Fully Fuzzy Neural Network For Solving Fuzzy Differential Equations

الشبكة العصبية الضبابية الكلية لحل المعادلات التفاضلية الضبابية

Mazin Hashim Suhhiem[1] ,Basim Nasih Abood[2]

1) Department of Statistics ,University of Sumer , Alrifaee , Iraq .
2) Department of Mathematics ,University of Wasit , Alkut , Iraq .
Mazin.suhhiem@yahoo.com ; Basim.nasih@yahoo.com

المستخلص

في هذا العمل قدمنا طريقة جديدة لحل المعادلات التفاضلية الضبابية.هذه الطريقة تستند على الشبكة العصبية الضبابية الكلية لإيجاد الحل العددي للمعادلات التفاضلية الضبابية من الرتبة الأولى.الحل التجريبي الضبابي لمسائل القيم الابتدائية الضبابية تمت كتابته على شكل مجموع حدين . الحد الأول يحقق الشروط الضبابية ولا يحتوي على معلمات ضبابية قابلة للتحديث . الحد الثاني يتضمن الشبكة العصبية الضبابية الكلية الأمامية والتي تحتوي على معلمات ضبابية قابلة للتحديث . بموجب بعض الشروط فأن الطريقة المقترحة في هذا العمل تعطي حلول عددية ذات دقة عالية

Abstract

In this work we have introduced a new method for solving fuzzy differential equations .This method based on the fully fuzzy neural network to find the numerical solution of the first order fuzzy differential equations . 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 fully fuzzy feed-forward neural networks containing fuzzy adjustable parameters. Under some conditions the proposed method provides numerical solutions with high accuracy .

Keywords : Fuzzy Differential Equation ,Fully Fuzzy Neural Network , Fuzzy Trial Solution, Minimized Error Function, Hyperbolic Tangent Activation Function .

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 [1].

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 [2]; Meade , Fernandez in [3,4] ;Lagaris , Likas , et al. in [5] ; Liu ,Jammes in [6] ; Ali ,Ucar , et al. in [7] ; Tawfiq in [8] ;malek , shekari in [9] ; Pattanaik , Mishra in [10]; Baymani ,Kerayechian , et al. in [11] ; and other researchers .

In 2010 researchers began using ANN for solving fuzzy differential equation such as : Effati and pakdaman in [12] ; Mosleh ,Otadi in [13] ;Ezadi ,Parandin , et al. [14].

In 2012 researchers began using partially (non fully) fuzzy artificial neural network(FANN) for solving fuzzy differential equation such as Mosleh ,Otadi in [15 ,16,17]. In (2016) Suhhiem[18] developed and used partially FANN for solving fuzzy and non-fuzzy differential equations.

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

2. Basic Definitions

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

Definition (1) , [18] : 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),[18]:Extension Principle

Let X be the Cartesian product of universes X_1 , X_2 , … , X_m and $\widetilde{A}_1$,$\widetilde{A}_2$, … , $\widetilde{A}_m$ be m - fuzzy subset in X_1 , X_2 , … , 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 , … , x_m)) . Then , the extension principle allows to define a fuzzy subset $\widetilde{B} = f(\widetilde{A})$ in Y by

$y = f(x_1 , x_2 , \dots , x_m)$, $(x_1 , x_2 , \dots , x_m) \in X\}$, where

$$\widetilde{B} = \{(y , \mu_{\widetilde{B}}(y)) :$$

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

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

For $m = 1$, the extension principle will be :

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

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

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

Definition (3), [18]: Fuzzy Number

A fuzzy number $\tilde{u}$ is completely determined by an ordered pair of functions $(\underline{u}(r) , \overline{u}(r))$, $0 \le r \le 1$, 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) , [12]: For arbitrary

$\tilde{u} = (\underline{u} , \overline{u})$, $\tilde{v} = (\underline{v} , \overline{v})$ 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 $K \ge 0$

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 $\tilde{u} = (\underline{u} , \overline{u})$ and $\tilde{v} = (\underline{v} , \overline{v})$ is given as :

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

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

Remark (4),[18]: In the following we have described the operations of fuzzy

numbers (in parametric form) which can be generalized from the crisp intervals. $\forall a_1, b_1, a_2, b_2 \in R$, $A = [a_1, b_1]$ and $B = [a_2, b_2]$.

