Available online at http://ijim.srbiau.ac.ir/
Int. J. Industrial Mathematics (ISSN 2008-5621)
Vol. 5, No. 4, 2013 Article ID IJIM-00263, 17 pages
Research Article

Numerical solution of fuzzy differential equations under generalized differentiability by fuzzy neural network

Maryam Mosleh [*] [†]

Abstract

In this paper, We interpret a fuzzy differential equation by using the strongly generalized differentiability concept. Utilizing the Generalized characterization Theorem. Then a novel hybrid method based on learning algorithm of fuzzy neural network for the solution of differential equation with fuzzy initial value is presented. Here neural network is considered as a part of large field called neural computing or soft computing. The model finds the approximated solution of fuzzy differential equation inside of its domain for the close enough neighborhood of the fuzzy initial point. We propose a learning algorithm from the cost function for adjusting of fuzzy weights.

Keywords : Fuzzy neural networks; Fuzzy differential equations; Feedforward neural network; Learning algorithm.

1 Introduction

PRoper design for engineering applications requires detailed information of the system-property distributions such as temperature, velocity, density, etc. in space and time domain [8, 9]. This information can be obtained by either experimental measurement or computational simulation. Although experimental measurement is reliable, it needs a lot of labor efforts and time. Therefore, the computational simulation has become a more and more popular method as a design tool since it needs only a fast computer with a large memory. Frequently, those engineering design problems deal with a set of differential equations (DEs), which have to numerically solved such as heat transfer, solid and fluid mechanics. Numerical methods of predictor-corrector, Runge-Kutta, finite difference, finite element, finite volume, boundary element, spectral and collocation provide a strategy by which we can attack many problems in applied mathematics, where we simulate a real-word problem with a differential equation, subject to some initial or boundary conditions. In the finite difference and finite element methods we approximate the solution by using the numerical operators of the function's derivatives and finding the solution at specific preassigned grids [56]. The linearity is assumed for the purposes of evaluating the derivatives. Although such an approximation method is conceptually easy to understand, it has a number of shortcomings. Obviously, it is diffi-

*Corresponding author. *mosleh@iaufb.ac.ir*

†Department of Mathematics, Firoozkooh Branch, Islamic Azad University, Firoozkooh, Iran.

cult to apply for systems with irregular geometry or unusual boundary conditions. Predictor-corrector and Runge-Kutta methods are widely applied over preassigned grid points to solve ordinary differential equations [35]. In the spectral and collocation approaches a truncated series of the specific orthogonal functions (basis functions) are used for finding the approximated solution of the DE. In the spectral and collocation techniques the role of trial functions as a basis function is important. The trial functions used in spectral methods are chosen from various classes of Jacobian polynomials [22], still the discretization meshes are preassigned. Neural network model is used to approximate the solutions of DEs for the entire domains. In 1990 the authors of [36] used parallel computers to solve a first order differential equation with Hopfield neural network models. Meade and Fernandez [39, 40] solved linear and nonlinear ordinary differential equations using feed forward neural networks architecture and B_1-splines. Leephakpreeda [37] applied neural network model and linguistic model as universal approximators for any nonlinear continuous functions. With this outstanding capability, the solution of DEs can be approximated by the appropriate neural network model and linguistic model within an arbitrary accuracy.

When a physical problem is transformed into a deterministic initial value problem

$$\begin{cases} \frac{dy(x)}{dx} = f(x, y), \\ y(a) = A, \end{cases} \tag{1.1}$$

We usually cannot be sure that this modelling is perfect. The initial value may not be known exactly and the function f may contain unknown parameters. If the nature of errors is random, then instead of a deterministic problem (1.1) we get a random differential equation with random initial value and/ or random coefficients. But if the underlying structure is not probabilistic, e.g. because of subjective choice, then it may be appropriate to use fuzzy numbers instead of real random variables.

The topic of Fuzzy Differential Equations (FDEs) has been rapidly growing in recent years. The fuzzy initial value problem have been studied by several authors [1, 2, 6, 7, 10, 50, 42, 43, 41, 13, 17, 51]. The concept of fuzzy derivative was first introduced by Chang and Zadeh [16], it was followed up by Dubois and Prade [19] who used the extension principle in their approach. Other methods have been discussed by Puri and Ralescu [49] and by Goetschel and Voxman [21]. Fuzzy differential equations were first formulated by Kaleva [32] and Seikkala [52] in time dependent form. Kaleva had formulated fuzzy differential equations, in terms of Hukuhara derivative [32]. Buckley and Feuring [14] have given a very general formulation of a fuzzy first-order initial value problem. They first find the crisp solution, make it fuzzy and then check if it satisfies the FDE. In [48, 20], investigated the existence and uniqueness of solution for fuzzy random differential equations with non-lipschitz coefficients and fuzzy differential equations with piecewise constant argument.

During the past few years, neural networks have received much attention Abbasbandy, Mosleh and *et al.* [3, 44, 45, 47]. In this work we propose a new solution method for the approximated solution of fuzzy differential equations under generalized differentiability using innovative mathematical tools and neural-like systems of computation. This hybrid method can result in improved numerical methods for solving fuzzy initial value problems. In this proposed method, fuzzy neural network model (FNNM) is applied as universal approximator. We use fuzzy trial function, this fuzzy trial function is a combination of two terms. A first term is responsible for the fuzzy initial while the second term contains the fuzzy neural network adjustable parameters to be calculated. The main aim of this paper is to illustrate how fuzzy connection weights are adjusted in the learning of fuzzy neural networks by the back-propagation-type learning algorithms [28, 31] for the approximated solution of fuzzy differential equations. Our fuzzy neural network in this paper is a three-layer feedforward neural network where connection weights and biases are fuzzy numbers. The remaining part of the paper is organized as follows. In Section 2, we discuss some basic definitions. Also, we briefly review relevant definition of the architecture of fuzzy neural networks. Section 3 gives details of problem formulation and the way to construct the fuzzy trial function and training of fuzzy neural network for finding the unknown adjustable coefficients. Also, training of partially fuzzy neural network for finding the unknown adjustable coefficients.

2 Preliminaries

In this section the most basic notations used in fuzzy calculus are introduced. We start by defining the fuzzy number.

Definition 2.1 *A fuzzy number is a fuzzy set $u : \mathbb{R}^1 \longrightarrow I = [0,1]$ which satisfies*

i. u is upper semi-continuous.

ii. $u(x) = 0$ outside some interval $[a, d]$.

iii. There are real numbers $b, c : a \le b \le c \le d$ for which

1. $u(x)$ is monotonic increasing on $[a, b]$,

2. $u(x)$ is monotonic decreasing on $[c, d]$,

3. $u(x) = 1, b, \le x \le c$.

The set of all the fuzzy numbers (as given by Definition 2.1) is denoted by E^1.

We briefly mention fuzzy number operations defined by the extension principle [57, 58]. Since input vector of feedforward neural network is fuzzy in this paper, the following addition, multiplication and nonlinear mapping of fuzzy numbers are necessary for defining our fuzzy neural network:

$$\mu_{A+B}(z) = max\{\mu_A(x) \wedge \mu_B(y) | z = x+y\}, \quad (2.2)$$

$$\mu_{AB}(z) = max\{\mu_A(x) \wedge \mu_B(y) | z = xy\}, \quad (2.3)$$

$$\mu_{f(Net)}(z) = max\{\mu_{Net}(x) | z = f(x)\}, \quad (2.4)$$

where A, B, Net are fuzzy numbers, $\mu_*(.)$ denotes the membership function of each fuzzy number, $\wedge$ is the minimum operator and $f(.)$ is a continuous activation function (like sigmoidal activation function) inside hidden neurons.

