Approximate solutions of dual fuzzy polynomials by feed-back neural networks

A. Jafarian [1*] , R. Jafari[2]

(1) *Department of Mathematics, Urmia Branch, Islamic Azad University, Urmia, Iran*

(2) *Department of Mathematics, Science and Research of Arak Branch, Islamic Azad University, Arak, Iran*

Abstract

Recently, artificial neural networks (ANNs) have been extensively studied and used in different areas such as pattern recognition, associative memory, combinatorial optimization, etc. In this paper, we investigate the ability of fuzzy neural networks to approximate solution of a dual fuzzy polynomial of the form $a_1x + ... + a_nx^n = b_1x + ... + b_nx^n + d$, where $a_j , b_j , d \epsilon E^1 \, (for \, j = 1, ..., n)$. Since the operation of fuzzy neural networks is based on Zadeh's extension principle. For this scope we train a fuzzified neural network by back-propagation-type learning algorithm which has five layer where connection weights are crisp numbers. This neural network can get a crisp input signal and then calculates its corresponding fuzzy output. Presented method can give a real approximate solution for given polynomial by using a cost function which is defined for the level sets of fuzzy output and target output. The simulation results are presented to demonstrate the efficiency and effectiveness of the proposed approach.

Keywords: Fuzzy feed-back neural networks; Dual fuzzy polynomials; Cost function; Learning algorithm

1 Introduction

The study of dual fuzzy polynomials forms a suitable setting for mathematical modeling of real-world problems in which uncertainties or vagueness pervade. In recent years, many approaches have been utilized to the study of these polynomials. One approach is using

*Corresponding author. Email address: jafarian5594@yahoo.com

fuzzy neural networks which is proposed in this paper. This is a new and a powerful method, to obtain approximate solutions of dual fuzzy polynomials.
Ishibuchi et al. [8] proposed a learning algorithm of fuzzy neural networks with triangular fuzzy weights and Hayashi et al. [7] used fuzzy delta learning rule for training fuzzy neural network with fuzzy signals and weights. Also Buckley and Eslami [5] considered neural net solutions for fuzzy problems. The topic of numerical solution of fuzzy polynomials by fuzzy neural network investigated by Abbasbandy et al. [2], consist of finding solution to polynomials like $a_1x + ... + a_nx^n = a_0$ where $x \epsilon \mathbb{R}$ and $a_0, a_1, \ldots, a_n$ are fuzzy numbers. Jafarian et al. [10] solved fuzzy polynomials by using the neural networks with a new learning algorithm, also they proposed a feed-back neural network for solving a system of fuzzy equations in [11]. Wang et al. [15] presented an iterative algorithm for solving dual linear systems of the form x = Ax + u, where A is a real $n \times n$ matrix, the unknown x and the constant u are all vectors whose components are fuzzy numbers. Abbasbandy et al. [3] investigated the existence of a minimal solution of a general dual fuzzy linear system of the form Ax + f = Bx + c, where A and B are two real $m \times n$ matrices and the unknown x and the known f and c are vectors whose components are fuzzy numbers.
In this paper, the feed-back neural networks method, one of the most effective method, is applied to obtain the approximate solution of the dual fuzzy polynomials to any desired degree of accuracy. The proposed neural network has five layers with the identity activation function that the input-output relation of each unit is defined by the extension principle of Zadeh [16]. In computer simulations, such a fuzzified neural network is trained. The important advantage of this method is that, it is capable of greatly reducing the size of calculations while still maintaining high accuracy of the numerical solution.
The paper is organized into five section. In section 2, the necessary preliminaries for defining the fuzzy numbers are briefly presented. In section 3, a cost function is defined for the level sets of fuzzy output and fuzzy target that measures the difference between the fuzzy target output and corresponding actual fuzzy output, afterward we describe how to find a crisp solution of the dual fuzzy polynomials by using FFNs. In section 4 we report our numerical results and demonstrate the efficiency and accuracy of the proposed numerical scheme by considering some numerical examples. Finally, conclusion is summarized in section 5.

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 fuzzy numbers (as given in 2.1) is denoted by E^1. An equivalent parametric is also given in [12] as follows.

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 $v = (v_m, v_l, v_u)$ where v_m denotes the modal value and the real values $v_l \geq 0$ and $v_u \geq 0$ represent the left and right fuzziness, respectively. The membership function of a triangular fuzzy number is defined as follows:

$$\mu_v(x) = \begin{cases} \frac{x-v_m}{v_l} + 1, & v_m - v_l \leq x \leq v_m, \\ \frac{v_m-x}{v_u} + 1, & v_m \leq x \leq v_m + v_u, \\ 0, & otherwise. \end{cases}$$

Its parametric form is:

$$\underline{v}(r) = v_m + v_l(r-1),\ \overline{v}(r) = v_m + v_u(1-r),\ 0 \leq r \leq 1.$$

Triangular fuzzy numbers are fuzzy numbers in LR representation where the reference functions L and R are linear.

2.1 Operation on fuzzy numbers

We briefly mention fuzzy number operations defined by the extension principle [16, 17].

