Available online at http://ijim.srbiau.ac.ir/
Int. J. Industrial Mathematics (ISSN 2008-5621)
Vol. 6, No. 2, 2014 Article ID IJIM-00371, 15 pages
Research Article

Numerical solution of hybrid fuzzy differential equations by fuzzy neural network

M. Otadi * †, M. Mosleh ‡

Abstract

The hybrid fuzzy differential equations have a wide range of applications in science and engineering. We consider the problem of finding their numerical solutions by using a novel hybrid method based on fuzzy neural network. Here neural network is considered as a part of large field called neural computing or soft computing. The proposed algorithm is illustrated by numerical examples and the results obtained using the scheme presented here agree well with the analytical solutions.

Keywords : Hybrid fuzzy differential equations; Fuzzy neural networks; Learning algorithm.

1 Introduction

The topic of Fuzzy Differential Equations (FDEs) has been rapidly growing in recent years. The concept of fuzzy derivative was first introduced by Chang and Zadeh [11], it was followed up by Dubois and Prade [13] who used the extension principle in their approach. Other methods have been discussed by Puri and Ralescu [35] and by Goetschel and Voxman [14]. Fuzzy differential equations were first formulated by Kaleva [20] and Seikkala [37] in time dependent form. Kaleva had formulated fuzzy differential equations, in terms of Hukuhara derivative [20]. Buckley and Feuring [10] 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. Also, the fuzzy initial value problem have been studied by several authors [1, 2, 6, 7, 8, 9, 12, 25, 36].

Hybrid system is a dynamic system that exhibits both continuous and discrete dynamic behavior. The hybrid systems are devoted to modeling, design, and validation of interactive systems of computer programs and continuous systems. The differential equations containing fuzzy value functions and interaction with a discrete time controller are named as hybrid fuzzy differential equations (HFDEs) [32].

In this work, we propose a new solution for the approximated solution of HFDEs using innovative mathematical tools and neural-like systems of computation based on the Hukuhara derivative. This hybrid method can result in improved numerical methods for solving HFDEs. In this proposed method, FNNM is applied as universal approximator. 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 [17, 19] for the approximated solution of HFDEs. In this paper, our fuzzy neural network is a three-layer feedforward neural network where connection weights and biases are fuzzy

*Corresponding author. otadi@iaufb.ac.ir
†Department of Mathematics, Firoozkooh Branch, Islamic Azad University, Firoozkooh, Iran.
‡Department of Mathematics, Firoozkooh Branch, Islamic Azad University, Firoozkooh, Iran.

numbers. Further, we show that the solutions obtained by fuzzy neural network is more accurate and well agree with the exact solutions. For many years this technology has been successfully applied to a wide variety of real-word applications [34]. Recently, fuzzy neural network model (FNNM) successfully used for solving fuzzy polynomial equation and systems of fuzzy polynomials [3, 4], approximate fuzzy coefficients of fuzzy regression models [27, 28, 29], approximate solution of fuzzy linear systems and fully fuzzy linear systems [30, 31] and fuzzy differential equations [26].

2 Preliminaries

Definition 2.1 *A fuzzy number u is a fuzzy subset of the real line with a normal, convex and upper semicontinuous membership function of bounded support.*

Definition 2.2 *[20] A fuzzy number u is a pair $(\underline{u}, \overline{u})$ of functions $\underline{u}(r), \overline{u}(r); 0 \leq r \leq 1$ which satisfy the following requirements:*

i. $\underline{u}(r)$ is a bounded monotonic increasing left continuous function on $(0, 1]$ and right continuous at 0.

ii. $\overline{u}(r)$ is a bounded monotonic decreasing left continuous function on $(0, 1]$ and right continuous at 0.

iii. $\underline{u}(r) \leq \overline{u}(r), 0 \leq r \leq 1$.

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

This fuzzy number space as shown in [38], can be embedded into the Banach space $B = \overline{C}[0, 1] \times \overline{C}[0, 1]$ where the metric is usually defined as

$$\|(u, v)\| = max\{sup_{0\leq r\leq 1}|u(r)|, sup_{0\leq r\leq 1}|v(r)|\},$$

for arbitrary $(u, v) \in \overline{C}[0, 1] \times \overline{C}[0, 1]$.

Before describing 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.

2.1 Operations of fuzzy numbers

We briefly mention fuzzy number operations defined by the extension principle [39]. 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:

$$\mu_{A+B}(z) = max\{\mu_A(x) \wedge \mu_B(y) | z = x+y\}, \quad (2.1)$$

$$\mu_{AB}(z) = max\{\mu_A(x) \wedge \mu_B(y) | z = xy\}, \quad (2.2)$$

$$\mu_{f(Net)}(z) = max\{\mu_{Net}(x) | z = f(x)\}, \quad (2.3)$$

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

$$[A]_h = \{x \in \mathbb{R} | \mu_A(x) \geq h\} \quad for \quad 0 < h \leq 1, \quad (2.4)$$

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

$$[A]_h = [[A]_h^L, [A]_h^U], \quad (2.5)$$

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:

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

$$\begin{aligned}[A]_h.[B]_h = [min\{&[A]_h^L.[B]_h^L, [A]_h^L.[B]_h^U,\\ &[A]_h^U.[B]_h^L, [A]_h^U.[B]_h^U\},\\ max\{&[A]_h^L.[B]_h^L, [A]_h^L.[B]_h^U,\\ &[A]_h^U.[B]_h^L, [A]_h^U.[B]_h^U\}],\end{aligned} \quad (2.7)$$

$$\begin{aligned}f([Net]_h) = f([[Net]_h^L, [Net]_h^U]) =\\ [f([Net]_h^L), f([Net]_h^U)],\end{aligned} \quad (2.8)$$

where f is increasing function. In the case of $0 \leq [B]_h^L \leq [B]_h^U$, Eq. (2.7) can be simplified as

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

2.2 Input-output relation of each unit

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 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. 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:

$$o_i = x_i, \qquad i = 1, 2, \ldots, n. \tag{2.9}$$

Hidden units:

$$Z_j = f(Net_j), \qquad j = 1, 2, \ldots, m, \tag{2.10}$$

$$Net_j = \sum_{i=1}^{n} o_i.W_{ji} + B_j. \tag{2.11}$$

Output units:

$$Y_k = f(Net_k), \qquad k = 1, 2, \ldots, s, \tag{2.12}$$

$$Net_k = \sum_{j=1}^{m} W_{kj}.Z_j + C_k, \tag{2.13}$$

where connection weights, biases, and targets are fuzzy and inputs are real numbers. The input-output relation in Eqs.(2.9)-(2.13) is defined by the extension principle.

