Annals of Pure and Applied Mathematics
Vol.19, No.2, 2019, 147-159
ISSN: 2279-087X (P), 2279-0888(online)
Published on 14 April 2019
www.researchmathsci.org
DOI: http://dx.doi.org/10.22457/apam.610v19n2a3

Annals of
Pure and Applied Mathematics

New Numerical Approach for Solving Fuzzy Boundary Value Problems

Mazin Hashim Suhhiem

Department of Statistics, University of Sumer, Alrifaee, Iraq
Email: mazin.suhhiem@yahoo.com

Received 2 March 2019; accepted 10 April 2019

Abstract. In this work we have introduced a modified method for solving second order fuzzy differential equations. This method based on the fully fuzzy neural network to find the numerical solution of the two- point fuzzy boundary value problems for the ordinary differential equations. The fuzzy trial solution of the two-point fuzzy boundary value problems is written based on the concepts of the fully fuzzy feed-forward neural networks which containing fuzzy adjustable parameters. In comparison with other numerical methods, the proposed method provides numerical solutions with high accuracy.

Keywords: Two-point fuzzy boundary value problem, fully fuzzy neural network, fuzzy trial solution, minimized error function, hyperbolic tangent activation function

AMS Mathematics Subject Classification (2010): 34A07

1. Introduction

Many methods have been developed so far for solving fuzzy differential equations (FDEs) since it is utilized widely for the purpose of modeling problems in science and engineering. Most of the practical problems require the solution of the FDE which satisfies fuzzy initial conditions or fuzzy boundary conditions, therefore, the FDE must be solved.Many FDE could not be solved exactly, thus considering their approximate solutions is becoming more important.

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

In 1990 researchers began using artificial neural network (ANN) for solving ordinary differential equation (ODE) and partial differential equation (PDE) such as : lee, Kang in [1]; Meade, Fernandez in [2,3]; Lagaris, Likas et al. in [4]; Liu, Jammes in

[5]; Tawfiq in [6]; malek, shekari in [7]; Pattanaik, Mishra in [8]; Baymani, Kerayechian, et al. in [9]; and other researchers.

In 2010 researchers began using ANN for solving fuzzy differential equation such as: Effati and pakdaman in [10]; Mosleh, Otadi in [11]; Ezadi, Parandin, et al. in [12].

In 2012 researchers began using partially (non fully) fuzzy artificial neural network (FANN) for solving fuzzy differential equation such as Mosleh, Otadi in [13,14,15]. In (2016) Suhhiem [16] developed and used partially FANN for solving fuzzy and non-fuzzy differential equations..

In this work we have used fully feed forward fuzzy neural network to find the numerical solution of the two- point fuzzy boundary value problems for the ordinary differential equations. The fuzzy trial solution of the fuzzy boundary value problem is written as a sum of two parts. The first part satisfies the fuzzy boundary condition, it contains no fuzzy adjustable parameters. The second part involves fully fuzzy feed-forward neural networks which containing fuzzy adjustable parameters.

2. Basic definitions

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

Definition 1. [16] The r - level (or r - cut) set of a fuzzy set $\widetilde{A}$ labeled by A_r, is the crisp set of all x in X(universal set) such that: $\mu_{\widetilde{A}}(x) \geq r$; i. e.

$$A_r = \{x \in X : \mu_{\widetilde{A}}(x) \geq r, r \in [0,1]\}. \quad (1)$$

Definition 2.[16, 17] **Extension principle**

Let X be the Cartesian product of universes $X_1, X_2, \ldots, X_m$ and $\widetilde{A}_1, \widetilde{A}_2, \ldots, \widetilde{A}_m$ be m - fuzzy subset in $X_1, X_2, \ldots, X_m$ respectively, with Cartesian product $\widetilde{A} = \widetilde{A}_1 \times \widetilde{A}_2 \times \ldots \times \widetilde{A}_m$ and f is a function from X to a universe Y, $(y = f(x_1, x_2, \ldots, x_m))$. Then, the extension principle allows to define a fuzzy subset $\widetilde{B} = f(\widetilde{A})$ in Y by**:**

$\widetilde{B} = \{(y, \mu_{\widetilde{B}}(y)) : y = f(x_1, x_2, \ldots, x_m), (x_1, x_2, \ldots, x_m) \in X\}$, where

$$\mu_{\widetilde{B}}(y) = \begin{cases} \sup_{(x_1,\ldots,x_m) \in f^{-1}(y)} \operatorname{Min}\left\{\mu_{\widetilde{A}_1}(x_1), \ldots, \mu_{\widetilde{A}_m}(x_m)\right\}, & \text{if } f^{-1}(y) \neq \emptyset \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

and f^{-1} is the inverse image off.

For m = 1, the extension principle will be :

$\widetilde{B} = f(\widetilde{A}) = \{(y, \mu_{\widetilde{B}}(y)) : y = f(x), x \in X\}$, where

$$\mu_{\widetilde{B}}(y) = \begin{cases} \sup_{x \in f^{-1}(y)} \mu_{\widetilde{A}}(x), & \text{if } f^{-1}(y) \neq \emptyset \\ 0, & \text{otherwise.} \end{cases} \quad (3)$$

which is one of the definitions of a fuzzy function Definition (4).