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

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

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

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

$[A]^\alpha = [[A]_l^\alpha, [A]_u^\alpha]$,

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 [4], 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

$$0 \le [A]_l^\alpha \le [A]_u^\alpha,$$

Eq. (2.2) can be simplified as

$$[A]^\alpha.[B]^\alpha = [\min\{[A]_l^\alpha.[B]_l^\alpha, [A]_l^\alpha.[B]_u^\alpha\}, \max\{[A]_u^\alpha.[B]_l^\alpha, [A]_u^\alpha.[B]_u^\alpha\}],$$

$$f([Net]^\alpha) = f([[Net]_l^\alpha, [Net]_u^\alpha]) = [f([Net]_l^\alpha), f([Net]_u^\alpha)], \tag{2.3}$$

$$k.[A]^\alpha = k.[[A]_l^\alpha, [A]_u^\alpha] = [k.[A]_l^\alpha, k.[A]_u^\alpha], \quad if\ k \ge 0,$$

$$k.[A]^\alpha = k.[[A]_l^\alpha, [A]_u^\alpha] = [k.[A]_u^\alpha, k.[A]_l^\alpha], \quad if\ k < 0. \tag{2.4}$$

For arbitrary $u = (\underline{u}, \overline{u})$ and $v = (\underline{v}, \overline{v})$ we define addition $(u + v)$ and multiplication by k as [13]:

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

$$\begin{aligned}&\overline{(ku)}(r) = k.\overline{u}(r),\ \underline{(kv)}(r) = k.\underline{u}(r), \quad if\ k \ge 0, \\ &\overline{(ku)}(r) = k.\underline{u}(r),\ \underline{(kv)}(r) = k.\overline{u}(r), \quad if\ k < 0.\end{aligned}$$

2.2 Input-output relation of each unit

Let fuzzify a five layer feed-back neural network with one input unit, 2 neurons in first hidden units, $2n$ neurons in second hidden units and one output unit. Input vector, target vector are fuzzified and weights are crisp. In order to derive a learning rule, we restrict fuzzy inputs and fuzzy target within triangular fuzzy numbers. The input-output relation of each unit of the fuzzified neural network can be written as follows:

• Input unit:

$$O = x_0. \tag{2.5}$$

• The first hidden units:

$$O_1 = f_1(net_{11}), \quad net_{11} = O, \tag{2.6}$$

and

$$O'_1 = f_2(net'_{11}), \quad net'_{11} = O. \tag{2.7}$$

• The second hidden units:

$$O_{2j} = f_{1j}(net_{2j}), \quad net_{2j} = O_1.w_j, \quad j = 1, ..., n, \tag{2.8}$$

and

$$O'_{2j} = f_{2j}(net'_{2j}), \quad net'_{2j} = O'_1.w_j, \quad j = 1, ..., n. \tag{2.9}$$

• The third hidden units:

$$O_3 = g_1(net_3), \quad net_3 = \sum_{j=1}^{n} a_j.O_{2j}, \tag{2.10}$$

and

$$O'_3 = g_2(net'_3), \quad net'_3 = \sum_{j=1}^{n} b_j.O'_{2j}. \tag{2.11}$$

• Output unit:

$$Y = f(Net), \tag{2.12}$$
$$Net = (O_3 + O'_3),$$

where $A = (a_1, ..., a_n)$ and $B = (b_1, ..., b_n)$ are fuzzy input vectors and w_j is a crisp weight. The relations between input unit and output unit in Eqs. (2.5)-(2.12) are defined by the extension principle [16] as in Hayashi et al. [7] and Ishibuchi et al. [9].

2.3 Calculation of fuzzy output

The fuzzy output from each unit in Eqs. (2.5)-(2.12) is numerically calculated for crisp weights and level sets of fuzzy inputs. The input-output relations of the neural network can be written for the α-level sets as follows:

• Input unit:

$$O = x_0. \tag{2.13}$$

• The first hidden units:

$$O_1 = f_1(net_{11}), \quad net_{11} = O, \tag{2.14}$$

and

$$O'_1 = f_2(net'_{11}), \quad net'_{11} = O. \tag{2.15}$$

• The second hidden units:

$$O_{2j} = f_{1j}(net_{2j}), \quad net_{2j} = O_1.w_j, \quad j = 1, ..., n, \tag{2.16}$$

and

$$O'_{2j} = f_{2j}(net'_{2j}), \quad net'_{2j} = O'_1.w_j, \quad j = 1, ..., n. \tag{2.17}$$

• The third hidden units:

$$[O_3]^\alpha = g_1([net_3]^\alpha), \quad [net_3]^\alpha = \sum_{j=1}^{n} a_j.O_{2j}, \tag{2.18}$$

and

$$[O\ '_3]^\alpha = g_2([net\ '_3]^\alpha), \quad [net\ '_3]^\alpha = \sum_{j=1}^{n} b_j.O\ '_{2j}. \tag{2.19}$$

• Output unit:

$$[Y]^\alpha = f([Net]^\alpha), \tag{2.20}$$
$$[Net]^\alpha = ([O_3]^\alpha + [O\ '_3]^\alpha).$$