2.3 Calculation of fuzzy output

The fuzzy output from each unit in Eqs.(2.9)-(2.13) 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:

$$o_i = x_i, \qquad i = 1, 2, \ldots, n. \tag{2.14}$$

Hidden units:

$$[Z_j]_h = f([Net_j]_h), \qquad j = 1, 2, \ldots, m, \tag{2.15}$$

$$[Net_j]_h = \sum_{i=1}^{n} o_i.[W_{ji}]_h + [B_j]_h. \tag{2.16}$$

Output unit:

$$[Y_k]_h = f([Net_k]_h), \qquad k = 1, 2, \ldots, s, \tag{2.17}$$

$$[Net_k]_h = \sum_{j=1}^{m} [W_{kj}]_h.[Z_j]_h + [C_k]_h. \tag{2.18}$$

From Eqs.(2.14)-(2.18), 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.(2.6)-(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:

$$o_i = x_i. \tag{2.19}$$

Hidden units:

$$[Z_j]_h = [[Z_j]^L_h, [Z_j]^U_h] =$$

$$[f([Net_j]^L_h), f([Net_j]^U_h)],$$

where f is increasing function.

$$[Net_j]^L_h = \sum_{i=1}^{n} o_i.[W_{ji}]^L_h + [B_j]^L_h, \tag{2.20}$$

$$[Net_j]^U_h = \sum_{i=1}^{n} o_i.[W_{ji}]^U_h + [B_j]^U_h. \tag{2.21}$$

Output units:

$$[Y_k]_h = [[Y_k]^L_h, [Y_k]^U_h] =$$

$$[f([Net_k]_h^L), f([Net_k]_h^U)], \tag{2.22}$$

where f is increasing function.

$$[Net_k]_h^L = \sum_{j\in a}[W_{kj}]_h^L.[Z_j]_h^L + \sum_{j\in b}[W_{kj}]_h^L.[Z_j]_h^U + [C_k]_h^L,$$
$$[Net_k]_h^U = \sum_{j\in c}[W_{kj}]_h^U.[Z_j]_h^U + \sum_{j\in d}[W_{kj}]_h^U.[Z_j]_h^L + [C_k]_h^U, \tag{2.23}$$

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 Hybrid fuzzy differential equations

In this paper, we will study the HFDE

$$\begin{aligned} &\tfrac{dy(x)}{dx} = f(x, y(x), \lambda_k(y_k)),\\ &y(a) = y_0, \quad x \in [x_k, x_{k+1}], \ k = 0, 1, 2, \cdots \end{aligned} \tag{3.24}$$

where y is a fuzzy function of x, y_k denotes $y(x_k)$, $f : [x_0, \infty) \times E \times E \to E$ is continuous, each $\lambda_k : E \to E$ is continuous and $\{t_k\}_{k=0}^{\infty}$ is strictly increasing and unbounded. A solution to Eq.(3.24) will be a fuzzy function $y : [x_0, \infty) \to E$ satisfying Eq.(3.24). For $k = 0, 1, 2, \cdots$, let $f_k : [x_k, x_{k+1}] \times E \to E$, where $f_k(x, y_k(x)) = f(x, y_k(x), \lambda_k(y_k))$. The HFDE (3.24) can be written in expanded form as

$$\frac{dy(x)}{dx} = \begin{cases} \frac{dy_0(x)}{dx} = f(x, y_0(x), \lambda_0(y_0)) \equiv \\ f_0(x, y_0(x)), \ x_0 \leq x \leq x_1, \\ \frac{dy_1(x)}{dx} = f(x, y_1(x), \lambda_1(y_1)) \equiv \\ f_1(x, y_1(x)), \ x_1 \leq x \leq x_2, \\ \vdots \\ \frac{dy_k(x)}{dx} = f(x, y_k(x), \lambda_k(y_k)) \equiv \\ f_k(x, y_k(x)), \ x_k \leq x \leq x_{k+1}, \\ \vdots \end{cases} \tag{3.25}$$

and a solution of (3.24) can be expressed as

$$y(x) = \begin{cases} y_0(x), \ x_0 \leq x \leq x_1, \\ y_1(x), \ x_1 \leq x \leq x_2, \\ \vdots \\ y_k(x), \ x_k \leq x \leq x_{k+1}, \\ \vdots \end{cases} \tag{3.26}$$

A solution y of (3.24) will be continuous and piecewise differentiable over $[x_0, \infty)$ and differentiable in each interval (x_k, x_{k+1}) for $k = 0, 1, 2, \cdots$.

Theorem 3.1 *[33] Consider the HFDE (3.24) expanded as (3.25) where for $k = 0, 1, 2, \cdots$, each $f_k : [x_k, x_{k+1}] \times E \to E$ is such that*

(i) $[f_k(x, y)]_h = [[f_k(x, [y]_h^L, [y]_h^U)]_h^L, [f_k(x, [y]_h^L, [y]_h^U)]_h^U]$,
(ii) $[f_k]_h^L$ and $[f_k]_h^U$ are equicontinuous and uniformly bounded on any bounded set,
(iii) There exists an $L_k > 0$ such that

$$|[f_k(x, y_1, z_1)]_h^{L,U} - [f_k(x, y_2, z_2)]_h^{L,U}| \leq L_k max\{|y_2 - y_1|, |z_2 - z_1|\}.$$

Then (3.24) and the hybrid system of ODEs

$$[\frac{dy_k(x)}{dx}]_h^L = [f_k(x, [y_k]_h^L, [y_k]_h^U)]_h^L,$$
$$[\frac{dy_k(x)}{dx}]_h^U = [f_k(x, [y_k]_h^L, [y_k]_h^U)]_h^U,$$
$$[y_k(x_k)]_h^L = [y_{k-1}(x_k)]_h^L \ if \ k > 0,$$
$$[y_0(x_0)]_h^L = [y_0]_h^L,$$
$$[y_k(x_k)]_h^U = [y_{k-1}(x_k)]_h^U \ if \ k > 0,$$
$$[y_0(x_0)]_h^U = [y_0]_h^U,$$

are equivalent.

Let us assume that a general approximation solution to Eq.(3.24) is in the form $y_{T,k}(x, P_k)$ in $[x_k, x_{k+1}], k = 0, 1, 2, \cdots$ where $y_{T,k}$ as a dependent variable to x and P_k, where P_k is an adjustable parameter involving weights and biases in the structure of the three-layered feed forward fuzzy neural network (see Fig. 1). The fuzzy trial solution y_{Tk} is an approximation solution to y_k for the optimized value of unknown weights and biases. Thus the problem of finding the approximated fuzzy solutions for Eq.(3.24) over some collocation points in $[x_k, x_{k+1}], k = 0, 1, 2, \cdots$ by a set of $M_k + 1$ regularly spaced grid points (including the endpoints) is equivalent to calculate the functional $y_{T,k}(x, P_k)$. The grid points on $[x_k, x_{k+1}]$ will be $x_{k,n} = x_k + nh_k$ where $h_k = \frac{x_{k+1}-x_k}{M_k}$ and $0 \leq n \leq M_k$. In order to obtain fuzzy approximate solution $y_{T,k}(x, P_k)$, we solve unconstrained optimization problem that is simpler to

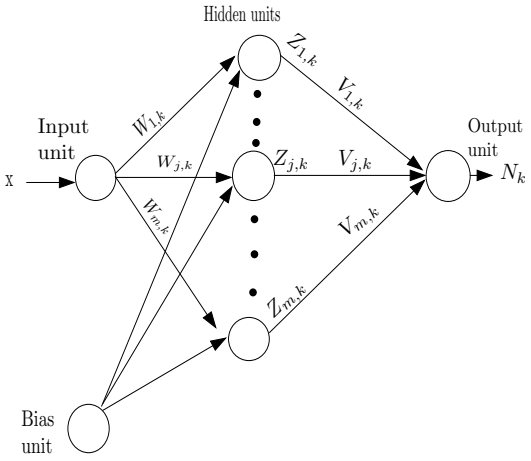

Figure 1: Three layer fuzzy neural network with one input and one output.

deal with, we define the fuzzy trial function to be in the following form:

$$y_{T,k}(x, P_k) = \alpha_k(x) + \beta_k[x, N_k(x, P_k)], \quad (3.27)$$

where the first term in the right hand side does not involve with adjustable parameters and satisfies the fuzzy initial conditions. 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 $N_k(x, P_k)$. In the next subsection, this FNNM with some weights and biases is considered and we train in order to compute the approximate solutions of problem (3.25).

