New Artificial Intelligence Approach for Solving Fuzzy Polynomial Equations

Ahmad Jafarian

Department of Mathematics
Urmia Branch, Islamic Azad University
Urmia, Iran
jafarian5594@yahoo.com

ABSTRACT

Recently, there has been a considerable amount of interest and practice in solving many problems of several applied fields by fuzzy polynomials. In this paper, we intend to offer a new method for finding a solution of fully fuzzy polynomial with degree n, by using an artificial fuzzified feed-forward neural network. This neural net has the ability to get fuzzy vector as an input, and calculates its corresponding fuzzy output. It is clear that the input-output relation for each unit of fuzzy neural network is defined by the extension principle of Zadeh. In this work, a cost function is also defined for the level sets of fuzzy output and fuzzy target. Then we derive a learning algorithm from the cost function for adjusting three parameters of each triangular fuzzy weight. Consequently, our approach is illustrated by computer simulations on numerical examples. It is worthwhile to mention that, the application of this method in fluid mechanics has been shown by an example.

Keywords: Fuzzy polynomials, Fuzzy feed-forward neural networks, Learning algorithm, Cost function, Fuzzy solution.

2000 Mathematics Subject Classification: 92B20,82C32.
Computing Classification System: 92B20,82C32.

1 Introduction

Artificial neural networks (ANNs) comprise a feature of learning both linear and nonlinear relationships directly from a data being modeled. So, ANNs have established an outstanding method in mathematic science. several successful applications of fuzzy interpolation problems, have been reported in (Caicedo Torres, Quintana and Pinzon, 2013; Joelianto, Anura and Priyanto, 2013; Martisius, Sidlauskas and Damasevicius, 2013; Purcaru, Precup, Iercan, Fedorovici and Dragan, 2013)
In (Caicedo Torres et al., 2013) the authors investigated artificial immune system for differential diagnosis of hemorrhagic fevers and in (Joelianto et al., 2013) they used Adaptive neuro fuzzy inference system (ANFIS) for Improving Transient Response of Controlled Systems. Purcaru et.al (Purcaru et al., 2013) designed a gravitational search algorithm to optimal robot path planning and MartiŽius et al (Martisius et al., 2013) used Brain-computer interface that is a kined

of artificial neural network for detaining and classification.
Fuzzy polynomials play a major role in various areas such as mathematics, engineering and social sciences. Recently various approaches for solving fuzzy polynomials have been proposed. One approach to indirect solution is using fuzzy neural networks. During the past few years, neural networks have received much attention(Han and Qiao, 2011; Melin and Castillo, 2007; Takemura and Ishii, 2011; Vassileiou, Maris, Kitikidou and Angelidis, 2012; Yoo, Park and Choi, 2007). Ishibuchi et al.(Ishibuchi, Kwon and Tanaka, 1995) defined a cost function for each pair of fuzzy output vector and its corresponding fuzzy target vector, and they have proposed a learning algorithm of fuzzy neural networks with triangular fuzzy weights. Hayashi et al.(Hayashi, Buckley and Czogala, 1993) used fuzzy delta learning rule for training fuzzy neural network with fuzzy signals and weights. Also Buckley and Eslami(Buckley and Eslami, 1997) considered neural net solutions for fuzzy problems. Linear and nonlinear fuzzy equations have been solved in(Asady, Abbasbandy and Alavi, 2005; Buckley and Qu, 1990). Some application of fuzzy polynomials have been considered by Abbasbandy and Amirfakhrian(Abbasbandy and Amirfakhrian, 2006). Also Abbasbandy and Otadi.(Abbasbandy and Otadi, 2006) have proposed an architecture of feed-forward neural network for finding solution to fuzzy polynomials, but the neural network that has been presented by Abbasbandy, was only able to find crisp solution of fuzzy polynomials, and this neural network was not able to find fuzzy solution. Jafarian et al.(Jafarian and Measoomynia, 2011) solved fuzzy polynomials by using the neural networks with a new learning algorithm. It is noted in that work(Jafarian and Measoomynia, 2011) that the presented neural network in comparison with Abbasbandy's was better in the speed of adjusting the weights, which caused the high speed of convergence. Jafarian and Jafari(Jafarian and Jafari, 2012) applied fuzzy feed-back neural network method for approximation of the crisp solution of dual fuzzy polynomials. Since the suggested neural networks by previous authors were not able to find fuzzy solution of fuzzy polynomials, we propose a fuzzified feed-forward neural network, being able to solve a fuzzy polynomial like

$$A_1 x + ... + A_n x^n = A_0, \tag{1.1}$$

where $A_0, A_1, ..., A_n$ and x are fuzzy numbers. Since neural networks possess the universal approximation capability, and also the fuzzy neural networks have the ability of approximating continuous functions, the fuzzy neural network for approximating fuzzy polynomial is a convergence method. In this paper, we propose a learning algorithm for training fuzzified feed-forward neural network with the identity activation function. This algorithm enables the fuzzy neural network to approximate a fuzzy solution of fuzzy polynomial to any desired degree of accuracy. The rest of the paper is organized as follows: In section 2, we briefly present the necessary preliminaries for defining the fuzzy arithmetic. In section 3 we define a cost function for the level sets of fuzzy output and fuzzy target then we describe how to find a fuzzy solution of the fuzzy polynomials by using feed forward fuzzy neural network. Finally, In section 4 some examples have been collected, one of which is an applied example in fluid mechanics.

2 Preliminaries

In this section the 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 $u : \mathbb{R}^1 \to I = [0, 1]$ such that

i u is upper semi-continuous .

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

iii There are real numbers b and c, $a \leq b \leq c \leq d$, for which

1. $u(x)$ is monotonically increasing on $[a, b]$,
2. $u(x)$ is monotonically decreasing on $[c, d]$,
3. $u(x) = 1, b \leq x \leq c$.

The set of all the fuzzy numbers (as given in definition 1) is denoted by E^1 (Goetschel and Voxman, 1986; Nguyen, 1978).

Definition 2.2. A fuzzy number v is a pair ($\underline{v}$,$\overline{v}$) of functions $\underline{v}(r)$ and $\overline{v}(r)$, $0 \leq r \leq 1$, which satisfy the following requirements:

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

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

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

A popular fuzzy number is the triangular fuzzy number $v = (m - \alpha, m, m + \beta) = (v_m, v_l, v_u)$ with membership function as follows:

$$\mu_v(x) = \begin{cases} \frac{x-m}{\alpha} + 1, & m - \alpha \leq x \leq m, \\ \frac{m-x}{\beta} + 1, & m \leq x \leq m + \beta, \\ 0, & otherwise, \end{cases}$$

for $\alpha, \beta > 0$ where $v_m = m - \alpha$, $v_l = m$ and $v_u = m + \beta$. Its parametric form is:

$$\underline{v}(r) = m + \alpha(r - 1),\ \overline{v}(r) = m + \beta(1 - r),\ 0 \leq r \leq 1.$$

Triangular fuzzy numbers are fuzzy numbers in LR representation where the reference functions L and R are linear (Fuller, 1995).