The above operations of fuzzy numbers are numerically performed on level sets (i.e.,α-cuts). The h-level set of a fuzzy number A is defined as

$$[A]_h = \{x \in \mathbb{R} | \mu_A(x) \ge h\} \quad for \quad 0 < h \le 1, \quad (2.5)$$

and $[A]_0 = \overline{\bigcup_{h \in (0,1]} [A]_h}$. Since level sets of fuzzy numbers become closed intervals, we denote $[A]_h$ as

$$[A]_h = [[A]_h^L, [A]_h^U], \quad (2.6)$$

where $[A]_h^L$ and $[A]_h^U$ are the lower limit and the upper limit of the h-level set $[A]_h$, respectively. From interval arithmetic [5], the above operations of fuzzy numbers are written for h-level sets as follows:

$$[A]_h + [B]_h = [[A]_h^L + [B]_h^L, [A]_h^U + [B]_h^U], \quad (2.7)$$

$$\begin{aligned} [A]_h.[B]_h = [&min\{[A]_h^L.[B]_h^L, [A]_h^L.[B]_h^U, \\ &[A]_h^U.[B]_h^L, [A]_h^U.[B]_h^U\}, max\{[A]_h^L.[B]_h^L, \\ &[A]_h^L.[B]_h^U, [A]_h^U.[B]_h^L, [A]_h^U.[B]_h^U\}], \quad (2.8) \end{aligned}$$

$$\begin{aligned} f([Net]_h) = f([[Net]_h^L, [Net]_h^U]) = \\ [f([Net]_h^L), f([Net]_h^U)], \quad (2.9) \end{aligned}$$

where f is increasing function.

Definition 2.2 *[21] For arbitrary fuzzy numbers U, V, we use the distance*

$$\begin{aligned} D(U,V) = sup_{0 \le h \le 1} max\{|[U]_h^L - [V]|_h^L, \\ |[U]_h^U - [V]_h^U|\} \end{aligned}$$

and it is shown that (E, D) is a complete metric space.

Definition 2.3 *Let $U, V \in E$. If there exists $W \in E$, such that $U = V + W$, then W is called the H-difference of U, V and it is denoted $U \ominus V$.*

In this paper the sign $\ominus$ always stands for the H-difference, and let us remark that $U \ominus V \neq U + (-1)V$. Usually we denote $U + (-1)V$ by $U - V$, while $U \ominus V$ stands for the H-difference. In what follows, we fix $X = (a, b)$, for $a, b \in \mathbb{R}$.

Definition 2.4 *Let $f : X \longrightarrow E$ be a fuzzy function. We say f is differentiable at $x_0 \in X$ if there exists an element $f'(x_0) \in E$ such that the limits*

$$lim_{h\longrightarrow 0^+}\frac{f(x_0+h)\ominus f(x_0)}{h}$$

and

$$lim_{h\longrightarrow 0^+}\frac{f(x_0)\ominus f(x_0-h)}{h},$$

exist and are equal to $F'(x_0)$.

The above definition is a straightforward generalization of the Hukuhara differentiability of a set-value function. From proposition 4.2.8 in [18], it follows that a Hukuhara differentiable function has increasing length of support. Note that this definition of a derivative is very restrictive [12]. The authors of [12] introduced a more general definition of a derivative for a fuzzy-number-valued function. In this paper we consider the following definition [15]:

Definition 2.5 *Let* $f : X \longrightarrow E$. *Fix* $x_0 \in X$. *We say* f *is differentiable at* x_0, *if there exists an element* $F'(x_0) \in E$ *such that*

(1) for all $h > 0$ *sufficiently close to* 0, *there exist* $f(x_0+h)\ominus F(x_0), f(x_0)\ominus f(x_0-h)$ *and the limits (in the metric D)*

$$lim_{h\longrightarrow 0^+}\frac{f(x_0+h)\ominus f(x_0)}{h} =$$

$$lim_{h\longrightarrow 0^+}\frac{f(x_0)\ominus f(t_0-h)}{h} = f'(t_0),$$

or

(2) for all $h > 0$ *sufficiently close to* 0, *there exist* $f(x_0+h)\ominus f(x_0), f(x_0)\ominus f(x_0-h)$ *and the limits (in the metric D)*

$$lim_{h\longrightarrow 0^-}\frac{f(x_0+h)\ominus f(x_0)}{h} =$$

$$lim_{h\longrightarrow 0^-}\frac{f(x_0)\ominus f(x_0-h)}{h} = f'(x_0).$$

Remark 2.1 *[12] This definition agrees with the one introduced in [12]. Indeed, if* f *is differentiable in the senses (1) and (2) simultaneously, then for* $h > 0$ *sufficiently small,, we have* $f(x_0+h) = f(x_0)+U_1, f(x_0) = f(x_0-h)+U_2, f(x_0) = f(x_0+h)+V_1$ *and* $f(x_0) = f(x_0+h)+v_2$, *with* $U_1, U_2, V_1, V_2 \in E$. *Thus,* $f(x_0) = f(x_0)+(U_2+V_1)$, *i.e.,*$U_2+V_1 = \chi_{\{0\}}$, *which implies two possibilities:* $U_2 = V_1 = \chi_{\{0\}}$ *if* $f'(x_0) = \chi_{\{0\}}$; *or* $U_2 = \chi_{\{a\}} = -V_1$, *with* $a \in \mathbb{R}$, *if* $f'(x_0) \in \mathbb{R}$. *Therefore, if there exists* $f'(x_0)$ *in the first form (second form) with* $f'(x_0)$ *is not in* $\mathbb{R}$, *then* $f'(x_0)$ *does not exist in the second form (first form, respectively).*

Remark 2.2 *In the previous definition, case (1) corresponds to the H-derivative introduced in [49], so this differentiability concept is a generalization of the H-derivative.*

Remark 2.3 *In [12], the authors consider four cases for derivatives. Here we only consider the two first cases od Definition 5 in [12]. In the other cases, the derivative is trivial because it is reduced to a crisp element (more precisely,* $f' \in \mathbb{R}$; *for details see Theorem 7 in [12]).*

Definition 2.6 *Let* $f : X \longrightarrow E$. *We say* f *is (1)-differentiable on* X *if* f *is differentiable in the sense (1) of Definition 7 and its derivative is denoted* D_1f, *and similarly for (2)-differentiability we have* D_2f.

The principal properties of defined derivatives are well known and can be found in [12, 15]. In this paper, we make use of the following Theorem [15].

Theorem 2.1 *Let* $f : X \longrightarrow E$ *and put* $[f(x)]_h = [[f(x)]^L_h, [f(x)]^U_h]$ *for each* $0 \leq h \leq 1$.
(i) If f *is (1)-differentiable then* $[f(x)]^L_h$ *and* $[f(x)]^U_h$ *are differentiable functions and* $[D_1f(x)]_h = [[f'(x)]^L_h, [f'(x)]^U_h]$.
(ii) If f *is (2)-differentiable then* $[f(x)]^L_h$ *and* $[f(x)]^U_h$ *are differentiable functions and we have* $[D_2f(x)]_h = [[f'(x)]^U_h, [f'(x)]^L_h]$.

Artificial neural networks are an exciting form of artificial intelligence which mimic the learning process of the human brain in order to extract patterns from historical data [53, 4]. Simple perceptrons need a teacher to tell the network what the desired output should be. These are supervised networks. In an unsupervised net, the network adapts purely in response to its inputs. Such networks can learn to pick out structure in their input. Fig. 1 shows typical three-layered perceptron. Multi-layered perceptrons with more than three layers, use more hidden layers [25, 33]. Multi-layered perceptrons correspond the input units to the output units by a specific nonlinear mapping [55]. From Kolmogorov existence theorem we know that a three-layered perceptron with $n(2n+1)$ nodes can compute any continuous function of n variables [26]. Before describ-

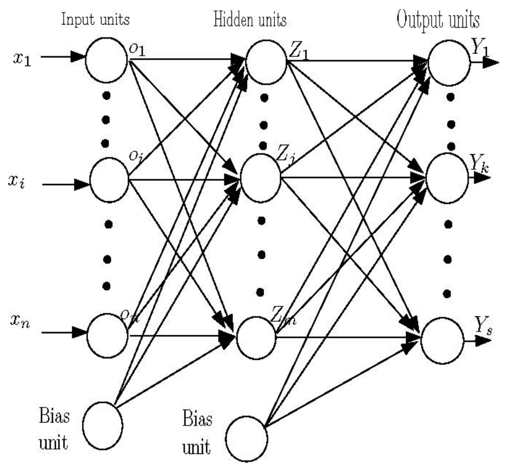

Figure 1: Multiple layer feed-forward FNNM.

ing a fuzzy neural network architecture, we denote real numbers and fuzzy numbers by lowercase letters ($e.g., a, b, c, \ldots$) and uppercase letters ($e.g., A, B, C, \ldots$), respectively.

