 0\right\}$ and

$a \cup b = \{1,2,3,\ldots m\}$ and $c \cup d = \{1,2,3,\ldots m\}$. Also we have

$$\frac{\partial\,[net_j]_r^L}{\partial x} = [w_j]_r^L \quad , \quad \frac{\partial\,[net_j]_r^U}{\partial x} = [w_j]_r^U \quad , \quad \frac{\partial\,[z_j]_r^L}{\partial\,[net_j]_r^L} = 1 - \left([z_j]_r^L\right)^2 \quad \text{and} \quad \frac{\partial\,[z_j]_r^U}{\partial\,[net_j]_r^U} = 1 - \left([z_j]_r^U\right)^2$$

Now differentiating from fuzzy trial function $[y_t(x\,,p)]_r$ in (39) and (40) we obtain :

$$\frac{[y_t\,(x\,,p)]_r^L}{\partial x} = [N\,(x\,,p)]_r^L + (x-a)\frac{\partial\,[N\,(x\,,p)]_r^L}{\partial x} \quad (43)$$

$$\frac{[y_t\,(x\,,p)]_r^U}{\partial x} = [N\,(x\,,p)]_r^U + (x-a)\frac{\partial\,[N\,(x\,,p)]_r^U}{\partial x} \quad (44)$$

Therefore , we get

$$E_{ir}^{L} = \left[\begin{array}{c} \sum_a [v_j]_r^L\,[z_j]_r^L + \sum_b [v_j]_r^L\,[z_j]_r^U + (x_i - a) \\ \left(\sum_a [v_j]_r^L\,[w_j]_r^L \left(1 - \left([z_j]_r^L\right)\right) + \sum_b [v_j]_r^L\,[w_j]_r^U \left(1 - \left([z_j]_r^U\right)\right)\right) \\ -F\left(x_i, [A]_r^L + (x_i - a)\left(\sum_a [v_j]_r^L\,[z_j]_r^L + \sum_b [v_j]_r^L\,[z_j]_r^U\right)\right) \end{array}\right]^2 \quad (45)$$

$$E_{ir}^{U} = \left[\begin{array}{c} \sum_c [v_j]_r^U\,[z_j]_r^U + \sum_d [v_j]_r^U\,[z_j]_r^L + (x_i - a) \\ \left(\sum_c [v_j]_r^U\,[w_j]_r^U \left(1 - \left([z_j]_r^U\right)\right) + \sum_d [v_j]_r^U\,[w_j]_r^L \left(1 - \left([z_j]_r^L\right)\right)\right) \\ -F\left(x_i, [A]_r^U + (x_i - a)\left(\sum_c [v_j]_r^U\,[z_j]_r^U + \sum_d [v_j]_r^U\,[z_j]_r^L\right)\right) \end{array}\right]^2 \quad (46)$$

Now we substitute (45) and (46) in (38) to find the error function that must be minimized for problem (36).

For the higher order fuzzy ordinary differential equations and fuzzy partial differential equations eq. (45) and eq. (46) will be very complex and the computations are very difficult.

Therefore, for reducing the complexity of the learning algorithm, we will propose a partially fuzzy neural network in the next section.

8. PARTIALLY FUZZY NEURAL NETWORKS

One drawback of the fully fuzzy neural networks with fuzzy connection weights is long computation time. Another drawback is that the learning algorithm is complicated. Therefore, for reducing the complexity of the learning algorithm, a partially fuzzy neural network (PFNN) architecture has been proposed where connection weights to the output unit are fuzzy numbers while connection weights and biases to the hidden units are real numbers .[23,24] .

The input – output relation of each unit of our partially fuzzy neural network in Eqs. (32-35) can be rewritten for r – level sets as follows :

Input unit : $o = x$

Hidden units : $z_j = s\left(net_j\right)$, $j = 1,2,3,\ldots m$

Where $net_j = o\, w_j + b_j$

Output unit : $[N]_r = [[N]_r^L\,,\ [N]_r^U] = \left[\sum_{j=1}^{m} [v_j]_r^L z_j \ , \sum_{j=1}^{m} [v_j]_r^U z_j\right]$

