Available online at http://ijim.srbiau.ac.ir

Int. J. Industrial Mathematics Vol. 2, No. 2 (2010) 129-142

A New Approach to the Numerical Solution of Dual Fully Fuzzy Polynomial Equations

M. Mosleh [a], M. Otadi [a*]

Department of Mathematics, Firuozkooh Branch, Islamic Azad University, Firuozkooh, Iran

Received 20 April 2010; revised 14 August 2010; accepted 29 August 2010.

Abstract

In this paper, we present a numerical method for solving fully fuzzy polynomials. The proposed method is based on approximating fuzzy neural network. This method can also lead to improving numerical methods. In this work, an architecture of fuzzy neural networks is also proposed to find a fuzzy root of a fuzzy polynomial (if exists) by introducing a learning algorithm. We propose a learning algorithm from the cost function for adjusting fuzzy weights. Finally, we illustrate our approach by numerical examples.

Keywords : Fuzzy number; Neural network; Fuzzy polynomial

1 Introduction

Polynomials play a major role in various areas such as pure and applied mathematics, engineering and social sciences. Previous papers [3, 4], tried to find the numerical solution $x \in \mathbb{R}$ (if exists) of a fuzzy polynomial equation such as $A_1x + A_2x^2 + \ldots + A_nx^n = A_0$ where $A_0, A_1, \ldots, A_n$ are fuzzy numbers and system F, where F denotes a system of s fuzzy polynomial equations such as:

$$\begin{aligned} f_1(x_1, x_2, \ldots, x_n) &= A_{10}, \\ &\vdots \\ f_l(x_1, x_2, \ldots, x_n) &= A_{l0}, \\ &\vdots \\ f_s(x_1, x_2, \ldots, x_n) &= A_{s0}, \end{aligned}$$

where $x_1, x_2, \ldots, x_n \in \mathbb{R}$ and all coefficients are fuzzy numbers. Also Allahviranloo et al [6] applied the Fixed point method for solving fuzzy nonlinear equations.

*Corresponding author. Email address: otadi@iaufb.ac.ir, Tel. +98 912 6964202

Now, consider the following fully fuzzy polynomial of the form $\acute{A}_1X + \acute{A}_2X^2 + \ldots + \acute{A}_nX^n = \acute{\acute{A}}_1X + \acute{\acute{A}}_2X^2 + \ldots + \acute{\acute{A}}_nX^n + A_0$ where X and all coefficients are fuzzy numbers. In this paper we are interested in finding a fuzzy root of such fuzzy polynomials.

We wish to find the answer to this question: How is the fuzzy neural network going to solve the fuzzy polynomials? In this paper, an architecture of fuzzy neural network is proposed to find a fuzzy root of a fuzzy polynomial by introducing a learning algorithm. Some applications of fuzzy polynomials are considered by [18, 24].

During the past few years, neural networks have received much attention [10, 13, 20, 21, 23, 25]. Ishibuchi *et al.* [14] proposed a learning algorithm of fuzzy neural networks with triangular fuzzy weights and Hayashi *et al.* [12] also fuzzified the delta rule. Linear and nonlinear fuzzy equations are solved by [1, 2, 8, 9], also Buckley and Eslami [7] considered neural net solutions to fuzzy problems.

In this paper, we first propose an architecture of fuzzy neural networks with fuzzy weights for fuzzy input vector and fuzzy target. The input-output relation of each unit is defined by the extension principle of Zadeh [26, 27]. The output from the fuzzy neural network, which is also a fuzzy number, is numerically calculated by the interval arithmetic [5] for fuzzy weights and level sets(i.e., α-cuts) of fuzzy inputs. Next, we define a cost function for the level sets of fuzzy output and fuzzy target. Then, a crisp learning algorithm is derived from the cost function to find a fuzzy root (if exists) of a fuzzy polynomial. The effectiveness of the proposed algorithm is proved by solving some examples in the last section.

2 Preliminaries

We represent an arbitrary fuzzy number by an ordered pair of functions $(\underline{u}(r), \overline{u}(r)), 0 \leq r \leq 1$, which satisfy the following requirements [19]:

1. $\underline{u}(r)$ is a bounded left continuous non decreasing function on $[0, 1]$.
2. $\overline{u}(r)$ is a bounded left continuous non increasing function on $[0, 1]$.
3. $\underline{u}(r) \leq \overline{u}(r), \quad 0 \leq r \leq 1$.