Definition 3. [**16, 17**] **Fuzzy number**

A fuzzy number $\tilde{u}$ is completely determined by an ordered pair of functions $\left(\underline{u}(r), \overline{u}(r)\right)$, $0 \leq r \leq 1$, which satisfy the following requirements :

1) $\underline{u}(r)$ is a bounded left continuous and non decreasing function on [0,1].
2) $\overline{u}(r)$ is a bounded left continuous and non increasing function on [0,1].
3) $\underline{u}(r) \leq \overline{u}(r)$, $0 \leq r \leq 1$.
The crisp number a is simply represented by :
$\underline{u}(r) = \overline{u}(r) = a$, $0 \leq r \leq 1$.
The set of all the fuzzy numbers is denoted by E^1.

Remark 1. [**10**] For arbitrary $\tilde{u}= \left(\underline{u}, \overline{u}\right)$, $\tilde{v}= \left(\underline{v}, \overline{v}\right)$ and $K \in R$, the addition and multiplication by K for all $r \in [0,1]$can be defined as:
1) $\underline{(u+v)}(r) = \underline{u}(r)+\underline{v}(r)$
2) $\overline{(u+v)}(r) = \overline{u}(r)+\overline{v}(r)$
3) $\underline{(Ku)}(r) = K\underline{u}(r)$, $\overline{(Ku)}(r) = K\overline{u}(r)$, if $K \geq 0$
4) $\underline{(Ku)}(r) = K\overline{u}(r)$, $\overline{(Ku)}(r) = K\underline{u}(r)$, if $K < 0$.

Remark 2. [**18**] The distance between two arbitrary fuzzy numbers $\tilde{u} = \left(\underline{u}, \overline{u}\right)$ and $\tilde{v} = \left(\underline{v}, \overline{v}\right)$ is given as :

$$D(\tilde{u}, \tilde{v}) = \left[\int_0^1 (\underline{u}(r) - \underline{v}(r))^2 dr + \int_0^1 (\overline{u}(r) - \overline{v}(r))^2 dr\right]^{\frac{1}{2}} \tag{4}$$

Remark 3. [**18**] (E^1, D) is a complete metric space.

Remark 4. [**13**] The operations of fuzzy numbers (in parametric form) can be generalized from that of crisp intervals. Let us have a look at the operations of intervals, $\forall a_1, b_1, a_2, b_2 \in R$, $A = [a_1, b_1]$ and $B = [a_2, b_2]$.
Assuming A and B numbers expressed as interval, main operations of intervals are :
1) Addition : $A + B = [a_1, b_1] + [a_2, b_2] = [a_1 + a_2, b_1 + b_2]$.
2) Subtraction : $A - B = [a_1, b_1] - [a_2, b_2] = [a_1 - b_2, b_1 - a_2]$.
3) Multiplication :
$A.B = [\min\{a_1 a_2, a_1 b_2, b_1 a_2, b_1 b_2\}, \max\{a_1 a_2, a_1 b_2, b_1 a_2, b_1 b_2\}]$
4)Division : $A/B = [\min\{a_1 / a_2, a_1 / b_2, b_1 / a_2, b_1 / b_2\}, \max\{a_1 / a_2, a_1 / b_2, b_1 / a_2, b_1 / b_2\}]$ excluding the case $a_2 = 0$ or $b_2 = 0$.
5) Inverse : $A^{-1} = [a_1, b_1]^{-1} = \left[\min\left\{\frac{1}{a_1}, \frac{1}{b_1}\right\}, \max\left\{\frac{1}{a_1}, \frac{1}{b_1}\right\}\right]$

excluding the case $a_1 = 0$ or $b_1 = 0$.

Definition 4. [16] **Fuzzy function**
A classical function F: $X \longrightarrow Y$ maps from a fuzzy domain $\tilde{A} \subseteq X$ into a fuzzy range $\tilde{B} \subseteq Y$ if and only if $\forall x \in X$, $\mu_{\tilde{B}}(F(x)) \geq \mu_{\tilde{A}}(x)$.

Remark 5. [10]
(1) The function $F : R \longrightarrow E^1$ is called a fuzzy function.
(2) We call every function defined in set $\tilde{A} \subseteq E^1$ to $\tilde{B} \subseteq E^1$ a fuzzy function.

Definition 5. [10] The fuzzy function $F: R \longrightarrow E^1$ is said to be continuous if :
For an arbitrary $t_1 \in R$ and $\epsilon > 0$ there exists a $\delta > 0$ such that :
$|t - t_1| < \delta \Rightarrow D(F(t), F(t_1)) < \epsilon$, where D is the distance between two fuzzy numbers.

Definition 6. [16] Let I be a real interval. The r-level set of the fuzzy function$y : I \rightarrow E^1$ can be denoted by :

$$[y(x)]^r = [y_1^r(x), y_2^r(x)] \qquad x \in I, r \in [0,1] \tag{5}$$

The Seikkala derivative $y'(x)$of the fuzzy function$y(x)$ is defined by :

$$[y'(x)]^r = [(y_1^r)'(x), (y_2^r)'(x)] \quad x \in I, r \in [0,1] \tag{6}$$

Definition 7. [10] let u, v $\in E^1$. If there exist $w \in E^1$ such that
u = v+w then w is called the H-difference (Hukuhara-difference) of u, v and it is denoted by w= $u \ominus v$.
In this work the $\ominus$ sign stands always for H-difference, and let us remark that $u \ominus v \neq u + (-1) v$.