ISSN 2347-1921

Now to find the minimized error function(with PFNN) for problem (36) :

$$\frac{\partial\,[\mathrm{N}]_{\mathrm{r}}^{\mathrm{L}}}{\partial \mathrm{x}} = \sum_{j=1}^{\mathrm{m}} [\mathrm{v}_j]_{\mathrm{r}}^{\mathrm{L}}\ \frac{\partial\,\mathrm{z}_j}{\partial\,\mathrm{net}_j}\ \frac{\partial\,\mathrm{net}_j}{\partial \mathrm{x}} = \sum_{j=1}^{\mathrm{m}} [\mathrm{v}_j]_{\mathrm{r}}^{\mathrm{L}}\ \mathrm{w}_j\ (1-\ \mathrm{z}_j{}^2) \qquad (47)$$

$$\frac{\partial\,[\mathrm{N}]_{\mathrm{r}}^{\mathrm{U}}}{\partial \mathrm{x}} = \sum_{j=1}^{\mathrm{m}} [\mathrm{v}_j]_{\mathrm{r}}^{\mathrm{U}}\ \frac{\partial\,\mathrm{z}_j}{\partial\,\mathrm{net}_j}\ \frac{\partial\,\mathrm{net}_j}{\partial \mathrm{x}} = \sum_{j=1}^{\mathrm{m}} [\mathrm{v}_j]_{\mathrm{r}}^{\mathrm{U}}\ \mathrm{w}_j\ (1-\ \mathrm{z}_j{}^2) \qquad (48)$$

By substituting Eqs (47 and 48) in Eqs (39 and 40) , we obtain :

$$\mathrm{E}_{\mathrm{ir}}^{\mathrm{L}} = \begin{bmatrix} \sum_{j=1}^{\mathrm{m}} \mathrm{z}_j\ [\mathrm{v}_j]_{\mathrm{r}}^{\mathrm{L}} + (\mathrm{x_i} - \mathrm{a})\ \sum_{j=1}^{\mathrm{m}} \mathrm{w}_j\ (1-\ \mathrm{z}_j{}^2)[\mathrm{v}_j]_{\mathrm{r}}^{\mathrm{L}} \\ -\mathrm{F}\left(\mathrm{x_i}, [A]_{\mathrm{r}}^{\mathrm{L}} + (\mathrm{x_i} - \mathrm{a})\ \sum_{j=1}^{\mathrm{m}} \mathrm{z}_j\ [\mathrm{v}_j]_{\mathrm{r}}^{\mathrm{L}}\right) \end{bmatrix}^2 \qquad (49)$$

$$\mathrm{E}_{\mathrm{ir}}^{\mathrm{U}} = \begin{bmatrix} \sum_{j=1}^{\mathrm{m}} \mathrm{z}_j\ [\mathrm{v}_j]_{\mathrm{r}}^{\mathrm{U}} + (\mathrm{x_i} - \mathrm{a})\ \sum_{j=1}^{\mathrm{m}} \mathrm{w}_j\ (1-\ \mathrm{z}_j{}^2)[\mathrm{v}_j]_{\mathrm{r}}^{\mathrm{U}} \\ -\mathrm{F}\left(\mathrm{x_i}, [A]_{\mathrm{r}}^{\mathrm{U}} + (\mathrm{x_i} - \mathrm{a})\ \sum_{j=1}^{\mathrm{m}} \mathrm{z}_j\ [\mathrm{v}_j]_{\mathrm{r}}^{\mathrm{U}}\right) \end{bmatrix}^2 \qquad (50)$$

And then we substitute (49) and (50) in (38) to find the error function that must be minimized for problem (36) (with respect to PFNN).

9. THE PROPOSED METHOD

In this section , we will discuss how we can find an approximate solution for the fuzzy differential equations by using the fuzzy neural networks based on Taylor series.In [12] , Ezadi,Parandin and et al. used the usual neural network based on the semi-Taylor series to solve the first order fuzzy differential equations . Here , we will use the same concepts in [12] ,but we will use fuzzy neural network instead of usual neural network .We will describe this new method for the first (and second) order fuzzy differential equation,and one can use the same procedure third (and more)order fuzzy differential equation and fuzzy partial differential equation .

