British Journal of Mathematics & Computer Science
12(2): 1-20, 2016, Article no.BJMCS.20504
*ISSN: 2231-0851*

SCIENCEDOMAIN *international*
www.sciencedomain.org

# Numerical Solution of Fuzzy Partial Differential Equations by Using Modified Fuzzy Neural Networks

**Eman A. Hussian$^{1*}$ and Mazin H. Suhhiem$^2$**

*[1]Department of Mathematics, College of Sciences, AL-Mustansiriyah University, Baghdad, Iraq.*
*[2]Department of Statistics, College of Adm. and Econ., University of Sumar, Alrefiey, Iraq.*

***Article Information***

DOI: 10.9734/BJMCS/2016/20504
*Editor(s):*
(1) Sheng Zhang, Department of Mathematics, Bohai University, Jinzhou, China.
*Reviewers:*
(1) Anonymous, Hong Kong Polytechnic University, Hong Kong, China.
(2) K. Kannan, SASTRA University, Kumbakonam, India.
(3) Grienggrai Rajchakit, Maejo University, Thailand.
Complete Peer review History: http://sciencedomain.org/review-history/11685

***Original Research Article***

*Received: 30 July 2015*
*Accepted: 31 August 2015*
*Published: 05 October 2015*

## Abstract

The aim of this work is to present a modified method for finding the numerical solutions of fuzzy partial differential equations by using fuzzy artificial neural networks. Using a fuzzy trial neural solution depending on the fuzzy initial values and the fuzzy boundary conditions of the problem. 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. In fact, This new method based on replacing each element in the training set by a polynomial of first degree. The fuzzy trial solution of fuzzy partial differential equation is written as a sum of two parts. The first part satisfies the fuzzy conditions, it contains no fuzzy adjustable parameters. The second part involves a feed-forward fuzzy neural network containing fuzzy adjustable parameters. In comparison with existing similar fuzzy neural networks, the proposed method provides solutions with high accuracy. Finally, we illustrate our approach by two problems.

*Keywords: Fuzzy partial differential equation; fuzzy neural network; feed-forward neural network; BFGS method; hyperbolic tangent function.*

## 1 Introduction

Many methods have been developed so far for solving fuzzy differential equations (FDEs). Most of the practical problems require the solution of a FDE which satisfies fuzzy initial or fuzzy boundary conditions. The theory of FDEs was treated by Kaleva [1], Ouyang and Wu [2], Nieto [3], Buckley and Feuring [4],

**Corresponding author: E-mail: dr_emansultan@yahoo.com, mz80m@yahoo.com;*

Seikkala also recently there appeared the papers of Bede, Bede and Gal [5], Diamond [6,7], Georgiou and Nieto and et al. [8], Nieto and Lopez [9].

In the following, we have mentioned some numerical solution which have proposed by other scientists. Abbasbandy and Allahviranloo have solved FDEs by Runge-Kuta and Taylor methods [10,11]. Also, Allahviranloo and et al. solved FDEs by predictor- corrector and transformation methods [12,13,14]. Ghazanfari and Shakerami developed Runge-Kuta like formula of order 4 for solving FDEs [15]. Nystrom method has been introduced for solving FDEs [16]. Allahviranloo and Kermani solved fuzzy linear partial differential equations under new definition of fuzzy derivative [17]. Dahalan and Muthuvalu and et al. developed the Performance of (Half-Sweep Alternating Group Explicit) method with Seikkala derivative for two dimensional fuzzy Poisson equation [18].

In recent years artificial neural networks for estimation of the ordinary differential equation (ODE) and partial differential equation (PDE) have been used. In (1990) lee and Kang [19] used parallel processor computers to solve a first order ODEs with Hopfield neural network models. In (1994) Meade and Fernandez [20,21] solved linear and non-linear ODEs by using feed-forward neural networks architecture and B-splines of degree one. In (1997) Lagaris and et al. [22,23] used artificial neural network for solving ODEs and PDEs with the initial/ boundary value problems. In (1999) Liu and Jammes [24] developed some properties of the trial solution to solve the ODEs by using artificial neural networks. In (2004) Tawfiq [25] presented and developed supervised and unsupervised algorithms for solving ODEs and PDEs. In (2006) malek and shekari [26] presented numerical method based on artificial neural network and optimization techniques which the higher-order ODE answers approximates by finding a package form analytical of specific functions. In (2008) Pattanaik and Mishra [27] applied and developed some properties of ANN for solution of PDE in RF Engineering. In (2011) Oraibi [28] design feed-forward neural networks for solving ordinary initial value problem. In (2015) Hussian and Suhhiem [29] used modified artificial neural networks for solving PDEs.

Numerical solution of FDEs by using artificial neural networks is the subject of a very modern because it only goes back to 2010. In (2010) Effati and pakdaman [30] used artificial neural network for solving FDEs, they used for the first time the artificial neural network to approximate fuzzy initial value problems. In (2012) Mosleh and Otadi [31] used artificial neural networks for solving fuzzy Fredholm integro-differential eauations. In (2013) Ezadi and et al. [32] used artificial neural networks based on semi-Taylor series to solve first order FDE. In (2015) Hussian and Suhhiem [33] used modified artificial neural networks for solving FDEs.

Numerical solution of FDEs by using fuzzy artificial neural networks is more modern than the previous subject, where it goes back to 2012. In (2012) Mosleh and Otadi [34] used fuzzy artificial neural network for solving first order FDEs, they used for the first time the fuzzy artificial neural network to approximate fuzzy initial value problems. In (2013) Mosleh [35] used fuzzy artificial neural network for solving a system of FDEs. In (2014) Mosleh and Otadi [36] used fuzzy artificial neural network for solving second order FDEs. In (2015) Hussian and Suhhiem [37] used modified fuzzy neural networks for solving Fuzzy ordinary differential equations.

In this work we proposed a new numerical method to find the approximate solution of fuzzy partial differential equations (FPDEs), this method can result in improved numerical methods for solving FPDEs. 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 conditions 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. This modified method is called modified fuzzy neural network (MFNN) for solving FPDEs. This new method based on replacing each x in the training set (where $x \in [a,b]$) by the polynomial $Q(x) = \epsilon(x+1)$ such that $Q(x) \in (a,b)$ by choosing a suitable $\epsilon \in (0,1)$. In this paper, we will illustrate this modified method by solving two numerical examples. In general, this modified method is effective for solving FPDEs.