Definition 8. [12, 19] **Fuzzy derivative**
Let F : (a,b)$\rightarrow E^1$ and $t_0 \in$ (a,b). We say that F is H-differential (Hukuhara-differential) at x_0, if there exists an element $F'(x_0) \in E^1$ such that for all h> 0 (sufficiently small), $\exists$ $F(x_0 + h) \ominus F(x_0)$, $F(x_0) \ominus F(x_0 - h)$ and the limits (in the metric D)

$$\lim_{h \to 0} \frac{F(x_0 + h) \ominus F(x_0)}{h} = \lim_{h \to 0} \frac{F(x_0) \ominus F(x_0 - h)}{h} = F'(x_0) \tag{7}$$

Then $F'(x_0)$ is called fuzzy derivative (H-derivative) of F at x_0, where D is the distance between two fuzzy numbers.

3. Fully fuzzy neural network [6,16]

Artificial neural networks are learning machines that can learn any arbitrary functional mapping between input and output. They are fast machines and can be implemented in parallel, either in software or in hardware. In fact, the computational complexity of ANN is polynomial in the number of neurons used in the network. Parallelism also brings with it the advantages of robustness and fault tolerance.

That is, ANN is a simplified mathematical model of the human brain. It can be implemented by both electric elements and computer software. It is a parallel distributed processor with large numbers of connections. It is an information processing system that has certain performance characters in common with biological neural networks.

A fuzzy neural network or neuro-fuzzy system is a learning machine that finds the parameters of a fuzzy system (i.e., fuzzy set, fuzzy rules) by exploiting approximation techniques from neural networks. Combining fuzzy systems with neural networks. Both neural networks and fuzzy systems have some things in common. They can be used for solving a problems (e. g. fuzzy differential equations, fuzzy integral equations, etc.).

If all the adjustable parameters (weights and biases) are fuzzy numbers then the fuzzy neural network is called fully fuzzy neural network, otherwise it is called partially fuzzy neural network.

4. Solution of FDEs by fully fuzzy neural network

To solve any fuzzy ordinary differential equation we consider a three – layered fully fuzzy neural network with one unit entry x, one hidden layer consisting of m activation functions and one unit output $N(x)$. The activation function for the hidden units of our fully fuzzy neural network is the hyperbolic tangent function ($s(\propto) = \tanh(\propto)$). Here, the dimension of fully fuzzy neural network is (1 × m × 1) (Fig. 1).

For every entry x (where $x \geq 0$) the mathematical operations in the fully fuzzy neural network can be described as :

Input unit: x = x, (8)

Hidden units :

$$[z_j]_r = \left[[z_j]_r^L,\ [z_j]_r^U\right] = \left[s\left([\mathrm{net}_j]_r^L\right), s\left([\mathrm{net}_j]_r^U\right)\right] \tag{9}$$

where

$$[\mathrm{net}_j]_r^L = x[w_j]_r^L + [b_j]_r^L \tag{10}$$

$$[\mathrm{net}_j]_r^U = x[w_j]_r^U + [b_j]_r^U \tag{11}$$

Output unit :

$$[N(x)]_r = [[N(x)]_r^L,\ [N(x)]_r^U] \tag{12}$$

where

$$[N(x)]_r^L = \sum_{j=1}^{m} \min\{[v_j]_r^L[z_j]_r^L, [v_j]_r^L[z_j]_r^U, [v_j]_r^U[z_j]_r^L, [v_j]_r^U[z_j]_r^U\} \tag{13}$$

$$[N(x)]_r^U = \sum_{j=1}^{m} \max\{[v_j]_r^L[z_j]_r^L, [v_j]_r^L[z_j]_r^U, [v_j]_r^U[z_j]_r^L, [v_j]_r^U[z_j]_r^U\} \tag{14}$$

where

$$[z_j]_r^L = s\left(x\,[w_j]_r^L + [b_j]_r^L\right) \quad (15)$$

$$[z_j]_r^U = s\left(x\,[w_j]_r^U + [b_j]_r^U\right) \quad (16)$$

Fully fuzzy neural network with crisp set inputs, fuzzy numbers adjustable parameters (weights and biases) and fuzzy numbers output solutions to the fuzzy ordinary differential equations is given in Fig. (1).