9.1. Solution of First Order Fuzzy Differential Equation

Again , if we consider the first order fuzzy differential equation :

$$\frac{\mathrm{d}\,y(x)}{\mathrm{d}x} = \mathrm{f}\,(x\,,\mathrm{y}) \quad ,\ x \in [a\,,\mathrm{b}]\ ,\ y\,(a) = \mathrm{A} \qquad (51)$$

Where A is a fuzzy number in E^1 with r – level sets :

$$[\mathrm{A}]_{\mathrm{r}} = [[\mathrm{A}]_{\mathrm{r}}^{\mathrm{L}}\ ,\ [\mathrm{A}]_{\mathrm{r}}^{\mathrm{U}}]\ ,\ \mathrm{r} \in [0,1]\ .$$

The fuzzy trail solution for this problem is :

$$[y_{\mathrm{t}}(x\,,\mathrm{p})]_{\mathrm{r}} = [\mathrm{U}(x)]_{\mathrm{r}} + (x-a)\ [\mathrm{N}(x,\mathrm{p})]_{\mathrm{r}} \qquad (52)$$

The Taylor series of a real or complex function $(y(x))$ that is infinitely differentiable in a neighborhood of a real or complex number x_0 is the power series : $y(x) = y(x_o) + y'(x_0)(x-x_0) + y''(x_0)\frac{(x-x_0)^2}{2!} + y'''(x_0)\frac{(x-x_0)^3}{3!} + \ldots\ldots$

Which can be written in the more compact sigma notation as :

$$y(x) = \sum_{n=0}^{\infty}\ y^{(n)}(x_0)\frac{(x-x_0)^n}{n!} \qquad (53)$$

where $n!$ denotes the factorial of n .

Exponential function $e^x = \sum_{n=0}^{\infty} \frac{y^{(n)}}{n!}$ for all .

Praise fuzzy function on basis Taylor series

$$y(x) = [y(x_0)]_r + [y(x_0)]_r(x-x_0) + [y(x_0)]_r\frac{(x-x_0)^2}{2!} + [y(x_0)]_r\frac{(x-x_0)^3}{3!} + \ldots\ldots$$

Therefore, we get

$$[U(x)]_r = (1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!})\ [y(x_0)]_r \qquad (54)$$

For $x_0 = a$, the fuzzy initial condition in eq.(51) will be

$[y(x_0)]_r = [y(a)]_r = [A]_r = [[A]_r^L\ , [A]_r^U]$, then eq.(54) will be :

$$[U(x)]_r = (1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!})\ [A]_r \qquad (55)$$

Then, the fuzzy trial solution (52) becomes (when $a = 0$) :

$$[y_t(x,p)]_r = (1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!})\ [A]_r + x[N(x,p)]_r \qquad (56)$$

The error function that must be minimized for the problem (51) is :

ISSN 2347-1921

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

$$E_{ir}^{L} = \left[\left[\frac{d\, y_t\,(x_i\,,p)}{dx}\right]_r^L - \left[f\left(x_i\,, y_t\,(x_i\,,p)\right)\right]_r^L\right]^2 \quad (57)$$

$$E_{ir}^{U} = \left[\left[\frac{d\, y_t\,(x_i\,,p)}{dx}\right]_r^U - \left[f\left(x_i\,, y_t\,(x_i\,,p)\right)\right]_r^U\right]^2 \quad (58)$$

where :

$$[y_t(x,p)]_r^L = (1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!})[A]_r^L + x[N(x,p)]_r^L \quad (59)$$

$$[y_t(x,p)]_r^L = (1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!})[A]_r^L + x[N(x,p)]_r^L \quad (60)$$

$$[y_t(x,p)]_r^U = (1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!})[A]_r^U + x[N(x,p)]_r^U \quad (61)$$

$$\frac{\partial[y_t(x,p)]_r^L}{\partial x} = \left(1 + x + \frac{x^2}{2} + \frac{x^3}{6}\right)[A]_r^L + x\frac{\partial[N(x,p)]_r^L}{\partial x} + [N(x,p)]_r^L \quad (62)$$

$$\frac{\partial[y_t(x,p)]_r^U}{\partial x} = \left(1 + x + \frac{x^2}{2} + \frac{x^3}{6}\right)[A]_r^U + x\frac{\partial[N(x,p)]_r^U}{\partial x} + [N(x,p)]_r^U \quad (63)$$

9.2. Solution of Second Order Fuzzy Differential Equation

Now, we consider the second order fuzzy differential equation : $y'' = f(x, y, y')$, $x \in [a, b]$ (64)

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

such that the functions :

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

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 fuzzy trial for problem (64) has the form (when $a = 0$) :

$$[y_t(x,p)]_r = [U(x)]_r + x[B]_r + x^2[N(x,p)]_r \quad (65)$$

Again , we write the power series :

$$y(x) = y(x_o) + y'(x_0)(x - x_0) + y''(x_0)\frac{(x-x_0)^2}{2!} + y'''(x_0)\frac{(x-x_0)^3}{3!} + \ldots\ldots$$

Hyperbolic cosine function $\cosh(x) = \sum_{n=0}^{\infty}\frac{y^{(2n)}}{(2n)!}$ for all .