Assume A and B are numbers expressed as interval, then we have :

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 :

$$A.B = \left[\min \begin{matrix} \{a_1a_2 , a_1b_2 , b_1a_2 , b_1b_2\} \\ , \max\{a_1a_2 , a_1b_2 , b_1a_2 , b_1b_2\} \end{matrix}\right]$$

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 $b_1 = 0$.

6)If $0 \le a_2 \le b_2$,then the multiplication operation can be described as:

$$A.B = \left[\begin{matrix} \min\{a_1a_2 , a_1b_2\} \\ , \max\{b_1a_2 , b_1b_2\} \end{matrix}\right]$$

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 : $A . B = [a_1, b_1] . [a_2, b_2] = [a_1a_2 , b_1b_2]$

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 (4) , [18] : Fuzzy Function

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

Definition (5) , [12]: 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 $\delta > 0$ such that :

$|t - t_1| < \delta \Rightarrow D(F(t) , F(t_1)) < \epsilon$, where D is the distance between two arbitrary fuzzy numbers.

Definition (6),[14]: Let I be a real interval . The r-level set of the fuzzy function $y : I \to E^1$ can be denoted by:

$$[y(t)]^r = [y_1^r(t), y_2^r(t)] \qquad t \in I , r \in [0,1] \qquad (5)$$

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

$$[y'(t)]^r = [(y_1^r)'(t), (y_2^r)'(t)] \qquad t \in I, r \in [0,1] \qquad (6)$$

Definition (7), [12]: 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 $w = u \ominus v$.

The sign $\ominus$ refers to H-difference, and we must note that $u \ominus v \neq u + (-1) v$.

Definition (8), [15]: Fuzzy Derivative

Let $F : (a,b) \to E^1$ and $t_0 \in (a,b)$.We say that F is H-differential (Hukuhara-differential) at t_0, if there exists an element $F'(t_0) \in E^1$ such that for all $h > 0$ (sufficiently small), $\exists\ F(t_0 + h) \ominus F(t_0)$, $F(t_0) \ominus F(t_0 - h)$ and the limits (in the metric D)

$$\lim_{h\to 0}\frac{F(t_0+h)\ominus F(t_0)}{h} = \lim_{h\to 0}\frac{F(t_0)\ominus F(t_0-h)}{h} = F'(t_0) \qquad (7)$$

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

where D is the distance between two arbitrary fuzzy numbers .

3. Fuzzy Neural Network [8,18]

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.

(i.e.) 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.).

Before 2005 a fuzzy neural network called FWNN (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.

4. Fully Fuzzy Neural Network [15,16,17]

Let us consider a fuzzy 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 numbers .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

For convenience in this discussion ,fully fuzzy neural network 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 fully fuzzy neural network 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 fully fuzzy neural network 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 fully fuzzy neural network , its input – output relations can be written as follows , where $F : R^n \longrightarrow E^s$:

Input units : $x_i = x_i$,

$i = 1 , 2 , 3 , \dots , n,$ (8)

Hidden units : $[z_j]_r = F\left([net_j]_r \right)$,

$j = 1 , 2 , 3 , \dots , m,$ (9)

where

$$[net_j]_r = \sum_{i=1}^{n} x_i [w_{ji}]_r + [b_j]_r \quad (10)$$

Output units :

