Ratio Mathematica Volume 38, 2020, pp. 287-311

Analytical and numerical solution of differential equations with generalized fuzzy derivative

Basim Nasih Abood*

Abstract

The aim of this work is to present a novel approach based on the fuzzy neural network for finding the numerical solution of the first order fuzzy differential equations under generalized H-derivation.The differentiability concept that used in this paper is the generalized differentiability since a first order fuzzy differential equation under this differentiability can have two solutions.The fuzzy trial solution of the fuzzy initial value problem is written as a sum of two parts. The first part satisfies the fuzzy condition, it contains no fuzzy adjustable parameters. The second part involves fuzzy feed-forward neural networks containing fuzzy adjustable parameters. This method, in comparison with existing numerical methods and the analytical solutions, shows that the use of fuzzy neural networks provides solutions with good generalization and high accuracy.

Keywords: Fuzzy differential equations; generalized H-derivation; fuzzy neural network; fuzzy trial solution; error function.

---

* Department of Mathematics, University of Wasit, Alkut, Iraq; basim.nasih@yahoo.com
Received on December 10th, 2019. Accepted on March 3rd, 2020. Published on June 30th, 2020. doi: 10.23755/rm.v38i0.519. ISSN: 1592-7415. eISSN: 2282-8214.

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 FDE which satisfies fuzzy initial or fuzzy boundary conditions, therefore, a fuzzy initial or fuzzy boundary problem should be solved. However, many fuzzy initial or fuzzy boundary value problems could not be solved exactly, sometimes it is even impossible to find their analytical solutions. Thus, considering their approximate solutions is becoming more important [1].

The theory of FDE was first formulated by Kaleva and Seikkala. Kaleva had 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 fuzzy differential equation [2].

In recent years artificial neural network (ANN) for estimation of the ordinary differential equation (ODE) and partial differential equation (PDE) has been used. We briefly review some articles in the literature concerning the differential equations. Lee and Kang in [3] used parallel processor computers to solve a first order differential equation with Hopfield neural network models. Meade and Fernandez in [4,5] solved linear and non-linear ODEs by using feed-forward neural networks (FFNN) architecture and B-splines of degree one. Lagaris and Likas in [6,7] used ANN for solving ODEs and PDEs with the initial / boundary value problems. Liu and Jammes in [8] developed some properties of the trial solution to solve the ODEs by using ANN. Ali and Ucar in [9] solved the vibration control problems by using ANN. Tawfiq in [10] presented and developed supervised and unsupervised algorithms for solving ODE and PDE. Malek and shekari in [11] presented numerical method based on ANN and optimization techniques which the higher-order ODE answers approximates by finding a package form analytical of specific functions. Pattanaik and Mishra in [12] applied and developed some properties of ANN for solution of PDE in RF Engineering**.** Baymani and Kerayechian in [13] proposed ANN approach for solving stokes problems. Oraibi in [14] designed FFNN for solving ordinary initial value problem. Ali in [15]

designed fast FFNN to solve two-point boundary value problems. Hussein in [16] designed fast FFNN to solve singular boundary value problems. Tawfiq and Al-Abrahemee in [17] designed ANN to solve singular perturbation problems, and other researchers.

Numerical solution of FDE by using ANN is the subject of a very modern because it only goes back to 2010. Effati and pakdaman in [18] used ANN for solving FDE, they used for the first time the ANN to approximate fuzzy initial value problems. Mosleh and Otadi in [19] used ANN for solving fuzzy Fredholm integro-differential equations. Ezadi and Parandin in [20] used ANN based on semi-Taylor series to solve first order FDE.

Numerical solution of FDE by using fuzzy artificial neural network (FANN) is more modern than the previous subject, where it goes back to 2012. Mosleh and Otadi in [21] used FANN for solving first order FDE, they used for the first time FANN to approximate fuzzy initial value problems. Mosleh in [22] used FANN for solving a system of FDE. Mosleh and Otadi in [23] used FANN for solving second order FDE. Suhhiem in [24] developed and used FANN for solving fuzzy and non-fuzzy differential equations.

In 2008, the concept of the generalized Hukuhara – differentiability is studied by Chalco-Cano and Roman Flores [25,26] to solve FDE.

In this work**,** for solving FDE Under Generalized H – Derivation, we present modified method which relies on the function approximation capabilities of fuzzy FFNN and results in the construction of a solution written in a differentiable, closed analytic form. This form employs fuzzy FFNN as the basic approximation element, whose fuzzy parameters (weights and biases) are adjusted to minimize an appropriate error function. To train the FANN which we design, we employ optimization techniques, which in turn require the computation of the gradient of the error with respect to the network parameters. In this proposed approach the model function is expressed as the sum of the two terms: the first term satisfies the fuzzy initial / fuzzy boundary conditions and contains no fuzzy adjustable parameters. The second term can be found by using fuzzy FFNN, which is trained so as to satisfy the FDE.

2 Basic Definitions

2.1 Fuzzy Concepts

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

Definition (2.1.1), [2] : 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(2.1.2),[2]: 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

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

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

For m = 1, the extension principle will be :

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

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

Definition (2.1.3), [24]: Fuzzy Number

A fuzzy number $\tilde{u}$ is completely determined by an ordered pair of functions $\left(\underline{u}(r), \overline{u}(r)\right)$, $0 \le r \le 1$, 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) $\underline{u}(r) \le \overline{u}(r)$, $0 \le r \le 1$.

The crisp number a is simply represented by:

$\underline{u}(r) = \overline{u}(r) = a$, $0 \le r \le 1$.