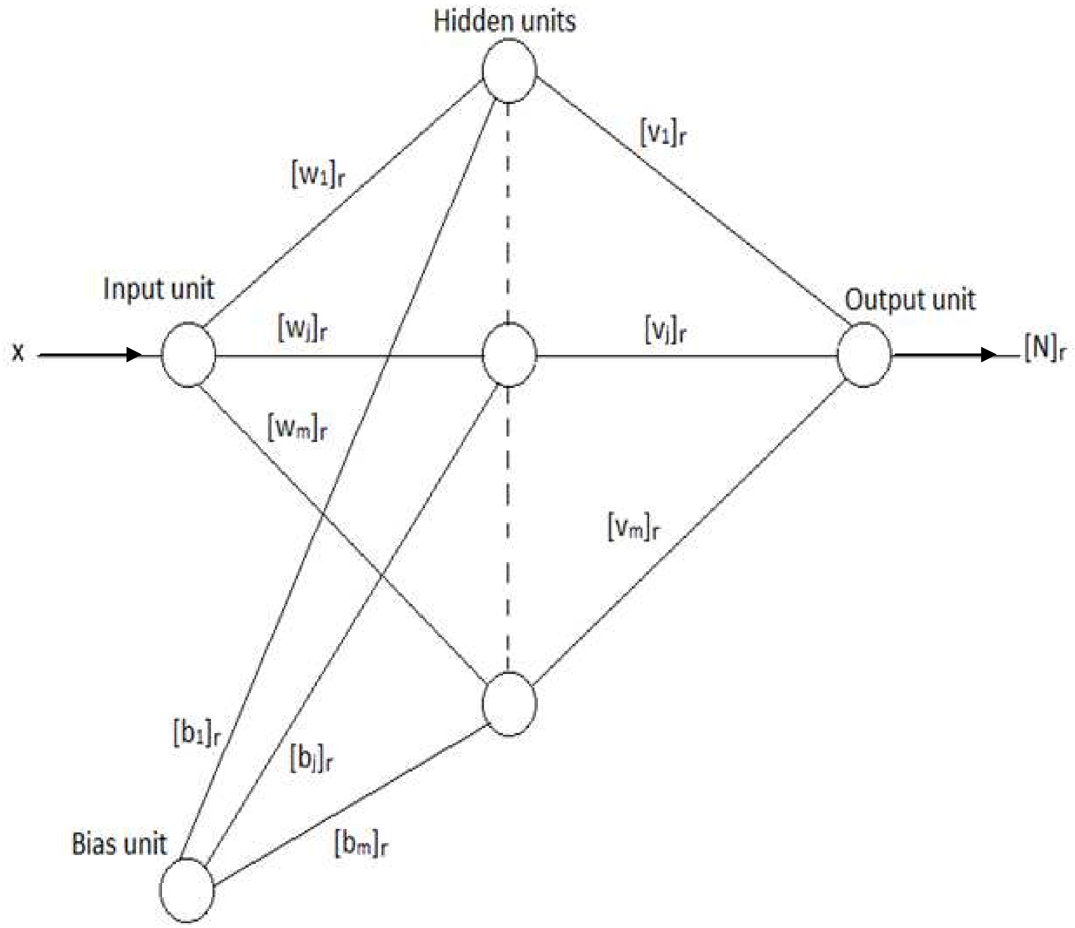

Figure 1: $(1 \times m \times 1)$ Fully fuzzy feed forward neural network.

5. Description of the proposed method

For illustration the proposed method, we will consider the two points fuzzy boundary value problems:

$$y''(x) = f\left(x\,,\, y(x),\ y'(x)\right), \quad x \in [a, b] \quad (17)$$

with the fuzzy boundary conditions:

$y(a) = A$ and $y(b) = B$, where A and B are fuzzy numbers in E^1 with r-level sets :

$[A]_r = [\underline{A}, \overline{A}]$ and $[B]_r = [\underline{B}, \overline{B}]$.

The fuzzy trial solution for this problem is :

$$[y_t(x)]_r = \frac{b-x}{b-a}[A]_r + \frac{x-a}{b-a}[B]_r + (x-a)(x-b)[N(x)]_r \quad (18)$$

This fuzzy trial solution by intention satisfies the fuzzy boundary conditions in (17).

The error function that must be minimized for problem (17) is in the form :

$$E = \sum_{i=1}^{g}\left(E_{ir}^{L} + E_{ir}^{U}\right) \quad (19)$$

where

$$E_{ir}^{L} = \left[\left[\frac{d^2y_t(x_i)}{dx^2}\right]_r^L - \left[f\left(x_i, y_t(x_i), \frac{d\,y_t(x_i)}{dx}\right)\right]_r^L\right]^2 \quad (20)$$

$$E_{ir}^{U} = \left[\left[\frac{d^2y_t(x_i)}{dx^2}\right]_r^U - \left[f\left(x_i, y_t(x_i), \frac{d\,y_t(x_i)}{dx}\right)\right]_r^U\right]^2 \quad (21)$$

where $\{x_i\}_{i=1}^{g}$ are discrete points belonging to the interval $[a, b]$ (training set) and in the cost function (19), E_r^L and E_r^U can be viewed as the squared errors for the lower limits and the upper limits of the r – level sets, respectively.

Now, to drive the minimized error function for problem (17) :

From (18) we can find :

$$[y_t(x)]_r^L = \frac{b-x}{b-a}[A]_r^L + \frac{x-a}{b-a}[B]_r^L + (x^2-(a+b)x+ab)[N(x)]_r^L \quad (22)$$

$$[y_t(x)]_r^U = \frac{b-x}{b-a}[A]_r^U + \frac{x-a}{b-a}[B]_r^U + (x^2-(a+b)x+ab)[N(x)]_r^U \quad (23)$$

Then we get :

$$\frac{d[y_t(x)]_r^L}{dx} = \frac{-1}{b-a}[A]_r^L + \frac{1}{b-a}[B]_r^L + (x^2-(a+b)x+ab)\frac{d[N(x)]_r^L}{dx} + (2x-a-b)[N(x)]_r^L. \quad (24)$$

$$\frac{d[y_t(x)]_r^U}{dx} = \frac{-1}{b-a}[A]_r^U + \frac{1}{b-a}[B]_r^U + (x^2-(a+b)x+ab)\frac{d[N(x)]_r^U}{dx} + (2x-a-b)[N(x)]_r^U. \quad (25)$$