Praise fuzzy function on basis Taylor series

$$y(x) = [y(x_0)]_r + [y(x_0)]_r\frac{(x-x_0)^2}{2!} + [y(x_0)]_r\frac{(x-x_0)^4}{4!} + \ldots\ldots \quad (66)$$

Therefore, we get

$$[U(x)]_r = (1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \frac{x^6}{6!} + \frac{x^8}{8!})\,[y(x_0)]_r \quad (67)$$

For $x_0 = a$, the first fuzzy initial condition in eq.(64) will be

$[y(x_0)]_r = [y(a)]_r = [A]_r = [[A]_r^L\,, [A]_r^U]$, then eq.(67) will be :

$$[U(x)]_r = (1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \frac{x^6}{6!})[A]_r \quad (68)$$

Then, the fuzzy trial solution(65) becomes :

$$[y_t(x,p)]_r = (1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \frac{x^6}{6!})[A]_r + x[B]_r + x^2[N(x,p)]_r \quad (69)$$

The error function that must be minimized for problem (64) is :

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

$$E_{ir}^{L} = ([y''_t(x_i\,,p)]_r^L - f(x_i\,, y_T(x_i,p),\ y'_t(x_i,p))]_r^L)^2 \quad (70)$$

$$E_{ir}^{U} = ([y''_t(x_i\,,p)]_r^U - f(x_i\,, y_T(x_i,p),\ y'_t(x_i,p))]_r^U)^2 \quad (71)$$

where :

$$[y_t(x,p)]_r^L = \left(1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \frac{x^6}{6!}\right)[A]_r^L + x[B]_r^L + x^2[N(x,p)]_r^L \quad (72)$$

$$[y_t(x,p)]_r^U = \left(1+\frac{x^2}{2!}+\frac{x^4}{4!}+\frac{x^6}{6!}\right)[A]_r^U + x[B]_r^U + x^2[N(x,p)]_r^U \quad (73)$$

$$\frac{\partial[y_t(x,p)]_r^L}{\partial x} = \left(x+\frac{x^3}{3!}+\frac{x^5}{5!}+\frac{x^7}{7!}\right)[A]_r^L + [B]_r^L + x^2\frac{\partial[N(x,p)]_r^L}{\partial x} + 2x[N(x,p)]_r^L \quad (74)$$

$$\frac{\partial[y_t(x,p)]_r^U}{\partial x} = \left(x+\frac{x^3}{3!}+\frac{x^5}{5!}+\frac{x^7}{7!}\right)[A]_r^U + [B]_r^U + x^2\frac{\partial[N(x,p)]_r^U}{\partial x} + 2x[N(x,p)]_r^U \quad (75)$$

$$\frac{\partial^2[y_t(x,p)]_r^L}{\partial x^2} = \left(1+\frac{x^2}{2!}+\frac{x^4}{4!}+\frac{x^6}{6!}\right)[A]_r^L + x^2\frac{\partial^2[N(x,p)]_r^L}{\partial x^2} + 4x\frac{\partial[N(x,p)]_r^L}{\partial x} + 2[N(x,p)]_r^L \quad (76)$$

$$\frac{\partial^2[y_t(x,p)]_r^U}{\partial x^2} = \left(1+\frac{x^2}{2!}+\frac{x^4}{4!}+\frac{x^6}{6!}\right)[A]_r^U + x^2\frac{\partial^2[N(x,p)]_r^U}{\partial x^2} + 4x\frac{\partial[N(x,p)]_r^U}{\partial x} + 2[N(x,p)]_r^U \quad (77)$$

and

$$[N(x,p)]_r^L = \sum_{j=1}^m [v_j]_r^L \; s(x\,W_j + B_j) \quad (78)$$

$$[N(x,p)]_r^U = \sum_{j=1}^m [v_j]_r^U \; s(x\,W_j + B_j) \quad (79)$$

$$\frac{\partial[N(x,p)]_r^L}{\partial x} = \sum_{j=1}^m W_j[v_j]_r^L \; s'(x\,W_j + B_j) \quad (80)$$

$$\frac{\partial[N(x,p)]_r^U}{\partial x} = \sum_{j=1}^m W_j[v_j]_r^U \; s'(x\,W_j + B_j) \quad (81)$$

$$\frac{\partial^2[N(x,p)]_r^L}{\partial x^2} = \sum_{j=1}^m W_j^2 \; [v_j]_r^L \; s''(x\,W_j + B_j) \quad (82)$$

$$\frac{\partial^2[N(x,p)]_r^U}{\partial x^2} = \sum_{j=1}^m W_j^2 \; [v_j]_r^U \; s''(x\,W_j + B_j) \quad (83)$$