The set of all the fuzzy numbers is denoted by E^1.

Remark (1), [24]: 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 can be defined as :

1) $\underline{(u+v)}\ (r) = \underline{u}(r) + \underline{v}(r)$ (4)

2) $\overline{(u+v)}\ (r) = \overline{u}(r) + \overline{v}(r)$ (5)

3) $\underline{(Ku)}\ (r) = K\,\underline{u}(r)$, $\overline{(Ku)}\ (r) = K\,\overline{u}(r)$, if $K \ge 0$ (6)

4) $\underline{(Ku)}\ (r) = K\,\overline{u}(r)$, $\overline{(Ku)}\ (r) = K\,\underline{u}(r)$, if $K < 0$. (7)

For all $r \in [0,1]$.

Remark (2), [1]:

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:

$$D(\tilde{u}, \tilde{v}) = \left[\int_0^1 \left(\underline{u}(r) - \underline{v}(r)\right)^2 dr + \int_0^1 \left(\overline{u}(r) - \overline{v}(r)\right)^2 dr\right]^{\frac{1}{2}} \quad (8)$$

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

Remark (4), [2]: 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 $b_1 = 0$.

In the case of $0 \leq a_2 \leq b_2$, multiplication operation can be simplified as:

$A.\ B = [\min\{a_1 a_2, a_1 b_2\}, \max\{b_1\ a_2, b_1 b_2\}]$

when previous sets A and B is defined in the positive real number R^+, the operations of multiplication, division and inverse are written as :

3′) Multiplication: $A.\ B = [a_1, b_1].\ [a_2, b_2] = [a_1 a_2, b_1 b_2]$

4′) Division: $A / B = [a_1, b_1] / [a_2, b_2] = \left[\frac{a_1}{b_2}, \frac{b_1}{a_2}\right]$.

5′) Inverse: $A^{-1} = [a_1, b_1]^{-1} = \left[\frac{1}{b_1}, \frac{1}{a_1}\right]$.

Definition (2.1.4), [24] : Triangular Fuzzy Number

Among the various shapes of fuzzy numbers, triangular fuzzy numbers is the most popular one. A triangular fuzzy number is a fuzzy number represented with three points as follows: $\widetilde{A} = (a_1, a_2, a_3)$, where $a_1 \leq a_2 \leq a_3$.

This representation is interpreted as membership functions:

$$\mu_{\widetilde{A}}(x) = \begin{cases} 0 & , \text{if} \quad x < a_1 \\ \frac{x - a_1}{a_2 - a_1} & , \text{if} \quad a_1 \le x \le a_2 \\ \frac{a_3 - x}{a_3 - a_2} & , \text{if} \quad a_2 \le x \le a_3 \\ 0 & , \text{if} \quad x > a_3 \end{cases} \tag{9}$$

Now if you get crisp interval by r - cut operation, interval $[A]_r$ shall be obtained as follows $\forall\, r \in [0,1]$ from: $\frac{\underline{A} - a_1}{a_2 - a_1} = r$, $\frac{a_3 - \overline{A}}{a_3 - a_2} = r$,

We get: $\underline{A} = (a_2 - a_1)r + a_1$, $\overline{A} = (a_2 - a_3)\, r + a_3$.

Thus: $[A]_r = \left[\underline{A}, \overline{A}\right] = [(a_2 - a_1)r + a_1,\ (a_2 - a_3)\, r + a_3]$ (10)

which is the parametric form of triangular fuzzy number $\widetilde{A}$.

Definition $(2.1.5)$, $[18]$: Fuzzy Function

A classical function $F : X \longrightarrow Y$ maps from a fuzzy domain $\widetilde{A} \subseteq X$ into a fuzzy range $\widetilde{B} \subseteq Y$ if and only if $\forall\, x \in X$, $\mu_{\widetilde{B}}(F(x)) \ge \mu_{\widetilde{A}}(x)$.

Remark (5) , $[18]$:

(1) The function $F : R \longrightarrow E^1$ is called a fuzzy function .

(2) We call every function defined in set $\widetilde{A} \subseteq E^1$ to $\widetilde{B} \subseteq E^1$ a fuzzy function.

Definition $(2.1.6)$, $[18]$: 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 (2.1.7),[25]: Let I be a real interval. The r-level set of the fuzzy function $y : I \to E^1$ can be denoted by :

$$[y(t)]^r = [y_1^r(t), y_2^r(t)] \quad t \in I \tag{11}$$

The Seikkala derivative $y'(t)$ of the fuzzy function $y(t)$ is defined by:

$$[y'(t)]^r = [(y_1^r)'(t), (y_2^r)'(t)] \quad t \in I \tag{12}$$

2.2 **H – Differentiability**

In this subsection, the basic definitions which are used in H – differentiability are introduced.

Definition (2.2.1), [18]**:** 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 ⊖ v.

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

Definition (2.2.2), [24]

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 $F'(t_0) \in E^1$ such that for all h> 0 (sufficiently small), ∃ F (t_0 +h)⊖$F(t_0)$, $F(t_0)$ ⊖ F (t_0 - h) and the limits (in the metric D)

$$\lim_{h\to 0} \frac{F(t_0+h) \ominus F(t_0)}{h} = \lim_{h\to 0} \frac{F(t_0) \ominus F(t_0-h)}{h} = F'(t_0) \tag{13}$$

Then $F'(t_0)$ is called fuzzy derivative (H-derivative) of F at t_0.

where D is the distance between two fuzzy numbers.

It is necessary to note that the definition (2.2.2) is the classical definition of the H-derivative (or differentiability in the sense of Hukuhara).