## 2 Preliminaries

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

### Definition 2.1. [38]:

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:

$$\underline{(u + v)}\,(r) = \underline{u}\,(r) + \underline{v}\,(r) \tag{1}$$

$$\overline{(u + v)}\,(r) = \bar{u}\,(r) + \bar{v}\,(r) \tag{2}$$

$$\underline{(k\,u)}\,(r) = K\,\underline{u}\,(r)\,,\ \overline{(k\,u)}\,(r) = K\,\bar{u}\,(r) \tag{3}$$

The collection of all fuzzy numbers with addition and multiplication as defined by $Eqs.\ (1) \longrightarrow (3)$ is denoted by $E^1$ . For r ∈ (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. [38]:

The function f: R ⟶ $E^1$ is called a fuzzy function. Now if, for an arbitrary fixed $t_1$ ∈ 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. [30]:

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

### Definition 2.4. [30]:

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.

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

## 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 the neural networks. Combining fuzzy system with neural network. Both neural network and fuzzy system have some things in common [37].

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 [39].

## 4 Operations of Fuzzy Numbers

We briefly on mention fuzzy numbers operation defined by the extension principle .Since output 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 [37]:

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

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

$$\mu_{f(\text{net})}(z) = \text{Max}\left\{\mu_{\text{net}}(x) \mid z = f(x)\right\} \tag{7}$$

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).

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

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

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:

$$[A]_r + [B]_r = \left[ [A]_r^L + [B]_r^L ,\ [A]_r^U + [B]_r^U \right] \tag{9}$$

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

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

## 5 Input – Output Relation 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 $n \times m \times s$, where n , m and s are the number of the neurons in the input layer, the hidden layer and the output layer , respectively [35,36] .

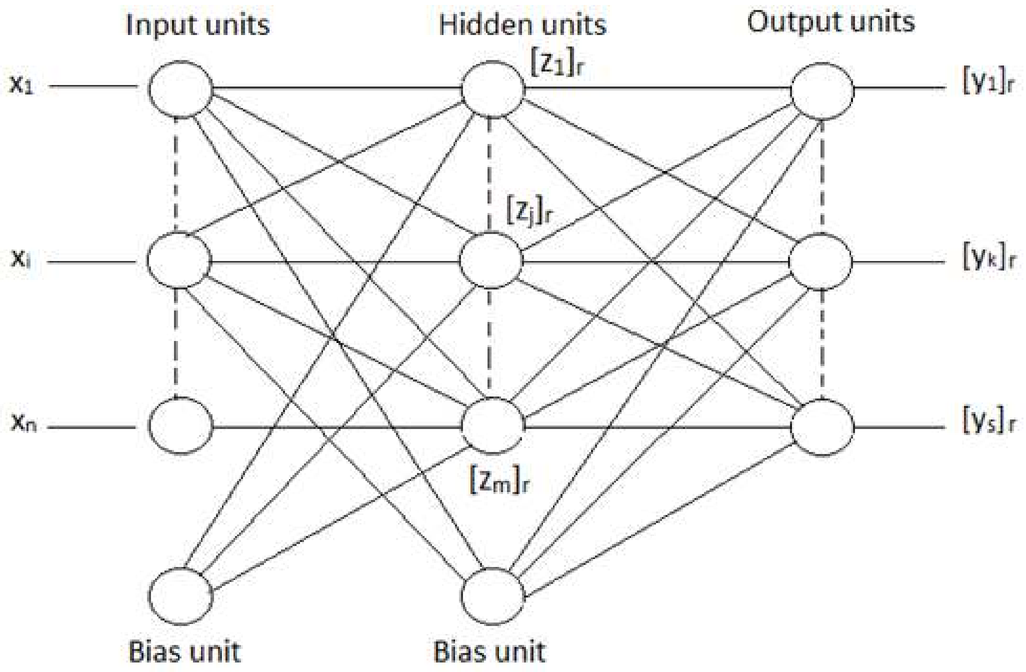

**Fig. 1. Three-layer feed-forward Fuzzy neural network**

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:

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

Hidden units:

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

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

Output units:

$$y_k = F\,(\mathrm{net}_k)\ ,\quad k = 1,2,3,\dots, s, \tag{15}$$

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

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 [34].

Input units:

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

Hidden units:

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

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

Output units:

$$[y_k]_r = F\,([\mathrm{net}_k]_r)\ ,\ k = 1,2,3,\dots, s, \tag{20}$$

$$[\mathrm{net}_k]_r = \sum_{j=1}^{m}[w_{kj}]_r\, [z_j]_r + [c_k]_r\ . \tag{21}$$

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:

$$o_i = x_i \tag{22}$$

Hidden units:

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

where

$$[\mathrm{net}_j]_r^L = \sum_{i=1}^{n} o_i\, [w_{ji}]_r^L + [b_j]_r^L \tag{24}$$

$$[\mathrm{net}_j]_r^U = \sum_{i=1}^{n} o_i\,[w_{ji}]_r^U + [b_j]_r^U \tag{25}$$

Output units:

$$[y_k]_r = F([\mathrm{net}_k]_r) = [[y_k]_r^L,\ [y_k]_r^U] = .\ [F([\mathrm{net}_k]_r^L)\,, F([\mathrm{net}_k]_r^U)] \tag{26}$$

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

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

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

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

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

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

## 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 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 [34,37]:

$$\mathrm{net}_j = x\, w_j + b_j\ \ ,\ j = 1,2,3,\dots,m, \tag{29}$$

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 $j$th unit in the hidden layer.