10. NUMERICAL EXAMPLES

To show the behavior and properties of the new method, two problem will be solved in this section . For each example, the accuracy of the method is illustrated by computing the deviations $\underline{E}(x,r)$ and $\overline{E}(x,r)$

where $\underline{E}(x,r) = \left|\underline{y_t}(x,r) - \underline{y_a}(x,r)\right|$, $\overline{E}(x,r) = \left|\overline{y_t}(x,r) - \overline{y_a}(x,r)\right|$.

and $y_a(x,r) = \left[\underline{y_a}(x,r), \overline{y_a}(x,r)\right]$, $y_t(x,r) = \left[\underline{y_t}(x,r), \overline{y_t}(x,r)\right]$ are the analytical and trial solutions respectively .

Note that, for all examples, a multilayer perceptron consisting of one hidden layer with 10 hidden units and one linear output unit is used.To minimize the error function,we used BFGS quasi-Newton method (For more details , see [33]) .

Example (1) : Consider the following first order FDE :

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

The fuzzy analytical solution for this problem is :

$$[y_a(x)]_r = \begin{bmatrix} 0.5\,(\sin x - \cos x + (2.92 + 0.08r)e^{-x}), \\ 0.5\,(\sin x - \cos x + (3.02 - 0.02r)e^{-x}) \end{bmatrix}$$

From(52),The fuzzy trial solution for this problem is :

$$[y_t(x,p)]_r = \left(1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!}\right)[0.96 + 0.04r, 1.01 - 0.01r] + x[N(x,p)]_r$$

Analytical and trial solutions for this example can be found in table (1) and table (2) .

Table 1. Numerical results for example (1) , for r = 0.5 .

ISSN 2347-1921

x	$[y_a(x)]_r^L$	$[y_t(x)]_r^L$	$\underline{E}(x,r)$	$[y_a(x)]_r^U$	$[y_t(x)]_r^U$	$\overline{E}(x,r)$
0	0.98	0.98	0	1.005	1.005	0
0.1	0.891574004	0.891574581	0.000000577	0.914194939	0.914195305	0.000000366
0.2	0.821022891	0.821023390	0.000000499	0.841491159	0.841491470	0.000000311
0.3	0.766502825	0.766502420	0.000000405	0.785023280	0.785023535	0.000000255
0.4	0.726252342	0.726252125	0.000000217	0.743010343	0.743010630	0.000000287
0.5	0.698586864	0.698586787	0.000000077	0.713750131	0.713749811	0.000000320
0.6	0.681894650	0.681894580	0.000000070	0.695614941	0.695614876	0.000000065
0.7	0.674633999	0.674634988	0.000000989	0.687048632	0.687048723	0.000000091
0.8	0.675331557	0.675332678	0.000001121	0.686564781	0.686565200	0.000000419
0.9	0.682581567	0.682579911	0.000001656	0.692745808	0.692746369	0.000000561
1	0.695045912	0.695046720	0.000000808	0.704242898	0.704236249	0.000006649

Table 2. Numerical results for example (1) , for x = 0.98 .

r	$[y_a(x)]_r^L$	$[y_t(x)]_r^L$	$\underline{E}(x,r)$	$[y_a(x)]_r^U$	$[y_t(x)]_r^U$	$\overline{E}(x,r)$
0	0.684691615	0.684691699	0.000000084	0.703457170	0.703457233	0.000000063
0.1	0.686192860	0.686193183	0.000000323	0.703081859	0.703081937	0.000000078
0.2	0.687694104	0.687694503	0.000000399	0.702706548	0.702707155	0.000000607
0.3	0.689195349	0.689194544	0.000000805	0.702331237	0.702330560	0.000000677
0.4	0.690696593	0.690695784	0.000000809	0701955926	0.701956698	0.000000772
0.5	0.692197837	0.692196904	0.000000933	0.701580615	0.701581567	0.000000952
0.6	0.693699082	0.693700032	0.000000950	0.701205304	0.701206300	0.000000996
0.7	0.695200326	0.695204660	0.000004334	0.700829993	0.700821546	0.000008447
0.8	0.696701571	0.696695912	0.000005659	0.700454682	0.700461711	0.000007029
0.9	0.698202815	0.698195114	0.000007701	0.700079370	0.700080145	0.000000775
1	0.699704059	0.699704180	0.000000121	0.699704059	0.699704968	0.000000909

Example (2) : Consider the following nonlinear FDE :

$y''(x) = -(y'(x))^2 \quad , \quad 0 \le x \le 0.1$

With the fuzzy initial conditions : $[y(0)]_r = [r, 2-r]$, $[y'(0)]_r = [1+r, 3-r]$.