Definition (2.2.3), [25, 26]**:**

Let $F : T \to E^1$ and $t_0 \in T \subseteq R$. F is differentiable at t_0, if

(1) there exist an element $F'(t_0) \in E^1$, such that for all $h > 0$ sufficiently small, there are $F(t_0+h) \ominus F(t_0), F(t_0) \ominus F(t_0-h)$ and the limits (in the metric D)

$$\lim_{h\to 0} \frac{F(t_0+h) \ominus F(t_0)}{h} = \lim_{h\to 0} \frac{F(t_0) \ominus F(t_0-h)}{h} = F'(t_0) \tag{14}$$

(in this case, F is called (1)-differentiable)

or

(2) there exist an element $F'(t_0) \in E^1$, such that for all $h > 0$ sufficiently small, there are $F(t_0) \ominus F(t_0 + h)$, $F(t_0 - h) \ominus F(t_0)$ and the limits (in the metric D)

$$\lim_{h\to 0} \frac{F(t_0) \ominus F(t_0 + h)}{-h} = \lim_{h\to 0} \frac{F(t_0 - h) \ominus F(t_0)}{-h} = F'(t_0) \qquad (15)$$

(in this case, F is called (2)-differentiable)

Where the relation (1) is the classical definition of the H-derivative.

Theorem (1) : Let $F : I \to E^1$ be a function and denote $[F(t)]^r = [f_r(t), g_r(t)]$, for each $r \in [0,1]$. Then

(i) If F is differentiable in the first form **(1)** of definition (2.2.3.), then f_r and g_r are differentiable functions and

$$[F'(t)]^r = [f_r'(t), g_r'(t)]$$

(ii) If F is differentiable in the second form **(2)** of definition (2.2.3), then f_r and g_r are differentiable functions and

$$[F'(t)]^r = [g_r'(t), f_r'(t)]$$

Proof: see [25] □

3 Fuzzy Neural Network [24]

A fuzzy neural network (FNN) 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 problem (e. g. fuzzy differential equations, fuzzy integral equations, etc.).

Before 2005 FNN called fuzzy weight neural networks was developed by Pabisek, Jakubek and et al. Membership functions of FWNN were formulated by the multi – layered perceptron (MLP) network training, separately for each learning pattern and then the interval arithmetic was applied to process crisp or fuzzy data.

3.1 Input – Output Relations of Each Unit [21,22]

Let us consider a three-layer fuzzy FFNN 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 numbers. For convenience in this discussion, FNN 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 FNN 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.

The architecture of the model shows how the FNN 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 $\left([z_1]_r, [z_2]_r, \dots [z_j]_r, \dots [z_m]_r\right)$, 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 FNN, its input – output relations can be written as follows, where $H : R^n \longrightarrow E^s$:

Input units: $o_i = x_i$, i = 1,2,3, …n (16)

Hidden units: $[z_j]_r = H\left([net_j]_r\right)$, j = 1,2,3, …,m, (17)

where

$[net_j]_r = \sum_{i=1}^{n} o_i\, [w_{ji}]_r + [b_j]_r$ (18)

Output units:

$$[y_k]_r = H([net_k]_r),\ k = 1,2,3,\ldots,s, \tag{19}$$

where

$$[net_k]_r = \sum_{j=1}^{m} [w_{kj}]_r\,[z_j]_r + [c_k]_r \tag{20}$$

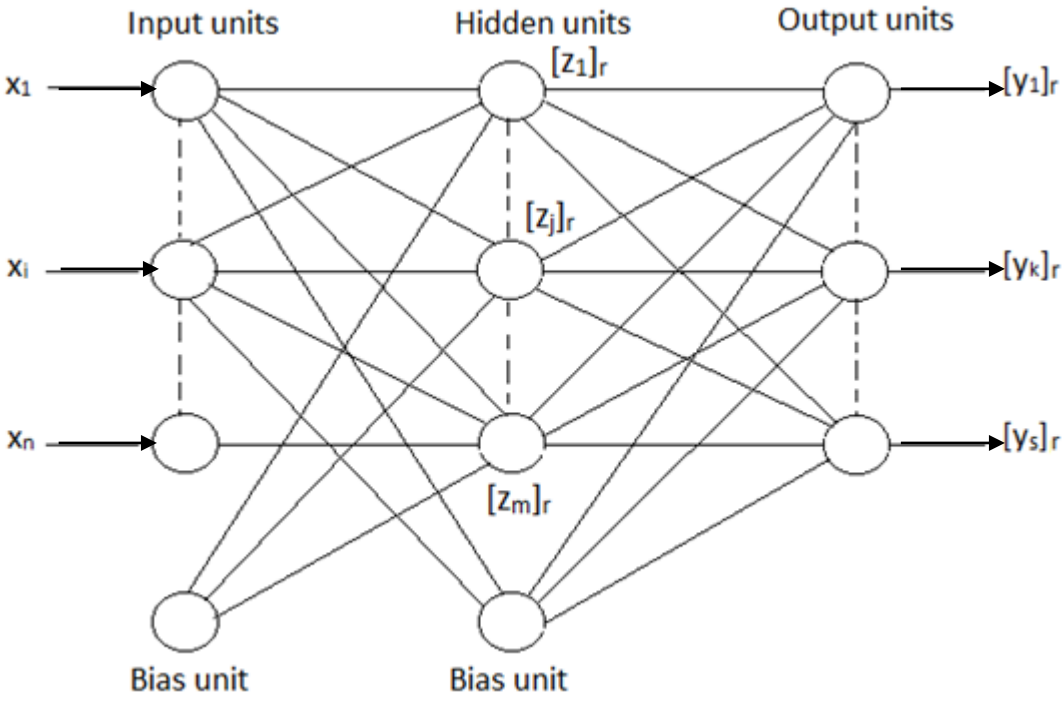

Fig. (1) Three-layer feed forward fuzzy neural network.