Therefore, we have :

$$\left[\frac{d^2y_t(x)}{dx^2}\right]_r^L = (x^2-(a+b)x+ab)\frac{d^2[N(x)]_r^L}{dx^2} + 2(2x-a-b)\frac{d[N(x)]_r^L}{dx} + 2[N(x)]_r^L \quad (26)$$

$$\left[\frac{d^2y_t(x)}{dx^2}\right]_r^U = (x^2-(a+b)x+ab)\frac{d^2[N(x)]_r^U}{dx^2} + 2(2x-a-b)\frac{d[N(x)]_r^U}{dx} + 2[N(x)]_r^U \quad (27)$$

Then (20) and (21) can be rewritten as :

$$E_{ir}^{L} = \Big[(x_i^2-(a+b)x_i+ab)\frac{d^2[N(x_i)]_r^L}{dx^2} + 2(2x_i-a-b)\frac{d[N(x_i)]_r^L}{dx} + 2[N(x_i)]_r^L - f\Big(x_i, \frac{b-x_i}{b-a}[A]_r^L + \frac{x_i-a}{b-a}[B]_r^L + (x_i^2-(a+b)x_i+ab)[N(x_i)]_r^L, \frac{-1}{b-a}[A]_r^L + \frac{1}{b-a}[B]_r^L + (x_i^2-(a+b)x_i+ab)\frac{d[N(x_i)]_r^L}{dx} + (2x_i-a-b)[N(x_i)]_r^L\Big)\Big]^2 \quad (28)$$

$$E_{ir}^{U} = \Big[\big(x_i^2 - (a+b)x_i + ab\big) \frac{d^2[N(x_i)]_r^U}{dx^2} + 2(2x_i - a - b)\frac{d[N(x_i)]_r^U}{dx} + 2[N(x_i)]_r^U - f\Big(x_i, \frac{b-x_i}{b-a}[A]_r^U + \frac{x_i-a}{b-a}[B]_r^U + (x_i^2 - (a+b)x_i + ab)[N(x_i)]_r^U, \frac{-1}{b-a}[A]_r^U + \frac{1}{b-a}[B]_r^U + (x_i^2 - (a+b)x_i + ab)\frac{d[N(x_i)]_r^U}{dx} + (2x_i - a - b)[N(x_i)]_r^U\Big)\Big]^2 \quad (29)$$

where

$$[N(x_i)]_r^L = \sum_{j=1}^{m} \min\left\{ [v_j]_r^L\, s\left(x_i[w_j]_r^L + [b_j]_r^L\right), [v_j]_r^L\, s\left(x_i[w_j]_r^U + [b_j]_r^U\right), [v_j]_r^U\, s\left(x_i[w_j]_r^L + [b_j]_r^L\right), [v_j]_r^U\, s\left(x_i[w_j]_r^U + [b_j]_r^U\right) \right\} \quad (30)$$

$$[N(x_i)]_r^L = \sum_{j=1}^{m} \max\left\{ [v_j]_r^L\, s\left(x_i[w_j]_r^L + [b_j]_r^L\right), [v_j]_r^L\, s\left(x_i[w_j]_r^U + [b_j]_r^U\right), [v_j]_r^U\, s\left(x_i[w_j]_r^L + [b_j]_r^L\right), [v_j]_r^U\, s\left(x_i[w_j]_r^U + [b_j]_r^U\right) \right\} \quad (31)$$

$$\frac{d[N(x_i)]_r^L}{dx} = \sum_{j=1}^{m} \min\left\{ [v_j]_r^L[w_j]_r^L s'\left(x_i[w_j]_r^L + [b_j]_r^L\right), [v_j]_r^L[w_j]_r^U s'\left(x_i[w_j]_r^U + [b_j]_r^U\right), [v_j]_r^U[w_j]_r^L s'\left(x_i[w_j]_r^L + [b_j]_r^L\right), [v_j]_r^U[w_j]_r^U s'\left(x_i[w_j]_r^U + [b_j]_r^U\right) \right\} \quad (32)$$

$$\frac{d[N(x_i)]_r^U}{dx} = \sum_{j=1}^{m} \max\left\{ [v_j]_r^L[w_j]_r^L s'\left(x_i[w_j]_r^L + [b_j]_r^L\right), [v_j]_r^L[w_j]_r^U s'\left(x_i[w_j]_r^U + [b_j]_r^U\right), [v_j]_r^U[w_j]_r^L s'\left(x_i[w_j]_r^L + [b_j]_r^L\right), [v_j]_r^U[w_j]_r^U s'\left(x_i[w_j]_r^U + [b_j]_r^U\right) \right\} \quad (33)$$

$$\frac{d^2[N(x_i)]_r^L}{dx^2} = \sum_{j=1}^{m} \min\left\{ [v_j]_r^L([w_j]_r^L)^2 s''\left(x_i[w_j]_r^L + [b_j]_r^L\right), [v_j]_r^L([w_j]_r^U)^2 s''\left(x_i[w_j]_r^U + [b_j]_r^U\right), [v_j]_r^U([w_j]_r^L)^2 s''\left(x_i[w_j]_r^L + [b_j]_r^L\right), [v_j]_r^U([w_j]_r^U)^2 s''\left(x_i[w_j]_r^U + [b_j]_r^U\right) \right\} \quad (34)$$