The fuzzy analytical solutions for this problem is :

$[y_a(x)]_r = [\mathrm{Ln}\,(e^r + xe^r + r\,xe^r), \mathrm{Ln}\,(e^{2-r} + 3xe^{2-r} - r\,xe^{2-r})]$.

From(69),the fuzzy trial solutions for this problem is :

$[y_t(x,p)]_r = (1+\frac{x^2}{2!}+\frac{x^4}{4!}+\frac{x^6}{6!})[r, 2-r] + [1+r, 3-r]x + x^2[N(x,p)]_r$

Analytical and trial solutions for this example can be found in table(3) and table(4) .

Table 3. Numerical results for example (2) , for r = 0.5 .

ISSN 2347-1921

x	$[y_a(x)]_r^L$	$[y_a(x)]_r^U$	$[y_t(x)]_r^L$	$[y_t(x)]_r^U$	$\underline{E}(x,r)$	$\overline{E}(x,r)$
0	0.5	1.5	0.5	1.5	0	0
0.01	0.514888612	1.524692613	0.514888619	1.524692617	7.232 e-9	4.440 e-9
0.02	0.529558802	1.548790164	0.529558809	1.548790160	7.287 e-9	4.828 e-9
0.03	0.544016885	1.572320662	0.544016878	1.572320667	7.898 e-9	5.110 e-9
0.04	0.558268908	1.595310180	0.558268900	1.595310185	8.001 e-9	5.909 e-9
0.05	0.572320661	1.617783036	0.572320669	1.617783030	8.088 e-9	6.559 e-9
0.06	0.586177696	1.639761942	0.586177630	1.639761936	6.663 e-8	6.313 e-9
0.07	0.599845335	1.661268148	0.599845280	1.661268141	5.555 e-8	7.122 e-9
0.08	0.613328685	1.682321557	0.613328694	1.682321551	9.443 e-9	6.997 e-9
0.09	0.626632650	1.702940844	0.626632659	1.702940769	9.033 e-9	7.544 e-8
0.10	0.639761942	1.723143551	0.639761933	1.723143623	9.221 e-9	7.208 e-8

Table 4. Numerical results for example (2) , for x = 0.1 .

r	$[y_a(x)]_r^L$	$[y_a(x)]_r^U$	$[y_t(x)]_r^L$	$[y_t(x)]_r^U$	$\underline{E}(x,r)$	$\overline{E}(x,r)$
0	0.095310179	2.262364264	0.095310173	2.262364272	6.623e-9	8.529e-9
0.1	0.204360015	2.154642218	0.204360073	2.154642213	5.895 e-8	5.397 e-9
0.2	0.313328685	2.046860078	0.313328736	2.046860082	5.179 e-8	4.530 e-9
0.3	0.422217632	1.939016900	0.422217636	1.939016910	4.478 e-9	1.001e-8
0.4	0.531028262	1.831111721	0.531028265	1.831111631	3.790e-9	9.081e-8
0.5	0.639761942	1.723143551	0.639761933	1.723143623	9.221 e-9	7.208 e-8
0.6	0.748420005	1.615111380	0.748420002	1.615111386	3.110e-9	6.743 e-9
0.7	0.857003748	1.507014169	0.857003746	1.507014163	2.422e-9	6.212e-9
0.8	0.965514438	1.398850859	0.965514439	1.398850862	1.702 e-9	3.421 e-9
0.9	1.073953307	1.290620360	1.073953398	1.290620365	9.138 e-8	5.080e-9
1	1.182321557	1.182321557	1.182321466	1.182321552	9.135 e-8	5.601 e-9

For the above example , we also solved it by using semi – Taylor series of the cosine function $\cos(x) = \sum_{n=0}^{\infty} (-1)^n \frac{y^{(2n)}}{(2n)!}$ for all x. Therefore , the fuzzy trial solutions for this case will be :

$[y_t(x,p)]_r = (1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!})[\,r\,, 2 - r] + [1 + r\,, 3 - r]x + x^2[N(x,p)]_r$

Analytical and trial solutions for this case can be found in tables(5) and table(6) .