$$[y_k]_r = F([net_k]_r), k = 1,2,3,\dots,s, \quad (11)$$

where

$$[net_k]_r = \sum_{j=1}^{m} [w_{kj}]_r [z_j]_r + [c_k]_r \quad (12)$$

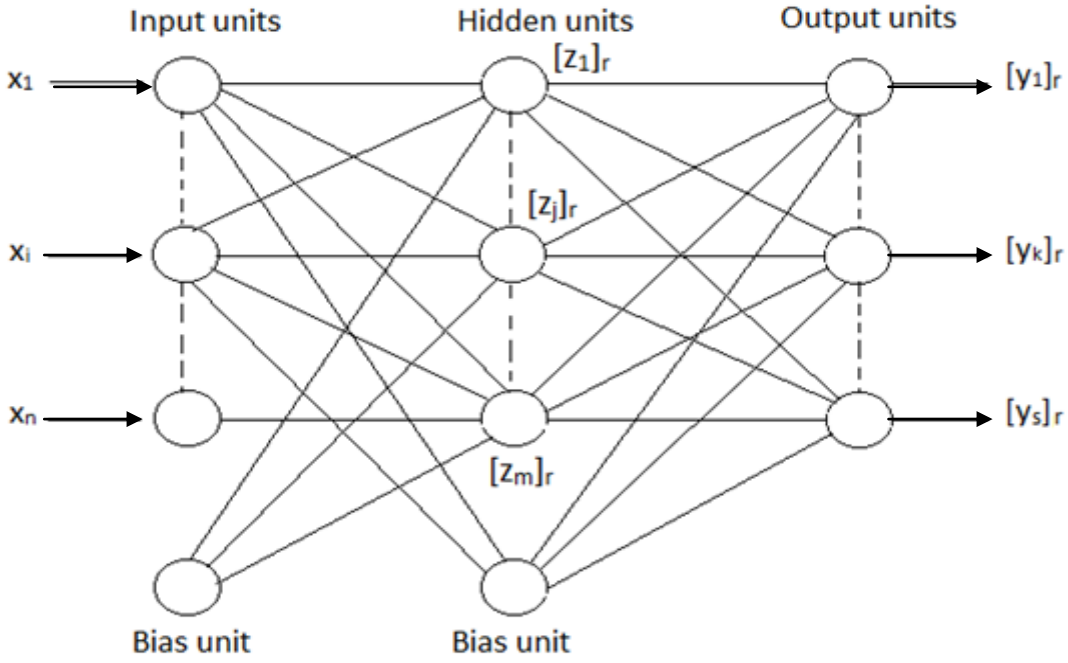

Fig.(1) Fully fuzzy feed forward neural network .

The architecture of our fully 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: $x_i = x_i$ (13)

Hidden units :

$[z_j]_r = F([net_j]_r) = \left[[z_j]_r^L , [z_j]_r^U\right] = \left[F\left([net_j]_r^L\right) , F\left([net_j]_r^U\right)\right]$ (14)

where

$[net_j]_r^L = \sum_{i=1}^{n} x_i [w_{ji}]_r^L + [b_j]_r^L$ (15)

$[net_j]_r^U = \sum_{i=1}^{n} x_i [w_{ji}]_r^U + [b_j]_r^U$ (16)

Output units :

$[y_k]_r = F([net_k]_r) = [[y_k]_r^L , [y_k]_r^U] = [F([net_k]_r^L) , F([net_k]_r^U)]$ (17)

where

$[net_k]_r^L = \sum_{j\in a} [w_{kj}]_r^L [z_j]_r^L + \sum_{j\in b} [w_{kj}]_r^L [z_j]_r^U + [c_k]_r^L$ (18)

$[net_k]_r^U = \sum_{j\in c} [w_{kj}]_r^U [z_j]_r^U + \sum_{j\in d} [w_{kj}]_r^U [z_j]_r^L + [c_k]_r^U$ (19)

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

5. 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 ($s(\propto) = \tanh(\propto)$) . The dimension of our fully fuzzy neural network in this research is (1 : m : 1) (Fig.2) .