From Eqs. (2.1)-(2.4), the above relations are written as follows when the α-level sets of the fuzzy coefficient a_j and b_j be nonnegative, i.e., $0 \leq [a_j]_l^\alpha \leq [a_j]_u^\alpha$ and $0 \leq [b_j]_l^\alpha \leq [b_j]_u^\alpha$ for all j's:

• Input unit:

$$O = x_0. \tag{2.21}$$

• The first hidden units:

$$O_1 = f_1(net_{11}), \quad net_{11} = O, \tag{2.22}$$

and

$$O\ '_1 = f_2(net\ '_{11}), \quad net\ '_{11} = O. \tag{2.23}$$

• The second hidden units:

$$O_{2j} = f_{1j}(net_{2j}), \quad net_{2j} = O_1.w_j, \quad j = 1, ..., n, \tag{2.24}$$

and

$$O\ '_{2j} = f_{2j}(net\ '_{2j}), \quad net\ '_{2j} = O\ '_1.w_j, \quad j = 1, ..., n. \tag{2.25}$$

• The third hidden units:

$$[O_3]^\alpha = [[O_3]_l^\alpha, [O_3]_u^\alpha] = g_1([[net_3]_l^\alpha, [net_3]_u^\alpha]), \tag{2.26}$$
$$g_1([[net_3]_l^\alpha, [net_3]_u^\alpha]) = [[net_3]_l^\alpha, [net_3]_u^\alpha],$$
$$[net_3]_l^\alpha = \sum_{j\epsilon M} O_{2j}.[a_j]_l^\alpha + \sum_{j\epsilon C} O_{2j}.[a_j]_u^\alpha,$$
$$[net_3]_u^\alpha = \sum_{j\epsilon M} O_{2j}.[a_j]_u^\alpha + \sum_{j\epsilon C} O_{2j}.[a_j]_l^\alpha,$$

where $M = \{O_{2j} \geq 0\}$, $C = \{O_{2j} < 0\}$ and $M \cup C = \{1, ..., n\}$,

$$[O\ '_3]^\alpha = [[O\ '_3]_l^\alpha, [O\ '_3]_u^\alpha] = g_2([[net\ '_3]_l^\alpha, [net\ '_3]_u^\alpha]), \tag{2.27}$$
$$g_2([[net\ '_3]_l^\alpha, [net\ '_3]_u^\alpha]) = [-[net\ '_3]_l^\alpha, -[net\ '_3]_u^\alpha],$$
$$[net\ '_3]_l^\alpha = \sum_{j\epsilon M} O\ '_{2j}.[b_j]_l^\alpha + \sum_{j\epsilon C} O\ '_{2j}.[b_j]_u^\alpha,$$
$$[net\ '_3]_u^\alpha = \sum_{j\epsilon M} O\ '_{2j}.[b_j]_u^\alpha + \sum_{j\epsilon C} O\ '_{2j}.[b_j]_l^\alpha,$$

where $M = \{O\ '_{2j} \geq 0\}$, $C = \{O\ '_{2j} < 0\}$ and $M \cup C = \{1, ..., n\}$.

• Output unit:

$$[Y]^{\alpha} = [[Y]_l^{\alpha}, [Y]_u^{\alpha}] = f([Net]_l^{\alpha}, [Net]_u^{\alpha}), \tag{2.28}$$
$$([Net]_l^{\alpha}, [Net]_u^{\alpha}) = ([O_3]_l^{\alpha} + [O\ '_3]_l^{\alpha}, [O_3]_u^{\alpha} + [O\ '_3]_u^{\alpha}).$$

3 Dual fuzzy polynomials

We are interested in finding the solution of dual fuzzy polynomials of the form

$$a_1 x + ... + a_n x^n = b_1 x + ... + b_n x^n + d, \tag{3.29}$$

where $a_j, b_j, d \in E^1$ $(for\ j = 1, ..., n)$. For getting an approximate solution, an architecture of FNN_2 (fuzzy neural network with fuzzy inputs, fuzzy output signal and crisp weights) equivalent to Eq. (3.29) is built. The network is shown in Fig. 1.
Usually, there is no inverse element for an arbitrary fuzzy number $u \in E^1$, i.e., there exists no element $v \in E^1$ such that

$$u + v = 0.$$

Actually, for all non-crisp fuzzy number $u \in E^1$ we have

$$u + (-u) \neq 0.$$

Therefore, the dual fuzzy polynomial in Eq. (3.29) cannot be equivalently replaced by

$$(a_1 - b_1)x + ... + (a_n - b_n)x^n = d,$$

which had been investigated.