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.

From the operations of fuzzy numbers (which we have described in section two), the above relations are rewritten as follows when the inputs x_i´s are non – negative, i.e., $x_i \geq 0$:

Input units: $o_i = x_i$ (21)

Hidden units:

$$[z_j]_r = H\left([net_j]_r\right) = \left[[z_j]_r^L,\ [z_j]_r^U\right] = \left[H\left([net_j]_r^L\right), H\left([net_j]_r^U\right)\right] \quad (22)$$ where

$$[net_j]_r^L = \sum_{i=1}^{n} o_i\,[w_{ji}]_r^L + [b_j]_r^L \quad (23)$$

$$[net_j]_r^U = \sum_{i=1}^{n} o_i\,[w_{ji}]_r^U + [b_j]_r^U \quad (24)$$

Output units:

$$[y_k]_r = H\left([net_k]_r\right) = [[y_k]_r^L,\ [y_k]_r^U] = [H([net_k]_r^L), H([net_k]_r^U)] \quad (25)$$

where

$$[net_k]_r^L = \sum_{j\in a}[w_{kj}]_r^L\ [z_j]_r^L + \sum_{j\in b}[w_{kj}]_r^L\ [z_j]_r^U + [c_k]_r^L \quad (26)$$

$$[net_k]_r^U = \sum_{j\in c}[w_{kj}]_r^U\ [z_j]_r^U + \sum_{j\in d}[w_{kj}]_r^U\ [z_j]_r^L + [c_k]_r^U \quad (27)$$

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

where

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

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

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

4 Technique of The Proposed Method

4.1 First Order Fuzzy Differential Equation

To solve any fuzzy ordinary differential equation we consider a three – layered fuzzy FFNN 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 FNN is the hyperbolic tangent function. Here, the dimension of FNNM is $(1 \times m \times 1)$.

For every entry x (where $x \geq 0$) equations (21-27) will be :

Input unit: o = x, (28)

Hidden units :

$$[z_j]_r = \left[[z_j]_r^L,\ [z_j]_r^U\right] = \left[s\left([net_j]_r^L\right), s\left([net_j]_r^U\right)\right] \quad (29)$$

where

$$[net_j]_r^L = o\,[w_j]_r^L + [b_j]_r^L \quad (30)$$

$$[net_j]_r^U = o\,[w_j]_r^U + [b_j]_r^U \quad (31)$$

Output unit:

$$[N]_r = [[N]_r^L,\ [N]_r^U] \quad (32)$$

where

$$[N]_r^L = \sum_{j\in a}[v_j]_r^L\,[z_j]_r^L + \sum_{j\in b}[v_j]_r^L\,[z_j]_r^U \quad (33)$$

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

For illustration the solution steps of our proposed method, we will consider the first order fuzzy differential equation :

$$\frac{dy(x)}{dx} = F(x,y) \quad ,\ x \in [a,b]\ ,\ y\ (a) = A \quad (35)$$

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:

$$[y_t(x,p)]_r^L = [A]_r^L + (x-a)[N(x,p)]_r^U$$

$$[y_t(x,p)]_r^U = [A]_r^U + (x-a)[N(x,p)]_r^L \quad (36)$$

This fuzzy solution by intention satisfies the fuzzy initial condition in (35).

In this work we use the error function : $E = \sum_{i=1}^{g}\left(E_{ir}^{L} + E_{ir}^{U}\right)$, where E_{ir}^{L} and E_{ir}^{U} can be viewed as the squared errors for the lower limits and the upper limits of the r-level sets, respectively.

Therefore, The error function that must be minimized for the problem (35) is in the form [23] :

$$E= \sum_{i=1}^{g}\left(E_{ir}^{L} + E_{ir}^{U}\right) \tag{37}$$

where

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

$$E_{ir}^{U} = \left[\left[\frac{d\, y_t\,(x_i,p)}{dx}\right]_r^{U} - \left[F\left(x_i, y_t\,(x_i,p)\right)\right]_r^{L} \right]^2 \tag{38}$$

where $\{x_i\}_{i=1}^{g}$ are discrete points belonging to the interval $[a, b]$ (training set) and in the cost function (37), 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.

It is easy to express the first derivative of $[N(x,p)]_r^U$ and $[N(x,p)]_r^L$ in terms of the derivative of the hyperbolic tangent activation function, i.e.,

$$\frac{\partial\,[N]_r^L}{\partial x} = \sum_a \left[v_j\right]_r^L \frac{\partial\,[z_j]_r^L}{\partial\,[net_j]_r^L}\ \frac{\partial\,[net_j]_r^L}{\partial x} + \sum_b \left[v_j\right]_r^L \frac{\partial\,[z_j]_r^U}{\partial\,[net_j]_r^U}\ \frac{\partial\,[net_j]_r^U}{\partial x} \tag{39}$$

$$\frac{\partial\,[N]_r^U}{\partial x} = \sum_c \left[v_j\right]_r^U \frac{\partial\,[z_j]_r^U}{\partial\,[net_j]_r^U}\ \frac{\partial\,[net_j]_r^U}{\partial x} + \sum_d \left[v_j\right]_r^U \frac{\partial\,[z_j]_r^L}{\partial\,[net_j]_r^L}\ \frac{\partial\,[net_j]_r^L}{\partial x} \tag{40}$$