2.1 Operations on fuzzy numbers

We briefly mentioned fuzzy number operations that have been defined by the extension principle (Zadeh, 2005).

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

where A and B are fuzzy numbers, $\mu_*(.)$ denotes the membership function of each fuzzy number and $\wedge$ is the minimum operator.
The above operations on fuzzy numbers are numerically performed on level sets (i.e. α-cuts). For $0 \leq \alpha \leq 1$,, a α-level set of a fuzzy number A is defined as:

$[A]_\alpha = \{x |\ \mu_A(x) \geq \alpha,\ x \in \mathbb{R}\}$,
$[A]_\alpha = [[A]^l_\alpha, [A]^u_\alpha]$,

where $[A]^l_\alpha$ and $[A]^u_\alpha$ are the lower and the upper limits of the α-level set $[A]_\alpha$, respectively.

From interval arithmetic (Alefeld and Herzberger, 1983), the above operations on fuzzy numbers are written for the α-level sets as follows:

$$[A]_\alpha + [B]_\alpha = [[A]^l_\alpha, [A]^u_\alpha] + [[B]^l_\alpha, [B]^u_\alpha] = [[A]^l_\alpha + [B]^l_\alpha, [A]^u_\alpha + [B]^u_\alpha], \tag{2.1}$$

$$\begin{aligned} [A]_\alpha.[B]_\alpha &= [[A]^l_\alpha, [A]^u_\alpha].[[B]^l_\alpha, [B]^u_\alpha] \\ &= [\min\{[A]^l_\alpha.[B]^l_\alpha, [A]^l_\alpha.[B]^u_\alpha, [A]^u_\alpha.[B]^l_\alpha, [A]^u_\alpha.[B]^u_\alpha\}, \\ &\quad \max\{[A]^l_\alpha.[B]^l_\alpha, [A]^l_\alpha.[B]^u_\alpha, [A]^u_\alpha.[B]^l_\alpha, [A]^u_\alpha.[B]^u_\alpha\}]. \end{aligned} \tag{2.2}$$

In the case of

$$0 \leq [A]^l_\alpha \leq [A]^u_\alpha,$$

Eq. (2.2) can be simplified as

$$[A]_\alpha.[B]_\alpha = [\min\{[A]^l_\alpha.[B]^l_\alpha, [A]^l_\alpha.[B]^u_\alpha\}, \max\{[A]^u_\alpha.[B]^l_\alpha, [A]^u_\alpha.[B]^u_\alpha\}],$$

$$f([Net]_\alpha) = f([Net]^l_\alpha, [Net]^u_\alpha]) = [f([Net]^l_\alpha), f([Net]^u_\alpha)],$$

$$\begin{aligned} k[A]_\alpha &= k[[A]^l_\alpha, [A]^u_\alpha] = [k[A]^l_\alpha, k[A]^u_\alpha],\ \ if\ k \geq 0, \\ k[A]_\alpha &= k[[A]^l_\alpha, [A]^u_\alpha] = [k[A]^u_\alpha, k[A]^l_\alpha],\ \ if\ k < 0. \end{aligned} \tag{2.3}$$

For arbitrary $u = (\underline{u}, \overline{u})$ and $v = (\underline{v}, \overline{v})$ we define addition $(u + v)$ and multiplication by k as (Goetschel and Voxman, 1986; Nguyen, 1978):

$$\overline{(u+v)}(r) = \overline{u}(r) + \overline{v}(r),$$
$$\underline{(u+v)}(r) = \underline{u}(r) + \underline{v}(r),$$

$$\overline{(ku)}(r) = k.\overline{u}(r),\ \underline{(kv)}(r) = k.\underline{u}(r), \quad k \geq 0,$$
$$\overline{(ku)}(r) = k.\underline{u}(r),\ \underline{(kv)}(r) = k.\overline{u}(r), \quad k < 0.$$

3 Neural Network and Fuzzy equations

In this section, we will focus on some general and powerful concepts and definitions that will be used commonly in solving second kind integral and integro-differential equations.

Definition 1. An artificial neural network is a mathematical model which attempts to simulate the computational model like the network of neurons of the central nervous system.

Definition 2. A feed-forward neural network is an artificial neural network where the signal flows from input to output unit, strictly in a forward direction. This means that, there is no connections extending from output of a unit to input of a unit in the same layer or previous layers. The multi-layer feed-forward neural network or multi-layer perceptron (MLP) that had been proposed Rosenblut, is very popular and is used more than other neural network types for a wide variety of tasks.

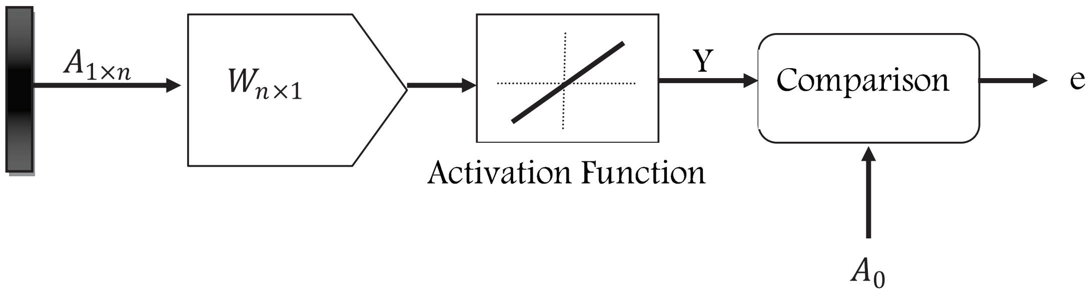

Figure 1: Schematic diagram of the proposed Fuzzy neural network to solve fuzzy polynomials.

In this part, we want to find fuzzy solution of

$$A_1x + ... + A_nx^n = A_0, \tag{3.1}$$

where $A_j, x \ \epsilon \ E^1 \ (for \ j = 0, ..., n)$. For getting an approximate solution, an architecture of FNN3 (fuzzy neural network with fuzzy input signals, fuzzy weights and fuzzy target) equivalent to Eq. (3.1) is built. Since the transfer function is identity, the input neurons make no change in their inputs, therefore the output neuron is $Y = A_1x + A_2x^2 + \ldots + A_nx^n$. For solving the fuzzy polynomials by using FNN3, we propose a learning algorithm from the cost function for adjusting weights.