The output, in the hidden neurons is:

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

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:

$$N = v_1\ z_1 + v_2\ z_2 + v_3\ z_3 + \dots + v_j\ z_j + \dots + v_m\ z_m = \sum_{j=1}^{m} v_j\ z_j \tag{31}$$

where $v_j$ is a weight parameter from $j$th 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. 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., $x \geq 0$ :

Input unit:

$$o = x, \tag{32}$$

Hidden units:

$$[z_j]_r = \left[[z_j]_r^L,\ [z_j]_r^U\right] = \left[s\left([\mathrm{net}_j]_r^L\right),\ s\left([\mathrm{net}_j]_r^U\right)\right] \tag{33}$$

Where

$$[\mathrm{net}_j]_r^L = o\,[w_j]_r^L + [b_j]_r^L \quad , \quad [\mathrm{net}_j]_r^U = o\,[w_j]_r^U + [b_j]_r^U$$

Output unit:

$$[N]_r = [[N]_r^L,\ [N]_r^U] \quad , \quad \text{where}$$

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

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

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

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

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

For illustration the solution steps, we will consider the first order FDE:

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

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:

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

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

The error function that must be minimized for the problem (36) is in the form:

$$E = \sum_{i=1}^{g}\left(E_{ir}^L + E_{ir}^U\right) \tag{38}$$

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

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

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.,

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

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

where

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

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

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

$$\frac{\partial\,[net_j]_r^L}{\partial x} = [w_j]_r^L \qquad , \qquad \frac{\partial\,[net_j]_r^U}{\partial x} = [w_j]_r^U$$

$$\frac{\partial\,[z_j]_r^L}{\partial\,[net_j]_r^L} = 1 - \left([z_j]_r^L\right)^2 \qquad , \qquad \frac{\partial\,[z_j]_r^U}{\partial\,[net_j]_r^U} = 1 - \left([z_j]_r^U\right)^2$$

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

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

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

Therefore, we get

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

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

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 FPDEs 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 [34,37].

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

Input unit: $o = x$

Hidden units: $z_j = s\left(\text{net}_j\right)$ , $j = 1,2,3,\dots m$

where $\text{net}_j = o\, w_j + b_j$

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

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

$$\frac{\partial\,[N]_r^L}{\partial x} = \sum_{j=1}^{m}\left[v_j\right]_r^L \frac{\partial\, z_j}{\partial\, \text{net}_j}\ \frac{\partial\, \text{net}_j}{\partial x} = \sum_{j=1}^{m}\left[v_j\right]_r^L\ w_j\left(1 - z_j^{\,2}\right) \tag{47}$$

$$\frac{\partial\,[N]_r^U}{\partial x} = \sum_{j=1}^{m}\left[v_j\right]_r^U \frac{\partial\, z_j}{\partial\, \text{net}_j}\ \frac{\partial\, \text{net}_j}{\partial x} = \sum_{j=1}^{m}\left[v_j\right]_r^U\ w_j\left(1 - z_j^{\,2}\right) \tag{48}$$

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

$$E_{ir}^L = \begin{bmatrix} \sum_{j=1}^{m} z_j\left[v_j\right]_r^L + (x_i - a)\ \sum_{j=1}^{m} w_j\left(1 - z_j^{\,2}\right)\left[v_j\right]_r^L \\ -F\left(x_i, [A]_r^L + (x_i - a)\ \sum_{j=1}^{m} z_j\ \left[v_j\right]_r^L\right) \end{bmatrix}^2 \tag{49}$$

$$E_{ir}^U = \begin{bmatrix} \sum_{j=1}^{m} z_j\left[v_j\right]_r^U + (x_i - a)\ \sum_{j=1}^{m} w_j\left(1 - z_j^{\,2}\right)\left[v_j\right]_r^U \\ -F\left(x_i, [A]_r^U + (x_i - a)\ \sum_{j=1}^{m} z_j\ \left[v_j\right]_r^U\right) \end{bmatrix}^2 \tag{50}$$

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 Fuzzy Neural Network Approach for Solving FPDEs

To solve any fuzzy partial differential equation , we consider a three – layered FNNM with two unit entries x and y , one hidden layer consisting of m activation functions , and one unit output N(x , y, p).

Here, the dimension of FNNM is (2 x m x 1).

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

$$\mathrm{net}_j = \mathrm{x}\ \mathrm{w}_{j1} + \mathrm{y}\ \mathrm{w}_{j2} + \mathrm{b}_j\ ,\ j = 1,2,3, \dots \mathrm{m} \tag{51}$$

where $\mathrm{w}_{j1}$ and $\mathrm{w}_{j2}$ are a weights from the input layer to the $j$th unit in the hidden layer, $\mathrm{b}_j$ is an $j$th bias for the $j$th unit in the hidden layer.

The output in the hidden neurons is:

$$\mathrm{z}_j = \mathrm{s}\left(\mathrm{net}_j\right),\ j = 1,2,3, \dots,\mathrm{m} \tag{52}$$

The output neuron make no changes in its input, so the input to the. Output neuron is equal to output:

$$\mathrm{N} = \sum_{j=1}^{\mathrm{m}} \mathrm{v}_j\ \mathrm{z}_j \tag{53}$$

From Eqs.(22-28) , we can see that the r – level sets of the Eqs. (51 – 53) 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 inputs x and y are non – negative. For reducing the complexity of the learning algorithm, the inputs x and y usually assumed as non – negative in the fuzzy neural network, i.e., x ≥ o and y ≥ o:

Input units:

$$\mathrm{x} = \mathrm{x},\ \ \mathrm{y} = \mathrm{y} \tag{54}$$

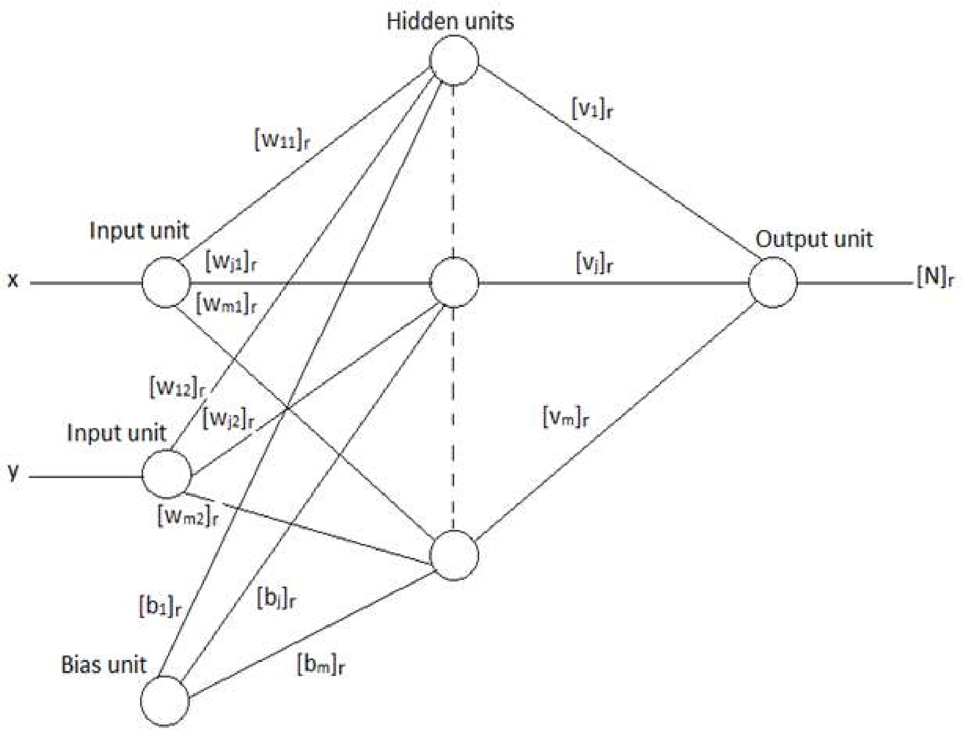

**Fig. 2. (2 x m x 1) feed-forward fuzzy neural network**

Hidden units:

$$[z_j]_r = \left[[z_j]_r^L ,\ [z_j]_r^U\right] = \left[s\left([net_j]_r^L\right) ,s\left([net_j]_r^U\right)\right] \tag{55}$$

$$[net_j]_r^L = x\ [w_{j1}]_r^L + y\ [w_{j2}]_r^L + [b_j]_r^L \tag{56}$$

$$[net_j]_r^U = x\ [w_{j1}]_r^U + y\ [w_{j2}]_r^U + [b_j]_r^U \tag{57}$$

Output unit:

$$[N]_r = [[N]_r^L ,\ [N]_r^U] \tag{58}$$

$$[N]_r^L = \sum_{j\in a}[v_j]_r^L\ [z_j]_r^L + \sum_{j\in b}[v_j]_r^L\ [z_j]_r^U \tag{59}$$

$$[N]_r^U = \sum_{j\in c}[v_j]_r^U\ [z_j]_r^U + \sum_{j\in d}[v_j]_r^U\ [z_j]_r^L \tag{60}$$

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

$c = \left\{j :\ [v_j]_r^U\ \geq\ o\right\}$ , $d = \left\{j :\ [v_j]_r^U\ <\ o\right\}$ ,

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

Also, the input – output of each unit of our PFNN in Eqs. (54-60) can be rewritten for r – level sets as follows:

Input unit:

$$x = x\ ,\quad y = y \tag{61}$$

Hidden units:

$$z_j = s\ (net_j)\ ,\ j = 1,2,3, \dots m \tag{62}$$

where

$$net_j = x\ w_{j1} + y\ w_{j2} + b_j \tag{63}$$

Output unit:

$$[N]_r = [[N]_r^L ,\ [N]_r^U] = \left[\sum_{j=1}^{m}[v_j]_r^L z_j\ \ ,\sum_{j=1}^{m}[v_j]_r^U z_j\ \right] \tag{64}$$

## 10 Description of the Method

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

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = f\ (x\ ,y)\ ,\ x\ ,y \in [a\ ,b] \tag{65}$$

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) :

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

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

The fuzzy trial solution $[U_T(x , y)]_r = [\ \underline{U}_T(x , y, r, \underline{p}) , \overline{U}_T(x , y, r, \overline{p})\ ]$ can be chosen as follows [29]:

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

$$\overline{U}_T(x , y, r, \overline{p}) = \overline{A}(x , y) + x\ y\ (1 - x)\ (1 - y)\ \overline{N}(x , y, r, \overline{p}) \qquad (66)$$

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

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

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

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

$$\left[\bar{g}_0(x) - \left[(1 - x)\ \bar{g}_0(0) + x\ \bar{g}_0(1)\right]\right] + y\left[\bar{g}_1(x) - \left[(1 - x)\ \bar{g}_1(0) + x\ \bar{g}_1(1)\right]\right] \qquad (67)$$

The minimized error function will be: $E = \sum_{i=1}^{g}\left(E_{ir}^{L} + E_{ir}^{U}\right)$, where

$$E_{ir}^{L} = \left[\left[\frac{\partial^2 U_T}{\partial x^2} + \frac{\partial^2 U_T}{\partial y^2}\right]_r^L - [f(x_i , y_i)]_r^L\right]^2$$

$$E_{ir}^{U} = \left[\left[\frac{\partial^2 U_T}{\partial x^2} + \frac{\partial^2 U_T}{\partial y^2}\right]_r^U - [f(x_i , y_i)]_r^U\right]^2 \qquad (68)$$

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

## 11 Proposed Method

In this section we will introduce a novel method to modify the fuzzy neural network. This new method based on replacing each x in the input vector (training set) $\vec{x} = (x_1 , x_2 , \dots , x_n)$ , $x_j \in [a, b]$ by a polynomial of first degree.

In [40] Ezadi and parandin used the function: $Q(x) = \epsilon\ (x + 1)$, $\epsilon \in (0,1)$.

then the input vector will be: $(Q(x_1) , Q(x_2) , \dots\ Q(x_n))$, $Q(x_j) \in (a , b)$. In this paper, we named this proposed method modified fuzzy neural network. Using modified fuzzy 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 fuzzy neural network are converted to similar points in the open interval (a , b) by using the new approach, then the fuzzy network is trained in these similar areas [21,22].