Table 5. Numerical results for example (2) , for r = 0.5 .

ISSN 2347-1921

x	$[y_a(x)]_r^L$	$[y_a(x)]_r^U$	$[y_t(x)]_r^L$	$[y_t(x)]_r^U$	$\underline{E}(x,r)$	$\overline{E}(x,r)$
0	0.5	1.5	0.5	1.5	0	0
0.01	0.514888612	1.524692613	0.514888664	1.524692642	5.211 e-8	2.922 e-8
0.02	0.529558802	1.548790164	0.529558895	1.548790173	9.312 e-8	9.289 e-9
0.03	0.544016885	1.572320662	0.544016957	1.572320735	7.287 e-8	7.303 e-8
0.04	0.558268908	1.595310180	0.558268981	1.595310228	7.378 e-8	4.886 e-8
0.05	0.572320661	1.617783036	0.572320655	1.617782979	6.344 e-9	5.785 e-8
0.06	0.586177696	1.639761942	0.586177782	1.639762042	8.604 e-8	1.002 e-7
0.07	0.599845335	1.661268148	0.599845428	1.661268326	9.344 e-8	1.781 e-7
0.08	0.613328685	1.682321557	0.613329669	1.682321916	9.844 e-7	3.596 e-7
0.09	0.626632650	1.702940844	0.626633508	1.702940788	8.589 e-7	5.670 e-8
0.10	0.639761942	1.723143551	0.639761864	1.723143499	7.865 e-8	5.219 e-8

Table 6. Numerical results for example (6.3.3.2) , for x = 0.1 .

r	$[y_a(x)]_r^L$	$[y_a(x)]_r^U$	$[y_t(x)]_r^L$	$[y_t(x)]_r^U$	$\underline{E}(x,r)$	$\overline{E}(x,r)$
0	0.095310179	2.262364264	0.095310212	2.262364267	3.359e-8	3.659e-9
0.1	0.204360015	2.154642218	0.204360032	2.154642225	1.757 e-8	7.182e-9
0.2	0.313328685	2.046860078	0.313328705	2.046860079	2.089 e-8	1.594e-9
0.3	0.422217632	1.939016900	0.422217623	1.939016873	9.052 e-9	2.788e-8
0.4	0.531028262	1.831111721	0.531028268	1.831111740	6.754e-9	1.992e-8
0.5	0.639761942	1.723143551	0.639762029	1.723144452	8.708 e-8	9.018 e-7
0.6	0.748420005	1.615111380	0.748420045	1.615112052	4.084e-8	6.725 e-7
0.7	0.857003748	1.507014169	0.857003766	1.507014188	1.886e-8	1.986 e-8
0.8	0.965514438	1.398850859	0.965515129	1.398850909	6.910 e-7	5.051 e-8
0.9	1.073953307	1.290620360	1.073953398	1.290620458	1.232 e-7	9.870e-8
1	1.182321557	1.182321557	1.182321618	1.182321564	6,151 e-8	7.667 e-9

11. CONCLUSIONS

In this paper, we presented a hybrid approach based on fuzzy neural networks and Taylor series for solving fuzzy differential equations. We demonstrate the ability of fuzzy neural networks to approximate the solutions of FDEs . By comparing our results with the results obtained by other 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 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] Abbasbandy,S. and Allahviranloo,T.2002. Numerical solution of fuzzy Differential equations by Taylor method. Journal of Computational Methods in Applied Mathematics. 2 , 113-124 .

[2] Abbasbandy,S. and Allahviranloo,T.2002. Numerical solution of fuzzy Differential equations by Runge-Kutta method.J. Sci. Teacher Training University. 1(3).

[3] Allahviranloo,T.,Ahmady,N. and et al.2007. Numerical solution of fuzzy Differential equations by predictor- corrector method. Information Sciences. 177 , 1633-1647 .

[4] Allahviranloo,T.,Ahmady,E. and et al.2008. Nth- order fuzzy linear Differential equations . Information Sciences.178 , 1309-1324 .

[5] Allahviranloo,T. ,Kiani,N. A. and et al.2009.Solving fuzzy differential equations by differential transformation method . Information Sciences.179 , 956-966 .

[6] Abbasbandy,S. and Otadi,M.2006. Numerical solution of fuzzy polynomials by fuzzy neural network.Appl. Math. Comput. 181,1084-1089 .

[7] Abbasbandy,S.,Otadi,M. and et al. 2008. Numerical solution of a system of fuzzy polynomials by fuzzy neural network. Information sciences .178, 1948 – 1960 .