3.1 Input-output relation of each unit

Let us fuzzify a two layer feed-forward neural network with n inputs and one output unit. Input vectors, target vector and connection weights are fuzzified (i.e., extended to fuzzy numbers). In order to derive a learning rule in the next section, we restrict fuzzy weights within triangular fuzzy numbers while we can use any type of fuzzy numbers for fuzzy inputs and fuzzy target. The input-output relation of each unit can be written as follows:

• Input units:

$$O_j = A_j, \quad j = 1, 2, ..., n. \tag{3.2}$$

• Output unit:

$$Y = f(Net),$$

$$Net = \sum_{j=1}^{n} w_j.O_j, \tag{3.3}$$

The relations between input neurons and output neuron in Eqs. (3.2)-(3.3) are defined by the extension principle (Zadeh, 2005) and Ishibuchi et al. (Ishibuchi, Okada and Tanaka, 1993).

3.2 Calculation of fuzzy output

The fuzzy output of neuron in the second layer is numerically calculated for fuzzy weights and level sets of fuzzy inputs. The input-output relations of the fuzzy neural network which is shown in Figure 1 can be written for the α-level sets as follows:

• Input units:

$$[O_j]_\alpha = [A_j]_\alpha, \quad j = 1, ..., n. \tag{3.4}$$

• Output unit:

$$[Y]_\alpha = f([Net]_\alpha),$$

$$[Net]_\alpha = \sum_{j=1}^{n} [w_j.O_j]_\alpha. \tag{3.5}$$

From Eqs. (3.4)-(3.5), we can see that the α-level sets of the fuzzy output Y are calculated from those of the fuzzy inputs and fuzzy weights. From Eqs. (2.1)-(2.3), the above relations are written as follows when the α-level sets of the fuzzy input A_j are nonnegative, i.e., $0 \le [A_j]_l^\alpha \le [A_j]_u^\alpha$ for all j:

• Input units:

$$[O_j]_\alpha = [[O_j]_\alpha^l, [O_j]_\alpha^u] = [[A_j]_\alpha^l, [A_j]_\alpha^u], \quad j = 1, ..., n.$$

• Output unit:

$$[Y]_\alpha = [[Y]_\alpha^l, [Y]_\alpha^u] = [f([Net]_\alpha^l), f([Net]_\alpha^u)], \tag{3.6}$$

$$[Net]_\alpha = [[Net]_\alpha^l, [Net]_\alpha^u],$$

$$[Net]_\alpha^l = \sum_{j \epsilon M} [w_j]_\alpha^l . [O_j]_\alpha^l + \sum_{j \epsilon C} [w_j]_\alpha^l . [O_j]_\alpha^u,$$

$$[Net]^u_\alpha = \sum_{j\epsilon M'} [w_j]^u_\alpha . [O_j]^u_\alpha + \sum_{j\epsilon C'} [w_j]^u_\alpha . [O_j]^l_\alpha,$$

where $M = \{j| \ [w_j]^l_\alpha \geq 0\}$, $C = \{j| \ [w_j]^l_\alpha < 0\}$, $M' = \{j| \ [w_j]^u_\alpha \geq 0\}$, $C' = \{j| \ [w_j]^u_\alpha < 0\}$, $M \cup C = \{1, ..., n\}$ and $M' \cup C' = \{1, ..., n\}$.

3.3 Cost function

Let the α-level sets of the fuzzy target output A_0 are denoted by

$$[A_0]_\alpha = [[A_0]^l_\alpha \ , \ [A_0]^u_\alpha], \quad \alpha \in [0, 1],$$

where $[A_0]^l_\alpha$ denotes the left-hand side and $[A_0]^u_\alpha$ denotes the right-hand side of the α-level sets of the desired output. A cost function to be minimized is defined for each α-level set as follows:

$$e_\alpha = e^l_\alpha + e^u_\alpha, \tag{3.7}$$

where

$$e^l_\alpha = \alpha . \frac{([A_0]^l_\alpha - [Y]^l_\alpha)^2}{2}, \tag{3.8}$$

$$e^u_\alpha = \alpha . \frac{([A_0]^u_\alpha - [Y]^u_\alpha)^2}{2}. \tag{3.9}$$

In the cost function, e^l_α and e^u_α can be viewed as the squared errors for the lower limits and the upper limits of the α-level sets of the fuzzy output Y and target output A_0, respectively. Then the total error of the given neural network is obtained as:

$$e = \sum_\alpha e_\alpha.$$

3.3.1 Learning algorithm of feed-forward fuzzy neural networks

Let us derive a learning algorithm of the fuzzy neural network from the cost function e defined for the α-level sets in the last subsection. If we modify the α-level set of a fuzzy weight independently of other level sets of that fuzzy weight in order to reduce e, this modification distorts the fuzzy weight (Ishibuchi et al., 1995). Therefore we should not change the α-level set of the fuzzy weight independently. In this paper, we update the first fuzzy weight in a manner that this fuzzy weight can move and change its width, but cannot destroy its triangular shape. Let us denote the triangular fuzzy weight W_j by its three parameters as $W_j = (w^l_j, w^c_j, w^u_j)$ where l, u, c denote the lower, center and upper limit of a triangular fuzzy number, respectively. Let us assume that the triangular fuzzy weight W_j has the symmetry property,i.e.,

$$w^c_j = \frac{w^l_j + w^u_j}{2}. \tag{3.10}$$

Our main aim is adjusting W_1 by using the learning algorithm which be introduced in below. The weight is adjusted by the following rule (Alefeld and Herzberger, 1983; Ishibuchi et al., 1995):

$$w^l_1(t + 1) = w^l_1(t) + \Delta w^l_1(t), \tag{3.11}$$

$$w_1^u(t+1) = w_1^u(t) + \Delta w_1^u(t),$$

$$\Delta w_1^l(t) = -\eta \frac{\partial e_\alpha}{\partial w_1^l} + \gamma . \Delta w_1^l(t-1), \tag{3.12}$$

$$\Delta w_1^u(t) = -\eta \frac{\partial e_\alpha}{\partial w_1^u} + \gamma . \Delta w_1^u(t-1), \tag{3.13}$$

where t is the number of adjustments, η is the learning rate and γ is the momentum term constant. The derivatives $\frac{\partial e_\alpha}{\partial w_1^l}$ and $\frac{\partial e_\alpha}{\partial w_1^u}$ can be calculated from the cost function e_α and using the input-output relation of our fuzzy neural network for the α-level sets in Eqs. (3.4)-(3.6). We calculate $\frac{\partial e_\alpha}{\partial w_1^l}$ and $\frac{\partial e_\alpha}{\partial w_1^u}$ as follows:

$$\frac{\partial e_\alpha}{\partial w_1^l} = \frac{\partial e_\alpha}{\partial [w_1]_\alpha^l} . \frac{\partial [w_1]_\alpha^l}{\partial w_1^l} + \frac{\partial e_\alpha}{\partial [w_1]_\alpha^u} . \frac{\partial [w_1]_\alpha^u}{\partial w_1^l}, \tag{3.14}$$

$$\frac{\partial e_\alpha}{\partial w_1^u} = \frac{\partial e_\alpha}{\partial [w_1]_\alpha^l} . \frac{\partial [w_1]_\alpha^l}{\partial w_1^u} + \frac{\partial e_\alpha}{\partial [w_1]_\alpha^u} . \frac{\partial [w_1]_\alpha^u}{\partial w_1^u}. \tag{3.15}$$