From (29-31) we can get

$$\frac{\partial [net_j]_r^L}{\partial x} = \left[w_j\right]_r^L \tag{41}$$

$$\frac{\partial [z_j]_r^L}{\partial\,[net_j]_r^L} = 1 - \left(\left[z_j\right]_r^L\right)^2 \tag{42}$$

$$\frac{\partial [net_j]_r^U}{\partial x} = \left[w_j\right]_r^U \tag{43}$$

$$\frac{\partial \left[z_j\right]_r^U}{\partial \left[net_j\right]_r^U} = 1 - \left(\left[z_j\right]_r^U\right)^2 \quad (44)$$

From (36) we can get

$$\frac{\partial [y_t\,(x,p)]_r^L}{\partial x} = [N\,(x,p)]_r^U + (x-a)\,\frac{\partial\,[N\,(x,p)]_r^U}{\partial x} \quad (45)$$

$$\frac{\partial [y_t\,(x,p)]_r^U}{\partial x} = [N\,(x,p)]_r^L + (x-a)\,\frac{\partial\,[N\,(x,p)]_r^L}{\partial x} \quad (46)$$

Then we have

$$E_{ir}^U = [\,\sum_a \left[v_j\right]_r^L \left[z_j\right]_r^L + \sum_b \left[v_j\right]_r^L \left[z_j\right]_r^U + (x_i - a)\,(\sum_a \left[v_j\right]_r^L \left[w_j\right]_r^L (1 - \left(\left[z_j\right]_r^L\right)^2) + \sum_b \left[v_j\right]_r^L \left[w_j\right]_r^U (1 - \left(\left[z_j\right]_r^U\right)^2)) - F(x_i, [A]_r^U + (x_i - a)\,(\sum_a \left[v_j\right]_r^L \left[z_j\right]_r^L + \sum_b \left[v_j\right]_r^L \left[z_j\right]_r^U))\,]^2 \quad (47)$$

$$E_{ir}^L = [\,\sum_c \left[v_j\right]_r^U \left[z_j\right]_r^U + \sum_d \left[v_j\right]_r^U \left[z_j\right]_r^L + (x_i - a)\,(\sum_c \left[v_j\right]_r^U \left[w_j\right]_r^U (1 - \left(\left[z_j\right]_r^U\right)^2) + \sum_d \left[v_j\right]_r^U \left[w_j\right]_r^L (1 - \left(\left[z_j\right]_r^L\right)^2)) - F(x_i, [A]_r^L + (x_i - a)\,(\sum_c \left[v_j\right]_r^U \left[z_j\right]_r^U + \sum_d \left[v_j\right]_r^U \left[z_j\right]_r^L))\,]^2 \quad (48)$$

Then we substitute (47) and (48) in (37) to find the error function that must be minimized for problem (35).

Note : For reducing the complexity of the learning algorithm in (37), we have used partially fuzzy neural network (PFNN) architecture where connection weights to the output unit are fuzzy numbers while connection weights and biases to the hidden units are real numbers .

The input – output relation of each unit of the PFNN can be rewritten for r – level sets as follows:

Input unit: $o = x$ (49)

Hidden units: $z_j = s\left(net_j\right)$, $j = 1,2,3, \ldots m$ (50)

where $net_j = o\, w_j + b_j$ (51)

Output unit:

$$[N]_r=[[N]_r^L,\ [N]_r^U]=\left[\sum_{j=1}^{m}\left[v_j\right]_r^L z_j\ ,\sum_{j=1}^{m}\left[v_j\right]_r^U z_j\ \right] \quad (52)$$

Now, to find the minimized error function for problem (35) :

$$\frac{\partial\,[N]_r^L}{\partial x}=\sum_{j=1}^{m}\left[v_j\right]_r^L\ \frac{\partial\,z_j}{\partial\,net_j}\ \frac{\partial\,net_j}{\partial x}=\sum_{j=1}^{m}\left[v_j\right]_r^L\ w_j\left(1-\ {z_j}^2\right) \quad (53)$$

$$\frac{\partial\,[N]_r^U}{\partial x}=\sum_{j=1}^{m}\left[v_j\right]_r^U\ \frac{\partial\,z_j}{\partial\,net_j}\ \frac{\partial\,net_j}{\partial x}=\sum_{j=1}^{m}\left[v_j\right]_r^U\ w_j\left(1-\ {z_j}^2\right) \quad (54)$$

Then we obtain :

$$E_{ir}^U=[\sum_{j=1}^{m}z_j\left[v_j\right]_r^L+(x_i-a)\ \sum_{j=1}^{m}\ w_j\,(1-\ {z_j}^2)\left[v_j\right]_r^L-F(x_i,[A]_r^U+(x_i-a)\ \sum_{j=1}^{m}z_j\ \left[v_j\right]_r^L)\ ]^2 \quad (55)$$

$$E_{ir}^L=[\sum_{j=1}^{m}z_j\left[v_j\right]_r^U+(x_i-a)\ \sum_{j=1}^{m}\ w_j\,(1-\ {z_j}^2)\left[v_j\right]_r^U-F(x_i,[A]_r^L+(x_i-a)\ \sum_{j=1}^{m}z_j\ \left[v_j\right]_r^U)\ ]^2 \quad (56)$$

Then we substitute (55) and (56) in (37) to find the error function that must be minimized for problem (35) under PFNN.

4.2 Reducing a FDE to a System of ODEs [24,25]

The solution of the fuzzy differential equation (35) is depend on the choice of the derivative (in the first form or in the second form of definition (10)).