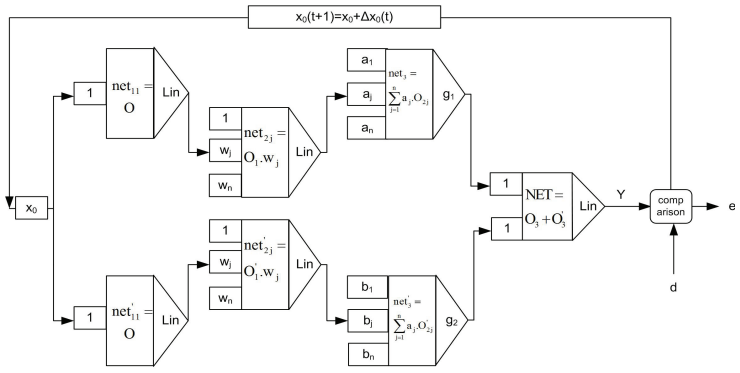

Fig. 1. Feed-back fuzzy neural network for solving dual fuzzy polynomials.

3.1 Cost function

Let d be the fuzzy target output corresponding to the fuzzy coefficient vectors (a_j, b_j). We want to introduce how to deduce a learning algorithm for training the connection weights. For this scope, we defined a cost function for α-level sets of the fuzzy output Y and the corresponding target output d as follows:

$$e^{\alpha} = e_l^{\alpha} + e_u^{\alpha}, \tag{3.30}$$

where

$$e_l^\alpha = \alpha.\frac{([d]_l^\alpha - [Y]_l^\alpha)^2}{2}, \tag{3.31}$$

$$e_u^\alpha = \alpha.\frac{([d]_u^\alpha - [Y]_u^\alpha)^2}{2}. \tag{3.32}$$

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 d, respectively. Then the total error of the given neural network is obtained as [1]:

$$e = \sum_\alpha e^\alpha. \tag{3.33}$$

Theoretically this cost function satisfies the following equation if we use infinite number of α-level sets in Eq. (3.33)

$$e \longrightarrow 0 \Longleftrightarrow [Y]^\alpha \longrightarrow [d]^\alpha.$$

*3.1.*1 Learning of fuzzy neural network

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. Our main aim is adjusting the crisp parameter x_0 by using the learning algorithm which is introduced in below. The weight is updated by the following rule [8, 14]

$$x_0(t+1) = x_0(t) + \Delta x_0(t), \tag{3.34}$$

$$\Delta x_0(t) = -\eta.\frac{\partial e^\alpha}{\partial x_0} + \gamma.\Delta x_0(t-1), \tag{3.35}$$

where t is the number of adjustments, η is the learning rate and γ is the momentum term constant. Thus our problem is to calculate the derivative $\frac{\partial e^\alpha}{\partial x_0}$ in Eq. (3.35). The derivative $\frac{\partial e^\alpha}{\partial x_0}$ can be calculated from the cost function e^α by using the input-output relation of the fuzzy neural network for the α-level sets in Eqs. (2.13)-(2.28). We calculated $\frac{\partial e^\alpha}{\partial x_0}$ as follows:

$$\frac{\partial e^\alpha}{\partial x_0} = \frac{\partial e_l^\alpha}{\partial x_0} + \frac{\partial e_u^\alpha}{\partial x_0}. \tag{3.36}$$

Thus our problem is calculating of the derivatives $\frac{\partial e_l^\alpha}{\partial x_0}$ and $\frac{\partial e_u^\alpha}{\partial x_0}$. So we have:

$$\frac{\partial e_l^\alpha}{\partial x_0} = \frac{\partial e_l^\alpha}{\partial [Y]_l^\alpha} \cdot \frac{\partial [Y]_l^\alpha}{\partial [Net]_l^\alpha} \cdot \frac{\partial [Net]_l^\alpha}{\partial [O_3]_l^\alpha} \cdot \frac{\partial [O_3]_l^\alpha}{\partial [net_3]_l^\alpha} \cdot \frac{\partial [net_3]_l^\alpha}{\partial x_0}$$

$$+\frac{\partial e_l^\alpha}{\partial [Y]_l^\alpha} \cdot \frac{\partial [Y]_l^\alpha}{\partial [Net]_l^\alpha} \cdot \frac{\partial [Net]_l^\alpha}{\partial [O\ '_3]_l^\alpha} \cdot \frac{\partial [O\ '_3]_l^\alpha}{\partial [net\ '_3]_l^\alpha} \cdot \frac{\partial [net\ '_3]_l^\alpha}{\partial x_0},$$

where

$$\frac{\partial [net_3]_l^\alpha}{\partial x_0} = \sum_{j=1}^{n} (\frac{\partial [net_3]_l^\alpha}{\partial O_{2j}} \cdot \frac{\partial O_{2j}}{\partial net_{2j}} \cdot \frac{\partial net_{2j}}{\partial O_1} \cdot \frac{\partial O_1}{\partial net_{11}} \cdot \frac{\partial net_{11}}{\partial x_0}),$$

$$\frac{\partial [net\ '_3]_l^\alpha}{\partial x_0} = \sum_{j=1}^{n} (\frac{\partial [net\ '_3]_l^\alpha}{\partial O\ '_{2j}} \cdot \frac{\partial O\ '_{2j}}{\partial net\ '_{2j}} \cdot \frac{\partial net\ '_{2j}}{\partial O\ '_1} \cdot \frac{\partial O\ '_1}{\partial net\ '_{11}} \cdot \frac{\partial net\ '_{11}}{\partial x_0}),$$