Our fuzzy neural network is a three-layer feedforward neural network where connection weights, biases and targets are given as fuzzy numbers and inputs are given as real numbers. For convenience in this discussion, FNNM with an input layer, a single hidden layer, and an output layer in Fig. 1 is represented as a basic structural architecture. Here, the dimension of FNNM is denoted by the number of neurons in each layer, that is $n \times m \times s$, where m, n and s are the number of neurons in the input layer, the hidden layer and the output layer, respectively. The architecture of the model shows how FNNM transforms the n inputs $(x_1, \ldots, x_i, \ldots, x_n)$ into the s outputs $(Y_1, \ldots, Y_k, \ldots, Y_s)$ throughout the m hidden neurons $(Z_1, \ldots, Z_j, \ldots, Z_m)$, where the cycles represent the neurons in each layer. Let B_j be the bias for neuron Z_j, C_k be the bias for neuron Y_k, W_{ji} be the weight connecting neuron x_i to neuron Z_j, and W_{kj} be the weight connecting neuron Z_j to neuron Y_k.

2.1 Input-output relation of each unit

Let us consider a fuzzy three-layer feedforward neural network with n input units, m hidden units and s output units. Target vector, connection weights and biases are fuzzy and input vector is real number. In order to derive a crisp learning rule, we restrict connection weights and biases by four types of (real numbers, symmetric triangular fuzzy numbers, asymmetric triangular fuzzy numbers and asymmetric trapezoidal fuzzy numbers) while we can use any type of fuzzy numbers for fuzzy targets. For example, an asymmetric triangular fuzzy connection weight is specified by its three parameters as $W_{kj} = (w^L_{kj}, w^C_{kj}, w^U_{kj})$.

When an n-dimensional input vector $(x_1, \ldots, x_i, \ldots, x_n)$ is presented to our fuzzy neural network, its input-output relation can be written as follows, where $f : \mathbb{R}^n \longrightarrow E^s$:

Input units:

$$o_i = x_i, \qquad i = 1, 2, \ldots, n. \tag{2.10}$$

Hidden units:

$$Z_j = f(Net_j), \qquad j = 1, 2, \ldots, m, \tag{2.11}$$

$$Net_j = \sum_{i=1}^{n} o_i.W_{ji} + B_j. \tag{2.12}$$

Output units:

$$Y_k = f(Net_k), \qquad k = 1, 2, \ldots, s, \tag{2.13}$$

$$Net_k = \sum_{j=1}^{m} W_{kj}.Z_j + C_k. \tag{2.14}$$

The architecture of our fuzzy neural network is shown in Fig. 1 where connection weights, biases, and targets are fuzzy and inputs are real numbers. The input-output relation in Eqs.(*2.1*)-(2.14) is defined by the extension principle [57] as in Hayashi et al.[24] and Ishibuchi *et al.*[30].

2.2 Calculation of fuzzy output

The fuzzy output from each unit in Eqs.(*2.1*)-(2.14) is numerically calculated for real inputs and level sets of fuzzy weights and fuzzy biases. The input-output relations of our fuzzy neural network can be written for the h-level sets:
Input units:

$$o_i = x_i, \qquad i = 1, 2, \ldots, n. \tag{2.15}$$

Hidden units:

$$[Z_j]_h = f([Net_j]_h), \qquad j = 1, 2, \ldots, m, \tag{2.16}$$

$$[Net_j]_h = \sum_{i=1}^{n} o_i.[W_{ji}]_h + [B_j]_h. \tag{2.17}$$

Output unit:

$$[Y_k]_h = f([Net_k]_h), \qquad k = 1, 2, \ldots, s, \tag{2.18}$$

$$[Net_k]_h = \sum_{j=1}^{m} [W_{kj}]_h.[Z_j]_h + [C_k]_h. \tag{2.19}$$

From Eqs.(*2.2*)-(2.19), we can see that the h-level sets of the fuzzy outputs Y_k's are calculated from those of the fuzzy weights, fuzzy biases and crisp inputs. From Eqs.(2.7)-(2.9), the above relations are rewritten as follows when the inputs x_i's are nonnegative, i.e., $0 \leq x_i$:
Input units:

$$o_i = x_i. \tag{2.20}$$

Hidden units:

$$[Z_j]_h = [[Z_j]_h^L, [Z_j]_h^U] = \\ [f([Net_j]_h^L), f([Net_j]_h^U)], \tag{2.21}$$

where f is increasing function.

$$[Net_j]_h^L = \sum_{i=1}^{n} o_i.[W_{ji}]_h^L + [B_j]_h^L, \tag{2.22}$$

$$[Net_j]_h^U = \sum_{i=1}^{n} o_i.[W_{ji}]_h^U + [B_j]_h^U. \tag{2.23}$$

Output units:

$$[Y_k]_h = [[Y_k]_h^L, [Y_k]_h^U] = \\ [f([Net_k]_h^L), f([Net_k]_h^U)], \tag{2.24}$$

where f is increasing function.

$$[Net_k]_h^L = \sum_{j \in a} [W_{kj}]_h^L.[Z_j]_h^L + \\ \sum_{j \in b} [W_{kj}]_h^L.[Z_j]_h^U + [C_k]_h^L,$$

$$[Net_k]_h^U = \sum_{j \in c} [W_{kj}]_h^U.[Z_j]_h^U + \\ \sum_{j \in d} [W_{kj}]_h^U.[Z_j]_h^L + [C_k]_h^U, \tag{2.25}$$

for $[Z_j]_h^U \geq [Z_j]_h^L \geq 0$, where $a = \{j \mid [W_{kj}]_h^L \geq 0\}$, $b = \{j \mid [W_{kj}]_h^L < 0\}$, $c = \{j \mid [W_{kj}]_h^U \geq 0\}$,$d = \{j \mid [W_{kj}]_h^U < 0\}$, $a \cup b = \{1, \ldots, m\}$ and $c \cup d = \{1, \ldots, m\}$.

3 FDEs under generalized differentiability

Let us consider the FDE

$$\frac{dy(x)}{dx} = f(x, y) \tag{3.26}$$

where y is a fuzzy function of x, $f(x, y)$ a fuzzy function of the crisp variable x and the fuzzy variable y, and y' is the fuzzy derivative of y. If an initial value $y(a) = A$ is given, we obtain a fuzzy Cauchy problem of first order:

$$\begin{cases} \frac{dy(x)}{dx} = f(x, y), & x \in [a, b], \\ y(a) = A, \end{cases} \tag{3.27}$$

where A is a fuzzy number in E with h-level sets

$$[A]_h = [[A]_h^L, [A]_h^U], \qquad 0 < h \leq 1.$$

Theorem 3.1 *[15] Let $f : X \times E \longrightarrow E$ be a continuous fuzzy function such that there exists $k > 0$ such that $D(f(x, y), f(x, z)) \leq kD(y, z)$, $\forall x \in X, y, z \in E$. Then problem (3.27) has two solution (one (1)-differentiable and the other one (2)-differentiable) on X.*

Definition 3.1 *Let $y : X \longrightarrow E$ be a fuzzy function such that D_1y or D_2y exists. If y and D_1y satisfy problem (3.27), we say y is a (1)-solution of problem (3.27). Similarly, if y and D_2y satisfy problem (3.27), we say y is a (2)-solution of problem (3.27).*

Let $[y(x)]_h = [[y(x)]_h^L, [y(x)]_h^U]$. If $y(x)$ is (1)-differentiable then $D_1y(x) = [[y'(x)]_h^L, [y'(x)]_h^U]$, and (3.27) translates into the following system of ODEs:

$$\begin{cases} [y'(x)]_h^L = [f(x, y)]_h^L, \quad [y(a)]_h^L = [A]_h^L, \\ \\ [y'(x)]_h^U = [f(x, y)]_h^U, \quad [y(a)]_h^U = [A]_h^U. \end{cases} \tag{3.28}$$

Also, if $y(x)$ is (2)-differentiable then $D_2y(x) = [[y'(x)]_h^U, [y'(x)]_h^L]$, and (3.27) translates into the following system of ODEs:

$$\begin{cases} [y'(x)]_h^L = [f(x, y)]_h^U, \quad [y(a)]_h^L = [A]_h^L, \\ \\ [y'(x)]_h^U = [f(x, y)]_h^L, \quad [y(a)]_h^U = [A]_h^U, \end{cases} \tag{3.29}$$

where $[f(x, y)]_h = [[f(x, y)]_h^L, [f(x, y)]_h^U]$. The authors of [15] state that if we ensure that the solution $[[y(x)]_h^L, [y(x)]_h^U]$ of the system (3.28) are valid level sets of a fuzzy number valued function and if $[[y(x)]_h^L, [y(x)]_h^U]$ are valid level sets of a fuzzy valued function, then by the stacking Theorem [32], it is possible to construct the (1)-solution of FDE (3.27). Also, for the (2)-solution, we can proceed in a similar way.