Since the weights are symmetric triangular fuzzy number, the following relation hold for its α-level set:

$$[w_1]_\alpha^l = w_1^l . (1 - \frac{\alpha}{2}) + w_1^u . \frac{\alpha}{2},$$

$$[w_1]_\alpha^u = w_1^l . \frac{\alpha}{2} + w_1^u . (1 - \frac{\alpha}{2}).$$

Therefore, Eq. (3.14) and Eq. (3.15) can be rewritten as

$$\frac{\partial e_\alpha}{\partial w_1^l} = \frac{\partial e_\alpha}{\partial [w_1]_\alpha^l} . (1 - \frac{\alpha}{2}) + \frac{\partial e_\alpha}{\partial [w_1]_\alpha^u} . \frac{\alpha}{2}, \tag{3.16}$$

$$\frac{\partial e_\alpha}{\partial w_1^u} = \frac{\partial e_\alpha}{\partial [w_1]_\alpha^l} . \frac{\alpha}{2} + \frac{\partial e_\alpha}{\partial [w_1]_\alpha^u} . (1 - \frac{\alpha}{2}). \tag{3.17}$$

The derivatives $\frac{\partial e_\alpha}{\partial w_1^l}$ and $\frac{\partial e_\alpha}{\partial w_1^u}$ in Eq. (3.12) and Eq. (3.13) can be calculated from the input-output relation of the fuzzy neural network, as follows:

$$\frac{\partial e_\alpha}{\partial w_1^l} = \frac{\partial e_\alpha}{\partial [Y]_\alpha^l} . \frac{\partial [Y]_\alpha^l}{\partial [Net]_\alpha^l} . \frac{\partial [Net]_\alpha^l}{\partial [w_j]_\alpha^l} . \frac{\partial [w_j]_\alpha^l}{\partial [w_1]_\alpha^l} (1 - \frac{\alpha}{2}) + \frac{\partial e_\alpha}{\partial [Y]_\alpha^u} . \frac{\partial [Y]_\alpha^u}{\partial [Net]_\alpha^u} . \frac{\partial [Net]_\alpha^u}{\partial [w_j]_\alpha^u} . \frac{\partial [w_j]_\alpha^u}{\partial [w_1]_\alpha^u} . \frac{\alpha}{2},$$

$$\frac{\partial e_\alpha}{\partial w_1^u} = \frac{\partial e_\alpha}{\partial [Y]_\alpha^l} . \frac{\partial [Y]_\alpha^l}{\partial [Net]_\alpha^l} . \frac{\partial [Net]_\alpha^l}{\partial [w_j]_\alpha^l} . \frac{\partial [w_j]_\alpha^l}{\partial [w_1]_\alpha^l} . \frac{\alpha}{2} + \frac{\partial e_\alpha}{\partial [Y]_\alpha^u} . \frac{\partial [Y]_\alpha^u}{\partial [Net]_\alpha^u} . \frac{\partial [Net]_\alpha^u}{\partial [w_j]_\alpha^u} . \frac{\partial [w_j]_\alpha^u}{\partial [w_1]_\alpha^u} . (1 - \frac{\alpha}{2}).$$

If $0 \leq [w_1]_\alpha^l \leq [w_1]_\alpha^u$, then

$$\frac{\partial e_\alpha}{\partial w_1^l} = -\alpha . ([A_0]_\alpha^l - [Y]_\alpha^l) . [A_j]_\alpha^l . j . ([w_1]_\alpha^l)^{j-1} . (1 - \frac{\alpha}{2}) - \alpha . ([A_0]_\alpha^u - [Y]_\alpha^u) . [A_j]_\alpha^u . j . ([w_1]_\alpha^u)^{j-1} . \frac{\alpha}{2},$$

$$\frac{\partial e_\alpha}{\partial w_1^u} = -\alpha . ([A_0]_\alpha^l - [Y]_\alpha^l) . [A_j]_\alpha^l . j . ([w_1]_\alpha^l)^{j-1} . \frac{\alpha}{2} - \alpha . ([A_0]_\alpha^u - [Y]_\alpha^u) . [A_j]_\alpha^u . j . ([w_1]_\alpha^u)^{j-1} . (1 - \frac{\alpha}{2}).$$

If $[w_1]_\alpha^l \leq [w_1]_\alpha^u < 0$, then

$$\frac{\partial e_\alpha}{\partial w_1^l} = -\alpha . ([A_0]_\alpha^l - [Y]_\alpha^l) . [A_j]_\alpha^u . j . ([w_1]_\alpha^l)^{j-1} . (1 - \frac{\alpha}{2}) - \alpha . ([A_0]_\alpha^u - [Y]_\alpha^u) . [A_j]_\alpha^l . j . ([w_1]_\alpha^u)^{j-1} . \frac{\alpha}{2},$$

$$\frac{\partial e_\alpha}{\partial w_1^u} = -\alpha.([A_0]_\alpha^l - [Y]_\alpha^l).[A_j]_\alpha^u.j.([w_1]_\alpha^l)^{j-1}.\frac{\alpha}{2} - \alpha.([A_0]_\alpha^u - [Y]_\alpha^u).[A_j]_\alpha^l.j.([w_1]_\alpha^u)^{j-1}.(1-\frac{\alpha}{2}).$$

If $[w_1]_\alpha^l < 0 \leq [w_1]_\alpha^u$ then

$$\frac{\partial e_\alpha}{\partial w_1^l} = -\alpha.([A_0]_\alpha^l - [Y]_\alpha^l).[A_j]_\alpha^u.j.([w_1]_\alpha^l)^{j-1}.(1-\frac{\alpha}{2}) - \alpha.([A_0]_\alpha^u - [Y]_\alpha^u).[A_j]_\alpha^u.j.([w_1]_\alpha^u)^{j-1}.\frac{\alpha}{2},$$

$$\frac{\partial e_\alpha}{\partial w_1^l} = -\alpha.([A_0]_\alpha^l - [Y]_\alpha^l).[A_j]_\alpha^u.j.([w_1]_\alpha^l)^{j-1}.\frac{\alpha}{2} - \alpha.([A_0]_\alpha^u - [Y]_\alpha^u).[A_j]_\alpha^u.j.([w_1]_\alpha^u)^{j-1}.(1-\frac{\alpha}{2}).$$

We can update other weights as W_1 adapted.