$$\frac{d^2[N(x_i)]_r^U}{dx^2} = \sum_{j=1}^{m} \max\left\{ [v_j]_r^L([w_j]_r^L)^2 s''\left(x_i[w_j]_r^L + [b_j]_r^L\right), [v_j]_r^L([w_j]_r^U)^2 s''\left(x_i[w_j]_r^U + [b_j]_r^U\right), [v_j]_r^U([w_j]_r^L)^2 s''\left(x_i[w_j]_r^L + [b_j]_r^L\right), [v_j]_r^U([w_j]_r^U)^2 s''\left(x_i[w_j]_r^U + [b_j]_r^U\right) \right\} \quad (35)$$

where s' and s'' are the first and second derivative of the hyperbolic tangent function. Then we substitute (28) and (29) in (19) to find the error function that must be minimized for problem (17).

6. Numerical example

In this section, we will solve two problems about two-point fuzzy boundary value problem. We have used$(1 \times 10 \times 1)$fully fuzzy feed forward neural network. The activation function of each hidden unit is the hyperbolic tangent activation function. The analytical solutions $[y_a(x)]_r^L$ and $[y_a(x)]_r^U$ has been known in advance. Therefore, we test the accuracy of the obtained solutions by computing the deviation :

$\overline{e}(x, r) = \left|[y_a(x)]_r^U - [y_t(x)]_r^U\right|$, $\underline{e}(x, r) = \left|[y_a(x)]_r^L - [y_t(x)]_r^L\right|$

To minimize the error function we have used BFGS quasi-Newton method (For more details, see [16]). The computer programs which we have used in this work are coded in MATLAB 2015.

Example 1. Consider the linear fuzzy boundary value problem:
$y''(x) - y'(x) = 1.$ with $x \in [0, 0.5]$
$y(0) = [2 + r,\ 4 - r]$,
$y(0.5) = [5 + r,\ 7 - r]$, where $r \in [0, 1]$.
The analytical solutions for this problem are :
$[y_a(x)]_r^L = \left(2 + r - \frac{3}{e^{0.5}-1}\right) + \left(\frac{3}{e^{0.5}-1}\right)e^x$
$[y_a(x)]_r^U = \left(4 - r - \frac{3}{e^{0.5}-1}\right) + \left(\frac{3}{e^{0.5}-1}\right)e^x$
The trial solutions for this problem are :
$[y_t(x)]_r^L = (1 - 2x)(2 + r) + 2x(4 - r) + (x^2 - 0.5x)[N(x)]_r^L$
$[y_t(x)]_r^U = (1 - 2x)(5 + r) + 2x(7 - r) + (x^2 - 0.5x)[N(x)]_r^U$
The fully fuzzy feed forward neural network has been trained by using a grid of ten equidistant points in [0, 0.5].
Numerical solutions for this problem can be found in table (1).
The error function that must be minimized for this problem will be :

$$E = \sum_{i=1}^{11}\left(E_{ir}^L + E_{ir}^U\right) \tag{36}$$

where

$$E_{ir}^L = [(x_i^2 - 0.5x_i)\frac{d^2[N(x_i)]_r^L}{dx^2} + (4x_i - 1)\frac{d[N(x_i)]_r^L}{dx} + 2[N(x_i)]_r^L - (x_i^2 - 0.5x_i)\frac{d[N(x_i)]_r^L}{dx} - (2x_i - 0.5)[N(x_i)]_r^L + 4r - 5\,]^2 \tag{37}$$

$$E_{ir}^U = [(x_i^2 - 0.5x_i)\frac{d^2[N(x_i)]_r^U}{dx^2} + (4x_i - 1)\frac{d[N(x_i)]_r^U}{dx} + 2[N(x_i)]_r^U - (x_i^2 - 0.5x_i)\frac{d[N(x_i)]_r^U}{dx} - (2x_i - 0.5)[N(x_i)]_r^U + 4r - 5\,]^2 \tag{38}$$

Then we use (36) to update the weights and biases.

Example 2. Consider the non-linear fuzzy boundary value problem:
$y''(x) = -\left(y'(x)\right)^2$, with $x \in [0, 2]$
$y(0) = [r, 2 - r]$, $y(2) = [1 + r, 3 - r]$ and $r \in [0, 1]$.
The analytical solutions for this problem are:
$[y_a(x)]_r^L = \ln\left(x + \frac{2}{e-1}\right) + r - \ln\frac{2}{e-1}$
$[y_a(x)]_r^U = \ln\left(x + \frac{2}{e-1}\right) + 2 - r - \ln\frac{2}{e-1}$
The trial solutions for this problem are:
$[y_t(x)]_r^L = r\frac{2-x}{2} + (1 + r)\frac{x}{2} + x(x - 2)[N(x)]_r^L$
$[y_t(x)]_r^U = (2 - r)\frac{2-x}{2} + (3 - r)\frac{x}{2} + x(x - 2)[N(x)]_r^U$