The Characterization Theorem [11] states that a FDE is equivalent to a system of ODEs under certain conditions.

Theorem 3.2 *[46] Let us consider the FDE (3.27) where $f : X \times E \longrightarrow E$ is such that*
(i) $[f(x, y)]_h = [[f(x, [y]_h^L, [y]_h^U)]_h^L, [f(x, [y]_h^L, [y]_h^U)]_h^U]$;
(ii) $[f(x, [y]_h^L, [y]_h^U)]_h^L$ and $[f(x, [y]_h^L, [y]_h^U)]_h^U$ are equicontinuous;
(iii) there exists $L > 0$ such that

$$|[f(x, y_1, z_1)]_h^L - [f(x, y_2, z_2)]_h^L| \leq$$

$$L max\{|y_1 - y_2|, |z_1 - z_2|\}, \quad \forall h \in [0, 1];$$

$$|[f(x, y_1, z_1)]_h^U - [f(x, y_2, z_2)]_h^U| \leq$$

$$L max\{|y_1 - y_2|, |z_1 - z_2|\}, \quad \forall h \in [0, 1].$$

Then, for (1)-differentiability, the FDE (3.27) and the system of ODEs (3.28) are equivalent and in (2)-differentiability, the FDE (3.27) and the system of ODEs (3.29) are equivalent.

Let us assume that a general approximation solution to Eq.(3.27) is in the form $y_T(x, P)$ for y_T as a dependent variable to x and P, where P is an adjustable parameter involving weights and biases in the structure of the three-layered feed forward FNNM (see Fig. 2). The fuzzy trial solution y_T is an approximation solution to y for the optimized value of unknown weights and biases. Thus the problem of finding the approximated fuzzy solution for Eq. (3.27) over some collocation points in $[a, b]$ by a set of discrete equally spaced grid points

$$a = x_1 < x_2 < \ldots < x_g = b,$$

is equivalent to calculate the functional $y_T(x, P)$. In order to obtain fuzzy approximate solution $y_T(x, P)$, we solve unconstrained optimization problem that is simpler to deal with, we define the fuzzy trial function to be in the following form:

$$y_T(x, P) = \alpha(x) + \beta[x, N(x, P)], \tag{3.30}$$

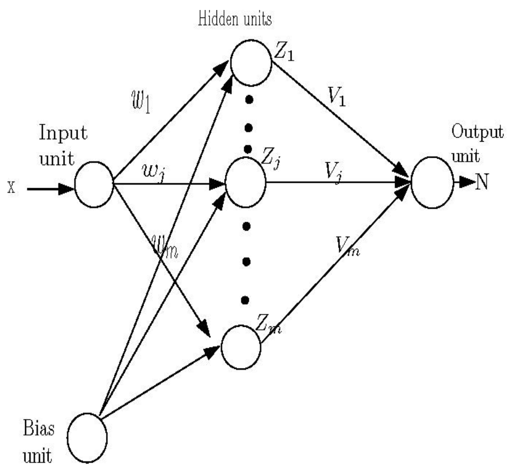

Figure 2: Three layer fuzzy neural network with one input and one output.

where the first term in the right hand side does not involve with adjustable parameters and satisfies the fuzzy initial condition. The second term in the right hand side is a feed forward three-layered fuzzy neural network consisting of an input x and the output $N(x, P)$. The crisp trial function was used in [38]. In the next subsection, this FNNM with some weights and biases is considered and we train in order to compute the approximate solution of problem (3.27).

Let us consider a three-layered FNNM (see Fig. 2) with one unit entry x, one hidden layer consisting of m activation functions and one unit output $N(x, P)$. In this paper, we use the sigmoidal activation function $s(.)$ for the hidden units of our fuzzy neural network.

Here, the dimension of FNNM is $1 \times m \times 1$.

One drawback of fully fuzzy neural networks with fuzzy connection weights is long computation time. Another drawback is that the learning algorithm is complicated. For reducing the complexity of the learning algorithm, we propose a partially fuzzy neural network (PFNN) architecture where connection weights to output unit are fuzzy numbers while connection weights and biases to hidden units are real numbers [29, 42]. Since we had good simulation results even from partially fuzzy three-layer neural networks, we do not think that the extension of our learning algorithm to neural networks with more than three layer is an attractive research direction.

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

$$net_j = x.w_j + b_j, \quad j = 1, \dots, m, \tag{3.31}$$

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

$$z_j = s(net_j), \quad j = 1, \dots, m, \tag{3.32}$$

where $s(.)$ is the activation function which is normally nonlinear function, the usual choices of the activation function [23] are the sigmoid transfer function, and the output neuron make no change its input, so the input to the output neuron is equal to output

$$N = V_1 z_1 + \ldots + V_j z_j + \ldots + V_m z_m, \tag{3.33}$$

where V_j is a weight parameter from jth unit in the hidden layer to the output layer. From Eqs. (2.20)-(2.25), we can be rewritten for h-level sets of the Eqs. (3.31)-(3.33). For reducing the complexity of the learning algorithm, input x usually assumed as non-negative in fully fuzzy neural networks, i.e., $0 \leq x$ [28]:
Input unit:

$$o = x. \tag{3.34}$$

Hidden units:

$$z_j = s(net_j), \quad j = 1, \dots, m, \tag{3.35}$$

$$net_j = o.w_j + b_j. \tag{3.36}$$

Output unit:

$$[N]_h = [[N]_h^L, [N]_h^U] =$$

$$[\sum_{j=1}^{m} [V_j]_h^L . z_j, \sum_{j=1}^{m} [V_j]_h^U . z_j]. \tag{3.37}$$

A $PFNN_4$ (partially fuzzy neural network with crisp set input signals, crisp number weights and biases to hidden units and fuzzy number weights to output unit) solution to Eq. (3.27) is given in Fig. 2. How is the $PFNN_4$ going to solve the fuzzy differential equations? The training data are $a = x_1 < x_2 < \ldots < x_g = b$ for input. We propose a learning algorithm from the cost function for adjusting weights.

Consider the following fuzzy initial value problem for a first order differential equation (3.27), the related trial function will be in the form

$$y_T(x, P) = A + (x - a)N(x, P), \tag{3.38}$$

this solution by intention satisfies the initial condition in (3.27). In [28], the learning of our fuzzy neural network is to minimize the difference between the fuzzy target vector $B = (B_1, \ldots, B_s)$ and the actual fuzzy output vector $O = (O_1, \ldots, O_s)$. The following cost function was used in [28, 3] for measuring the difference between B and O:

$$e = \sum_h e_h = \sum_h h.\{\sum_{k=1}^{s}([B_k]_h^L - [O_k]_h^L)^2/2 + \sum_{k=1}^{s}([B_k]_h^U - [O_k]_h^U)^2/2\}, \tag{3.39}$$

where e_h is the cost function for the h-level sets of B and O. The squared errors between the h-level sets of B and O are weighted by the value of h in (3). In [29], it is shown by computer simulations that their paper, the fitting of fuzzy outputs to fuzzy targets is not good for the h-level sets with small values of h when we use the cost function in (3). This is because the squared errors for the h-level sets are weighted by h in (3). Krishnamraju et al. [34] used the cost function without the weighting scheme:

$$e = \sum_h e_h = \sum_h \{\sum_{k=1}^{s}([B_k]_h^L - [O_k]_h^L)^2/2 + \sum_{k=1}^{s}([B_k]_h^U - [O_k]_h^U)^2/2\}. \tag{3.40}$$

In the computer simulations included in this paper, we mainly use the cost function in (3) without the weighting scheme.