Let us explain the proposed method, if we denote

$$[y(x)]^r=[y_1^r(x),y_2^r(x)],\qquad [y_0]^r=[y_{01}^r\,,\,y_{02}^r]$$

and

$$[F(x,y(x))]^r=\left[\ F_1^r\left(x,y_1^r(x),y_2^r(x)\right),F_2^r\left(x,y_1^r(x),y_2^r(x)\right)\ \right] \quad (57)$$

we have the following results:

Case I. If we consider $y'(x)$ in the first form **(1)** of definition (2.2.3), then we have to solve the following system of ODEs

$$\frac{d}{dx}\left(y_1^r(x)\right) = F_1^r\left(x, y_1^r(x), y_2^r(x)\right) \qquad y_1^r(a) = y_{01}^r$$

$$\frac{d}{dx}\left(y_2^r(x)\right) = F_2^r\left(x, y_1^r(x), y_2^r(x)\right) \qquad y_2^r(a) = y_{02}^r$$

Case II. If we consider $y'(t)$ in the second form **(2)** of definition (2.2.3) then we have to solve the following system of ODEs

$$\frac{d}{dx}\left(y_1^r(x)\right) = F_2^r\left(x, y_1^r(x), y_2^r(x)\right) \qquad y_1^r(a) = y_{01}^r$$

$$\frac{d}{dx}\left(y_2^r(x)\right) = F_1^r\left(x, y_1^r(x), y_2^r(x)\right) \qquad y_2^r(a) = y_{02}^r$$

The existence and uniqueness of the two solutions (for problem (35)) which described above are given by the following theorem

Theorem (2) : Let $F : I \times E^1 \to E^1$ be a continuous fuzzy function such that there exists $k > 0$ such that

$D\left(F(x,w), F(x,z)\right) \leq k\, D(w,z)$ for all $t \in I$ and $w, z \in E^1$ then the problem (35) has two solutions (one (1)-differentiable and the other one (2)-differentiable) on I, where $I = [a,b]$.

Proof: see [26] □

To illustrate how we can find the two solutions for a fuzzy differential equation under generalized H-derivation, we present the following problems:

Problem (1): Consider the fuzzy initial value problem

$$y' = -y(x) \ , \quad y(0) = [\,0.96 + 0.04r, 1.01 - 0.01r\,]$$

(1) According to subsection (4.2), **Case I.**, after reducing the above problem , we have the following system of ODEs

$$\frac{d}{dx}\left(y_1^r(x)\right) = -y_1^r(x), \qquad y_1^r(0) = 0.96 + 0.04r$$

$$\frac{d}{dx}\left(y_2^r(x)\right) = -y_2^r(x), \qquad y_2^r(0) = 1.01 - 0.01r$$

Which gives the following fuzzy analytical solution

$y(x,r) = [\ (0.96 + 0.04r)e^{-x}, (1.01 - 0.01r)e^{-x}\ ]$

(2) According to subsection (4.2), **Case II.**, after reducing the above problem , we have the following system of ODEs

$$\frac{d}{dx}\left(y_1^r(x)\right) = -y_2^r(x), \qquad y_1^r(0) = 0.96 + 0.04r$$

$$\frac{d}{dx}\left(y_2^r(x)\right) = -y_1^r(x), \qquad y_2^r(0) = 1.01 - 0.01r$$

Which gives the following fuzzy analytical solution

$y(x,r) = [\ (0.985 + 0.015r)e^{-x} - (1-r)0.025e^{x}, (0.985 + 0.015r)e^{-x} + (1-r)0.025e^{x}\ ]$.

Problem (2): Consider the fuzzy initial value problem

$$y' = -3y(x) + e^{2x}, \quad y(0) = [\ 0.75 + 0.25r, 1.25 - 0.25r\ ]$$

(1) According to subsection (4.2), **Case I.**, after reducing the above problem , we have the following system of ODEs

$$\frac{d}{dx}\left(y_1^r(x)\right) = -3y_1^r(x) + e^{2x}, \qquad y_1^r(0) = 0.75 + 0.25r$$

$$\frac{d}{dx}\left(y_2^r(x)\right) = -3y_2^r(x) + e^{2x}, \qquad y_2^r(0) = 1.25 - 0.25r$$

Which gives the following fuzzy analytical solution

$y(x,r) = [\ (0.55 + 0.25r)e^{-3x} + 0.2e^{2x}, (1.05 - 0.25r)e^{-3x} + 0.2e^{2x}\ ]$

(2) According to subsection (4.2), **Case II.**, after reducing the above problem , we have the following system of ODEs

$$\frac{d}{dx}\left(y_1^r(x)\right) = -3y_2^r(x) + e^{2x}, \qquad y_1^r(0) = 0.75 + 0.25r$$

$$\frac{d}{dx}\left(y_2^r(x)\right) = -3y_1^r(x) + e^{2x}, \qquad y_2^r(0) = 1.25 - 0.25r$$

Which gives the following fuzzy analytical solution

$y(x,r) = [\,(0.75+0.25r)\cosh 3x-(1.25-0.25r)\sinh 3x+0.2e^{2x}-0.2e^{-3x},\ (1.25-0.25r)\cosh 3x-(0.75+0.25r)\sinh 3x+0.2e^{2x}-0.2e^{-3x}\,]$.