## 12 Numerical Examples

To show the behavior and properties of the new method, two problems will be solved in this section. For each example, the accuracy of the method is illustrated by computing the deviations $[\text{error}]_r^L$ , $[\text{error}]_r^U$ where

$$[\text{error}]_r^L=\left|[U_a(x\,,y)]_r^L-[U_T(x\,,y)]_r^L\right|\,,\,[\text{error}]_r^U=\left|[U_a(x\,,y)]_r^U-[U_T(x\,,y)]_r^U\right|$$

and

$[U_a(x\,,y)]_r=[[U_a(x\,,y)]_r^L\,,[U_a(x\,,y)]_r^U]$ the analytical solution

$[U_T(x\,,y)]_r=[[U_T(x\,,y)]_r^L\,,[U_T(x\,,y)]_r^U]$ the trial solution

For all examples, multilayer perceptron consisting of one hidden layer with 10 units and one linear output unit is used. To minimize the error function, we used BFGS (Broyden-Fletcher-Goldfarb-Shanno) quasi-Newton method (For more details, see [41,42]).

**Example (1): Consider the fuzzy Poisson problem:**

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

where $\tilde{K}(r)=[0.75+0.25\,r\,,\ 1.25-0.25\,r]$ with the fuzzy conditions:

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

and

$$\tilde{U}\,(x\,,0)=\tilde{K}\,x\,,\ \tilde{U}\,(x\,,1)=\tilde{K}\,ex\,,\ 0\le x\le 1\,.$$

The fuzzy analytical solution for this problem is :

$$[U_a(x\,,y)]_r=[(0.75+0.25r)\,xe^y\,,(1.25-0.25r\,)\,xe^y].$$

The fuzzy trial solution for this problem is :

$$[U_T(x\,,y)]_r=[(0.75+0.25r)\,xe^y\,,(1.25-0.25r\,)\,xe^y]$$
$$+\,xy\,(1-x)\ \ (1-y)\ [N(x\,,y,p)]_r\,.$$

The error function for m = 10 units in the hidden layer and for g = 11 equally spaced points inside the interval [0 , 1] for each variable x and y is trained.

For $\epsilon=0.4$, the training set will be:

$$0.4\,(x+1)\ \ ,\forall\,x\in[0\,,1]\text{ and }0.4\,(y+1)\ \ ,\forall\,y\in[0\,,1].$$

Analytical and trial solutions for this problem can be found in Table (1) and Table (2).

For this problem, the minimized error function is:

$$E = \sum_{i=1}^{11}\left(E_{ir}^{L} + E_{ir}^{U}\right)$$

where

$$E_{ir}^{L} = \left[\begin{array}{c} (y_i - y_i^2)\left(\begin{array}{c}(x_i - x_i^2)\sum_{j=1}^{10}[v_j]_r^L\, w_{j1}^2 s''\left(Q(x_i)\, w_{j1} + Q(y_i)\, w_{j2} + b_j\right) \\ +(2-4x_i)\sum_{j=1}^{10}[v_j]_r^L w_{j1} s'\left(Q(x_i)\, w_{j1} + Q(y_i)\, w_{j2} + b_j\right) \\ -\,2\sum_{j=1}^{10}[v_j]_r^L s\left(Q(x_i)\, w_{j1} + Q(y_i)\, w_{j2} + b_j\right)\end{array}\right) \\ + \\ (x_i - x_i^2)\left(\begin{array}{c}(y_i - y_i^2)\sum_{j=1}^{10}[v_j]_r^L\, w_{j2}^2 s''\left(Q(x_i)\, w_{j1} + Q(y_i)\, w_{j2} + b_j\right) \\ +(2-4y_i)\sum_{j=1}^{10}[v_j]_r^L w_{j2} s'\left(Q(x_i)\, w_{j1} + Q(y_i)\, w_{j2} + b_j\right) \\ -\,2\sum_{j=1}^{10}[v_j]_r^L s\left(Q(x_i)\, w_{j1} + Q(y_i)\, w_{j2} + b_j\right)\end{array}\right)\end{array}\right]^2$$

$$E_{ir}^{U} = \left[\begin{array}{c} (y_i - y_i^2)\left(\begin{array}{c}(x_i - x_i^2)\sum_{j=1}^{10}[v_j]_r^U\, w_{j1}^2 s''\left(Q(x_i)\, w_{j1} + Q(y_i)\, w_{j2} + b_j\right) \\ +(2-4x_i)\sum_{j=1}^{10}[v_j]_r^U w_{j1} s'\left(Q(x_i)\, w_{j1} + Q(y_i)\, w_{j2} + b_j\right) \\ -\,2\sum_{j=1}^{10}[v_j]_r^U s\left(Q(x_i)\, w_{j1} + Q(y_i)\, w_{j2} + b_j\right)\end{array}\right) \\ + \\ (x_i - x_i^2)\left(\begin{array}{c}(y_i - y_i^2)\sum_{j=1}^{10}[v_j]_r^U\, w_{j2}^2 s''\left(Q(x_i)\, w_{j1} + Q(y_i)\, w_{j2} + b_j\right) \\ +(2-4y_i)\sum_{j=1}^{10}[v_j]_r^U w_{j2} s'\left(Q(x_i)\, w_{j1} + Q(y_i)\, w_{j2} + b_j\right) \\ -\,2\sum_{j=1}^{10}[v_j]_r^U s\left(Q(x_i)\, w_{j1} + Q(y_i)\, w_{j2} + b_j\right)\end{array}\right)\end{array}\right]^2$$

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

| **x** | **y** | $[\mathbf{U_a}(\mathbf{x},\mathbf{y})]_r^L$ | $[\mathbf{U_a}(\mathbf{x},\mathbf{y})]_r^U$ | $[\mathbf{U_T}(\mathbf{x},\mathbf{y})]_r^L$ | $[\mathbf{U_T}(\mathbf{x},\mathbf{y})]_r^U$ | $[\mathbf{error}]_r^L$ | $[\mathbf{error}]_r^U$ |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0.1 | 0.1 | 0.096702455 | 0.124331728 | 0.096702461 | 0.124331737 | 6.6854 e-9 | 9.3281 e-9 |
| 0.2 | 0.2 | 0.213745482 | 0.274815620 | 0.213745474 | 0.274815626 | 8.5672 e-9 | 6.5423 e-9 |
| 0.3 | 0.3 | 0.354337937 | 0.455577347 | 0.354337929 | 0.455577342 | 8.3301 e-9 | 5.7664 e-9 |
| 0.4 | 0.4 | 0.522138644 | 0.671321113 | 0.522138646 | 0.671321109 | 2.4376 e-9 | 4.9806 e-9 |
| 0.5 | 0.5 | 0.721315555 | 0.927405714 | 0.721315505 | 0.927405718 | 5.0158 e-8 | 4.5307 e-9 |
| 0.6 | 0.6 | 0.956612370 | 1.229930190 | 0.956612401 | 1.229930266 | 3.1330 e-8 | 7.6771 e-8 |
| 0.7 | 0.7 | 1.233423533 | 1.585830257 | 1.233423539 | 1.585830346 | 6.7012 e-9 | 8.9137 e-8 |
| 0.8 | 0.8 | 1.557878650 | 2.002986836 | 1.557878654 | 2.002986840 | 4.8833 e-9 | 4.0264 e-9 |
| 0.9 | 0.9 | 1.936937450 | 2.490348150 | 1.936937457 | 2.490348154 | 7.6347 e-9 | 4.3421 e-9 |
| 1 | 1 | 2.378496600 | 3.058067057 | 2.378496600 | 3.058067057 | 0 | 0 |