and

$$\frac{\partial e_u^\alpha}{\partial x_0} = \frac{\partial e_u^\alpha}{\partial [Y]_u^\alpha} \cdot \frac{\partial [Y]_u^\alpha}{\partial [Net]_u^\alpha} \cdot \frac{\partial [Net]_u^\alpha}{\partial [O_3]_u^\alpha} \cdot \frac{\partial [O_3]_u^\alpha}{\partial [net_3]_u^\alpha} \cdot \frac{\partial [net_3]_u^\alpha}{\partial x_0}$$
$$+ \frac{\partial e_u^\alpha}{\partial [Y]_u^\alpha} \cdot \frac{\partial [Y]_u^\alpha}{\partial [Net]_u^\alpha} \cdot \frac{\partial [Net]_u^\alpha}{\partial [O\ '_3]_u^\alpha} \cdot \frac{\partial [O\ '_3]_u^\alpha}{\partial [net\ '_3]_u^\alpha} \cdot \frac{\partial [net\ '_3]_u^\alpha}{\partial x_0},$$

where

$$\frac{\partial [net_3]_u^\alpha}{\partial x_0} = \sum_{j=1}^{n} (\frac{\partial [net_3]_u^\alpha}{\partial O_{2j}} \cdot \frac{\partial O_{2j}}{\partial net_{2j}} \cdot \frac{\partial net_{2j}}{\partial O_1} \cdot \frac{\partial O_1}{\partial net_{11}} \cdot \frac{\partial net_{11}}{\partial x_0}),$$

$$\frac{\partial [net\ '_3]_u^\alpha}{\partial x_0} = \sum_{j=1}^{n} (\frac{\partial [net\ '_3]_u^\alpha}{\partial O\ '_{2j}} \cdot \frac{\partial O\ '_{2j}}{\partial net\ '_{2j}} \cdot \frac{\partial net\ '_{2j}}{\partial O\ '_1} \cdot \frac{\partial O\ '_1}{\partial net\ '_{11}} \cdot \frac{\partial net\ '_{11}}{\partial x_0}).$$

If $O_{2j} \geq 0$ and $O\ '_{2j} \geq 0$,

$$\frac{\partial e_l^\alpha}{\partial x_0} = \sum_{j=1}^{n} \left(-\alpha.([d]_l^\alpha - [Y]_l^\alpha).[a_j]_l^\alpha.j.x_0^{j-1} + \alpha.([d]_l^\alpha - [Y]_l^\alpha).[b_j]_l^\alpha.j.x_0^{j-1} \right),$$

$$\frac{\partial e_u^\alpha}{\partial x_0} = \sum_{j=1}^{n} \left(-\alpha.([d]_u^\alpha - [Y]_u^\alpha).[a_j]_u^\alpha.j.x_0^{j-1} + \alpha.([d]_u^\alpha - [Y]_u^\alpha).[b_j]_u^\alpha.j.x_0^{j-1} \right).$$

Otherwise,

$$\frac{\partial e_l^\alpha}{\partial x_0} = \sum_{j \epsilon M} \left(-\alpha.([d]_l^\alpha - [Y]_l^\alpha).[a_j]_l^\alpha.j.x_0^{j-1} + \alpha.([d]_l^\alpha - [Y]_l^\alpha).[b_j]_l^\alpha.j.x_0^{j-1} \right)$$
$$+ \sum_{j \epsilon C} \left(-\alpha.([d]_l^\alpha - [Y]_l^\alpha).[a_j]_u^\alpha.j.x_0^{j-1} + \alpha.([d]_l^\alpha - [Y]_l^\alpha).[b_j]_u^\alpha.j.x_0^{j-1} \right),$$

$$\frac{\partial e_u^\alpha}{\partial x_0} = \sum_{j \epsilon M} \left(-\alpha.([d]_u^\alpha - [Y]_u^\alpha).[a_j]_u^\alpha.j.x_0^{j-1} + \alpha.([d]_u^\alpha - [Y]_u^\alpha).[b_j]_u^\alpha.j.x_0^{j-1} \right)$$
$$+ \sum_{j \epsilon C} \left(-\alpha.([d]_u^\alpha - [Y]_u^\alpha).[a_j]_l^\alpha.j.x_0^{j-1} + \alpha.([d]_u^\alpha - [Y]_u^\alpha).[b_j]_l^\alpha.j.x_0^{j-1} \right),$$

where $M = \{j|\ j\ be\ an\ even\ number\}$, $C = \{j|\ j\ be\ an\ odd\ number\}$ and $M \cup C = \{1, ..., n\}$.

Now we can update the connection weights w_j by using Eq. (2.4) as follows:

$$w_j = x_0^{j-1},\ j = 1, ..., n. \tag{3.37}$$

Learning algorithm

Step 1: $\eta > 0$, $\gamma > 0$, and $Emax > 0$ are chosen. Then the crisp quantity x_0 is initialized

at a small random value.
Step 2: Let $t := 0$ where t is the number of iterations of the learning algorithm. Then the running error E is set to 0.
Step 3: Calculate the corresponding connection weights to the hidden layer as $w_j(t) = x_0(t)^{j-1}$, (for $j = 1, ..., n$).
Step 4: Let $t := t + 1$. Repeat *Step 3* for $\alpha = \alpha_1, ..., \alpha_m$.
Step 5: The following procedures are calculated:

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

[ii] Back-propagation: Adjust crisp parameter x_0 by using the cost function for the α-level sets of the fuzzy output Y and the target output d then update. Then update the other connection weights as has been described in Eq. (3.37).