For every entry x (where $x \geq 0$) equations (13-19) will be :

Input unit : $x = x$, (20) **Hidden units** :

$[z_j]_r = \left[[z_j]_r^L , [z_j]_r^U\right] = \left[s\left([net_j]_r^L\right) , s\left([net_j]_r^U\right)\right]$ (21)

where

$[net_j]_r^L = x[w_j]_r^L + [b_j]_r^L$ (22)

$[net_j]_r^U = x[w_j]_r^U + [b_j]_r^U$ (23)

Output unit :

$[N]_r = [[N]_r^L , [N]_r^U]$ (24)

where

$[N]_r^L = \sum_{j\in a} [v_j]_r^L [z_j]_r^L + \sum_{j\in b} [v_j]_r^L [z_j]_r^U$ (25)

$[N]_r^U = \sum_{j\in c} [v_j]_r^U [z_j]_r^U + \sum_{j\in d} [v_j]_r^U [z_j]_r^L$ (26)

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

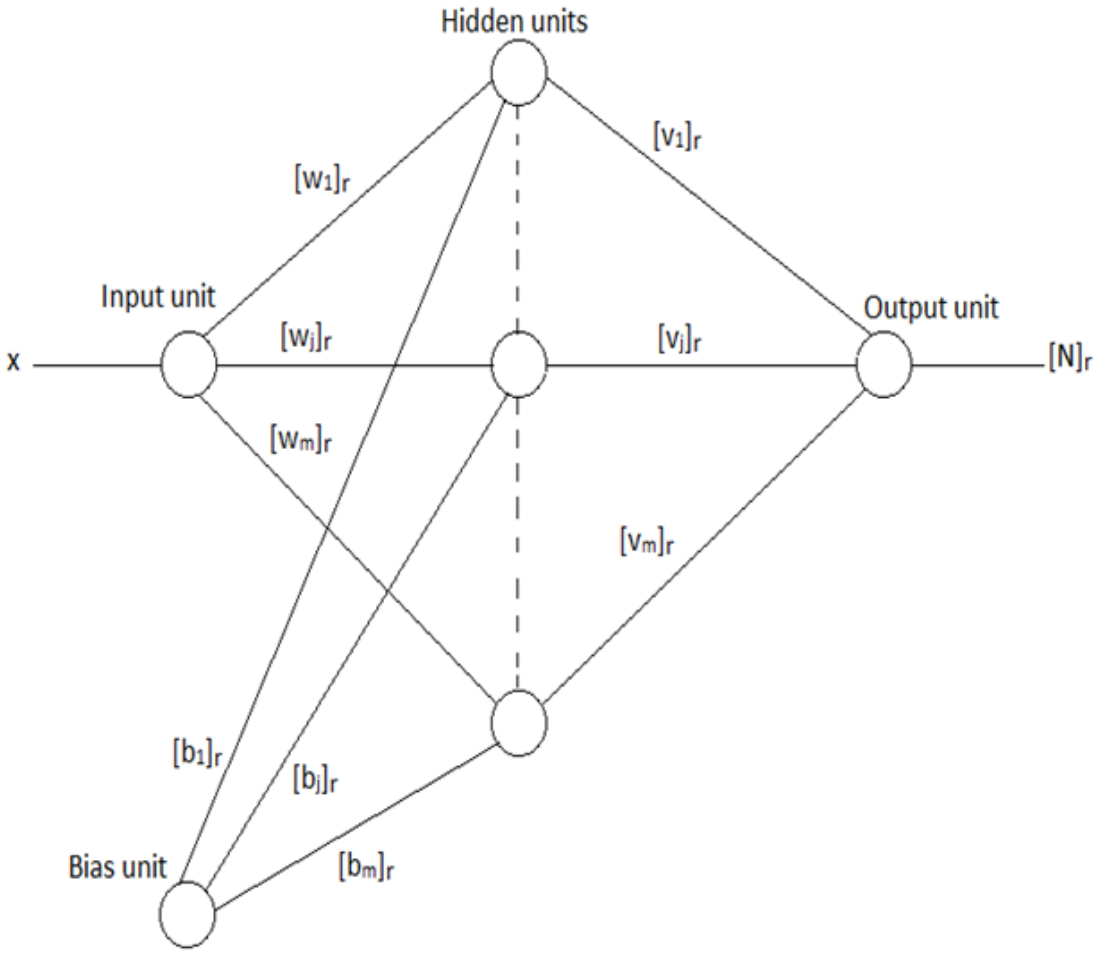

Fig. (2) (1 : m : 1) Fully fuzzy feed forward neural network .