**Example (2): Consider the fuzzy Wave problem:**

$$\frac{\partial^2 \tilde{U}}{\partial t^2}\,(x, t) - 4\,\frac{\partial^2 \tilde{U}}{\partial x^2}\,(x, t) = 0, \quad x, t \in [0, 1],$$

with the following fuzzy boundary and initial conditions:

$\tilde{U}(0,t) = 0$ , $\tilde{U}(1,t) = 0$ , 0 , $\tilde{U}(x,0) = \tilde{K}\sin(\pi x)$ and $\frac{\partial}{\partial t}\tilde{U}(x,0) = 0$.

where $\tilde{K}(r) = [0.75 + 0.25\,r,\ 1.25 - 0.25\,r]$ .

The fuzzy analytical solution for this problem is:

$[U_a(x,t)]_r = [(0.75 + 0.25r)\sin(\pi x)\cos(2\pi t), (1.25 - 0.25r)\sin(\pi x)\cos(2\pi t)]$.

The fuzzy trial solution for this problem is:

$[U_T(x,t)]_r = [(0.75 + 0.25r)(1 - t^2)\sin(\pi x), (1.25 - 0.25r)(1 - t^2)\sin(\pi x)] + x(1 - x)t^2\,[N(x,t,p)]_r$.

**Table 2. Numerical results for example (1), for x = y = 0.7**

| $\mathbf{r}$ | $[\mathbf{U_a(x,y)}]_r^L$ | $[\mathbf{U_a(x,y)}]_r^U$ | $[\mathbf{U_T(x,y)}]_r^L$ | $[\mathbf{U_T(x,y)}]_r^U$ | $[\mathbf{error}]_r^L$ | $[\mathbf{error}]_r^U$ |
|---|---|---|---|---|---|---|
| 0 | 1.057220171 | 1.762033619 | 1.057220173 | 1.762033614 | 2.1229 e-9 | 5.0107 e-9 |
| 0.1 | 1.092460844 | 1.726792947 | 1.092460840 | 1.726792941 | 4.7724 e-9 | 6.7036 e-9 |
| 0.2 | 1.127701516 | 1.691552274 | 1.127701508 | 1.691552268 | 8.3444 e-9 | 6.7939 e-9 |
| 0.3 | 1.162942189 | 1.656311602 | 1.162942196 | 1.656311611 | 7.0643 e-9 | 9.4339 e-9 |
| 0.4 | 1.198182861 | 1.621070930 | 1.198182810 | 1.621070921 | 5.1151 e-8 | 9.0052 e-9 |
| 0.5 | 1.233423533 | 1.585830257 | 1.233423539 | 1.585830346 | 6.7012 e-9 | 8.9137 e-8 |
| 0.6 | 1.268664206 | 1.550589585 | 1.268664248 | 1.550589588 | 4.2159 e-8 | 3.9211 e-9 |
| 0.7 | 1.303904878 | 1.515348912 | 1.303904925 | 1.515348894 | 4.7441 e-8 | 1.8195 e-8 |
| 0.8 | 1.339145550 | 1.480108240 | 1.339145556 | 1.480108230 | 6.2102 e-9 | 1.0122 e-8 |
| 0.9 | 1.374386223 | 1.444867568 | 1.374386230 | 1.444867571 | 7.7177 e-9 | 3.9898 e-9 |
| 1 | 1.409626895 | 1.409626895 | 1.409626903 | 1.409626898 | 8.0020 e-9 | 3.7220 e-9 |

In [17], Allahviranloo and Kermani solved this problem by using Finite Difference method for h=0.1 and k=0.001. The max absolute error at the point (0.1,0.001), ∀ r ∈[0,1] is 7.6247 e−6 .

Analytical and trial solutions for this problem can be found in Tables 3 and 4.

The minimized error function is: $E = \sum_{i=1}^{11}\left(E_{ir}^{L} + E_{ir}^{U}\right)$, where

**Table 3. Numerical results for example (2), for x =0.1, t=0.001**

| $\mathbf{r}$ | $[\mathbf{U_a(x,t)}]_r^L$ | $[\mathbf{U_a(x,t)}]_r^U$ | $[\mathbf{U_T(x,t)}]_r^L$ | $[\mathbf{U_T(x,t)}]_r^U$ | $[\mathbf{error}]_r^L$ | $[\mathbf{error}]_r^U$ |
|---|---|---|---|---|---|---|
| 0 | 0.231758171 | 0.386263618 | 0.231758196 | 0.386263624 | 2.5029e-8 | 0.6369e-8 |
| 0.1 | 0.239483443 | 0.378538345 | 0.239483480 | 0.378538333 | 3.7617e-8 | 1.2021e-8 |
| 0.2 | 0.247208715 | 0.370813073 | 0.247208676 | 0.370813168 | 3.9641e-8 | 9.5407e-8 |
| 0.3 | 0.254933988 | 0.363087801 | 0..254933954 | 0.363087888 | 3.4237e-8 | 8.7311e-8 |
| 0.4 | 0.262659260 | 0.355362528 | 0.262659284 | 0.355361709 | 2.4999e-8 | 8.1995e-7 |
| 0.5 | 0.270384532 | 0.347637256 | 0.270384547 | 0.347638049 | 1.5216e-8 | 7.9307e-7 |
| 0.6 | 0.278109805 | 0.339911984 | 0.278110262 | 0.339911194 | 4.5775e-7 | 7.9054e-7 |
| 0.7 | 0.285835077 | 0.332186711 | 0.285835487 | 0.332186630 | 4.1010e-7 | 8.1042e-8 |
| 0.8 | 0.293560349 | 0.324461439 | 0.293560292 | 0.324461524 | 5.7922e-8 | 8.5142e-8 |
| 0.9 | 0.301285622 | 0.316736167 | 0.301285692 | 0.316736258 | 7.0355e-8 | 9.1373e-8 |
| 1 | 0.309010894 | 0.309010894 | 0.309010824 | 0.309010990 | 7.0922e-8 | 9.6334e-8 |