Learning algorithm

Step 1: $\eta > 0,\ \gamma > 0$ and $Emax > 0$ are chosen, where $Emax$ is stoping condition. Then the fuzzy number W_1 is initialized with a random value.

Step 2: Let $t := 0$ where t is the number of learning iterations and also the running error E is set to 0.

Step 3: Let $t := t + 1$. Repeat until *Step 5* for $\alpha = 0, 0.1, ..., 1$.

Step 4: The following procedures are calculated:

[i] Forward calculation: Calculate the α-level set of the fuzzy output Y by presenting the α-level sets of the fuzzy coefficients vector A and the fuzzy connection weights.

[ii] Back-propagation: Adjust fuzzy parameter W_1 by using the cost function for the α-level sets of the fuzzy output Y and the fuzzy target output A_0. Then update the other connection weights as has been described in Eq. (3.15).

Step 5: Update the corresponding connection weights $[W_j(t+1)]_\alpha,\ (for$j=2, ..., n).

Step 6: Cumulative cycle error is computed by adding the present error to E.

Step 7: The training cycle is completed. For $E < Emax$ terminate the training session. If $E > Emax$ then E is set to 0 and a new training cycle is initiated by going back to *Step 3*.

3.3.2 Convergence analysis

For the convergence analysis of FNN3(it is the general case of FNN) it is sufficient to show that the neural network is Universal approximators. In (Feuring, 1996) they show that these neural networks can approximate any fuzzy continues monotonic function on a compact domain to any degree of accuracy. We should mention that our definition of fuzzy function is as Feuring,(Feuring, 1996), these functions can be describe as a spatial form and it lead us to present a universal approximation theorem for these neural networks.

Theorem 1. Let U be a compact set in E, (The set of all fuzzy number) and $F(U)$ the set of all fuzzy functions on U. then for every $f \in F(U)$ and for every $\epsilon > 0$ there exists a p in all fuzzy pscudopolynoms such that

$$d(f(x), p(x)) < \epsilon, \forall x \in U. \tag{3.18}$$

Proof. See (Feuring, 1996).

The corollary of this theorem is the universal approximation of fuzzy neural network.
Monotony of fuzzy neural networks has some further advantages Fuzzy neural networks operate monotonic relative to the supports of the triangular fuzzy numbers This means that for the output $b = (b_1, ..., b_n)$ of fuzzy neural network with input data $a = (a_1, ..., a_m)$ For all input data whose supports are subsets of the support of a the support of of the corresponding output data will be subsets of the support of b This fact enables us to gain information about the reaction of the net on yet unknown input data We only have to cover the input space of the net with the supports of the data of the training set This effect can be used to diminish the risk of overtraining.(Feuring and Lippe, 1995)

4 Numerical examples

To illustrate the technique proposed in this paper, consider the following examples. All examples programs written in Matlab 2012 and run with computer CORE I7.

Example 4.1. Consider the following fuzzy equation problem:

$$(2,3,5)x + (1,2,4)x^2 + (5,6,7)x^3 + (2,3,4,6)x^4 = A_0,$$

where

$$(\underline{A}_0(r), \overline{A}_0(r)) =$$

$$(r^5 + 5r^4 + 32r^3 + 68r^2 + 239r - 201\ ,\ -2r^5 + 29r^4 - 190r^3 + 537r^2 - 871r + 29),\ \ 0 \leq r \leq 1.$$

The exact solution is $x = (-3, -2, -1)$. This problem is solved with the help of fuzzy neural network as described in this paper. Let $x_0 = (-5, -4, -3)$, $\eta = 2 \times 10^{-3}$ and $\gamma = 2 \times 10^{-3}$. Table 1 shows the approximated solution over a number of iterations and Figure 2 shows the accuracy of the solution $x_0(t)$ where t is the number of iterations, in this figure by increasing the iterations the cost function goes to zero. Figure 3 shows the convergence of the approximated solution, in this figure by increasing the iterations the calculated solution goes to exact one. Figure 4 shows the comparison between the approximate solution and the exact one.

Table 1. The approximated solutions with error analysis for Example 1

t	$x_0(t)$	e
1	(-4.4950 -3.7920 -2.8102)	6636101.47150
2	(-3.9015 -3.2645 -2.3122)	756232.856421
3	(-3.6145 -2.8099 -1.9960)	65236.0215487
4	(-3.4013 -2.5113 -1.6355)	7025.00245125
5	(-3.2003 -2.2315 -1.3221)	1535.98542120
⋮	⋮	⋮
58	(-2.9999 -2.0030 -1.0050)	0.27225465210
59	(-2.9999 -2.0025 -1.0043)	0.20835650250
60	(-2.9998 -2.0019 -1.0038)	0.15903245801
61	(-2.9998 -2.0012 -1.0030)	0.12161365420
62	(-2.9998 -2.0008 -1.0022)	0.09296002015

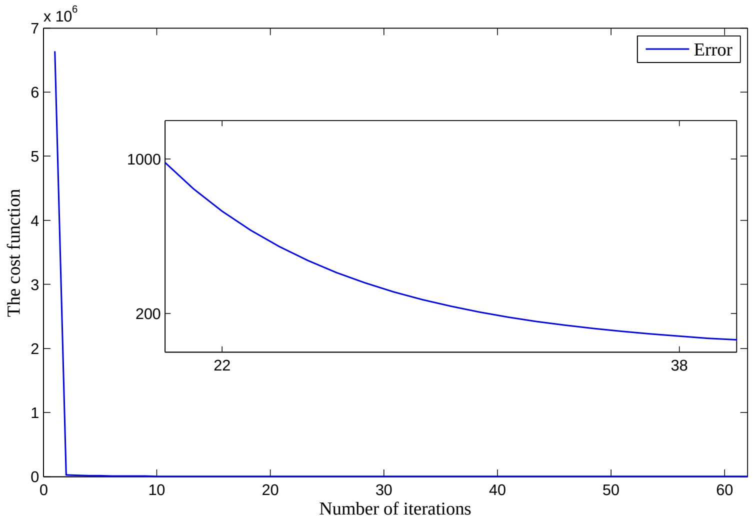

Figure 2: The cost function for Example 1 on the number of iterations.