Fully fuzzy neural network with crisp set inputs , fuzzy number weights and fuzzy number output solution to the fuzzy ordinary differential equations is given in Fig. (2) .

6. Description of The Proposed Method

For illustration the proposed method , we will consider the first order fuzzy differential equation :

$$\frac{dy(x)}{dx} = f(x, y), \quad x \in [a_1, a_2] \quad , \; y(a_1) = A \qquad (27)$$

where A is a fuzzy number in E^1 under the r – cut sets :

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

For this problem we can write the fuzzy trial solution as :

$$[y_t(x)]_r = [A]_r + (x - a_1)[N(x)]_r \qquad (28)$$

It is clear that the fuzzy initial condition in (27)is satisfied by the fuzzy trial solution in (28).

The error function that must be minimized for the problem (27) is in the form :

$$E = \sum_{i=1}^{g} \left(E_{ir}^L + E_{ir}^U \right) \qquad (29)$$

where

$$E_{ir}^{L}=\left[\left[\frac{d\,y_t(x_i)}{dx}\right]_r^{L}-\left[f\left(x_i,y_t(x_i)\right)\right]_r^{L}\right]^2 \quad (30)$$

$$E_{ir}^{U}=\left[\left[\frac{d\,y_t(x_i)}{dx}\right]_r^{U}-\left[f\left(x_i,y_t(x_i)\right)\right]_r^{U}\right]^2 \quad (31)$$

where $\{x_i\}_{i=1}^{h}$ are discrete points in the interval $[a_1, a_2]$ (training set) and in the error function (29), E_r^L and E_r^U represent the square errors of the lower and upper limits of the r – level sets, respectively.

We can find the first lower and upper derivative of the $[N(x)]_r$ in terms of the derivative of the activation function, i.e.,

$$\frac{\partial [N]_r^L}{\partial x}=\sum_a [v_j]_r^L \frac{\partial [z_j]_r^L}{\partial [net_j]_r^L}\frac{\partial [net_j]_r^L}{\partial x}+ \sum_b [v_j]_r^L \frac{\partial [z_j]_r^U}{\partial [net_j]_r^U}\frac{\partial [net_j]_r^U}{\partial x} \quad (32)$$

$$\frac{\partial [N]_r^U}{\partial x}=\sum_c [v_j]_r^U \frac{\partial [z_j]_r^U}{\partial [net_j]_r^U}\frac{\partial [net_j]_r^U}{\partial x}+ \sum_d [v_j]_r^U \frac{\partial [z_j]_r^L}{\partial [net_j]_r^L}\frac{\partial [net_j]_r^L}{\partial x} \quad (33)$$

$$\frac{\partial [net_j]_r^L}{\partial x}=[w_j]_r^L \quad (34)$$

$$\frac{\partial [z_j]_r^L}{\partial [net_j]_r^L}=1-\left([z_j]_r^L\right)^2 \quad (35)$$

$$\frac{\partial [net_j]_r^U}{\partial x}=[w_j]_r^U \quad (36)$$

$$\frac{\partial [z_j]_r^U}{\partial [net_j]_r^U}=1-\left([z_j]_r^U\right)^2 \quad (37)$$

Note that if $s(\propto)=\tanh(\propto)$ then $s'(\propto)=1-s^2(\propto)$

From the lower and upper derivative of the fuzzy trial function $[y_t(x)]_r$ in (28), we can obtain :

$$\frac{[y_t(x)]_r^L}{\partial x}=[N(x)]_r^L+(x-a_1)\frac{\partial [N(x)]_r^L}{\partial x} \quad (38)$$

$$\frac{[y_t(x)]_r^U}{\partial x}=[N(x)]_r^U+(x-a_1)\frac{\partial [N(x)]_r^U}{\partial x} \quad (39)$$