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 we initiate a new training cycle by going back to *Step 4*.

The following theorem illustrates the convergence properties of the neural networks.

Theorem 3.1. *If the presented fuzzy problem has solutions then the perceptron learning algorithm will find one of them.*
Proof. [6].

4 Numerical examples

To illustrate the technique proposed in this paper, consider the following examples.

Example 4.1. Consider the following dual fuzzy polynomial:

$$(2, 4, 5)x + (2, 8, 9, 12)x^2 = (1, 2, 3)x + (1, 7, 8, 10)x^2 + d,$$

where

$$(\underline{d}(r), \overline{d}(r)) = (3r + 12\ ,\ -9r + 24), \quad 0 \leq r \leq 1.$$

where the exact solution is $x = 3$. This problem is solved with the help of fuzzy neural network as described in this paper. Let $x_0 = 0.25$, $\eta = 2 \times 10^{-3}$ and $\gamma = 2 \times 10^{-3}$. Table 1 shows the approximated solution over a number of iterations and Fig. 2 shows the accuracy of the solution $x_0(t)$ where t is the number of iterations, its noticeable that by increasing the iterations the cost function goes to zero. Fig. 3 shows the convergence of the approximated solution, in this figure by increasing the iterations the calculated solution goes to exact one.

Table 1
The approximated solutions with error analysis for Example 4.1.

t	$x_0(t)$	e	t	$x_0(t)$	e
1	0.2605	3280.83271	14	2.9556	3.26918351
2	0.6026	2806.78532	15	2.9707	1.43256751
3	0.9734	2479.27170	16	2.9807	0.62312887
4	1.2653	2083.46845	17	2.9874	0.26971343
5	1.5653	1643.47612	18	2.9917	0.11636229
6	1.8568	1201.95976	19	2.9946	0.05009436

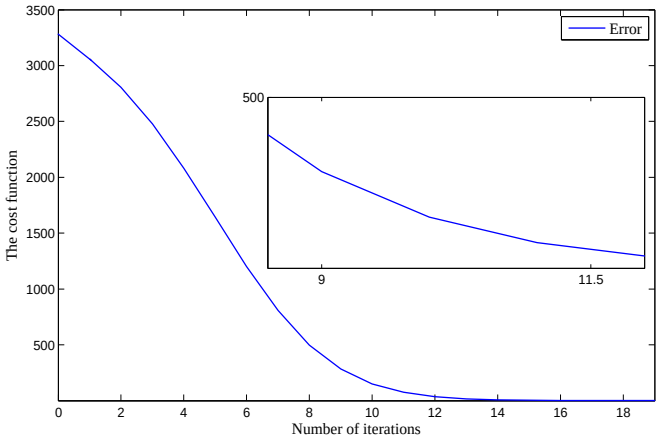

Fig. 2. The cost function for Example 4.1 over the number of iterations.

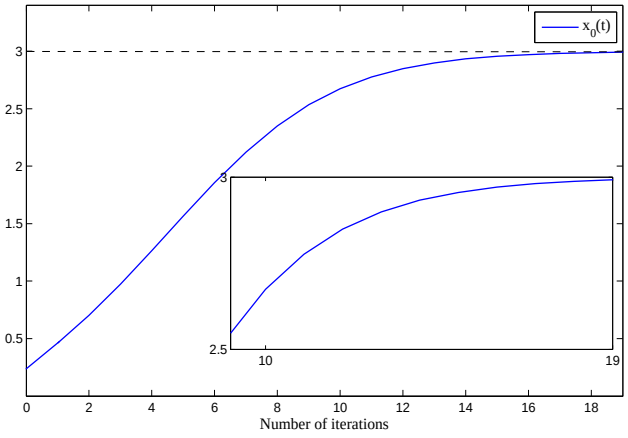

Fig. 3. Convergence of the approximated solution for Example 4.1.

Example 4.2. Let fuzzy equation

$$(7,8,10)x + (6,7,8)x^2 + (1,2,4)x^3 = (2,3,4)x + (4,6,7)x^2 + (1,2,3)x^3 + d,$$

where

$$(\underline{d}(r), \overline{d}(r)) = (-16r + 52\ ,\ -68r + 104), \quad 0 \leq r \leq 1.$$

The exact solution is $x = 4$. This problem is solved with the help of fuzzy neural network as described in this paper. Let $x_0 = 6.5$, $\eta = 2 \times 10^{-3}$ and $\gamma = 2 \times 10^{-3}$. Table 2 shows the approximated solution over a number of iterations and Fig. 4 shows the accuracy of the solution $x_0(t)$ where t is the number of iterations, its noticeable that by increasing the iterations the cost function goes to zero. Fig. 5 shows the convergence of the approximated solution, in this figure by increasing the iterations the calculated solution goes to exact one.