Example 4.2. Let fuzzy equation

$$(2,4,5)x + (2,3,5)x^2 + (1,2,3,5)x^3 + (2,3,4)x^4 + (1,3,4)x^5 = A_0,$$

where

$$(\underline{A}_0(r), \overline{A}_0(r)) = (2r^6 + 52r^5 + 548r^4 + 2957r^3 + 7154r^2 + 11260r + 7560\ ,$$
$$r^6 - 40r^5 + 664r^4 - 5840r^3 + 28706r^2 - 74317r + 78827),\ \ 0 \leq r \leq 1.$$

The exact solution is $x = (5,6,7)$. Before starting calculations, we assumed that $x_0 = (3,4,5)$, $\eta = 2 \times 10^{-3}$ and $\gamma = 2 \times 10^{-3}$. Numerical result can be found in Table 2. Figure 5 shows the accuracy of the solution $x_0(t)$ where t is the number of iterations, in this figure by increasing the iterations the cost function goes to zero and Figure 6 shows the convergence of the approximated solution, in this figure by increasing the iterations the calculated solution goes to

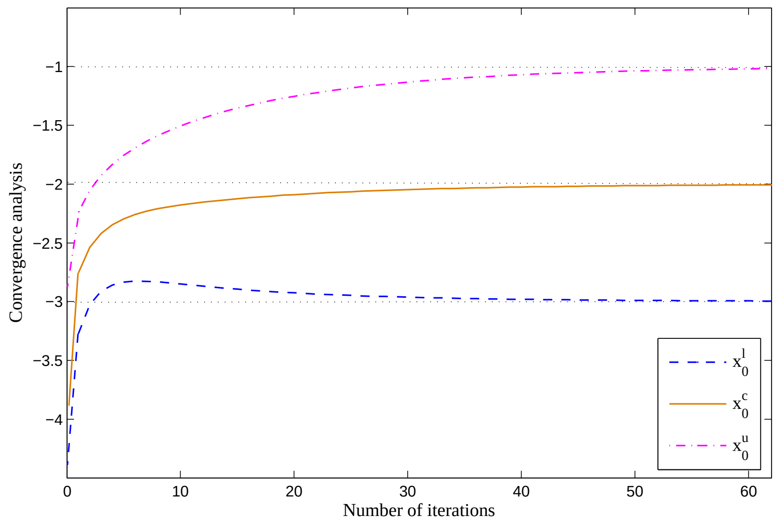

Figure 3: Convergence of the approximated solution for Example 1.

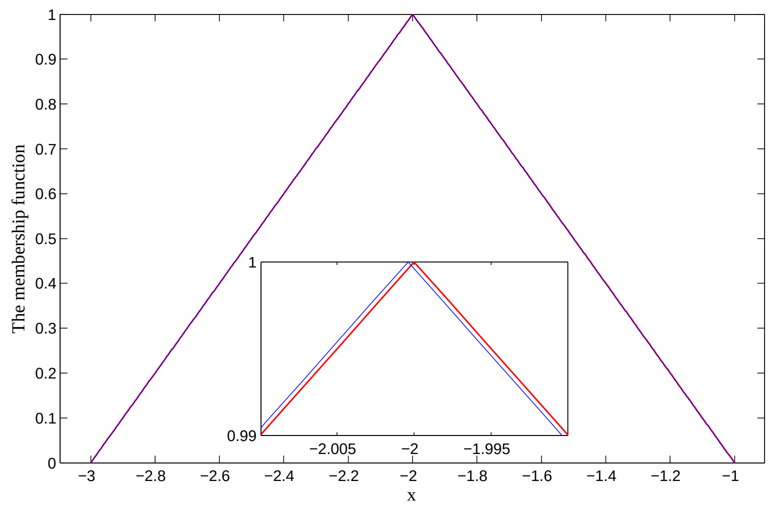

Figure 4: The comparison between approximate solution and exact one.

exact one. Figure 7 shows the comparison between the approximate solution and the exact one.

Table 2. The approximate solutions with error analysis for Example 2

t	$x_0(t)$	e
1	(3.3054 4.2941 5.3456)	3480262.39392
2	(3.6852 4.6501 5.7652)	854736.864660
3	(4.0365 4.9982 6.1002)	91020.7904408
4	(4.3521 5.3650 6.4560)	5288.59996121
5	(4.6021 5.6802 6.7250)	1007.55333026
⋮	⋮	⋮
45	(4.9995 5.9994 7.0001)	0.27986919161
46	(4.9997 5.9995 7.0001)	0.20042845406
47	(4.9998 5.9997 7.0002)	0.14353525162
48	(4.9999 5.9998 7.0002)	0.10279061631
49	(4.9999 5.9999 7.0003)	0.07361133997

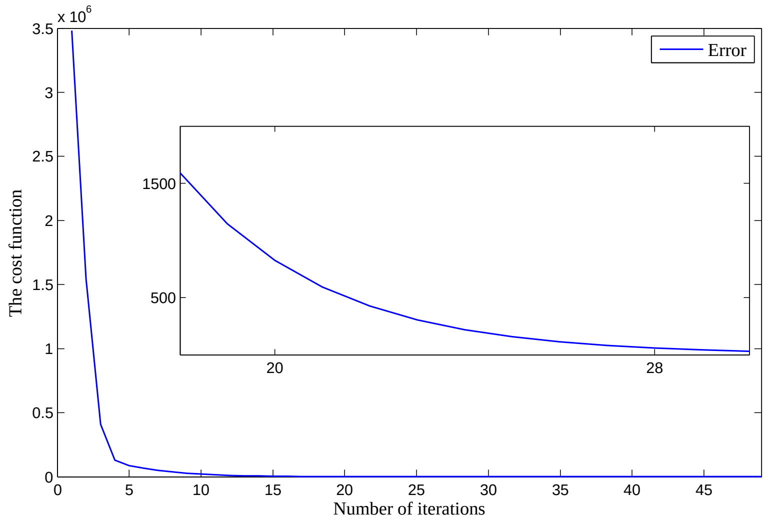

Figure 5: The cost function for Example 1 on the number of iterations.