The fully fuzzy feed forward neural network has been trained by using a grid of ten equidistant points in [0, 2].
Numerical solution for this problem can be found in table (2).
The error function that must be minimized for this problem will be :

$$E= \sum_{i=1}^{11}\left(E_{ir}^{L} + E_{ir}^{U}\right) \tag{39}$$

where

$$E_{ir}^{L}=[(x_i{}^2 - 2x_i)\frac{d^2[N(x_i)]_r^L}{dx^2} + (4x_i - 4)\frac{d[N(x_i)]_r^L}{dx} + 2[N(x_i)]_r^L + ((x_i{}^2 - 2x_i)\frac{d[N(x_i)]_r^L}{dx} + (2x_i - 2)[N(x_i)]_r^L + 0.5)^2]^2 \tag{40}$$

$$E_{ir}^{L}=[(x_i{}^2 - 2x_i)\frac{d^2[N(x_i)]_r^U}{dx^2} + (4x_i - 4)\frac{d[N(x_i)]_r^U}{dx} + 2[N(x_i)]_r^U + ((x_i{}^2 - 2x_i)\frac{d[N(x_i)]_r^U}{dx} + (2x_i - 2)[N(x_i)]_r^U + 0.5)^2]^2 \qquad 41)$$

Then we use (39) to update the weights and biases.
For the above two problems we have

$$[N(x_i)]_r^L = \sum_{j=1}^{10} \min\{ [v_j]_r^L s\left(x_i[w_j]_r^L + [b_j]_r^L\right), [v_j]_r^L s\left(x_i[w_j]_r^U + [b_j]_r^U\right), [v_j]_r^U s\left(x_i[w_j]_r^L + [b_j]_r^L\right), [v_j]_r^U s\left(x_i[w_j]_r^U + [b_j]_r^U\right) \}$$

$$[N(x_i)]_r^L = \sum_{j=1}^{10} \max\{ [v_j]_r^L s\left(x_i[w_j]_r^L + [b_j]_r^L\right), [v_j]_r^L s\left(x_i[w_j]_r^U + [b_j]_r^U\right), [v_j]_r^U s\left(x_i[w_j]_r^L + [b_j]_r^L\right), [v_j]_r^U s\left(x_i[w_j]_r^U + [b_j]_r^U\right) \}$$

$$\frac{d[N(x_i)]_r^L}{dx} = \sum_{j=1}^{10} \min\{ [v_j]_r^L[w_j]_r^L s'\left(x_i[w_j]_r^L + [b_j]_r^L\right), [v_j]_r^L[w_j]_r^U s'\left(x_i[w_j]_r^U + [b_j]_r^U\right), [v_j]_r^U[w_j]_r^L s'\left(x_i[w_j]_r^L + [b_j]_r^L\right), [v_j]_r^U[w_j]_r^U s'\left(x_i[w_j]_r^U + [b_j]_r^U\right) \}$$

$$\frac{d[N(x_i)]_r^U}{dx} = \sum_{j=1}^{10} \max\{ [v_j]_r^L[w_j]_r^L s'\left(x_i[w_j]_r^L + [b_j]_r^L\right), [v_j]_r^L[w_j]_r^U s'\left(x_i[w_j]_r^U + [b_j]_r^U\right), [v_j]_r^U[w_j]_r^L s'\left(x_i[w_j]_r^L + [b_j]_r^L\right), [v_j]_r^U[w_j]_r^U s'\left(x_i[w_j]_r^U + [b_j]_r^U\right) \}$$

$$\frac{d^2[N(x_i)]_r^L}{dx^2} = \sum_{j=1}^{10} \min\{ [v_j]_r^L([w_j]_r^L)^2 s''\left(x_i[w_j]_r^L+[b_j]_r^L\right), [v_j]_r^L([w_j]_r^U)^2 s''\left(x_i[w_j]_r^U + [b_j]_r^U\right), [v_j]_r^U([w_j]_r^L)^2 s''\left(x_i[w_j]_r^L+[b_j]_r^L\right), [v_j]_r^U([w_j]_r^U)^2 s''\left(x_i[w_j]_r^U + [b_j]_r^U\right) \}$$

$$\frac{d^2[N(x_i)]_r^U}{dx^2} = \sum_{j=1}^{10} \max\{ [v_j]_r^L([w_j]_r^L)^2 s''\left(x_i[w_j]_r^L+[b_j]_r^L\right), [v_j]_r^L([w_j]_r^U)^2\, s''\left(x_i[w_j]_r^U + [b_j]_r^U\right), [v_j]_r^U([w_j]_r^L)^2 s''\left(x_i[w_j]_r^L+[b_j]_r^L\right), [v_j]_r^U([w_j]_r^U)^2 s''\left(x_i[w_j]_r^U + [b_j]_r^U\right) \}$$

7. Conclusion

In this work, we have introduced a modified method to findthe numerical solution of the two- point fuzzy boundary value problems for the ordinary differential equations. This method based on the fully fuzzy neural network to approximate the solution of the second order fuzzy differential equations. For future studies, one can extend this method to find a numerical solution of the fuzzy partial differential equation.