**Table 4. Numerical results for example (2), for x =0.001, t=0.1**

| $\mathbf{r}$ | $[\mathbf{U_a(x\,,t)}]_\mathbf{r}^\mathbf{L}$ | $[\mathbf{U_a(x\,,t)}]_\mathbf{r}^\mathbf{U}$ | $[\mathbf{U_T(x\,,t)}]_\mathbf{r}^\mathbf{L}$ | $[\mathbf{U_T(x\,,t)}]_\mathbf{r}^\mathbf{U}$ | $[\mathbf{error}]_\mathbf{r}^\mathbf{L}$ | $[\mathbf{error}]_\mathbf{r}^\mathbf{U}$ |
|---|---|---|---|---|---|---|
| 0 | 0.001906198 | 0.003176997 | 0.001906962 | 0.003177081 | 7.6442e-7 | 8.4491e-8 |
| 0.1 | 0.001969738 | 0.003113457 | 0.001969130 | 0.003113543 | 6.0894e-7 | 8.6409e-8 |
| 0.2 | 0.002033278 | 0.003049917 | 0.002033340 | 0.003049851 | 6.2334e-8 | 6.6226e-8 |
| 0.3 | 0.002096818 | 0.002986377 | 0.002096748 | 0.002987041 | 7.0810e-8 | 6.6492e-7 |
| 0.4 | 0.002160358 | 0.002922837 | 0.002160426 | 0.002923537 | 6.8840e-8 | 7.0076e-7 |
| 0.5 | 0.002223897 | 0.002859297 | 0.002223827 | 0.002859920 | 7.0098e-8 | 6.2343e-7 |
| 0.6 | 0.002287437 | 0.002795757 | 0.002287382 | 0.002795806 | 5.5665e-8 | 4.9970e-8 |
| 0.7 | 0.002350977 | 0.002732217 | 0.002351027 | 0.002732264 | 5.0226e-8 | 4.7977e-8 |
| 0.8 | 0.002414517 | 0.002668677 | 0.002414570 | 0.002668647 | 5.3307e-8 | 3.0222e-8 |
| 0.9 | 0.002478057 | 0.002605137 | 0.002478014 | 0.002605161 | 4.3366e-8 | 2.4430e-8 |
| 1 | 0.002541597 | 0.002541597 | 0.002541638 | 0002541570 | 4.1121e-8 | 2.7786e-8 |

$$E_{ir}^{L}=\left[\begin{array}{c} (-4t_i^2)\begin{pmatrix} (x_i-x_i^2)\sum_{j=1}^{10}[v_j]_r^L\, w_{j1}^2 s''\left(Q(x_i)\,w_{j1}+Q(t_i)\,w_{j2}+b_j\right) \\ +(2-4x_i)\sum_{j=1}^{10}[v_j]_r^L w_{j1} s'\left(Q(x_i)\,w_{j1}+Q(t_i)\,w_{j2}+b_j\right) \\ -\,2\sum_{j=1}^{10}[v_j]_r^L s\left(Q(x_i)\,w_{j1}+Q(y_i)\,w_{j2}+b_j\right) \end{pmatrix} \\ + \\ (x_i-x_i^2)\begin{pmatrix} (t_i^2)\sum_{j=1}^{10}[v_j]_r^L\, w_{j2}^2 s''\left(Q(x_i)\,w_{j1}+Q(t_i)\,w_{j2}+b_j\right) \\ +(4t_i)\sum_{j=1}^{10}[v_j]_r^L w_{j2} s'\left(Q(x_i)\,w_{j1}+Q(t_i)\,w_{j2}+b_j\right) \\ +\,2\sum_{j=1}^{10}[v_j]_r^L s\left(Q(x_i)\,w_{j1}+Q(t_i)\,w_{j2}+b_j\right) \end{pmatrix} \\ +\,(4\pi^2(1-t_i^2)-2)(0.75+0.25r)\sin\pi x_i \end{array}\right]^2$$

$$E_{ir}^{U}=\left[\begin{array}{c} (-4t_i^2)\begin{pmatrix} (x_i-x_i^2)\sum_{j=1}^{10}[v_j]_r^U\, w_{j1}^2 s''\left(Q(x_i)\,w_{j1}+Q(t_i)\,w_{j2}+b_j\right) \\ +(2-4x_i)\sum_{j=1}^{10}[v_j]_r^U w_{j1} s'\left(Q(x_i)\,w_{j1}+Q(t_i)\,w_{j2}+b_j\right) \\ -\,2\sum_{j=1}^{10}[v_j]_r^U s\left(Q(x_i)\,w_{j1}+Q(y_i)\,w_{j2}+b_j\right) \end{pmatrix} \\ + \\ (x_i-x_i^2)\begin{pmatrix} (t_i^2)\sum_{j=1}^{10}[v_j]_r^U\, w_{j2}^2 s''\left(Q(x_i)\,w_{j1}+Q(t_i)\,w_{j2}+b_j\right) \\ +(4t_i)\sum_{j=1}^{10}[v_j]_r^U w_{j2} s'\left(Q(x_i)\,w_{j1}+Q(t_i)\,w_{j2}+b_j\right) \\ +\,2\sum_{j=1}^{10}[v_j]_r^U s\left(Q(x_i)\,w_{j1}+Q(t_i)\,w_{j2}+b_j\right) \end{pmatrix} \\ +\,(4\pi^2(1-t_i^2)-2)(1.25-0.25r)\sin\pi x_i \end{array}\right]^2$$

## 13 Conclusions

In this paper, we presented a novel approach based on fuzzy neural networks for solving fuzzy partial differential equations. We demonstrate the ability of fuzzy neural networks to approximate the solutions of FPDEs. From the two examples it is clear that the modified fuzzy neural network method gives best results and better accuracy comparison with usual fuzzy neural network. As well, we can conclude that the method we proposed can handle effectively all types of FPDEs 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. better results may be possible if one uses more neurons or more 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 three-dimensional fuzzy partial differential equations.