The crisp number λ is simply represented by $\underline{u}(r) = \overline{u}(r) = \lambda, \ 0 \leq r \leq 1$. The set of all the fuzzy numbers is denoted by E^1. A popular fuzzy number is the triangular fuzzy number $u = (m, \alpha, \beta)$ with membership function

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

where $\alpha > 0$ and $\beta > 0$. Its parametric form is

$$\underline{u}(r) = m + \alpha(r - 1), \quad \overline{u}(r) = m + \beta(1 - r).$$

Triangular fuzzy numbers are fuzzy numbers in LR representation where the reference functions L and R are linear. The set of all triangular fuzzy numbers on $\mathbb{R}$ is called $\hat{FZ}$. Let n be a fuzzy number with membership function $\mu(x|n)$. The membership function is

partially specified by (n_1, n_2, n_3, n_4) where: (1) $n_1 < n_2 < n_3 < n_4$; (2) $\mu(x|n) = 0$ outside (n_1, n_4) and equals one at (n_2, n_3); (3) $\mu(x|n)$ is continuous and monotonically increasing from zero to one on $[n_1, n_2]$; and (4) $\mu(x|n)$ is continuous and monotonically decreasing from one to zero on $[n_3, n_4]$. If $\mu(x|n)$ consists of a straight line segment on (n_1, n_2) and on (n_3, n_4) we will write $n = (n_1/n_2/n_3/n_4)$.

Let $f_1(\alpha|n)$ be the inverse of $\mu(x|n)$ on $[n_1, n_2]$ and let $f_2(\alpha|n)$ be the inverse of $\mu(x|n)$ on $[n_3, n_4]$. Therefore, $n_1 = f_1(0|n)$, $n_2 = f_1(1|n)$, $n_3 = f_2(1|n)$, and $n_4 = f_2(0|n)$. If $n = (n_1/n_2/n_3/n_4)$, then $f_1(\alpha|n) = (n_2 - n_1)\alpha + n_1$ and $f_2(\alpha|n) = (n_3 - n_4)\alpha + n_4$. The inverse notation is very handy for performing multiplication and addition of fuzzy numbers.

The $\alpha-$cut of a fuzzy number n is

$$n^\alpha = \{x|\mu(x|n) \geq \alpha\}$$

for $0 < \alpha \leq 1$. We see that $n^\alpha = [f_1(\alpha|n), f_2(\alpha|n)]$ which we write as $[n_1^\alpha, n_2^\alpha]$, for $0 < \alpha \leq 1$. We will use the notation $\dot{n}_1^\alpha$ and $\dot{n}_2^\alpha$ for the derivative of n_1^α and n_2^α, respectively, with respect to α.

Theorem 2.1. *Let a and c are fuzzy numbers. The equation $a + x = c$ has a solution x if and only if $c_1 - a_1 < c_2 - a_2 < c_3 - a_3 < c_4 - a_4$.*

Proof: Taking α-cuts we obtain $a_i^\alpha + x_i^\alpha = c_i^\alpha, i = 1, 2$. Then

$$x_1 < x_2 < x_3 < x_4,$$

and $\dot{x}_1^\alpha > 0, \dot{x}_2^\alpha < 0$ if and only if $c_1 - a_1 < c_2 - a_2 < c_3 - a_3 < c_4 - a_4$.

2.1 Operations on fuzzy numbers

Operations on fuzzy numbers are numerically performed on level sets (i.e., h-cuts). For $0 < h \leq 1$, a h-level set of a fuzzy number X is defined as

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

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

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

where $[X]_h^L$ and $[X]_h^U$ are the lower and the upper limits of the h-level set $[X]_h$, respectively.

From interval arithmetic [5], the above operations on fuzzy numbers are written for h-level sets as follows:

$$A = B \Longleftrightarrow [A]_h = [B]_h \quad for \quad 0 < h \leq 1, \tag{2.1}$$

$$[A + B]_h = [[A]_h^L + [B]_h^L, [A]_h^U + [B]_h^U], \tag{2.2}$$

$$\begin{aligned}[A.B]_h &= [[A]_h^L, [A]_h^U].[[B]_h^L, [B]_h^U] \\ &= [min\{[A]_h^L.[B]_h^L, [A]_h^L.[B]_h^U, [A]_h^U.[B]_h^L, [A]_h^U.[B]_h^U\}, \\ &\quad max\{[A]_h^L.[B]_h^L, [A]_h^L.[B]_h^U, [A]_h^U.[B]_h^L, [A]_h^U.[B]_h^U\}],\end{aligned} \tag{2.3}$$