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

Here, the dimension of FNNM is $1 \times m \times 1$. For every entry x the input neuron makes no changes in its input, so the input to the hidden neurons is

$$Net_{j,k} = x.W_{j,k} + B_{j,k},$$

$$j = 1, \cdots, m, \ k = 0, 1, 2, \cdots \quad (3.28)$$

where $W_{j,k}$ is a weight parameter from input layer to the jth unit in the hidden layer, $B_{j,k}$ is an jth bias for the jth unit in the hidden layer. The output, in the hidden neurons is

$$Z_{j,k} = s(Net_{j,k}),$$

$$j = 1, \cdots, m, \ k = 0, 1, 2, \cdots \quad (3.29)$$

where s is the activation function which is normally nonlinear function, the usual choices of the activation function [15] 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

$$N_k = V_{1,k}Z_{1,k} + \cdots + V_{j,k}Z_{j,k} + \cdots + V_{m,k}Z_{m,k}, \quad (3.30)$$

where $V_{j,k}$ is a weight parameter from jth unit in the hidden layer to the output layer.

From Eqs.(2.19)-(2.23), we can see that the h-level sets of the Eqs.(3.28)-(3.30) are calculated from those of the fuzzy weights, fuzzy biases and crisp inputs. For our fuzzy neural network, we can derive the learning algorithm without assuming that the input x is non-negative. For reducing the complexity of the learning algorithm, input x_0 usually assumed as non-negative in fully fuzzy neural networks, i.e., $0 \leq x_0$ [17]:
Input unit:

$$o = x. \quad (3.31)$$

Hidden units:

$$[Z_{j,k}]_h = [[Z_{j,k}]_h^L, [Z_{j,k}]_h^U] =$$

$$[s([Net_{j,k}]_h^L), s([Net_{j,k}]_h^U)],$$

$$[Net_{j,k}]_h^L = o.[W_{j,k}]_h^L + [B_{j,k}]_h^L,$$

$$[Net_{j,k}]_h^U = o.[W_{j,k}]_h^U + [B_{j,k}]_h^U.$$

Output unit:

$$[N_k]_h = [[N_k]_h^L, [N_k]_h^U], \quad (3.32)$$

$$[N_k]_h^L = \sum_{j \in a} [V_{j,k}]_h^L.[Z_{j,k}]_h^L + \sum_{j \in b} [V_{j,k}]_h^L.[Z_{j,k}]_h^U,$$

$$[N_k]_h^U = \sum_{j \in c} [V_{j,k}]_h^U.[Z_{j,k}]_h^U + \sum_{j \in d} [V_{j,k}]_h^U.[Z_{j,k}]_h^L,$$

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

A FNN_4 (fuzzy neural network with crisp set input signals, fuzzy number weights and fuzzy number output) [16] solution to Eq.(3.25) is given in Figure 1. How is the FNN_4 going to solve the fuzzy differential equations? The training data in $[x_k, x_{k+1}]$ are $x_k < x_k + h_k < \cdots < x_k + M_k h_k = x_{k+1}$ 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 (3.25), the related trial function will be in the form

$$\begin{aligned} &y_{T,0}(x,P_0) = y_0 + (x-x_0)N_0(x,P_0),\\ &x_0 \le x \le x_1,\\ &y_{T,k}(x,P_k) = y_{k-1}(x_k) + (x-x_k)N_k(x,P_k),\\ &x_k \le x \le x_{k+1},\ k=1,2,\cdots. \end{aligned} \tag{3.33}$$

In [17], 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 [3, 17, 26] for measuring the difference between B and O:

$$e = \sum_h e_h = \sum_h h.\{\sum_{k=1}^{s}([B_k]_h^L - [O_k]_h^L)^2/2 + \sum_{k=1}^{s}([B_k]_h^U - [O_k]_h^U)^2/2\}, \tag{3.34}$$

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 (3.34).

In [18], 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 (3.34). This is the squared errors for the h-level sets are weighted by h in (3.34). Krishnamraju et al. [22] used the cost function without the weighting scheme:

$$e = \sum_h e_h = \sum_h \{\sum_{k=1}^{s}([B_k]_h^L - [O_k]_h^L)^2/2 + \sum_{k=1}^{s}([B_k]_h^U - [O_k]_h^U)^2/2\}. \tag{3.35}$$

In the computer simulations included in this paper, we mainly use the cost function in (3.35) without the weighting scheme.

The error function that must be minimized for problem (3.25) is in the form

$$e_k = \sum_{n=0}^{M_k} e_{n,k} = \sum_{n=0}^{M_k}\sum_h e_{n,h,k} = \sum_{n=0}^{M_k}\sum_h \{e_{n,h,k}^L + e_{n,h,k}^U\}, \tag{3.36}$$

where

$$e_{n,h,k}^L = \frac{([\frac{dy_{T,k}(x_{k,n},P_k)}{dx}]_h^L - [f_k]_h^L)^2}{2}, \tag{3.37}$$

$$e_{n,h,k}^U = \frac{([\frac{dy_{T,k}(x_{k,n},P_k)}{dx}]_h^U - [f_k]_h^U)^2}{2}, \tag{3.38}$$

where $\{x_{k,n}\}_{n=0}^{M_k}$ are discrete points belonging to the interval $[x_k, x_{k+1}]$ and in the cost function (3.36), $e_{n,h,k}^L$ and $e_{n,h,k}^U$ can be viewed as the squared errors for the lower limits and the upper limits of the h-level sets, respectively.

It is easy to express the first derivative of $N_k(x,P_k)$ in terms of the derivative of the sigmoid function, i.e.

$$\frac{\partial[N_k]_h^L}{\partial x} = \sum_a [V_{j,k}]_h^L.\frac{\partial[Z_{j,k}]_h^L}{\partial[Net_{j,k}]_h^L}.\frac{\partial[Net_{j,k}]_h^L}{\partial x} + \sum_b [V_{j,k}]_h^L.\frac{\partial[Z_{j,k}]_h^U}{\partial[Net_{j,k}]_h^U}.\frac{\partial[Net_{j,k}]_h^U}{\partial x} \tag{3.39}$$

$$\frac{\partial[N_k]_h^U}{\partial x} = \sum_c [V_{j,k}]_h^U.\frac{\partial[Z_{j,k}]_h^U}{\partial[Net_{j,k}]_h^U}.\frac{\partial[Net_{j,k}]_h^U}{\partial x} + \sum_d [V_{j,k}]_h^U.\frac{\partial[Z_{j,k}]_h^L}{\partial[Net_{j,k}]_h^L}.\frac{\partial[Net_{j,k}]_h^L}{\partial x} \tag{3.40}$$

where $a = \{j \mid [V_{j,k}]_h^L \ge 0\}$, $b = \{j \mid [V_{j,k}]_h^L < 0\}$, $c = \{j \mid [V_{j,k}]_h^U \ge 0\}$,$d = \{j \mid [V_{j,k}]_h^U < 0\}$, $a \cup b = \{1,\ldots,m\}$ and $c \cup d = \{1,\ldots,m\}$ and

$$\frac{\partial[Net_{j,k}]_h^L}{\partial x} = [W_{j,k}]_h^L,$$

$$\frac{\partial[Z_{j,k}]_h^L}{\partial[Net_{j,k}]_h^L} = [Z_{j,k}]_h^L.(1-[Z_{j,k}]_h^L),$$

$$\frac{\partial[Net_{j,k}]_h^U}{\partial x} = [W_{j,k}]_h^U,$$

$$\frac{\partial[Z_{j,k}]_h^U}{\partial[Net_{j,k}]_h^U} = [Z_{j,k}]_h^U.(1-[Z_{j,k}]_h^U).$$