The error function that must be minimized for problem (3.28) is in the form

$$e_1 = \sum_{i=1}^{g} e_{1i} = \sum_{i=1}^{g}\sum_h e_{1ih} = \sum_{i=1}^{g}\sum_h \{e_{1ih}^L + e_{1ih}^U\}, \tag{3.41}$$

where

$$e_{1ih}^L = \frac{([\frac{dy_T(x_i,P)}{dx}]_h^L - [f(x_i, y_T(x_i, P))]_h^L)^2}{2},$$

$$e_{1ih}^U = \frac{([\frac{dy_T(x_i,P)}{dx}]_h^U - [f(x_i, y_T(x_i, P))]_h^U)^2}{2}.$$

Also, the error function that must be minimized for problem (3.29) is in the form

$$e_2 = \sum_{i=1}^{g} e_{2i} = \sum_{i=1}^{g}\sum_h e_{2ih} = \sum_{i=1}^{g}\sum_h \{e_{2ih}^L + e_{2ih}^U\}, \tag{3.42}$$

where

$$e_{2ih}^L = \frac{([\frac{dy_T(x_i,P)}{dx}]_h^L - [f(x_i, y_T(x_i, P))]_h^U)^2}{2},$$

$$e_{2ih}^U = \frac{([\frac{dy_T(x_i,P)}{dx}]_h^U - [f(x_i, y_T(x_i, P))]_h^L)^2}{2}.$$

and $\{x_i\}_{i=1}^g$ are discrete points belonging to the interval $[a, b]$ and in the cost functions (3.41) and (3.42) e_{1ih}^L, e_{2ih}^L can be viewed as the squared errors for the lower limits and e_{1ih}^U, e_{2ih}^U the upper limits of the h-level sets. It is easy to express the first derivative of $N(x, P)$ in terms of the derivative of the sigmoid function, i.e.

$$\frac{\partial [N]_h^L}{\partial x} = \sum_{j=1}^{m} [V_j]_h^L . \frac{\partial z_j}{\partial net_j} . \frac{\partial net_j}{\partial x} =$$

$$\sum_{j=1}^{m}[V_j]_h^L.z_j.(1-z_j).w_j, \tag{3.43}$$

$$\frac{\partial[N]_h^U}{\partial x} = \sum_{j=1}^{m}[V_j]_h^U.\frac{\partial z_j}{\partial net_j}.\frac{\partial net_j}{\partial x} =$$

$$\sum_{j=1}^{m}[V_j]_h^U.z_j.(1-z_j).w_j. \tag{3.44}$$

Now differentiating from trial function $y_T(x,P)$ in (3.42), we obtain

$$\frac{\partial[y_T(x,P)]_h^L}{\partial x} =$$

$$[N(x,P)]_h^L + (x-a).\frac{\partial[N(x,P)]_h^L}{\partial x},$$

$$\frac{\partial[y_T(x,P)]_h^U}{\partial x} =$$

$$[N(x,P)]_h^U + (x-a).\frac{\partial[N(x,P)]_h^U}{\partial x},$$

thus the expressions in (3.43) and (3.44) are applicable here. A learning algorithm is derived in Appendix.

4 Example

In this section, we apply FNNM to an example. Consider the following FDE

Example 4.1 *Consider the nuclear decay equation*

$$\begin{cases} \frac{dy(x)}{dx} = -\lambda y(x), \\ y(0) = A, \end{cases}$$

where $y(x)$ is the number of radionuclides present in a given radioactive material, λ is the decay constant and A is the initial number of radionuclides. In the model, uncertainty is introduced if we have uncertain information on the initial number A of radionuclides present in the material. Note that the phenomenon of nuclear disintegration is considered a stochastic process, uncertainty being introduced by the lack of information on the radioactive material under study. In order to take into account the uncertainty we consider A to be a fuzzy number.

Let $\lambda = 1$, $x \in [0, 0.1]$ and $[A]_h = [h-1, 1-h]$. By using the Eq. (3.28) we get the exact solution

$$[y_1(x)]_h = [(h-1)e^x, (1-h)e^x],$$

that is a (1)-differentiable solution of the problem (3.27). Using the Eq. (3.29) we get the exact solution

$$[y_2(x)]_h = [(h-1)e^{-x}, (1-h)e^{-x}],$$

is a (2)-differentiable solution of the problem (3.27).

In the interval $[0, 0.1]$ we consider a set of discrete equally spaced grid points $0 = x_1 < 0.01 < ... < 0.1$ and for Eq. (3.28) and Eq. (3.29) the approximate solutions are denoted by $y_{1T}(x,P)$ and $y_{2T}(x,P)$, respectively.

Here, the dimension of PFNN is $1\times5\times1$. The error function for the $m = 5$ sigmoid units in the hidden layer and for $g = 11$ equally spaced points inside the interval $[0, 0.1]$ is trained. In the computer simulation of this section, we use the following specifications of the learning algorithm.

(1) Number of hidden units: five units.

(2) Stopping condition: 100 iterations of the learning algorithm.

(3) Learning constant: $\eta = 0.1$.

(4) Momentum constant: $\alpha = 0.2$.

(5) Initial value of the weights and biases of PFNN are shown in table 1, that we suppose $V_i = (v_i^{(1)}, v_i^{(2)}, v_i^{(3)})$ for $i = 1, \dots, 5$.

We apply the proposed method to the approximate realization of solution of problem (3.27). A comparison between the exact and approximate solutions at $x = 0.1$ is shown in the Figures 3 and 4. Now, if we consider the same differential equation under Hukuhara differentiability, then the (1)-solution (it exists and is unique by theorems in [54]) has an increasing length of its support, which leads us to the conclusion that there is a possibility that the radioactivity of the system increases as time goes on and even a non-zero possibility that it is negative. Fortunately, the real situation is different, and the radioactivity of a material always decreases with time and it cannot be negative. So, the (2)-solution models the

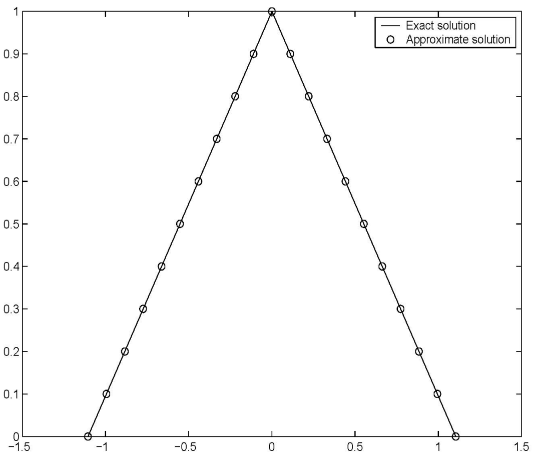

Figure 3: Analytical solution y_1 and approximate solution y_{1T}.

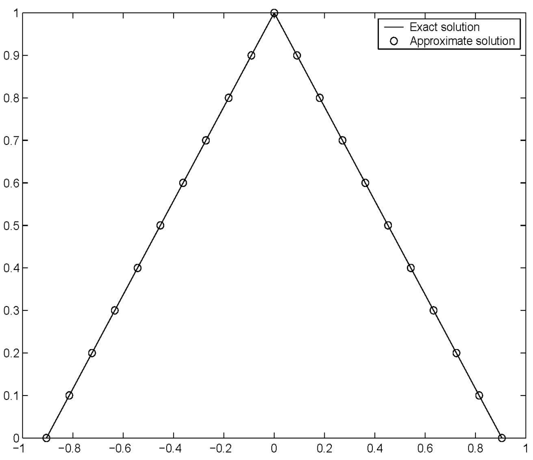

Figure 4: Analytical solution y_2 and approximate solution y_{2T}.

radioactive decay better. Now, if we consider the same differential equation under Hukuhara differentiability, then the (1)-solution (it exists and is unique by theorems in [54]) has an increasing length of its support, which leads us to the conclusion that there is a possibility that the radioactivity of the system increases as time goes on and even a non-zero possibility that it is negative. Fortunately, the real situation is different, and the radioactivity of a material always decreases with time and it cannot be negative. So, the (2)-solution models the radioactive decay better.

heighti	*1*	*2*	*3*	*4*	*5*
$v_i^{(1)}$	*-0.5*	*-0.5*	*-0.5*	*-0.5*	*-0.5*
$v_i^{(2)}$	*0*	*0*	*0*	*0*	*0*
$v_i^{(3)}$	*0.5*	*0.5*	*0.5*	*0.5*	*0.5*
w_i	*0*	*0*	*0*	*0*	*0*
b_i	*0*	*0*	*0*	*0*	*0*

Table 4.1. The initial values of weights

5 Conclusion

Solving fuzzy differential equations (FDEs) under generalized differentiability by using universal approximators (UA), that is, FNNM is presented in this paper. The problem formulation of the proposed UAM is quite straightforward. To obtain the "Best-approximated" solution of FDEs, the adjustable parameters of FNNM are systematically adjusted by using the learning algorithm.