Table 2
The approximated solutions with error analysis for Example 4.2.

t	$x_0(t)$	e	t	$x_0(t)$	e
1	6.2031	58756.6573	9	4.0169	5.25800333
2	5.7529	6479.79005	10	4.0108	2.15630978
3	5.1686	1741.48377	11	4.0070	0.88820022
4	4.8779	577.759761	12	4.0045	0.36688350
5	4.4654	210.882238	13	4.0029	0.15181854
6	4.2714	81.0081877	14	4.0019	0.06289576

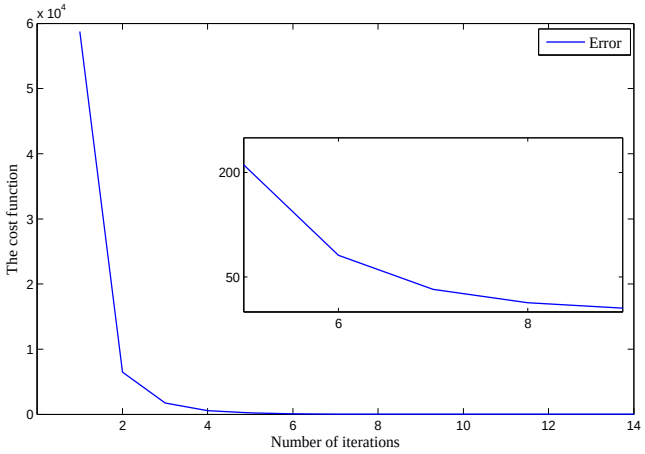

Fig. 4. The cost function for Example 4.2 over the number of iterations.

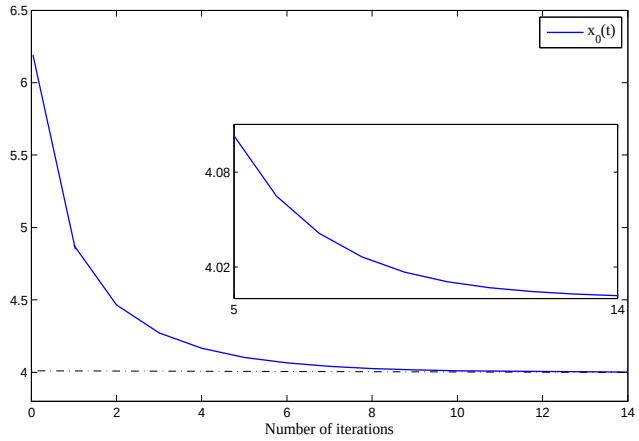

Fig. 5. Convergence of the approximated solution for Example 4.2.

Example 4.3. Consider the following dual fuzzy polynomial:

$$(2,4,8)x + (4,5,6)x^2 = (1,2,8)x + (1,3,7)x^2 + d,$$

where

$$(\underline{d}(r), \overline{d}(r)) = (-2r + 14\ ,\ 16r - 4), \quad 0 \leq r \leq 1.$$

where the exact solution is $x = 2$. This problem is solved with the help of fuzzy neural network as described in this paper. Let $x_0 = 5$, $\eta = 2 \times 10^{-3}$ and $\gamma = 2 \times 10^{-3}$. Table 3 shows the approximated solution over a number of iterations and Fig. 6 shows the accuracy of the solution $x_0(t)$ where t is the number of iterations, its noticeable that by increasing the iterations the cost function goes to zero. Fig. 7 shows the convergence of the approximated solution, in this figure by increasing the iterations the calculated solution goes to exact one.

Table 3
The approximated solutions with error analysis for Example 4.3.

t	$x_0(t)$	e	t	$x_0(t)$	e
1	4.7105	10579.0969	8	2.0472	3.31740504
2	3.9026	1098.42115	9	2.0321	1.51112723
3	3.1825	295.362973	10	2.0219	0.69584803
4	2.6308	102.803950	11	2.0149	0.32276080
5	2.3786	40.3674858	12	2.0102	0.15044391
6	2.2398	16.9441451	13	2.0070	0.07035772

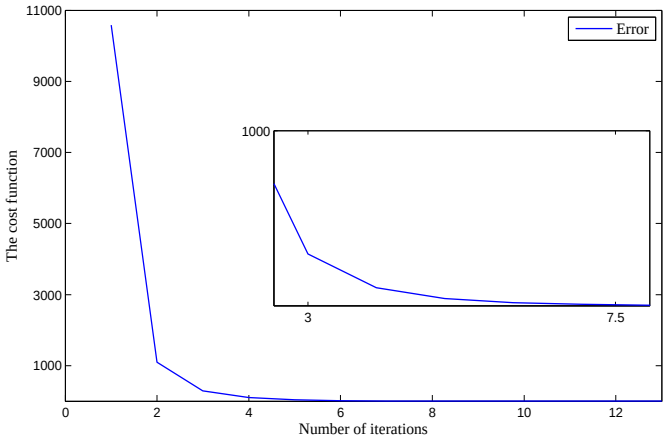

Fig. 6. The cost function for Example 4.3 over the number of iterations.