Now differentiating from trial function $y_{T,k}(x,P_k)$ in (3.37) and (3.38), we obtain

$$\frac{\partial[y_{T,k}(x,P_k)]_h^L}{\partial x} = [N_k(x,P_k)]_h^L + (x-x_k).\frac{\partial[N_k(x,P_k)]_h^L}{\partial x},$$

$$\frac{\partial[y_{T,k}(x,P_k)]_h^U}{\partial x} = [N_k(x,P_k)]_h^U + (x-x_k).\frac{\partial[N_k(x,P_k)]_h^U}{\partial x},$$

thus the expression in (3.39) and (3.40) is applicable here. A learning algorithm is derived in Appendix A.

3.1 Partially fuzzy neural networks

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 [18, 26]. 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.

The input-output relation of each unit of our partially fuzzy neural network in (3.31)-(3.32) can be rewritten for h-level sets as follows:

Input unit: Let $x \in [x_k, x_{k+1}]$

$$o = x.$$

Hidden units:

$$z_{j,k} = s(net_{j,k}), j = 1, \cdots, m, \ k = 0, 1, \cdots$$

$$net_{j,k} = o.w_{j,k} + b_{j,k}.$$

Output unit:

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

$$[\sum_{j=1}^{m} [V_{j,k}]_h^L . z_{j,k}, \sum_{j=1}^{m} [V_{j,k}]_h^U . z_{j,k}].$$

The error function that must be minimized for problem (3.25) is in the form

$$e_k = \sum_{n=0}^{M_k} e_{n,k} =$$

$$\sum_{n=0}^{M_k} \sum_{h} e_{n,h,k} = \sum_{n=0}^{M_k} \sum_{h} \{e_{n,h,k}^L + e_{n,h,k}^U\}, \quad (3.41)$$

where $\{x_{k,n}\}_{n=0}^{M_k}$ are discrete points belonging to the interval $[x_k, x_{k+1}]$ and in the cost function (3.41), $e_{n,h,k}^L$ and $e_{n,h,k}^U$ can be viewed as the squared errors for the lower limits and the upper limits of the h-level sets, respectively.

It is easy to express the first derivative of $N_k(x, P_k)$ in terms of the derivative of the sigmoid function, i.e.

$$\frac{\partial [N_k]_h^L}{\partial x} = \sum_{j=1}^{m} [V_{j,k}]_h^L . \frac{\partial z_{j,k}}{\partial net_{j,k}} . \frac{\partial net_{j,k}}{\partial x} =$$

$$\sum_{j=1}^{m} [V_{j,k}]_h^L . z_{j,k} . (1 - z_{j,k}) . w_{j,k}, \quad (3.42)$$

$$\frac{\partial [N_k]_h^U}{\partial x} = \sum_{j=1}^{m} [V_{j,k}]_h^U . \frac{\partial z_{j,k}}{\partial net_{j,k}} . \frac{\partial net_{j,k}}{\partial x} =$$

$$\sum_{j=1}^{m} [V_{j,k}]_h^U . z_{j,k} . (1 - z_{j,k}) . w_{j,k}. \quad (3.43)$$

Now differentiating from trial function $y_{T,k}(x, P_k)$, we obtain

$$\frac{\partial [y_{T,k}(x, P_k)]_h^L}{\partial x} =$$

$$[N_k(x, P_k)]_h^L + (x - x_k) . \frac{\partial [N_k(x, P_k)]_h^L}{\partial x},$$

$$\frac{\partial [y_{T,k}(x, P_k)]_h^U}{\partial x} =$$

$$[N_k(x, P_k)]_h^U + (x - x_k) . \frac{\partial [N_k(x, P_k)]_h^U}{\partial x},$$

thus the expressions in (3.42) and (3.43) are applicable here. A learning algorithm is derived in Appendix B.

Pederson and Sambandham [32, 33] numerically solved the HFDEs by using the Euler, Runge-Kutta, improved Euler, Adams-Bashforth and Adams-Moulton method. A similar example was recently considered in [21] for HFDEs using improved Predictor-corrector method. The FNNM implemented in this paper gives better approximation.

Consider the following HFDE [32]

$$\begin{cases} \frac{dy(x)}{dx} = y(x) + m(x)\lambda_k(y(x_k)), \\ [y(0)]_h = [0.75 + 0.25h, 1.125 - 0.125h], \\ x \in [x_k, x_{k+1}], \ x_k = k, \ k = 0, 1, 2, \cdots \end{cases} \quad (3.44)$$

where

$$m(x) = \begin{cases} 2(x(mod\ 1)) \ if \ x(mod\ 1) \leq 0.5, \\ 2(1 - x(mod\ 1)) \\ \qquad if \ x(mod\ 1) > 0.5, \end{cases}$$

$$\lambda_k(\mu) = \begin{cases} \hat{0} & if \ k = 0, \\ \mu & if \ k \in \{1, 2, \cdots\}, \end{cases}$$

$$\hat{0}(x) = \begin{cases} 1 & if \ x = 0, \\ 0 & if \ x \neq 0. \end{cases}$$

By applying Example 6.1 0f Kaleva [20], (3.44) has a unique solution. By [23] for $x \in [0,1]$ the exact solution of (3.44) is given by

$$[y(x)]_h =$$

$$[(0.75+0.25h)e^x, (1.125-0.125h)e^x].$$

By [32] for $x \in [1,2]$ the exact solution of (3.44) is given by

$$[y(x)]_h = \begin{cases} [y(1)]_h(3e^{x-1}-2x), \\ \\ x \in [1,1.5], \\ [y(1)]_h(2x-2+ \\ \\ e^{x-1.5}(3\sqrt{e}-4)), x \in [1.5,2]. \end{cases}$$

Using the FNNM developed in Section 3, we will solve the HFDE (3.44) to obtain a numerical solution. Suppose $k=0$ and $x \in [x_0, x_1]$. Then (3.44) is equivalent to

$$\begin{cases} \frac{dy(x)}{dx} = y(x), \\ [y(0)]_h = [0.75+0.25h, 1.125-0.125h]. \end{cases} \tag{3.45}$$

The fuzzy trial function for this problem is

$$y_{T,0}(x,P_0) = (0.75,1,1.125) + x.N_0(x,P_0).$$

Here, the dimension of PFNNM is $1 \times 5 \times 1$. In the computer simulation of this section, we use the following specifications of the learning algorithm:

(1) $M_0 = 5$.

(2) Number of hidden units: five units.

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

(4) Learning constant: $\eta_0 = 0.3$.

(5) Momentum constant: $\alpha_0 = 0.2$.

() h=0,0.2,. . . ,h=1.

(7) Initial value of the weights and biases of PFNNM are shown in Table 1, that we suppose $V_{i,0} = (v_{i,0}^{(1)}, v_{i,0}^{(2)}, v_{i,0}^{(3)})$ for $i = 1, \ldots, 5$.