In this paper, we derived a learning algorithm of fuzzy weights of tree-layer feedforward fuzzy neural networks whose input-output relations were defined by extension principle. The effectiveness of the derived learning algorithm was demonstrated by computer simulation on numerical example. Since we had good simulation result even from partially fuzzy three-layer neural networks, we do not think that the extension of our learning algorithm to neural networks with more than three layers is an attractive research direction. Good simulation result was obtained by this neural network in shorter computation times than fully fuzzy neural networks in our computer simulations.

Appendix

Derivation of a learning algorithm in PFNN

Let us denote the fuzzy connection weight V_j to the output unit by its parameter values as $V_j = (v_j^{(1)}, \ldots, v_j^{(q)}, \ldots, v_j^{(r)})$. The amount of modification of each parameter value for problem (3.28) is written as [27]

$$v_j^{(q)}(t+1) = v_j^{(q)}(t) + \triangle v_j^{(q)}(t),$$

$$\triangle v_j^q(t) = -\eta \sum_{i=1}^{g} \frac{\partial e_{1ih}}{\partial v_j^{(q)}} + \alpha. \triangle v_j^{(q)}(t-1),$$

and the amount of modification of each parameter value for problem (3.29) is written as:

$$v_j^{(q)}(t+1) = v_j^{(q)}(t) + \triangle v_j^{(q)}(t),$$

$$\triangle v_j^q(t) = -\eta \sum_{i=1}^{g} \frac{\partial e_{2ih}}{\partial v_j^{(q)}} + \alpha. \triangle v_j^{(q)}(t-1),$$

where t indexes the number of adjustments, η is a learning rate (positive real number) and α is a momentum term constant (positive real number).

Thus our problem is to calculate the derivatives $\frac{\partial e_{1ih}}{\partial v_j^{(q)}}$ and $\frac{\partial e_{2ih}}{\partial v_j^{(q)}}$. Let us rewrite $\frac{\partial e_{1ih}}{\partial v_j^{(q)}}$ and $\frac{\partial e_{2ih}}{\partial v_j^{(q)}}$ as follows:

$$\frac{\partial e_{1ih}}{\partial v_j^{(q)}} = \frac{\partial e_{1ih}}{\partial [V_j]_h^L}.\frac{\partial [V_j]_h^L}{\partial v_j^{(q)}} + \frac{\partial e_{1ih}}{\partial [V_j]_h^U}.\frac{\partial [V_j]_h^U}{\partial v_j^{(q)}},$$

$$\frac{\partial e_{2ih}}{\partial v_j^{(q)}} = \frac{\partial e_{2ih}}{\partial [V_j]_h^L}.\frac{\partial [V_j]_h^L}{\partial v_j^{(q)}} + \frac{\partial e_{2ih}}{\partial [V_j]_h^U}.\frac{\partial [V_j]_h^U}{\partial v_j^{(q)}}.$$

In this formulation, $\frac{\partial [V_j]_h^L}{\partial v_j^{(q)}}$ and $\frac{\partial [V_j]_h^U}{\partial v_j^{(q)}}$ are easily calculated from the membership function of the fuzzy connection weight V_j.

On the other hand, the derivatives $\frac{\partial e_{1ih}}{\partial [V_j]_h^L}$, $\frac{\partial e_{1ih}}{\partial [V_j]_h^U}$, $\frac{\partial e_{2ih}}{\partial [V_j]_h^L}$ and $\frac{\partial e_{2ih}}{\partial [V_j]_h^U}$ are independent of the shape of the fuzzy connection weight. They can be calculated from the cost functions e_{1ih} and e_{2ih} using the input-output relation of our fuzzy neural network for the h-level sets. When we use the cost function with the weighting scheme in (3.41) and (3.42), $\frac{\partial e_{1ih}}{\partial [V_j]_h^L}$, $\frac{\partial e_{1ih}}{\partial [V_j]_h^U}$, $\frac{\partial e_{2ih}}{\partial [V_j]_h^L}$ and $\frac{\partial e_{2ih}}{\partial [V_j]_h^U}$ are calculated as follows:

[Calculation of $\frac{\partial e_{1ih}}{\partial [V_j]_h^L}$]

$$\frac{\partial e_{1ih}}{\partial [V_j]_h^L} = \delta_1^L.[\frac{\partial [N(x_i,P)]_h^L}{\partial [V_j]_h^L} + (x_i-a).\frac{\partial z_j}{\partial x}$$
$$-\frac{\partial [f(x,y_T(x_i,P))]_h^L}{\partial [y_T(x_i,P)]_h^L}.\frac{\partial [y_T(x_i,P))]_h^L}{\partial [V_j]_h^L}],$$

where

$$\delta_1^L = ([\frac{dy_T(x_i,P)}{dx}]_h^L - [f(x_i,y_T(x_i,P))]_h^L),$$

$$\frac{\partial [y_T(x_i,P))]_h^L}{\partial [V_j]_h^L}] = (x_i-a).z_j,$$

$$\frac{\partial N(x_i,P)]_h^L}{\partial [V_j]_h^L} = z_j.$$

[Calculation of $\frac{\partial e_{1ih}}{\partial [V_j]_h^U}$]

$$\frac{\partial e_{1ih}}{\partial [V_j]_h^U} = \delta_1^U.[\frac{\partial N(x_i,P)]_h^U}{\partial [V_j]_h^U} + (x_i-a).\frac{\partial z_j}{\partial x}$$
$$-\frac{\partial [f(x,y_T(x_i,P))]_h^U}{\partial [y_T(x_i,P)]_h^U}.\frac{\partial [y_T(x_i,P))]_h^U}{\partial [V_j]_h^U}],$$

where

$$\delta_1^U = ([\frac{dy_T(x_i,P)}{dx}]_h^U - [f(x_i,y_T(x_i,P))]_h^U),$$

$$\frac{\partial [y_T(x_i,P))]_h^U}{\partial [V_j]_h^U}] = (x_i-a).z_j,$$

$$\frac{\partial N(x_i,P)]_h^U}{\partial [V_j]_h^U} = z_j.$$

[Calculation of $\frac{\partial e_{2ih}}{\partial [V_j]_h^L}$]

$$\frac{\partial e_{2ih}}{\partial [V_j]_h^L} = \delta_2^L.[\frac{\partial [N(x_i,P)]_h^U}{\partial [V_j]_h^L} + (x_i-a).\frac{\partial z_j}{\partial x}$$
$$-\frac{\partial [f(x,y_T(x_i,P))]_h^L}{\partial [y_T(x_i,P)]_h^L}.\frac{\partial [y_T(x_i,P))]_h^L}{\partial [V_j]_h^L}],$$

where

$$\delta_2^L = ([\frac{dy_T(x_i,P)}{dx}]_h^U - [f(x_i,y_T(x_i,P))]_h^L),$$

$$\frac{\partial [y_T(x_i,P))]_h^L}{\partial [V_j]_h^L}] = (x_i-a).z_j,$$

$$\frac{\partial N(x_i,P)]_h^L}{\partial [V_j]_h^L} = z_j.$$

[Calculation of $\frac{\partial e_{2ih}}{\partial [V_j]_h^U}$]

$$\frac{\partial e_{2ih}}{\partial [V_j]_h^U} = \delta_2^U.[\frac{\partial N(x_i,P)]_h^L}{\partial [V_j]_h^U} + (x_i-a).\frac{\partial z_j}{\partial x}$$
$$-\frac{\partial [f(x,y_T(x_i,P))]_h^U}{\partial [y_T(x_i,P)]_h^U}.\frac{\partial [y_T(x_i,P))]_h^U}{\partial [V_j]_h^U}],$$

where

$$\delta_2^U = ([\frac{dy_T(x_i,P)}{dx}]_h^L - [f(x_i, y_T(x_i,P))]_h^U),$$

$$\frac{\partial[y_T(x_i,P))]_h^U}{\partial[V_j]_h^U}] = (x_i - a).z_j,$$

$$\frac{\partial N(x_i,P)]_h^U}{\partial[V_j]_h^U} = z_j.$$

In our partially fuzzy neural network, the connection weights and biases to the hidden units are real numbers. The non-fuzzy connection weight w_j to the jth hidden unit is updated in the same manner as the parameter values of the fuzzy connection weight V_j as follows:

$$w_j(t+1) = w_j(t) + \triangle w_j(t),$$

$$\triangle w_j(t) = -\eta \sum_{i=1}^{g} \frac{\partial e_{ih}}{\partial w_j} + \alpha \triangle w_j(t-1).$$