## Competing Interests

Authors have declared that no competing interests exist.

## References

[1] Kaleva O. Fuzzy differential equations. Fuzzy Sets and Systems. 1987;24:301–317.

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

[3] Nieto JJ. The Cauchy problem for continuous fuzzy differential equations. Fuzzy Sets and Systems. 1999;102:259–262.

[4] Buckley JJ, Feuring T. Fuzzy differential equations. Fuzzy Sets and Systems. 2000;110:43–54.

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

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

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

[8] Georgiou DN, Nieto JJ, et al. Initial value problems for higher-order fuzzy differential equations. Nonlinear Anal. 2005;63:587-600.

[9] Nieto JJ, Lopez R. Bounded solutions for fuzzy differential and integral equations. Chaos, Solitons and Fractals. 2006;27:1376–1386.

[10] Abbasbandy S, Allahviranloo T. Numerical solution of fuzzy differential equations by Taylor method. Journal of Computational Methods in Applied Mathematics. 2002;2:113-124.

[11] Abbasbandy S, Allahviranloo T. Numerical solution of fuzzy differential equations by Runge-Kutta method. J. Sci. Teacher Training University. 2002:1:3.

[12] Allahviranloo T, Ahmady N, et al. Numerical solution of fuzzy differential equations by predictor-corrector method. Information Sciences. 2007;177:1633-1647.

[13] Allahviranloo T, Ahmady E, et al. Nth- order fuzzy linear differential equations. Information Sciences. 2008;178:1309-1324 .

[14] Allahviranloo T, Kiani NA, et al. Solving fuzzy differential equations by differential transformation method. Information Sciences. 2009;179:956-966.

[15] Ghazanfari B, Shakerami A. Numerical solution of fuzzy differential equations extended Runge – Kutta- like formulae of order 4. Fuzzy Sets and Systems. 2011;189:74–91.

[16] Kastan A, Ivaz K. Numerical solution of fuzzy differential equations by Nystrom method. Chaos, Solitons and Fractals. 2009;41:859–868.

[17] Allahviranloo T, Kermani MA. Numerical methods for fuzzy linear partial differential equations under new definition for derivative. Iranian Journal of Fuzzy Systems. 2010;7(3):33-50.

[18] Dahalan AA, Muthuvalu MS, et al. Performance of HSAGE method with Seikkala derivative for 2-D fuzzy Poisson equation. Applied Mathematical Sciences. 2014;8(18):885-899.

[19] Lee H, Kang IS. Neural algorithms for solving differential equations. Journal of Computational Physics. 1990;91:110-131.

[20] Meade AJ, Fernandes AA. Solution of nonlinear ordinary differential equations by feed-forward neural networks. Mathematical and Computer Modelling. 1994;20(9):19-44.

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

[22] Lagaris IE, Likas A, et al. Artificial neural networks for solving ordinary and partial differential equations. Comput. Phys. Commun. 1997;104:1-26.

[23] Lagaris IE, Likas A, et al. Artificial neural networks for solving ordinary and partial differential equations. IEEE Transaction on Neural Networks. 1998;9(5):987-1000.

[24] Liu B, Jammes B. Solving ordinary differential equations by neural networks. Warsaw, Poland; 1999.

[25] Tawfiq LNM. On design and training of artificial neural network for solving differential equations. Ph.D. Thesis, College of Education Ibn AL-Haitham, University of Baghdad, Iraq; 2004.

[26] Malek A, Shekari R. Numerical solution for high order differential equations by using a hybrid neural network optimization method. Applied Mathematics and Computation. 2006;183:260-271.

[27] Pattanaik S, Mishra RK. Application of ANN for solution of PDE in RF engineering. International Journal on Information Sciences and Computing. 2008;2(1):74-79.

[28] Oraibi YA. Design feed-forward neural networks for solving ordinary initial value problem. M.Sc. Thesis, College of Education Ibn Al-Haitham, University of Baghdad, Iraq; 2011.

[29] Hussian EA, Suhhiem MH. Numerical solution of partial differential equations by using modified artificial neural network. Network and Complex Systems. 2015;5(6):11-21.

[30] Effati S, Pakdaman M. Artificial neural network approach for solving fuzzy differential equations. Information Sciences. 2010;180:1434-1457.

[31] Mosleh M, Otadi M. Fuzzy Fredholm integro-differential equations with artificial neural networks. Communications in Numerical Analysis. 2012;Article ID cna-00128:1-13.

[32] Ezadi S, Parandin N, et al. Numerical solution of fuzzy differential equations based on semi-Taylor by using neural network. Journal of Basic and Applied Scientific Research. 2013;3(1s):477-482.

[33] Hussian EA, Suhhiem MH. Modified artificial neural networks for solving fuzzy differential equations. Mathematical Theory and Modeling. 2015;5(6):176-192.

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

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

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

[37] Hussian EA, Suhhiem MH. Numerical solution of fuzzy differential equations by using modified fuzzy neural network. International Journal of Mathematical Archive. 2015;6(6):84-94.

[38] Otadi M, Mosleh M, et al. Solving fuzzy linear system by neural network and applications in Economics. Journal of Mathematics Extension. 2011;47-66.

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

[40] Ezadi S, Parandin N. An application of neural networks to solve ordinary differential equations. International Journal of Mathematical Modelling and Computations. 2013;3(3):245-252.

[41] Blomgren P. Numerical optimization. Quasi-Newton Methods; Convergence Analysis, Computational Sciences Research Center, San Diego State University, San Diego; 2014.

[42] Blomgren P. Numerical optimization. Quasi-Newton Methods; Convergence Analysis, Computational Sciences Research Center, San Diego State University, San Diego; 2014.