5 Numerical Examples

To show the behavior and properties of the proposed method, two problem will be solved in this section. We have used a multilayer perceptron having one hidden layer with ten hidden units and one output unit. The activation function of each hidden unit is hyperbolic tangent activation function. The analytical solution $[y_a(x)]_r^L$ and $[y_a(x)]_r^U$ has been known in advance. Therefore, we test the accuracy of the obtained solutions by computing the deviation (absolute error):

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

In order to obtain better results, more hidden units or training points may be used. To minimize the error function we have used BFGS quasi-Newton method (For more details, see [24]). The computer programs which we have used in this work are coded in MATLAB 2015.

Example 1: Consider the following linear fuzzy initial value problem:

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

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

The analytical solution (According to subsection (4.2), Case II.) are :

$[y_a(x)]_r^L = x + (0.985+0.015r)e^{-x} - (1-r)0.025e^{x}$

$[y_a(x)]_r^U = x + (0.985+0.015r)e^{-x} + (1-r)0.025e^{x}$

The trial solution (According to the proposed method in this work) are:

$[y_t(x)]_r^L = (0.96 + 0.04r) + x\,[N(x,p)]_r^U$

$[y_t(x)]_r^U = (1.01 - 0.01r) + x\,[N(x,p)]_r^L$

The ANN trained using a grid of ten equidistant points in [0, 1].

The error function that must be minimized for this problem will be:

$$E = \sum_{i=1}^{11}([x_i \sum_{j=1}^{10}[v_j]_r^L w_j s'(x_i w_j + b_j) + (1 + x_i) \sum_{j=1}^{10} [v_j]_r^L s(x_i w_j + b_j) - x_i - 0.01r + 0.01]^2 + [x_i \sum_{j=1}^{10}[v_j]_r^U w_j s'(x_i w_j + b_j) + (1 + x_i) \sum_{j=1}^{10}[v_j]_r^U s(x_i w_j + b_j) - x_i + 0.04 r - 0.04]^2) \quad (58)$$

Where s′ is first derivative of hyperbolic tangent activation function.

Then we use (58) to update the weights and biases.

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

Example 2: Consider the following non-linear fuzzy initial value problem:

$y' = \cos(x^2 + y^2)$, with $x \in [0, 1]$

$y(0) = [0.75 + 0.25r, 1.25 - 0.25r]$, where $r \in [0, 1]$.

The trial solutions (According to the proposed method in this work) for this problem are :

$$[y_t(x)]_r^L = (0.75 + 0.25r) + x\,[N(x, p)]_r^U$$

$$[y_t(x)]_r^U = (1.25 - 0.25r) + x\,[N(x, p)]_r^L$$

The ANN trained using a grid of ten equidistant points in [0, 1].

The error function that must be minimized for this problem will be $E = \sum_{i=1}^{11}([x_i \sum_{j=1}^{10}[v_j]_r^L w_j s'(x_i w_j + b_j) + \sum_{j=1}^{10} [v_j]_r^L s(x_i w_j + b_j) - \cos(x_i^2 + (1.25 - 0.25r + x_i \sum_{j=1}^{10} [v_j]_r^L s(x_i w_j + b_j))^2)]^2 + [$

$$x_i \sum_{j=1}^{10}[v_j]_r^U w_j s'(x_i w_j + b_j) + \sum_{j=1}^{10}[v_j]_r^U s(x_i w_j + b_j) - \cos(x_i^2 + (0.75 + 0.25r + x_i \sum_{j=1}^{10}[v_j]_r^U s(x_i w_j + b_j))^2)]^2) \quad (59)$$

Then we use (59) to update the weights and biases.

Trial (numerical) solutions for this problem can be found in table (3).

r	$[y_t(x)]_r^L$	$\underline{e}(x,r)$	$[y_t(x)]_r^U$	$\overline{e}(x,r)$
0	1.948579274	0.000000422	2.318032417	0.000000761
0.1	1.967254967	0.000000472	2.299762662	0.000000643
0.2	1.985930525	0.000000387	2.281492466	0.000000085
0.3	2.004605986	0.000000205	2.263223455	0.000000711
0.4	2.023282053	0.000000629	2.244953513	0.000000406
0.5	2.041957703	0.000000635	2.226683566	0.000000096
0.6	2.060632805	0.000000094	2.208413946	0.000000114
0.7	2.079308402	0.000000048	2.190144557	0.000000362
0.8	2.097984071	0.000000074	2.171875271	0.000000713
0.9	2.116660365	0.000000725	2.153604974	0.000000054
1	2.135335791	0.000000508	2.135336111	0.000000828

Table (1): Trial solutions for example (1), x = 2.

x	$[y_t(x)]_r^L$	$\underline{e}(x,r)$	$[y_t(x)]_r^U$	$\overline{e}(x,r)$
0	0.98	0	1.005	0
0.1	0.984236574	0.000000074	1.011865864	0.000000090
0.2	0.997322850	0.000000112	1.027858634	0.000000827
0.3	1.018388968	0.000000119	1.052136129	0.000000810
0.4	1.046645251	0.000000414	1.083941124	0.000000670
0.5	1.081372866	0.000000202	1.122591654	0.000000958
0.6	1.121919133	0.000000069	1.167472118	0.000000084
0.7	1.167689342	0.000000337	1.218032891	0.000000068
0.8	1.218140638	0.000000903	1.273778484	0.000000226
0.9	1.272775429	0.000000081	1.334265671	0.000000245
1	1.331142019	0.000000196	1.399099381	0.000000513

Table (2): Trial solutions for example (1), r = 0.5.

x	$[y_t(x)]_r^L$	$[y_t(x)]_r^U$
0	0.8125	1.1875
0.1	0.814603338	1.216504331
0.2	0.814821036	1.244468346
0.3	0.812924175	1.270982832
0.4	0.808261184	1.295659825
0.5	0.799681021	1.318165427
0.6	0.785827891	1.338238511
0.7	0.765526122	1.355644313
0.8	0.737875907	1.370043258
0.9	0.702086306	1.380817214
1	0.657325883	1.386926291

Table (3): Trial solutions for example (2), r = 0.25.