The derivative $\frac{\partial e_{1ih}}{\partial w_j}$ and $\frac{\partial e_{2ih}}{\partial w_j}$ can be calculated from the cost function e_{ih} using the input-output relation of our PFNN for the h-level sets. When we use the cost function with the weighting scheme, $\frac{\partial e_{1ih}}{\partial w_j}$ and $\frac{\partial e_{2ih}}{\partial w_j}$ are calculated as follows:

$$\frac{\partial e_{1ih}}{\partial w_j} = \delta_1^L.[\frac{\partial[N(x_i,P)]_h^L}{\partial w_j} + (x_i - a).[V_j]_h^L.z_j$$

$$+(x_i - a).x_i.[V_j]_h^L.z_j(1-z_j)w_j$$

$$-(x_i - a).[V_j]_h^L.z_j^2 - 2(x_i - a).x_i.[V_j]_h^L.z_j^2$$

$$(1-z_j)w_j - (\frac{\partial[f(x_i, y_T(x_i,P))]_h^L}{\partial[y_T(x_i,P))]_h^L}$$

$$.\frac{\partial[y_T(x_i,P)]_h^L}{\partial}$$

w˙j+$\partial[f(x_i, y_T(x_i,P))]_h^L \frac{}{\partial[y_T(x,P))]_h^U}.$

$$\frac{\partial[y_T(x_i,P)]_h^U}{\partial w_j})] + \delta_1^U.[\frac{\partial N(x_i,P)]_h^U}{\partial w_j} +$$

$$(x_i - a).[V_j]_h^U.z_j + (x_i - a).x_i.[V_j]_h^U.z_j$$

$$(1-z_j)w_j - (x_i - a).[V_j]_h^U.z_j^2 -$$

$$2(x_i - a).x_i.[V_j]_h^U.z_j^2(1-z_j)w_j -$$

$$(\frac{\partial[f(x_i, y_T(x_i,P))]_h^U}{\partial[y_T(x_i,P))]_h^L}.\frac{\partial[y_T(x_i,P)]_h^L}{\partial}$$

w˙j+

$$\frac{\partial[f(x_i, y_T(x_i,P))]_h^U}{\partial[y_T(x_i,P))]_h^U}.\frac{\partial[y_T(x_i,P)]_h^U}{\partial w_j})],$$

and

$$\frac{\partial e_{2ih}}{\partial w_j} = \delta_2^U.[\frac{\partial[N(x_i,P)]_h^U}{\partial w_j} + (x_i - a).[V_j]_h^U.z_j$$

$$+(x_i - a).x_i.[V_j]_h^U.z_j(1-z_j)w_j$$

-(x˙i-a).

$$[V_j]_h^U.z_j^2 - 2(x_i - a).x_i.[V_j]_h^U.z_j^2(1-z_j)w_j -$$

$$(\frac{\partial[f(x_i, y_T(x_i,P))]_h^L}{\partial[y_T(x_i,P))]_h^L}.\frac{\partial[y_T(x_i,P)]_h^L}{\partial}$$

w˙j+

$$\frac{\partial[f(x_i, y_T(x_i,P))]_h^L}{\partial[y_T(x,P))]_h^U}.\frac{\partial[y_T(x_i,P)]_h^U}{\partial w_j})]+$$

$$\delta_2^U.[\frac{\partial N(x_i,P)]_h^L}{\partial w_j} + (x_i - a).[V_j]_h^L.z_j + (x_i - a).$$

$$x_i.[V_j]_h^L.z_j(1-z_j)w_j - (x_i - a).[V_j]_h^L.z_j^2 -$$

$$2(x_i - a).x_i.[V_j]_h^L.z_j^2(1-z_j)w_j -$$

$$(\frac{\partial[f(x_i, y_T(x_i,P))]_h^U}{\partial[y_T(x_i,P))]_h^L}.\frac{\partial[y_T(x_i,P)]_h^L}{\partial}$$

w˙j

$$+\frac{\partial[f(x_i, y_T(x_i,P))]_h^U}{\partial[y_T(x_i,P))]_h^U}.\frac{\partial[y_T(x_i,P)]_h^U}{\partial}$$

w˙j)],

where

$$\frac{\partial[N(x_i,P)]_h^L}{\partial w_j}=\frac{\partial[N(x_i,P)]_h^L}{\partial z_j}.\frac{\partial z_j}{\partial net_j}.$$

$$\frac{\partial net_j}{\partial w_j}=[V_j]_h^L.z_j.(1-z_j).x_i,$$

$$\frac{\partial[N(x_i,P)]_h^U}{\partial w_j}=\frac{\partial[N(x_i,P)]_h^U}{\partial z_j}.\frac{\partial z_j}{\partial net_j}.$$

$$\frac{\partial net_j}{\partial w_j}=[V_j]_h^U.z_j.(1-z_j).x_i,$$

$$\frac{\partial[y_T(x_i,P))]_h^L}{\partial w_j}=(x_i-a).\frac{\partial[N(x_i,P)]_h^L}{\partial w_j},$$

$$\frac{\partial[y_T(x_i,P))]_h^U}{\partial w_j}=(x_i-a).\frac{\partial[N(x_i,P)]_h^U}{\partial w_j}.$$

The non-fuzzy biases to the hidden units are updated in the same manner as the non-fuzzy connection weights to the hidden units.

References

[1] S. Abbasbandy, J. J. Nieto, M. Alavi, *Tuning of reachable set in one dimensional fuzzy differential inclusions*, Chaos, Solitons & Fractals 26 (2005) 1337-1341.

[2] S. Abbasbandy, T. Allaviranloo, O. Lopez-Pouso, J. J. Nieto, *Numerical methods for fuzzy differential inclusions*, Computers & mathematics with applications 48 (2004) 1633-1641.

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

[4] S. Abbasbandy, M. Otadi, M. Mosleh, *Numerical solution of a system of fuzzy polynomials by fuzzy neural network*, Information Sciences 178 (2008) 1948-1960.

[5] G. Alefeld, J. Herzberger, *Introduction to Interval Computations*, Academic Press, New York, (1983).

[6] T. Allahviranloo, E. Ahmady, N. Ahmady, *Nth-order fuzzy linear differential eqations*, Information Sciences 178 (2008) 1309-1324.

[7] T. Allahviranloo, N. Ahmady, E. Ahmady, *Numerical solution of fuzzy differential eqations by predictor-corrector method*, Information Sciences 177 (2007) 1633-1647.

[8] H. Md. Azamathulla, A. Ab. Ghani, N. A. Zakaria, *ANFIS based approach for predicting maximum scour location of spillway*, Water Management, ICE London 162 (6) 399-407.

[9] H. Md. Azamathulla, C. C. Kiat, A. Ab. Ghani, Z. A. Hasan, N. A. Zakaria, *An ANFIS-based approach for predicting the bed load for moderately-sized rivers*, Journal of Hydro-Environment Research, Elsevier & KWRA 3 (2009) 35-44.

[10] M. Barkhordari Ahmadi, N. A. Kiani, *Differential transformation method for solving fuzzy differential inclusions by fuzzy partitions*, International Journal of Industrial Mathematics 5 (2013) 237-249.

[11] B. Bede, *Note on "Numerical solutions of fuzzy differential equations by predictor corrector method"*, Information Sciences 178 (2008) 1917-1922.

[12] B. Bede, S. G. Gal, *Generalizations of the differentiability of fuzzy number value functions with applications to fuzzy differential equations*, Fuzzy Sets and Systems 151 (2005) 581-599.