We apply the proposed method to the approximate realization of solution of problem (3.46). Analytical solution and fuzzy trial function are shown in Table 2 and Table 3 for $x = 0.1$.

Suppose $k=1$ and $x \in [x_1, x_2]$. Then (3.44) is equivalent to

$$\begin{cases} \frac{dy(x)}{dx} = y(x) + m(x)y(1), \\ [y(1)]_h = [[y(1)]_h^L, [y(1)]_h^U]. \end{cases} \tag{3.46}$$

The fuzzy trial function for this problem is

$$[y_{T,1}(x,P_1)]_h = [y(1)]_h + (x-x_1).[N_1(x,P_1)]_h.$$

Here, the dimension of PFNNM is $1 \times 5 \times 1$. In the computer simulation of this section, we use the above conditions for the learning algorithm. Analytical solution and fuzzy trial function are shown in Table 4 and Table 5 for $x = 1.1$.

4 Conclusion

Solving HFDEs by using FNNM is presented in this paper. To obtain the "Best-approximated" solution of HFDEs, the adjustable parameters of FNNM are systematically adjusted by using the learning algorithm of fuzzy weights whose input-output relations were defined by extension principle. The effectiveness of the derived learning algorithm was demonstrated by computer simulation on numerical examples. Our computer simulation in this paper were performed for three-layer feedforward neural networks using the back-propagation-type learning algorithm. 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 A. Derivation of a learning algorithm in fuzzy neural networks

Let us denote the fuzzy connection weight $V_{j,k}$ by its parameter values as $V_{j,k} = (v_{j,k}^{(1)}, \ldots, v_{j,k}^{(q)}, \ldots, v_{j,k}^{(r)})$ where $V_{j,k}$ is a weight parameter from jth unit in the hidden layer to the output layer. The amount of modification of each parameter value is written as [16, 26]

$$v_{j,k}^{(q)}(t+1) = v_{j,k}^{(q)}(t) + \triangle v_{j,k}^{(q)}(t),$$

Table 1: The initial values of weights.

i	1	2	3	4	5
$v_{i,0}^{(1)}$	-0.5	-0.5	-0.5	-0.5	-0.5
$v_{i,0}^{(2)}$	0	0	0	0	0
$v_{i,0}^{(3)}$	0.5	0.5	0.5	0.5	0.5
$w_{i,0}$	0	0	0	0	0
$b_{i,0}$	0	0	0	0	0

Table 2: Comparision of exact and approximate solution in $x = 0.1$.

Exact	PFNNM	Error
0.82887818	0.82885512	$0.2306e^{-4}$
0.88413673	0.88411123	$0.2550e^{-4}$
0.93939528	0.93932341	$0.7187e^{-4}$
0.99465382	0.99467892	$0.2510e^{-4}$
1.04991237	1.04991021	$0.2160e^{-5}$
1.10517091	1.10517223	$0.1320e^{-5}$

Table 3: Comparision of exact and approximate solution in $x = 0.1$.

Exact	PFNNM	Error
1.24331728	1.24333321	$0.1593e^{-4}$
1.21568801	1.21562378	$0.6423e^{-4}$
1.18805873	1.18802134	$0.3739e^{-4}$
1.16042946	1.16046721	$0.3775e^{-4}$
1.13280019	1.13282155	$0.2136e^{-4}$
1.10517091	1.10517335	$0.2440e^{-5}$

Table 4: Comparision of exact and approximate solution in $x = 1.1$.

Exact	PFNNM	Error
2.51338913	2.51301225	$0.37688e^{-3}$
2.68094841	2.68022015	$0.72826e^{-3}$
2.84850768	2.84843865	$0.69030e^{-3}$
3.01606696	3.01623415	$0.16719e^{-3}$
3.18362624	3.18311259	$0.51365e^{-3}$
3.35118551	3.35116745	$0.18060e^{-4}$

Table 5: Comparision of exact and approximate solution in $x = 1.1$.

Exact	PFNNM	Error
3.77008370	3.77045218	$0.36848e^{-3}$
3.68630406	3.68698257	$0.67851e^{-3}$
3.60252442	3.60298781	$0.46339e^{-3}$
3.51874479	3.51801258	$0.73221e^{-3}$
3.43496515	3.43428894	$0.67621e^{-3}$
3.35118551	3.35110245	$0.83060e^{-4}$

$$\triangle v_{j,k}^{q}(t) = -\eta_k \sum_{n=0}^{M_k} \frac{\partial e_{n,h,k}}{\partial v_{j,k}^{(q)}} + \alpha_k . \triangle v_{j,k}^{(q)}(t-1),$$

where t indexes the number of adjustments, η is a learning rate and α is a momentum term con-

stant.

Thus our problem is to calculate the derivatives $\frac{\partial e_{n,h,k}}{\partial v_{j,k}^{(q)}}$. Let us rewrite $\frac{\partial e_{n,h,k}}{\partial v_{j,k}^{(q)}}$ as follows:

$$\frac{\partial e_{n,h,k}}{\partial v_{j,k}^{(q)}} = \frac{\partial e_{n,h,k}}{\partial [V_{j,k}]_h^L}.\frac{\partial [V_{j,k}]_h^L}{\partial v_{j,k}^{(q)}} + \frac{\partial e_{n,h,k}}{\partial [V_{j,k}]_h^U}.\frac{\partial [V_{j,k}]_h^U}{\partial v_{j,k}^{(q)}}.$$

In this formulation, $\frac{\partial [V_{j,k}]_h^L}{\partial v_{j,k}^{(q)}}$ and $\frac{\partial [V_{j,k}]_h^U}{\partial v_{j,k}^{(q)}}$ are easily calculated from the membership function of the fuzzy connection weight $V_{j,k}$. For example, when the fuzzy connection weight $V_{j,k}$ is trapezoidal (i.e., $V_{j,k} = (v_{j,k}^{(1)}, v_{j,k}^{(2)}, v_{j,k}^{(3)}, v_{j,k}^{(4)})$,, $\frac{\partial [V_{j,k}]_h^L}{\partial v_{j,k}^{(q)}}$ and $\frac{\partial [V_{j,k}]_h^U}{\partial v_{j,k}^{(q)}}$ are calculated as follows:

$$\frac{\partial [V_j]_h^L}{\partial v_{j,k}^{(1)}} = 1-h, \quad \frac{\partial [V_{j,k}]_h^U}{\partial v_{j,k}^{(1)}} = 0,$$

$$\frac{\partial [V_{j,k}]_h^L}{\partial v_{j,k}^{(2)}} = h, \quad \frac{\partial [V_{j,k}]_h^U}{\partial v_{j,k}^{(2)}} = 0,$$

$$\frac{\partial [V_{j,k}]_h^L}{\partial v_{j,k}^{(3)}} = 0, \quad \frac{\partial [V_{j,k}]_h^U}{\partial v_{j,k}^{(3)}} = h,$$

$$\frac{\partial [V_{j,k}]_h^L}{\partial v_{j,k}^{(4)}} = 0, \quad \frac{\partial [V_{j,k}]_h^U}{\partial v_{j,k}^{(4)}} = 1-h.$$