$$f([Net]_h) = f([[Net]_h^L, [Net]_h^U]) = [f([Net]_h^L), f([Net]_h^U)], \tag{2.4}$$

where f is an increasing function. In the case of $0 \leq [A]_h^L \leq [A]_h^U$, (2.3) can be simplified as

$$[A.B]_h = [min\{[A]_h^L.[B]_h^L, [A]_h^L.[B]_h^U\}, max\{[A]_h^U.[B]_h^L, [A]_h^U.[B]_h^U\}]. \tag{2.5}$$

The result of a fuzzy addition of triangular fuzzy numbers is a triangular fuzzy number again. So we only have to compute the following equation:

$$(a_m, a_l, a_r) + (b_m, b_l, b_r) = (a_m + b_m, a_l + b_l, a_r + b_r) \tag{2.6}$$

Considering the fuzzy multiplication, some computational expense problems can be investigated. The result of a fuzzy multiplication is a fuzzy number in LR representation, but it is difficult to compute the new functions L and R because they are not necessarily linear. We approximate this fuzzy multiplication such that it computes a triangular fuzzy number too. This fuzzy multiplication is denoted by $\hat{*}$ [11].

This fuzzy multiplication is based on the extension principle but it is a bit different from the classical fuzzy multiplication. We compute our operation by the following equation:

$$(a_m, a_l, a_r)\hat{*}(b_m, b_l, b_r) = (c_m, c_l, c_r) \tag{2.7}$$

with

$$c_m = a_m.b_m, c_l = c_m - c_\lambda, c_r = c_\rho - c_m,$$
$$c_\lambda := min(a_\lambda.b_\lambda, a_\lambda.b_\rho, a_\rho.b_\lambda, a\rho.b_\rho),$$
$$c_\rho := max(a_\lambda.b_\lambda, a_\lambda.b_\rho, a_\rho.b_\lambda, a\rho.b_\rho)$$

where $a_\lambda = a_m - a_l, a_\rho = a_m + a_r, b_\lambda = b_m - b_l, b_\rho = b_m + b_r$. a_λ and b_λ denote the left limits of the support of the fuzzy numbers a and b respectively and a_ρ and b_ρ denote the right limits of the support of the fuzzy numbers a and b respectively.

The use of these fuzzy operations has some advantages:

- The distributivity of these operations is retained. This is very important for our theoretical examinations.
- The computational expense is acceptable.
- The idea of fuzzy sets is preserved even if a fuzzy number is characterized by only three values.

We describe the classical definition of distance between fuzzy numbers [11]:

Definition 2.1. *The mapping $\hat{D} : \hat{FZ} \times \hat{FZ} \longrightarrow \mathbb{R}^+$ is defined by*

$$\hat{D}(A, B) = max(|a_m - b_m|, |a_\lambda - b_\lambda|, |a_\rho - b_\rho|),$$

where $A = (a_m, a_l, a_r)$ and $B = (b_m, b_l, b_r)$. It can be proved that $\hat{D}$ is a metric on $\hat{FZ}$ and so $(\hat{FZ}, \hat{D})$ becomes a metric space.

2.2 Input-output relation of each unit

Let's fuzzify a two-layer feedforward neural network with n input units and one output unit. Input vector, target and connection weights are fuzzified (i.e., extended to fuzzy numbers). In order to derive a crisp learning rule, we restrict fuzzy weights, 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 units:

$$O_i = A_i, \qquad i = 1, 2, \ldots, n. \tag{2.8}$$

Output unit:

$$Y = f(Net), \tag{2.9}$$

$$Net = \sum_{j=1}^{n} W_j.O_j, \tag{2.10}$$

where A_i and W_j are fuzzy input and fuzzy weight respectively (see Fig. 1).

The input-output relation in Eqs. (2.8)-(2.10) is defined by the extension principle[26, 27] as in Hayashi et al. [12] and Ishibuchi et al. [16].