Now, by substituting eq. (25), eq. (26), eq. (28) and eqs(32 – 39) in eq. (30) and eq. (31), we obtain :

$$\begin{aligned}E_{ir}^{L}=&\Big[\sum_a [v_j]_r^L[z_j]_r^L+\sum_b [v_j]_r^L[z_j]_r^U+(x_i-a_1)\Big(\sum_a [v_j]_r^L[w_j]_r^L(1-\left([z_j]_r^L\right)^2)+\sum_b [v_j]_r^L[w_j]_r^U(1-\left([z_j]_r^U\right)^2)\Big)\\&-f(x_i,[A]_r^L+(x_i-a_1)\Big(\sum_a [v_j]_r^L[z_j]_r^L+\sum_b [v_j]_r^L[z_j]_r^U\Big)\Big]^2\end{aligned} \quad (40)$$

$$\begin{aligned}E_{ir}^{U}=&\Big[\sum_a [v_j]_r^U[z_j]_r^U+\sum_b [v_j]_r^U[z_j]_r^L+(x_i-a_1)\Big(\sum_a [v_j]_r^U[w_j]_r^U(1-\left([z_j]_r^U\right)^2)+\sum_b [v_j]_r^U[w_j]_r^L(1-\left([z_j]_r^L\right)^2)\Big)\\&-f(x_i,[A]_r^U+(x_i-a_1)\Big(\sum_a [v_j]_r^U[z_j]_r^U+\sum_b [v_j]_r^U[z_j]_r^L\Big)\Big]^2\end{aligned} \quad (41)$$

Then we substitute (40) and (41) in (29) to find the error function that must be minimized for problem (27).

7. Numerical Examples

To show the behavior and properties of proposed method, two problem will be solved in this section. We have used a multi layer perceptron having one hidden layer with ten hidden units and one output unit. The activation function of each hidden

unit is the hyperbolic tangent 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):

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

To minimize the error function we have used BFGS quasi-Newton method (For more details, see [18]) . The computer programs used in this work are coded in MATLAB 2015 .

Example1: Consider the following 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 solutions for this problem are:

$[y_a(x)]_r^L = x + (0.96 + 0.04r)e^{-x}$

$[y_a(x)]_r^U = x + (1.01 - 0.01r)e^{-x}$

The trial solutions for this problem are:

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

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

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 :

$E=\sum_{i=1}^{11}(E_{ir}^L + E_{ir}^U)$ (42)

where

$$E_{ir}^{L}=[x_i(\sum_{j\in a}[v_j]_r^L[w_j]_r^L s'\left(x_i[w_j]_r^L+[b_j]_r^L\right)+\sum_{j\in b}[v_j]_r^L[w_j]_r^U s'\left(x_i[w_j]_r^U+[b_j]_r^U\right))+(1+x_i)(\sum_{j\in a}[v_j]_r^L\, s\left(x_i[w_j]_r^L+[b_j]_r^L\right)+\sum_{j\in b}[v_j]_r^L\, s\left(x_i[w_j]_r^U+[b_j]_r^U\right))-x_i+0.04r-0.04\,]^2$$

$$E_{ir}^{U}=[x_i(\sum_{j\in c}[v_j]_r^U[w_j]_r^U s'\left(x_i[w_j]_r^U+[b_j]_r^U\right)+\sum_{j\in d}[v_j]_r^U[w_j]_r^L s'\left(x_i[w_j]_r^L+[b_j]_r^L\right))+(1+x_i)(\sum_{j\in c}[v_j]_r^U\, s\left(x_i[w_j]_r^U+[b_j]_r^U\right)+\sum_{j\in d}[v_j]_r^U\, s\left(x_i[w_j]_r^L+[b_j]_r^L\right))-x_i-0.01r+0.01\,]^2$$

Then we use (42) to update the weights and biases .