These derivatives are calculated from the following relation between the h-level set of the fuzzy connection weight $V_{j,k}$ and its parameter values (see Fig. 3):

$$[V_{j,k}]_h^L = (1-h).v_{j,k}^{(1)} + h.v_{j,k}^{(2)},$$

$$[V_{j,k}]_h^U = h.v_{j,k}^{(3)} + (1-h).v_{j,k}^{(4)}.$$

When the fuzzy connection weight $V_{j,k}$ is a symmetric triangular fuzzy number, the following relations hold for its h-level set $[V_{j,k}]_h = [[V_{j,k}]_h^L, [V_{j,k}]_h^U]$:

$$[V_{j,k}]_h^L = (1-\frac{h}{2}).v_{j,k}^{(1)} + \frac{h}{2}.v_{j,k}^{(3)},$$

$$[V_{j,k}]_h^U = \frac{h}{2}.v_{j,k}^{(1)} + (1-\frac{h}{2}).v_{j,k}^{(3)}.$$

Therefore,

$$\frac{\partial [V_{j,k}]_h^L}{\partial v_{j,k}^{(1)}} = 1-\frac{h}{2}, \quad \frac{\partial [V_{j,k}]_h^U}{\partial v_{j,k}^{(1)}} = \frac{h}{2},$$

$$\frac{\partial [V_{j,k}]_h^L}{\partial v_{j,k}^{(3)}} = \frac{h}{2}, \quad \frac{\partial [V_{j,k}]_h^U}{\partial v_{j,k}^{(3)}} = 1-\frac{h}{2},$$

and $v_{j,k}^{(2)}(t+1)$ is updated by the following rule:

$$v_{j,k}^{(2)}(t+1) = \frac{v_{j,k}^{(1)}(t+1) + v_{j,k}^{(3)}(t+1)}{2}.$$

On the other hand, the derivatives $\frac{\partial e_{n,h,k}}{\partial [V_{j,k}]_h^L}$ and $\frac{\partial e_{n,h,k}}{\partial [V_{j,k}]_h^U}$ are independent of the shape of the fuzzy connection weight. They can be calculated from the cost function $e_{n,h,k}$ 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 (3.36), $\frac{\partial e_{i,h,k}}{\partial [V_{j,k}]_h^L}$ and $\frac{\partial e_{n,h,k}}{\partial [V_{j,k}]_h^U}$ are calculated as follows:

[Calculation of $\frac{\partial e_{n,h,k}}{\partial [V_{j,k}]_h^L}$]

(i) If $[V_{j,k}]_h^L \geq 0$ then

$$\frac{\partial e_{n,h,k}}{\partial [V_{j,k}]_h^L} = \delta_k^L.([Z_{j,k}]_h^L + (x_{k,n} - x_k) \frac{\partial [Z_{j,k}]_h^L}{\partial x} - \frac{\partial [f_k(x, y_{T,k}(x_i, P_k))]_h^L}{\partial [y_{T,k}(x_{k,n}, P_k)]_h^L} (x_{k,n} - x_k).[Z_{j,k}]_h^L),$$

where

$$\delta_k^L = ([\frac{dy_{T,k}(x_{k,n}, P_k)}{dx}]_h^L - [f_k]_h^L).$$

(ii) If $[V_{j,k}]_h^L < 0$ then

$$\frac{\partial e_{n,h,k}}{\partial [V_{j,k}]_h^L} = \delta_k^L.([Z_{j,k}]_h^U + (x_{k,n} - x_k). \frac{\partial [Z_{j,k}]_h^U}{\partial x} - \frac{\partial [f_k]_h^L}{\partial [y_{T,k}(x_{k,n}, P_k)]_h^L}. (x_{k,n} - x_k).[Z_{j,k}]_h^U).$$

[Calculation of $\frac{\partial e_{n,h,k}}{\partial [V_{j,k}]_h^U}$]

(i) If $[V_{j,k}]_h^U \geq 0$ then

$$\frac{\partial e_{n,h,k}}{\partial [V_{j,k}]_h^U} = \delta_k^U.([Z_{j,k}]_h^U + (x_{k,n} - x_k). \frac{\partial [Z_{j,k}]_h^U}{\partial x} - \frac{\partial [f_k(x, y_{T,k}(x_{k,n}, P_k))]_h^U}{\partial [y_{T,k}(x_{k,n}, P_k)]_h^U} (x_{k,n} - x_k).[Z_{j,k}]_h^U),$$

where

$$\delta_k^U = ([\frac{dy_{T,k}(x_{k,n}, P_k)}{dx}]_h^U - [f_k]_h^U).$$

(ii) If $[V_{j,k}]_h^U < 0$ then

$$\frac{\partial e_{n,h,k}}{\partial [V_{j,k}]_h^U} = \delta_k^U.([Z_{j,k}]_h^L + (x_{k,n} - x_k).$$

$$\frac{\partial [Z_{j,k}]_h^L}{\partial x} - \frac{\partial [f_k]_h^U}{\partial [y_{T,k}(x_{k,n}, P_k)]_h^U}.$$

$$(x_{k,n} - x_k).[Z_{j,k}]_h^L).$$

In our fuzzy neural network, the connection weights and biases to the hidden units are updated in the same manner as the parameter values of the fuzzy connection weights $V_{j,k}$ as follows:

$$w_{j,k}^{(q)}(t+1) = w_{j,k}^{(q)}(t) + \triangle w_{j,k}^{(q)}(t),$$

$$\triangle w_{j,k}^{q}(t) = -\eta \sum_{n=0}^{M_k} \frac{\partial e_{i,h,k}}{\partial w_{j,k}^{(q)}} + \alpha. \triangle w_{j,k}^{(q)}(t-1).$$

Thus our problem is to calculate the derivatives $\frac{\partial e_{i,h,k}}{\partial w_{j,k}^{(q)}}$. Let us rewrite $\frac{\partial e_{i,h,k}}{\partial w_{j,k}^{(q)}}$ as follows:

$$\frac{\partial e_{i,h,k}}{\partial w_{j,k}^{(q)}} = \frac{\partial e_{i,h,k}}{\partial [W_{j,k}]_h^L}.\frac{\partial [W_{j,k}]_h^L}{\partial w_{j,k}^{(q)}} +$$

$$\frac{\partial e_{i,h,k}}{\partial [W_{j,k}]_h^U}.\frac{\partial [W_{j,k}]_h^U}{\partial w_{j,k}^{(q)}}.$$

In this formulation, $\frac{\partial [W_{j,k}]_h^L}{\partial w_{j,k}^{(q)}}$ and $\frac{\partial [W_{j,k}]_h^U}{\partial w_{j,k}^{(q)}}$ are easily calculated from the membership function of the fuzzy connection weight $W_{j,k}$. Derivatives $\frac{\partial e_{n,h,k}}{\partial [W_{j,k}]_h^L}$ and $\frac{\partial e_{n,h,k}}{\partial [W_{j,k}]_h^U}$ can be calculated from the cost function $e_{i,h,k}$ 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 (3.36), $\frac{\partial e_{n,h,k}}{\partial [W_{j,k}]_h^L}$ is calculated as follows:

[Calculation of $\frac{\partial e_{n,h,k}}{\partial [W_{j,k}]_h^L}$]