Example 4.3. An example of fluid mechanics

A pump is in a pipeline from a suction reservoir to a junction to which 3 reservoirs are connected with pipes at a constant H, as in Figure 8, where H is the height of the pump. There is a check valve at the pump. Assume that the pump is operating with flow through the pump. The pump equation is given by

$$H = A_0 + A_1 Q + A_2 Q^2 + A_3 Q^3,$$

where A_0 ,A_1 ,A_2 ,A_3 are the characteristic coefficients of the pump.
Where

$$A_0 = (90, 100, 110), \quad A_1 = (0.1, 0.2, 0.3), \quad A_2 = (0.02, 0.03, 0.04),$$

$$A_3 = (0.006, 0.007, 0.008), \quad H = (99.768831, 175.232772, 378.97805),$$

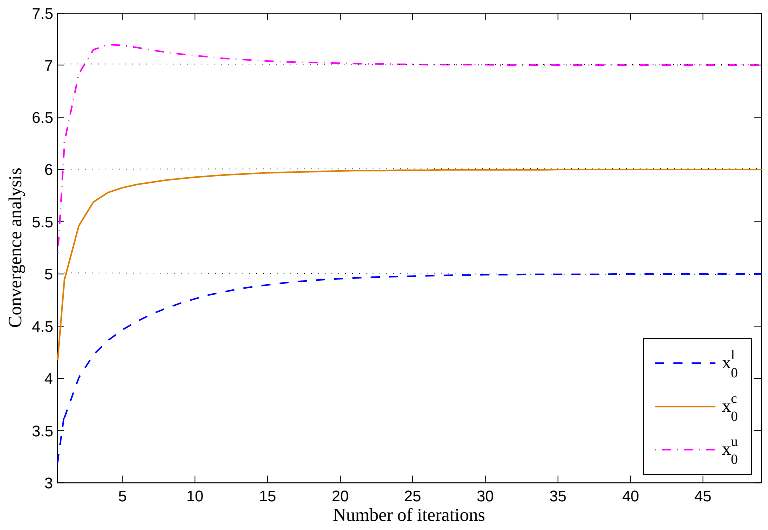

Figure 6: Convergence of the approximated solution for Example 2.

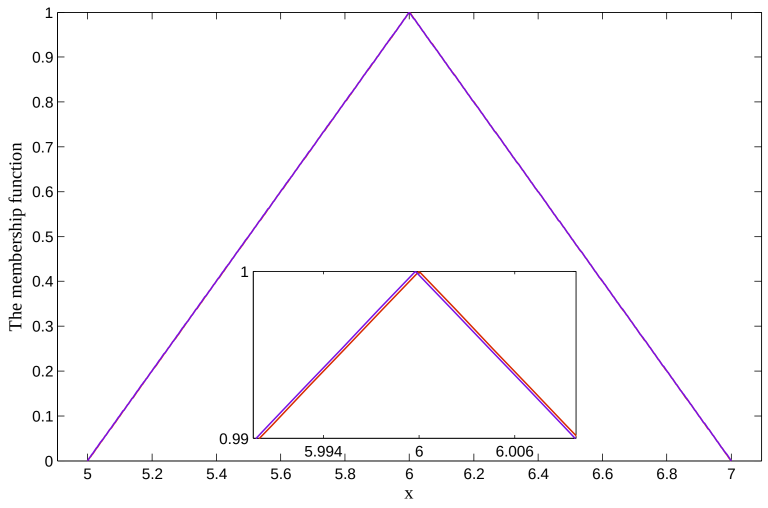

Figure 7: The comparison between approximate solution and exact one.

and the exact solution is $Q = (10.325, 20.325, 30.325)$. Before starting calculations, we assumed that $Q_0 = (6.325, 17.325, 28.325)$, $\eta = 2 \times 10^{-2}$ and $\gamma = 2 \times 10^{-2}$. Numerical result can be found in Table 3. Figure 9 shows the accuracy of the solution $Q_0(t)$ where t is the number of iterations, in this figure by increasing the iterations the cost function goes to zero.

Table 3. The approximate solutions with error analysis for Example 3

t	$Q_0(t)$	e
1	(6.95332 17.8538 28.7942)	22461.429800
2	(7.45831 18.3542 29.2925)	3633.7638100
3	(7.95332 18.8032 29.5751)	734.04810800
4	(8.26779 19.3128 29.7403)	470.69450300
5	(8.67627 19.7272 29.8017)	396.85044700
⋮	⋮	⋮
41	(10.2902 20.3126 30.3029)	0.1989814390
42	(10.2938 20.3139 30.3041)	0.1602383160
43	(10.2970 20.3150 30.3050)	0.1290331660
44	(10.2999 20.3161 30.3065)	0.1039009320
45	(10.3024 20.3170 30.3072)	0.0836608686

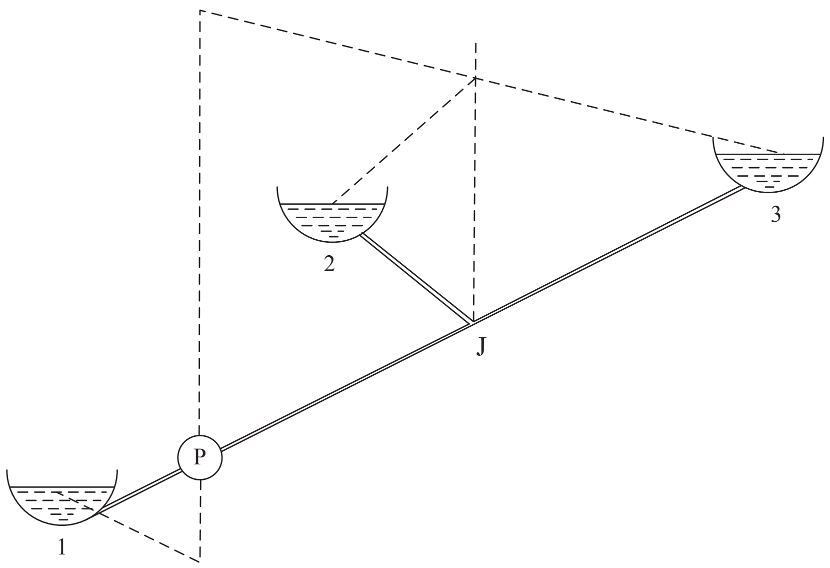

Figure 8: Pumping from one reservoir to two other reservoirs.

5 Conclusions

The topics of fuzzy neural networks which attracted growing interest for some time, have been developed in recent years. In this paper, a FFNN3 model equivalent to the fuzzy polynomial is built, and a learning algorithm of fuzzy neural network is introduced on the basis of the input-output relations defined by the extension principle, to find the fuzzy root of fuzzy polynomial. The proposed neural network is a two layer feed-forward neural network where connection weights are triangular symmetric fuzzy numbers while we can use any type of fuzzy numbers for fuzzy inputs and fuzzy target. The effectiveness of the derived learning algorithm was demonstrated by computer simulation of numerical examples. The analyzed examples illustrated the ability and reliability of the present method. The obtained solutions, in comparison with the exact solutions admitted a remarkable accuracy.