6 Conclusion

In this paper, we have presented numerical method based on fuzzy neural network for solving first order fuzzy initial value problem under generalized H-derivation. we have demonstrated the ability of the fuzzy neural network to approximate the solution of the fuzzy differential equations. Therefore, we can conclude that the method which we proposed can handle effectively all types of the fuzzy differential equations and provide accurate approximate solution throughout the whole domain and not only at the training set. As well, one can use the interpolation techniques to find the approximate solution at points between the training points or at points outside the training set. Further research is in progress to apply and extend this method to solve higher order fuzzy differential equations and partial fuzzy differential equations.

References

[1] Abbasbandy S., Allahviranloo T., et al.,"Numerical Solution of N-Order Fuzzy Differential Equations by Runge - Kutta Method ", Mathmatical and Computational Application, 16(4),935-946, 2011.

[2] Buckley J. J., Feuring T.,"Fuzzy Differential Equations",Fuzzy Sets and Systems, 110, 69-77, 2000.

[3] Lee H., Kang I. S., "Neural Algorithms For Solving Differential Equations", Journal of Computational Physics, 91,110-131,1990 .

[4] Meade A. J., Fernandes A. A.,"The Numerical Solution of Linear Ordinary Differential Equations by Feed-Forward Neural Networks", Mathematical and Computer Modelling, Vol. 19, No. 12, 1-25, 1994.

[5] Meade A. J., Fernandes A. A.," Solution of Nonlinear Ordinary Differential Equations by Feed-Forward Neural Networks", Mathematical and Computer Modelling, Vol. 20, No. 9, 19-44, 1994.

[6] Lagaris I. E., Likas A., et al.,"Artificial Neural Networks For Solving Ordinary and Partial Differential Equations", Journal of Computational Physics, 104, 1-26, 1997.

[7] Lagaris I. E., Likas A., et al.,"Artificial Neural Networks For Solving Ordinary and Partial Differential Equations", IEEE Transaction on Neural Networks, Vol. 9, No. 5, 987-1000, 1998.

[8] Liu B., Jammes B.,"Solving Ordinary Differential Equations by Neural Networks", Warsaw, Poland, 1999.

[9] Alli H., Ucar A., et al., "The Solutions of Vibration Control Problems Using Artificial Neural Networks",Journal of the Franklin Institute, 340, 307-325, 2003.

[10] Tawfiq L. N. M.,"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.

[11] Malek A., Shekari R.,"Numerical Solution For High Order Differential Equations by Using a Hybrid Neural Network Optimization Method ",Applied Mathematics and Computation, 183, 260-271, 2006.

[12] Pattanaik S., Mishra R. K.,"Application of ANN For Solution of PDE in RF Engineering",International Journal on Information Sciences and Computing, Vol. 2, No. 1, 74-79, 2008.

[13] Baymani M., Kerayechian A., et al.,"Artificial Neural Networks Approach For Solving Stokes Problem", Applied Mathematics, 1, 288-292,2010.

[14] Oraibi Y. A.,"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.

[15] Ali M. H.,"Design Fast Feed-Forward Neural Networks to Solve Two Point Boundary Value Problems", M.Sc. Thesis, College of Education Ibn Al-Haitham,University of Baghdad, Iraq, 2012.

[16] Hussein A. A. T., "Design Fast Feed-Forward Neural Networks to Solve Singular Boundary Value Problems", M.Sc. Thesis, College of Education Ibn Al-Haitham,University of Baghdad, Iraq, 2013.

[17] Tawfiq L. N. M., Al-Abrahemee K. M. M.,"Design Neural Network to Solve Singular Perturbation Problems", Applied and Computational Mathematics, Vol. 3, No. 3, 1-5, 2014 .

[18] Effati S., Pakdaman M.,"Artificial Neural Network Approach For Solving Fuzzy Differential Equations",Information Sciences, 180, 1434-1457, 2010.

[19] Mosleh M., Otadi M.,"Fuzzy Fredholm Integro-Differential Equations with Artificial Neural Networks",Communications in Numerical Analysis, Article ID cna-00128, 1-13, 2012.

[20] 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, 3(1s), 477-482, 2013.

[21] Mosleh M., Otadi M,"Simulation and Evaluation of Fuzzy Differential Equations by Fuzzy Neural Network ", Applied Soft Computing, 12, 2817-2827, 2012.

[22] Mosleh M.,"Fuzzy Neural Network For Solving a System of Fuzzy Differential Equations ", Applied Soft Computing, 13, 3597-3607, 2013.

[23] Mosleh M., Otadi M.,"Solving the Second Order Fuzzy Differential Equations by Fuzzy Neural Network, Journal of Mathematical Extension, Vol. 8, No. 1, 11-27, 2014.

[24] Suhhiem M. H.,"Fuzzy Artificial Neural Network For Solving Fuzzy and Non-Fuzzy Differential Equations ",Ph.D. Thesis, College of Sciences, AL-Mustansiriyah University, Iraq, 2016.

[25] Cano Y. C., Flores H. R.,"On New Solutions of Fuzzy Differential Equations ", Chaos, Solitons and Fractals, 38, 112-119, 2008.

[26] Cano Y. C., Flores H. R.,et al., "Fuzzy Differential Equations with Generalized Derivative",Fuzzy Sets and Systems, 160, 1517-1527,2008.