[13] B. Bede, I. J. Rudas, A. L. Bencsik, *First order linear fuzzy differential eqations under generalized differentiability*, Information Sciences 177 (2007) 1648-1662.

[14] J. J. Buckley, T. Feuring, *Fuzzy differential equations*, Fuzzy Sets and Systems 110 (2000) 69-77.

[15] Y. Chalco-Cano, H. Roman-Flores, *On new solutions of fuzzy differential equations*, Chaos Solitons & Fractals 38 (2008) 112-119.

[16] S. L. Chang, L. A. Zadeh, *On fuzzy mapping and control*, IEEE Trans. Systems Man Cybemet 2 (1972) 30-34.

[17] M. Chen, C. Wu, X. Xue, G. Liu, *On fuzzy boundary value problems*, Information Sciences 178 (2008) 1877-1892.

[18] P. Diamond, P. Kloeden, *Metric spaces of fuzzy sets*, World scientific, Singapore, (1994).

[19] D. Dubois, H. Prade, *Towards fuzzy differential calculus: Part3, differentiation*, Fuzzy Sets and Systems 8 (1982) 225-233.

[20] W. Fei, *Existence and uniqueness of solution for fuzzy random differential equations with non-lipschitz coefficients*, Information Sciences 177 (2007) 4329-4337.

[21] R. Goetschel, W. Voxman, *Elementary fuzzy calculus*, Fuzzy Sets and Systems 18 (1986) 31-43.

[22] D. Gottlieb, S.A. Orszag, *Numerical analysis of spectral methods: theory and applications*, CBMS-NSF Regional Conference Series in Applied Mathematics 26, SIAM, Philadelphia, (1977).

[23] M. T. Hagan, H. B. Demuth, M. Beale, *Neural Network Design*, PWS publishing company, Massachusetts, (1996).

[24] Y. Hayashi, J. J. Buckley, E. Czogala, *Fuzzy neural network with fuzzy signals and weights*, Internat. J. Intelligent Systems 8 (1993) 527-537.

[25] S. Haykin, *Neural Networks:A Comprehensive Foundation*, Prentice Hall, New Jersey, (1999).

[26] K. Hornick, M. Stinchcombe, H. White, *Multilayer feedforward networks are universal approximators*, Neural Networks 2 (1989) 359-366.

[27] H. Ishibuchi, K. Kwon, H. Tanaka, *A learning algorithm of fuzzy neural networks with triangular fuzzy weights*, Fuzzy Sets and Systems 71 (1995) 277-293.

[28] H. Ishibuchi, K. Morioka, I. B. Turksen, *Learning by fuzzified neural networks*, Int. J. Approximate Reasoning 13 (1995) 327-358.

[29] H. Ishibuchi, M. Nii, *Numerical analysis of the learning of fuzzified neural networks from fuzzy if-then rules*, Fuzzy Sets and Systems 120 (2001) 281-307.

[30] H. Ishibuchi, H. Okada, H. Tanaka, *Fuzzy neural networks with fuzzy weights and fuzzy biases*, Proc. ICNN 93 (1993) 1650-1655.

[31] H. Ishibuchi, H. Tanaka, H. Okada, *Fuzzy neural networks with fuzzy weights and fuzzy biases*, Proceedings of 1993 IEEE International Conferences on Neural Networks, 1993, pp. 1650-1655.

[32] O. Kaleva, *Fuzzy differential equations*, Fuzzy Sets and Systems 24 (1987) 301-317.

[33] T. Khanna, *Foundations of Neural Networks*, Addison-Wesly, Reading, MA, (1990).

[34] P. V. Krishnamraju, J. J. Buckley, K. D. Relly, Y. Hayashi, *Genetic learning algorithms for fuzzy neural nets*, Proceedings of 1994 IEEE International Conference on Fuzzy Systems, 1994, pp. 1969-1974.

[35] J. D. Lamber, *Computational Methods in Ordinary Differential Equations*, John Wiley & Sons, New York, (1983).

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

[37] T. Leephakpreeda, *Novel determination of differential-equation solutions: universal approximation method*, Computational and Applied Mathematics 146 (2002) 443-457.

[38] A. Malek, R. Shekari Beidokhti, *Numerical solution for high order differential equations*

using a hybrid neural network-Optimization method, Appl. Math. Comput. 183 (2006) 260-271.

[39] A. J. Meade Jr, A. A. Fernandez, *The numerical solution of linear ordinary differential equations by feedforward neural networks*, Mathematical and Computer Modelling 19 (1994) 1-25.

[40] A. J. Meade Jr, A. A. Fernandez, *Solution of nonlinear ordinary differential equations by feedforward neural networks*, Mathematical and Computer Modelling 20 (1994) 19-44.

[41] M. T. Mizukoshi, L. C. Barros, Y. Chalco-Cano, H. Romn-Flores, R. C. Bassanezi, *Fuzzy differential equations and the extention principle*, Information Sciences 177 (2007) 3627-3635.

[42] M. Mosleh, M. Otadi, *Simulation and evaluation of fuzzy differential equations by fuzzy neural network*, Applied Soft Computing 12 (2012) 2817-2827.

[43] M. Mosleh, M. Otadi, *Minimal solution of fuzzy linear system of differential equations*, Neural Computing and Applications 21 (2012) 329-336.

[44] M. Mosleh, M. Otadi, S. Abbasbandy, *Evaluation of fuzzy regression models by fuzzy neural network*, Journal of Computational and Applied Mathematics 234 (2010) 825-834.

[45] M. Mosleh, M. Otadi, S. Abbasbandy, *Evaluation of fuzzy polynomial regression model by fuzzy neural network*, Applied mathematical modelling 35 (2011) 5400-5412.

[46] J. J. Nieto, A. Khastan, K. Ivaz, *Numerical solution of fuzzy differential equation under generalized differentiability*, Nonlinear Analisis: Hybrid Systems 3 (2009) 700-707.

[47] M. Otadi, M. Mosleh, *Simulation and evaluation of dual fully fuzzy linear systems by fuzzy neural network*, Applied mathematical modelling 35 (2011) 5026-5039.

[48] G. Papaschinopoulos, G. Stefanidou, P. Efraimidis, *Existence, uniquencess and asymptotic behavior of the solutions of a fuzzy differential equation with piecewise constant argument*, Information Sciences 177 (2007) 3855-3870.

[49] M. L. Puri, D. A. Ralescu, *Differentials of fuzzy functions*, J. Math. Anal. Appl. 91 (1983) 552-558.

[50] R. Rodriguez-Lopez, *Comparison results for fuzzy differential eqations*, Information Sciences 178 (2008) 1756-1779.

[51] O. Sedaghatfar, P. Darabi, S. Moloudzadeh, *A method for solving first order fuzzy differential equation*, International Journal of Industrial Mathematics 5 (2013) 251-257.

[52] S. Seikkala, *On the fuzzy initial value problem*, Fuzzy Sets and Systems 24 (1987) 319-330.

[53] R. J. Schalkoff, *Artificial Neural Networks*, McGraw-Hill, New York, (1997).

[54] S. Song, C. Wu, *Existence and uniqueness of solutions to the Cauchy problem of fuzzy differential equations*, Fuzzy Sets and Systems 110 (2000) 55-67.

[55] J. Stanley, *Introduction to Neural Networks*, third ed., Sierra Mardre, (1990).

[56] J. Store, R. Bulirsch, *Introduction to Numerical Analysis*, second ed., Springer-Verlag, New York, (1993).

[57] L. A. Zadeh, *The concept of a liguistic variable and its application to approximate reasoning*, Information Sciences 8 (1975) 199-249.

[58] L. A. Zadeh, *Is there a need for fuzzy logic?*, Information Sciences 178 (2008) 2751-2779.

Maryam Mosleh borned in 1979 in Iran. She received the B.S., M.S., and Ph.D. degrees in applied mathematics from Islamic Azad University, Iran, in 2001, 2003 and 2009, respectively. She is currently an Associate Professor in the Department of Mathematics, Firoozkooh Branch, Islamic Azad University, Firoozkooh, Iran. She is actively pursuing research in fuzzy modeling, fuzzy neural network, fuzzy linear system, fuzzy differential equations and fuzzy integral equations. She has published numerous papers in this area.