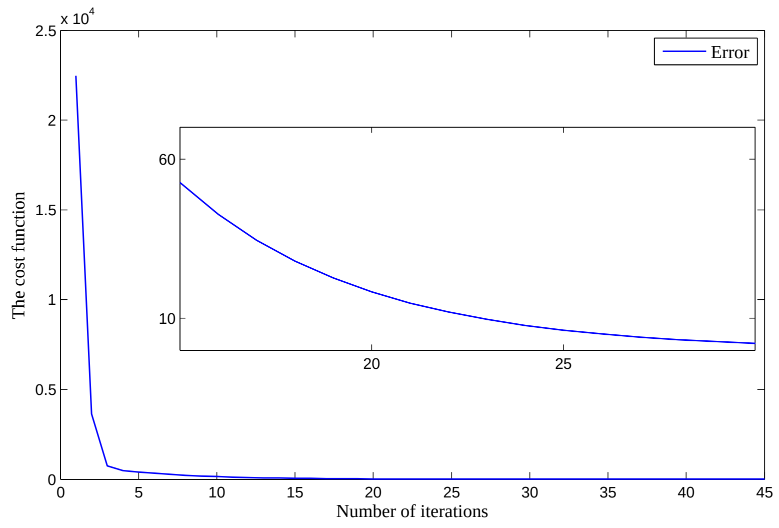

Figure 9: The cost function for Example 3 on the number of iterations.

Acknowledgment

The respected anonymous referees have carefully reviewed this paper. As a result of their careful analysis, our paper has been improved. The authors would like to express their thankfulness to them for their helpful constructive comments.

References

Abbasbandy, S. and Amirfakhrian, M.,2006. Numerical approximation of fuzzy functions by fuzzy polynomials, *Appl. Math. Comput.* **174**: 669–675.

Abbasbandy, S. and Otadi, M.,2006. Numerical approximation of fuzzy functions by fuzzy polynomials, *Appl. Math. Comput.* **181**: 1084–1089.

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

Asady, B., Abbasbandy, S., and Alavi, M.2005. Fuzzy general linear systems, *Appl. Math. Comput.* **169**: 34–40.

Buckley, J.J. and Eslami, E.,1997. Neural net solutions to fuzzy problems: The quadratic equation, *Fuzzy Sets Syst.* **86**: 289–298.

Buckley, J.J. and Qu, Y.,1990. Solving linear and quadratic fuzzy equations, *Fuzzy Sets Syst.* **35**: 43–59.

Caicedo Torres, W., Quintana, M. and Pinzon, H.,2013. Differential diagnosis of hemorrhagic fevers using ARTMAP and an artificial immune system , *Int. J. Artif. Intell.* **11**: 150–169.

Feuring , Th. and Lippe, Y.,1995. Fuzzy neural networks are universal approximators, *World congres on Artificial neural network. Sanpaulo, Brazil,***11**: 150–160.

Feuring, Th.,1996. Fuzzy Neuronale Netze Von kooperativen uber hybride zu fusionierten vage konnektionistischen Systemen, *Ph.D Thesis Westf. Wilhelms-University-Munster Germany.*

Fuller, R.,1995. Neural fuzzy systems, *Abo Akademi University, Finland.*

Joelianto, E., Anura, D. C. and Priyanto, M. P.,2013. ANFIS hybrid reference control for improving transient response of controlled systems using PID controller, *Int. J. Artif. Intell.* **10**: 88–111.

Goetschel, R. and Voxman, W.,1986. Elementary fuzzy calculus, *Fuzzy Sets Syst.* **18**: 31–34.

Han, H. and Qiao, J.,2011. An efficient algorithm for feedforward neural network reconstructing and its application, *Int. J. Artif. Intell.* **7**: 125–141.

Hayashi, Y., Buckley, J.J., and Czogala, E.,1993. Fuzzy neural network with fuzzy signals and weights, *Int. J. Intell. Syst.* **8**: 527–537.

Ishibuchi, H., Kwon, K., and Tanaka, H.,1995. A learning algorithm of fuzzy neural networks with triangular fuzzy weights, *Fuzzy Sets Syst.* **71**: 277–293.

Ishibuchi, H., Okada, H., and Tanaka, H.,1993. Fuzzy neural networks with fuzzy weights and fuzzy biases, *in: Proc, ICNN 93 (San Francisco)* **4**: 1650–1655.

Jafarian, A. and Jafari, R.,2012. Approximate solutions of dual fuzzy polynomials by feed-back neural networks, *j. of Soft Compu. and Appli.* doi:10.5899/2012/jsca-00005.

Jafarian, A., and Measoomynia, S.,2011. Solving fuzzy polynomials using neural nets with a new learning algorithm, *Aust. J. Basic and Appl. Sci.* **5**: 2295–2301.

Martisius, I., Sidlauskas, K. and Damasevicius, R.,2013. Real-time training of voted perceptron for classification of EEG data, *Int. J. Artif. Intell.* **10**: 41–50.

Melin, P., and Castillo, O.,2007. An intelligent hybrid approach for industrial quality control combining neural networks, *Inform. Sci.* **177**: 1543–1557.

Nguyen, H.T.,1978. A note on the extension principle for fuzzy sets, *J. Math. Anal. Appl.* **64**: 369–380.

Purcaru, C., Precup, R.-E., Iercan, D., Fedorovici, L.-O., David, R.-C. and Dragan, F., 2013. Optimal robot path planning using gravitational search algorithm, *Int. J. Artif. Intell.* **10**: 1–20.

Takemura, Y. and Ishii, K.,2011. Auto color calibration algorithm using neural networks and its application to RoboCup robot vision, *Int. J. Artif. Intell.* **7**: 360–367.

Vassileiou, A., Maris, F., Kitikidou, K., and Angelidis, P.,2012. Artificial neural networks for improved predictions in flow estimation, *Int. J. Artif. Intell.* **9**: 186–201.

Yoo, S.J., Park, J.B., and Choi, Y.H.,2007. Indirect adaptive control of nonlinear dynamic systems using self recurrent wavelet neural networks via adaptive learning rates, *Int. J. Artif. Intell.* **177**: 3047–3098.

Zadeh, L.A.,1975. The concept of a liguistic variable and its application to approximate reasoning, *Inform. Sci.* **8**: 199–249.

Zadeh, L.A.,2005. Toward a generalized theory of uncertainty (GTU) an outline, *Inform. Sci.* **172**: 1–40.