in table (1) we can found the numerical solution for this problem

Example 2 : Consider the following fuzzy initial value problem :

$y' = e^{x^2}$, with $x \in [0 , 1]$

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

For this problem the trial solutions are :

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

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

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

$$E=\sum_{i=1}^{11}\left(E_{ir}^{L}+E_{ir}^{U}\right) \qquad (43)$$

where

$$E_{ir}^{L}=[x_i(\sum_{j\in a}[v_j]_r^L[w_j]_r^L s'\left(x_i[w_j]_r^L+[b_j]_r^L\right)+\sum_{j\in b}[v_j]_r^L[w_j]_r^U s'\left(x_i[w_j]_r^U+[b_j]_r^U\right))+(\sum_{j\in a}[v_j]_r^L\, s\left(x_i[w_j]_r^L+[b_j]_r^L\right)+\sum_{j\in b}[v_j]_r^L\, s\left(x_i[w_j]_r^U+[b_j]_r^U\right))-e^{x_i^2}\,]^2$$

$$E_{ir}^{U}=[x_i(\sum_{j\in c}[v_j]_r^U[w_j]_r^U s'\left(x_i[w_j]_r^U+[b_j]_r^U\right)+\sum_{j\in d}[v_j]_r^U[w_j]_r^L s'\left(x_i[w_j]_r^L+[b_j]_r^L\right))+(\sum_{j\in c}[v_j]_r^U\, s\left(x_i[w_j]_r^U+[b_j]_r^U\right)+\sum_{j\in d}[v_j]_r^U\, s\left(x_i[w_j]_r^L+[b_j]_r^L\right))-e^{x_i^2}\,]^2$$

Then we use (43) to update the weights and biases .

in table (2) we can found the numerical solution for this problem .

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

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	0.986741141	4.72 e-7	1.009362248	6.43 e-7
0.2	1.002356525	3.87 e-7	1.022824322	0.85 e-7
0.3	1.026002061	2.05 e-7	1.044523023	7.11 e-7
0.4	1.056913016	6.29 e-7	1.073671240	4.06 e-7
0.5	1.094399412	6.35 e-7	1.109563409	0.96 e-7
0.6	1.137835497	0.94 e-7	1.151555790	1.14 e-7
0.7	1.186653646	0.48 e-7	1.199068592	3.62 e-7
0.8	1.240342311	0.74 e-7	1.251574896	7.13 e-7
0.9	1.298438992	7.25 e-7	1.308603048	0.54 e-7
1	1.360521344	5.08 e-7	1.369719666	8.28 e-7

Table (2):Numerical solution for example (2) , r =0.5 .

x	$[y_t(x)]_r^L$	$[y_t(x)]_r^U$
0	0.875	1.125
0.1	0.975502508	1.225502508
0.2	1.078045555	1.328045555
0.3	1.184794808	1.434794808
0.4	1.298179066	1.548179066
0.5	1.421055880	1.671055880
0.6	1.556923622	1.806923622
0.7	1.710205904	1.960205904
0.8	1.886645759	2.136645759
0.9	2.093865202	2.343865202
1	2.342174693	2.592174693

8. Conclusion

In this work, we have introduced numerical method based on fully fuzzy neural network for solving first order fuzzy initial value problems. we have explained the efficiency of the feed forward fully fuzzy neural network to find the numerical solution of the first order fuzzy differential equations. In comparison with the exact solutions and with other numerical methods, we can concludes that the proposed method which we have used in this work provides numerical solutions with high accuracy . For future studies , one can apply and extend this method to solve higher order fuzzy differential equations .

References

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

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

[3] 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.

[4] 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 .

[5] 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 .

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

[7] 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 .

[8] 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 .

[9] 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 .

[10] 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 .

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

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

[13] Mosleh M. , Otadi M. ,"Fuzzy FredholmIntegro-Differential Equations with Artificial Neural

Networks",Communications in Numerical Analysis , Article ID cna-00128 , 1-13 , 2012 .

[14] 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 .

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

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

[17] 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 .

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