Table 1: Numerical result for example (1), x=1.

r	$[y_t(x)]_r^L$	$\underline{e}(x, r)$	$[y_t(x)]_r^U$	$\overline{e}(x, r)$
0	9.946164141	3.29137e-7	11.94616425	4.33916e-7
0.1	10.04616401	1.96846e-7	11.84616411	2.93475e-7
0.2	10.14616481	9.95565e-7	11.74616478	9.70548e-7
0.3	10.24616458	7.63284e-7	11.64616385	3.95104e-8
0.4	10.34616447	6.60993e-7	11.54616387	5.67802e-8
0.5	10.44616422	4.09513e-7	11.44616389	7.56011e-8
0.6	10.54616396	1.47232e-7	11.34616391	9.53493e-8
0.7	10.64616391	9.75941e-8	11.24616382	1.15291e-8
0.8	10.74616385	3.39072e-8	11.14616384	2.63433e-8
0.9	10.84616389	7.52383e-8	11.04616386	5.26859e-8
1	10.94616389	7.39070e-8	10.94616386	4.56782e-8

Table 2: Numerical result for example (2), x=1.

r	$[y_t(x)]_r^L$	$\underline{e}(x, r)$	$[y_t(x)]_r^U$	$\overline{e}(x, r)$
0	0.620114507	3.24734e-10	2.620114507	8.46634e-10
0.1	0.720114507	4.66221-10	2.520114507	9.79602e-10
0.2	0.820114507	2.03208e-10	2.420114507	6.85555e-10
0.3	0.920114507	3.80684e-10	2.320114513	6.62032e-9
0.4	1.020114507	4.09557e-10	2.220114514	7.59010e-9
0.5	1.120114507	3.50405e-10	2.120114508	1.74006e-9

0.6	1.220114507	4.59008e-10	2.020114507	9.00817e-10
0.7	1.320114516	9.46681e-9	1.920114507	9.21604e-10
0.8	1.420114512	5.06564e-9	1.820114507	4.99811e-10
0.9	1.520114507	8.21899e-10	1.720114514	7.15955e-9
1	1.620114514	7.88763e-9	1.620114508	1.02988e-9

Acknowledgements. The author thank the anonymous referee and the Editor for their valuable comments and suggestions, which were very helpful in improving the presentation of this work.

REFERENCES

1. H.Lee and I.S.Kang, Neural algorithms for solving differential equations, *Journal of Computational Physics*, 91 (1990) 110-131.
2. A.J.Meade and A.A.Fernandes, The numerical solution of linear ordinary differential equations by feed-forward neural networks, *Mathematical and Computer Modeling*, 19(12) (1994) 1-25.
3. A.J.Meade and A.A.Fernandes, Solution of nonlinear ordinary differential equations by feed-forward neural networks, *Mathematical and Computer Modeling,* 20(9) (1994) 19-44.
4. I.E.Lagaris and A.Likas, Artificial neural networks for solving ordinary and partial differential equations, *Journal of Computational Physics*, 104 (1997) 1-26.
5. B.Liu and B.Jammes, *Solving Ordinary Differential Equations by Neural Networks,* Warsaw, Poland, 1999.
6. L.N.Tawfiq, 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.
7. A.Malek and R.Shekari, Numerical solution for high order differential equations by using a hybrid neural network optimization method, *Applied Mathematics and Computation,* 183 (2006) 260-271.
8. S.Pattanaik and R.K.Mishra, Application of ANN for solution of PDE in RF engineering, *International Journal on Information Sciences and Computing,* 2(1) (2008) 74-79.
9. M.Baymani and A.Kerayechian, Artificial neural networks approach for solving stokes problem, *Applied Mathematics*, 1 (2010) 288-292.
10. S.Effati and M.Pakdaman, Artificial neural network approach for solving fuzzy differential equations, *Information Sciences*, 180 (2010) 1434-1457.
11. M.Mosleh and M.Otadi, Fuzzy Fredholm integro-differential equations with artificial neural networks, *Communications in Numerical Analysis,* Article ID 00128 (2012) 1-13.
12. S.Ezadi and N.Parandin, Numerical solution of fuzzy differential equations based on semi-Taylor by using neural network, *Journal of Basic and Applied Scientific Research*, 3(1s) (2013) 477-482.

13. M.Mosleh and M.Otadi, Simulation and evaluation of fuzzy differential equations by fuzzy neural network, *Applied Soft Computing*, 12 (2012) 2817-2827.
14. M.Mosleh, Fuzzy neural network for solving a system of fuzzy differential equations, *Applied Soft Computing*, 13 (2013) 3597-3607.
15. M.Mosleh and M.Otadi, Solving the second order fuzzy differential equations by fuzzy neural network, *Journal of Mathematical Extension*, 8(1) (2014) 11-27.
16. M.S.Suhhiem, *Fuzzy artificial neural network for solving fuzzy and non-fuzzy differential equations*, Ph.D. Thesis, College of Sciences, AL-Mustansiriyah University, Iraq, 2016.
17. T.Selvakumar and M.V.Suresh, Interval valued Q-fuzzy quasi-ideals in a semigroups, *Annals of Pure and Applied Mathematics*, 18(1) (2018) 83-90.
18. R.Sharma and B.A.Deole, Fuzzy semi – open sets and fuzzy pre – open sets in fuzzy quad topological space, *Annals of Pure and Applied Mathematics*, 17(1) (2018)123-134.
19. M.Saradha, Second order hybrid fuzzy fractional differential equations by Runge-Kutta 6th order Fehlberg method, *Annals of Pure and Applied Mathematics*, 17(2) (2018) 123-134.