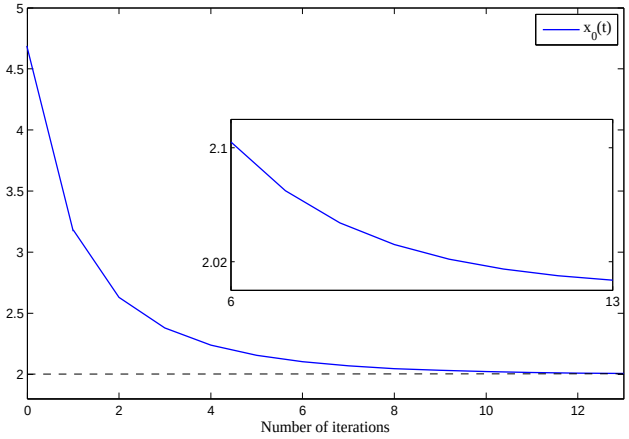

Fig. 7. Convergence of the approximated solution for Example 4.3.

5 Conclusions

The topics of fuzzy neural networks which attracted growing interest for some time, have been developed in recent years. In this paper we propose a Perceptron Neural Network consisting of a learning algorithm based on the gradient descent method applied in order to approximate the solution of dual fuzzy polynomials. The proposed neural network is a five layer feed-back neural network where connection weights are crisp numbers and its input-output relation was defined by the extension principle. Due to the application of gradient descent learning, this method presents a rapid convergence for the solutions. The approach is simulated in MATLAB. The simulation shows the method is effective; fast in response, minimal in overshoot, robust and very powerful technique in finding analytical solutions for dual fuzzy polynomials.

References

[1] S. Abbasbandy, M. Amirfakhrian, Numerical approximation of fuzzy functions by fuzzy polynomials, Appl. Math. Comput. 174 (2006) 669-675.
http://dx.doi.org/10.1016/j.amc.2005.05.028

[2] S. Abbasbandy, M. Otadi, Numerical solution of fuzzy polynomials by fuzzy neural network, Appl. Math. Comput. 181 (2006) 1084-1089.
http://dx.doi.org/10.1016/j.amc.2006.01.073

[3] S. Abbasbandy, M. Otadi, M. Mosleh, Minimal solution of general dual fuzzy linear systems, Chaos Solitons Fract. 178 (2008) 11131124.
http://dx.doi.org/10.1016/j.chaos.2006.10.045

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

[5] J.J. Buckley, E. Eslami, Neural net solutions to fuzzy problems: The quadratic equation, Fuzzy Sets and Syst. 86 (1997) 289-298.
http://dx.doi.org/10.1016/S0165-0114(95)00412-2

[6] R. Fuller, Neural fuzzy systems, Department of Information Technologies, Abo Akademi University, (1995).

[7] Y. Hayashi, J.J. Buckley, E. Czogala, Fuzzy neural network with fuzzy signals and weights, Int. J. Intell. Syst. 8 (1993) 527-537.
http://dx.doi.org/10.1002/int.4550080405

[8] H. Ishibuchi, K. Kwon, H. Tanaka, A learning of fuzzy neural networks with triangular fuzzy weghts, Fuzzy Sets and Syst. 71 (1995) 277-293.
http://dx.doi.org/10.1016/0165-0114(94)00281-B

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

[10] A. Jafarian, S. Measoomynia, Solving fuzzy polynomials using neural nets with a new learning algorithm, Appl. Math. Sci. 5 (2011) 2295-2301.

[11] A. Jafarian, S. Measoomynia, Solving system of fuzzy equations using neural net, Commun. Numer. Anal. 10 (2012) 5899.

[12] M. Ma, M. Friedman, A. Kandel, A new fuzzy arithmetic, Fuzzy Sets Syst. 108 (1999) 83-90.
http://dx.doi.org/10.1016/S0165-0114(97)00310-2

[13] H.T. Nguyen, A note on the extension principle for fuzzy sets, J. Math. Anal. Appl. 64 (1978) 369-380.
http://dx.doi.org/10.1016/0022-247X(78)90045-8

[14] D.E. Rumelhart, J.L. McClelland, The PDP Research Group, Parallel Distributed Processing, vol. 1, MIT Press. Cambridge. MA. (1986).

[15] X. Wang, Z. Zhong, M. Ha, Iteration algorithms for solving a system of fuzzy linear equations, Fuzzy Sets and Syst. 119 (2001) 121128.
http://dx.doi.org/10.1016/S0165-0114(98)00284-X

[16] L.A. Zadeh, The concept of a liguistic variable and its application to approximate reasoning, Inform. Sci. 8 (1975) 199-249.
http://dx.doi.org/10.1016/0020-0255(75)90036-5

[17] L.A. Zadeh, Toward a generalized theory of uncertainty (GTU) an outline, Inform. Sci. 172 (2005) 1-40.
http://dx.doi.org/10.1016/j.ins.2005.01.017