(i) If $[V_{j,k}]_h^L \geq 0$ then

$$\frac{\partial e_{n,h,k}}{\partial [W_{j,k}]_h^L} = \delta_k^L.[[V_{j,k}]_h^L.[Z_{j,k}]_h^L.(1 - [Z_{j,k}]_h^L).$$

$$x_{n,k} + (x_{k,n} - x_k).[V_{j,k}]_h^L.[Z_{j,k}]_h^L$$
$$+(x_{k,n} - x_k).x_{k,n}.[V_{j,k}]_h^L.[Z_{j,k}]_h^L.$$

$$(1 - [Z_{j,k}]_h^L)w_{j,k} - (x_{k,n} - x_k).[V_{j,k}]_h^L.([Z_{j,k}]_h^L)^2$$

$$-2(x_{k,n} - x_k).x_{k,n}.[V_{j,k}]_h^L.([Z_{j,k}]_h^L)^2$$

$$(1 - [Z_{j,k}]_h^L)w_{j,k} - (\frac{\partial [f_k(x, y_{T,k}(x_{k,n}, P_k))]_h^L}{\partial [y_{T,k}(x_{k,n}, P_k)]_h^L}.$$

$$(x_{k,n} - x_k).[V_{j,k}]_h^L.[Z_{j,k}]_h^L.(1 - [Z_{j,k}]_h^L).x_{k,n})].$$

(ii) If $[V_{j,k}]_h^U < 0$ then

$$\frac{\partial e_{n,h,k}}{\partial [W_{j,k}]_h^L} = \delta_k^U.[[V_{j,k}]_h^U.[Z_{j,k}]_h^L.$$

$$(1 - [Z_{j,k}]_h^L).x_{k,n} + (x_{k,n} - x_k).[V_{j,k}]_h^U.$$

$$[Z_{j,k}]_h^L + (x_{k,n} - x_k).x_{k,n}.[V_{j,k}]_h^U.[Z_{j,k}]_h^L.$$

$$(1 - [Z_{j,k}]_h^L)w_{j,k} - (x_{k,n} - x_k).[V_{j,k}]_h^U.$$

$$([Z_{j,k}]_h^L)^2 - 2(x_{k,n} - x_k).x_{k,n}.[V_{j,k}]_h^U.([Z_{j,k}]_h^L)^2$$

$$(1 - [Z_{j,k}]_h^L)w_{j,k} - (\frac{\partial [f_k(x, y_{T,k}(x_{k,n}, P_k))]_h^U}{\partial [y_{T,k}(x_{k,n}, P_k)]_h^U}$$

$$(x_{k,n} - x_k).[V_{j,k}]_h^U.[Z_{j,k}]_h^L.(1 - [Z_{j,k}]_h^L).x_{k,n})].$$

In the other cases, $\frac{\partial e_{n,h,k}}{\partial [W_{j,k}]_h^U}$ and the fuzzy biases to the hidden units are updated in the same manner as the fuzzy connection weights to the hidden units and fuzzy connection to the output unit.

Appendix B. Derivation of a learning algorithm in partially fuzzy neural networks

Let us denote the fuzzy connection weight $V_{j,k}$ to the output unit by its parameter values as $V_{j,k} = (v_{j,k}^{(1)}, \ldots, v_{j,k}^{(q)}, \ldots, v_{j,k}^{(r)})$. The amount of modification of each parameter value is written as [16, 26]

$$v_{j,k}^{(q)}(t+1) = v_{j,k}^{(q)}(t) + \triangle v_{j,k}^{(q)}(t),$$

$$\triangle v_{j,k}^{q}(t) = -\eta_k \sum_{n=0}^{M_k} \frac{\partial e_{n,h,k}}{\partial v_{j,k}^{(q)}} + \alpha_k. \triangle v_{j,k}^{(q)}(t-1)$$

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 $\frac{\partial e_{n,h,k}}{\partial v_{j,k}^{(q)}}$. Let us rewrite $\frac{\partial e_{n,h,k}}{\partial v_{j,k}^{(q)}}$ as follows:

$$\frac{\partial e_{n,h,k}}{\partial v_{j,k}^{(q)}} = \frac{\partial e_{n,h,k}}{\partial [V_{j,k}]_h^L}.\frac{\partial [V_{j,k}]_h^L}{\partial v_{j,k}^{(q)}}$$

$$+\frac{\partial e_{n,h,k}}{\partial [V_{j,k}]_h^U}.\frac{\partial [V_{j,k}]_h^U}{\partial v_{j,k}^{(q)}}.$$

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

On the other hand, the derivatives $\frac{\partial e_{n,h,k}}{\partial [V_{j,k}]_h^L}$ and $\frac{\partial e_{n,h,k}}{\partial [V_{j,k}]_h^U}$ are independent of the shape of the fuzzy connection weight. They can be calculated from the cost function $e_{n,h,k}$ 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), $\frac{\partial e_{n,h,k}}{\partial [V_{j,k}]_h^L}$ and $\frac{\partial e_{n,h,k}}{\partial [V_{j,k}]_h^U}$ are calculated as follows:

[Calculation of $\frac{\partial e_{n,h,k}}{\partial [V_{j,k}]_h^L}$]

$$\frac{\partial e_{n,h,k}}{\partial [V_{j,k}]_h^L} = \delta_k^L.[\frac{\partial [N_k(x_{k,n},P_k)]_h^L}{\partial [V_{j,k}]_h^L}$$

$$+(x_{k,n}-x_k).\frac{\partial z_{j,k}}{\partial x} - \frac{\partial [f_k]_h^L}{\partial [y_{T,k}(x_{k,n},P_k)]_h^L}$$

$$\frac{\partial [y_{T,k}(x_{k,n},P_k))]_h^L}{\partial [V_{j,k}]_h^L}],$$

where

$$\delta_k^L = ([\frac{dy_{T,k}(x_{k,n},P_k)}{dx}]_h^L -$$

$$[f_k(x_{k,n},y_{T,k}(x_{k,n},P_k))]_h^L),$$

$$\frac{\partial [y_{T,k}(x_{k,n},P_k))]_h^L}{\partial [V_{j,k}]_h^L}] = (x_{k,n}-x_k).z_{j,k},$$

$$\frac{\partial N_k(x_{k,n},P_k)]_h^L}{\partial [V_{j,k}]_h^L} = z_{j,k}.$$

[Calculation of $\frac{\partial e_{n,h,k}}{\partial [V_{j,k}]_h^U}$]

$$\frac{\partial e_{n,h,k}}{\partial [V_{j,k}]_h^U} = \delta^U.[\frac{\partial N_k(x_{k,n},P_k)]_h^U}{\partial [V_{j,k}]_h^U}$$

$$+(x_{k,n}-x_k).\frac{\partial z_{j,k}}{\partial x} - \frac{\partial [f_k]_h^U}{\partial [y_{T,k}(x_{k,n},P_k)]_h^U}$$

$$.\frac{\partial [y_{T,k}(x_{k,n},P_k))]_h^U}{\partial [V_{j,k}]_h^U}],$$

where

$$\delta_k^U = ([\frac{dy_{T,k}(x_{k,n},P_k)}{dx}]_h^U - [f_k]_h^U),$$

$$\frac{\partial [y_{T,k}(x_{k,n},P_k))]_h^U}{\partial [V_{j,k}]_h^U}] = (x_{k,n}-x_k).z_{j,k},$$

$$\frac{\partial N_k(x_{k,n},P_k)]_h^U}{\partial [V_{j,k}]_h^U} = z_{j,k}.$$

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

$$w_{j,k}(t+1) = w_{j,k}(t) + \triangle w_{j,k}(t),$$

$$\triangle w_{j,k}(t) = -\eta \sum_{n=0}^{M_k} \frac{\partial e_{n,h,k}}{\partial w_{j,k}} + \alpha \triangle w_{j,k}(t-1).$$