[8] Bede,B. and Gal,S. G.2005. Generalizations of the differentiability of fuzzy number- valued functions with applications to fuzzy differential equations.Fuzzy Sets and Systems. 151 , 581 – 599 .

[9] Buckley,J. J. and Feuring,T.2000. Fuzzy differential equations. Fuzzy Sets and Systems. 110, 43 – 54.

[10] Diamond,P.2000. Stability and periodicity in fuzzy differential equations .IEEE Trans Fuzzy Systems .8 , 583 – 590 .

[11] Diamond,P.2002. Brief note on the variation of constants formula for fuzzy differential equations.Fuzzy Sets and Systems, 129 ,65 – 71 .

[12] Ezadi,S. ,Parandin,N. and et al.2013. Numerical solution of fuzzy differential equations based on semi-Taylor by using neural network. Journal of basic and applied scientific research . 477-482 .

[13] Ghazanfari,B. and Shakerami,A.2001. Numerical solution of fuzzy differential equations extended Runge – Kutta- like formulae of order 4. Fuzzy Sets and Systems. 189 , 74 – 91 .

[14] Georgiou,D. N.,Nieto,J. J. and et al.2005. Initial value problems for higher-order fuzzy differential equations . Nonlinear Anal.63, 587-600 .

[15] Hornick,K.,Stinchcombe,M.1989. Multi-layer feed forward networks are universal approximators. Neural Networks .2, 359-366 .

[16] Kaleva,O.1987.Fuzzy differential equations. Fuzzy Sets and Sydtems. 24 , 301 – 317 .

[17] TKhanna ,T.1990. Foundations of neural networks. Addison-Wesly, Reading, MA.

[18] Kastan,A. and Ivaz,K.2009. Numerical solution of fuzzy differential equations by Nystrom method. Chaos,Solitons and Fractals .4 , 859 – 868 .

[19] Lee,H. and Kang,I. S.1990. Neural algorithms for solving differential equations .Journal of Computational Physics.91,110-131 .

[20] Mosleh,M.2013.Fuzzy neural network for solving a system of fuzzy differential equations.Applied Soft Computing .13,3597-3607 .

[21] Mosleh.M.,Allahviranloo,T. and et al.2012.Evaluation of fully fuzzy regression models by fuzzy neural network. Neural Comput and Applications.21, 105 – 112 , .

[22] Meade,A. J. and Fernandez,A. A.1994. The numerical solution of linear ordinary differential equations by feed-forward neural network.Mathematical and Computer Modelling.19(12) , 1 – 25 .

[23] Mosleh.M. and Otadi,M.2012. Simulation and evaluation of fuzzy differential equation by fuzzy neural network. Applied soft computing 12, 2817-2827 .

[24] Mosleh,M. and Otadi, M.2014.Solving the second order fuzzy differential equations by fuzzy neural network. Journal of Mathematical Extension. 81 , 11 – 27 .

[25] Mosleh,M.,Otadi ,M.and et al.2010.Evaluation of fuzzy regression models by fuzzy neural network.Journal of Computational and Applied Mathematics.234 , 825 – 834 .

[26] Mosleh,M. , Otadi ,M. and et al.2011. Fuzzy polynomial regression with fuzzy neural network. Applied Mathematical Modeling .35, 5400 – 5412 .

[27] Malek,A. and Shekari ,R.2006. Numerical solution for high order differential equations using a hybrid neural network-Optimization method. Appl. Math. Comput. 183 , 260 – 271 .

[28] Nieto,J. J.1999. The Cauchy problem for continuous fuzzy differential equations .Fuzzy Sets and Systems.102 , 259 – 262 .

[29] Nieto,J. J. ,Rodriguez-Lopez,R.2006. Bounded solutions for fuzzy differential and integral equations.Chaos, Solitons and Fractals. 27 , 1376 – 1386 .

[30] Otadi,M.,Mosleh,M. and et al.2011. solving fuzzy linear system by neural network and applications in Economics. Journal of mathematics extension. 47-66 .

[31] Otadi,M. ,Mosleh,M. and et al.2011.Numerical solution of fully fuzzy linear systems by fuzzy neural network,Soft Computing, 15 , 1513 – 1522 .

[32] Ouyang,H. and Wu,Y.1989. On fuzzy differential equations. Fuzzy Sets and Systems.32 , 321 – 325 .

[33] Vandenberghe,L. 2013. Quasi-Newton Methods .EE236C (Spring 2013-14) , 1-15 .