The derivative $\frac{\partial e_{n,h,k}}{\partial w_{j,k}}$ can be calculated from the cost function $e_{n,h,k}$ using the input-output relation of our partially fuzzy neural network for the h-level sets. When we use the cost function with the weighting scheme in (3.36), $\frac{\partial e_{n,h,k}}{\partial w_{j,k}}$ is calculated as follows:

$$\frac{\partial e_{n,h,k}}{\partial w_{j,k}} = \delta_k^L.[\frac{\partial [N_k(x_{k,n},P_k)]_h^L}{\partial w_{j,k}} + (x_{k,n}-x_k)$$

$$.[V_{j,k}]_h^L.z_{j,k} + (x_{k,n}-x_k).x_{k,n}.[V_{j,k}]_h^L.$$

$$z_{j,k}(1-z_{j,k})w_{j,k}$$

$$-(x_{k,n}-x_k).[V_{j,k}]_h^L.z_{j,k}^2 - 2(x_{k,n}-x_k).x_{k,n}$$

$$[V_{j,k}]_h^L.z_{j,k}^2(1-z_{j,k})w_{j,k} -$$

$$(\frac{\partial [f_k]_h^L}{\partial [y_{T,k}(x_{k,n},P_)]_h^L}\frac{\partial [y_{T,k}(x_{k,n},P_k)]_h^L}{\partial w_{j,k}} +$$

$$\frac{\partial [f_k]_h^L}{\partial [y_{T,k}(x,P_k))]_h^U}\frac{\partial [y_{T,k}(x_{k,n},P_k)]_h^U}{\partial w_{j,k}})] +$$

$$\delta_k^U.[\frac{\partial N_k(x_{k,n},P_k)]_h^U}{\partial w_{j,k}} + (x_{k,n}-x_k).[V_{j,k}]_h^U$$

$$z_{j,k} + (x_{k,n}-x_k)x_{k,n}[V_{j,k}]_h^U.z_{j,k}$$

$$(1-z_{j,k})w_{j,k} - (x_{k,n}-x_k).[V_{j,k}]_h^U.$$

$$z_{j,k}^2 - 2(x_{k,n} - x_k)$$

$$x_{k,n}.[V_{j,k}]_h^U.z_{j,k}^2(1 - z_{j,k})w_{j,k} -$$

$$(\frac{\partial[f_k]_h^U}{\partial[y_{T,k}(x_{,k,n}P_k))]_h^L}.\frac{\partial[y_{T,k}(x_{k,n}, P_k)]_h^L}{\partial w_{j,k}}$$

$$+\frac{\partial[f_k(x_{k,n}, y_{T,k}(x_{k,n}, P_k))]_h^U}{\partial[y_{T,k}(x_{k,n}, P_k))]_h^U}$$

$$.\frac{\partial[y_{T,k}(x_{k,n}, P_k)]_h^U}{\partial w_{jk,}})],$$

where

$$\frac{\partial[N_k(x_{k,n}, P_k)]_h^L}{\partial w_{j,k}} = \frac{\partial[N_k(x_{k,n}, P_k)]_h^L}{\partial z_{j,k}}$$

$$\frac{\partial z_{j,k}}{\partial net_{j,k}}\frac{\partial net_{j,k}}{\partial w_{j,k}} = [V_{j,k}]_h^L$$

$$z_{j,k}(1 - z_{j,k})x_{k,n},$$

$$\frac{\partial[N_k(x_{k,n}, P_k)]_h^U}{\partial w_{j,k}} = \frac{\partial[N_k(x_{k,n}, P_k)]_h^U}{\partial z_{j,k}}$$

$$\frac{\partial z_{j,k}}{\partial net_{j,k}}\frac{\partial net_{j,k}}{\partial w_{j,k}} = [V_{j,k}]_h^U z_{j,k}(1 - z_{j,k})x_{k,n},$$

$$\frac{\partial[y_{T,k}(x_{k,n}, P_k))]_h^L}{\partial w_{j,k}} = (x_{k,n} - x_k)$$

$$\frac{\partial[N_k(x_{k,n}, P_k)]_h^L}{\partial w_{j,k}},$$

$$\frac{\partial[y_{T,k}(x_{k,n}, P_k))]_h^U}{\partial w_{j,k}} = (x_{k,n} - x_k)$$

$$\frac{\partial[N_k(x_{k,n}, P_k)]_h^U}{\partial w_{j,k}}.$$

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 and 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 and J. J. Nieto, *Numerical methods for fuzzy differential inclusions*, Computers & mathematics with applications 48 (2004) 1633-1641.

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

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

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

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

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

[8] T. Allahviranloo, N. A. Kiani, N. Motamedi, *Solving fuzzy differential equations by differential transformation method*, Information Sciences 179 (2009) 956-966.

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

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

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

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

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

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

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

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

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

[18] 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.

[19] 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.

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

[21] H. Kim and R. Sakthivel, *Numerical solution of hybrid fuzzy differential equations using improved predictor-corrector method*, Commun nonlinear Sci. numer.v simulat 17 (2012) 3788-3794.

[22] 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.

[23] M. Ma, M. Friedman and A. Kandel, *Numerical solutions of fuzzy differential equations*, Fuzzy Sets and Systems 105 (1999) 133-138.

[24] 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.

[25] M. Mosleh, *Numerical solution of fuzzy differential equations under generalized differentiability by fuzzy neural network*, Int. J. Industrial Mathematics 5 (2013) 281-297.

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

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

[28] M. Mosleh, M. Otadi and S. Abbasbandy, *Fuzzy polynomial regression with fuzzy neural networks*, Applied Mathematical Modelling 35 (2011) 5400-5412.

[29] M. Mosleh, T. Allahviranloo and M. Otadi, *Evaluation of fully fuzzy regression models by fuzzy neural network*, Neural Comput and Applications 21 (2012) 105-112.

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

[31] M. Otadi, M. Mosleh and S. Abbasbandy, *Numerical solution of fully fuzzy linear systems by fuzzy neural network*, Soft Computing 15 (2011) 1513-1522.

[32] S. Pederson, M. Sambandham, *Numerical solution to hybrid fuzzy systems*, Mathematical and Computer Modelling 45 (2007) 1133-1144.

[33] S. Pederson, M. Sambandham, *Numerical solution of hybrid fuzzy differential equation IVPs by a characterization theorem*, Inf. Sci. 179 (2009) 319-328.

[34] P. Picton, *Neural Networks*, second ed., Palgrave, Great Britain, 2000.

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

[36] O. Sedaghatfar, P. Darabi and S. Moloudzadeh, *A method for solving first order fuzzy differential equation*, Int. J. Industrial Mathematics 5 (2013) 251-257.

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

[38] Wu Congxin, Ma Ming, *On embedding problem of fuzzy number space*, Fuzzy Sets and Systems 44 (1991) 33-38.

[39] L. A. Zadeh, *The concept of a liguistic variable and its application to approximate reasoning: Parts 1-3*, Inform. Sci. 9 (1975) 43-80.

Mahmood Otadi borned in 1978 in Iran. He received the B.S., M.S., and Ph.D. degrees in applied mathematics from Islamic Azad University, Iran, in 2001, 2003 and 2008, respectively. He is currently an Associate Professor in the Department of Mathematics, Firoozkooh Branch, Islamic Azad University, Firoozkooh, Iran. He is actively pursuing research in fuzzy modeling, fuzzy neural network, fuzzy linear system, fuzzy differential equations and fuzzy integral equations. He has published numerous papers in this area.

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.