2.3 Calculation of fuzzy output

The fuzzy output from each unit in Eqs. (2.8)-(2.10) is numerically calculated for crisp weights and level sets of fuzzy inputs. The input-output relations of our fuzzy neural network can be written for the h-level sets as follows:
Input units:

$$[O_i]_h = [A_i]_h, \qquad i = 1, 2, \ldots, n. \tag{2.11}$$

Output unit:

$$[Y]_h = f([Net]_h), \tag{2.12}$$

$$[Net]_h = \sum_{j=1}^{n} [W_j.O_j]_h. \tag{2.13}$$

From Eqs. (2.11)-(2.13), we can see that the h-level sets of the fuzzy output Y is calculated from those of the fuzzy inputs and fuzzy weights. From Eqs. (2.2)-(2.7), the above relations are written as follows when the h-level sets of the fuzzy inputs A_i's are nonnegative, i.e., $0 \leq [A_i]_h^L \leq [A_i]_h^U$ for all i's:
Input units:

$$[O_i]_h = [[O_i]_h^L, [O_i]_h^U] = [[A_i]_h^L, [A_i]_h^U], \qquad i = 1, 2, \ldots, n. \tag{2.14}$$

Output unit:

$$[Y]_h = [[Y]_h^L, [Y]_h^U] = [f([Net]_h^L), f([Net]_h^U)], \tag{2.15}$$

where f is an increasing function.

$$[Net]_h^L = \sum_{i \in a} [O_i]_h^U.[W_i]_h^L + \sum_{i \in b} [O_i]_h^L.[W_i]_h^L + \sum_{i \in c} [O_i]_h^U.[W_i]_h^L, \tag{2.16}$$

$$[Net]_h^U = \sum_{i\in a}[O_i]_h^L.[W_i]_h^U + \sum_{i\in b}[O_i]_h^U.[W_i]_h^U + \sum_{i\in c}[O_i]_h^U.[W_i]_h^U, \tag{2.17}$$

where $a = \{i \mid [W_i]_h^U \leq 0\}$, $b = \{i \mid 0 \leq [W_i]_h^L\}$ and $c = \{i \mid [W_i]_h^L < 0, 0 \leq [W_i]_h^U\}$.

3 Fuzzy polynomials

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 fuzzy polynomial equation of the form

$$\acute{A}_1X + \acute{A}_2X^2 + \ldots + \acute{A}_nX^n = \acute{\acute{A}}_1X + \acute{\acute{A}}_2X^2 + \ldots + \acute{\acute{A}}_nX^n + A_0, \tag{3.18}$$

cannot be equivalently replaced by the fuzzy polynomial equation

$$(\acute{A}_1 - \acute{\acute{A}}_1)X + \ldots + (\acute{A}_n - \acute{\acute{A}}_n)X^n = A_0,$$

which had been investigated. In the sequel, we will call the fuzzy polynomial equation

$$\acute{A}_1X + \acute{A}_2X^2 + \ldots + \acute{A}_nX^n = \acute{\acute{A}}_1X + \acute{\acute{A}}_2X^2 + \ldots + \acute{\acute{A}}_nX^n + A_0, \tag{3.19}$$

where $X^2 = X\hat{*}X, X^3 = X\hat{*}X\hat{*}X, \ldots$ and $X, A_0, A_1, \cdots, A_n$ are fuzzy numbers, a dual fully fuzzy polynomial equation. Therefore, we find solution Eq.(3.19) by the neural network, with suppose $A_{i1}^{'} - A_{i1}^{''} < A_{i2}^{'} - A_{i2}^{''} < A_{i3}^{'} - A_{i3}^{''}$ for $i = 1, \ldots, n$. Therefore, we have of the Eq.(3.19)

$$A_1x + A_2x^2 + \ldots + A_nx^n = A_0, \tag{3.20}$$

If

$$A_1\hat{*}X + A_2\hat{*}X^2 + \ldots + A_n\hat{*}X^n = Y, \tag{3.21}$$

then $Y \neq A_0$ generally. In this case, we try to find $\hat{X}$ such that $A_1\hat{*}\hat{X} + A_2\hat{*}\hat{X}^2 + \ldots + A_n\hat{*}\hat{X}^n$ approximates A_0 closely enough according to,

$$min\ |[Y]_h^L - [A_0]_h^L| \ \ and \ \ min\ |[Y]_h^U - [A_0]_h^U|, \quad h \in [0,1], \tag{3.22}$$

therefore,

$$min \ \ \hat{D}(Y, A_0), \tag{3.23}$$

where $\hat{D}$ is a distance between fuzzy numbers [11]. Then, it becomes a problem of optimization.

A FNN_3 (fuzzy neural network with fuzzy inputs, fuzzy output and fuzzy weights) solution to Eq. (3.20) is given in Figure 1.

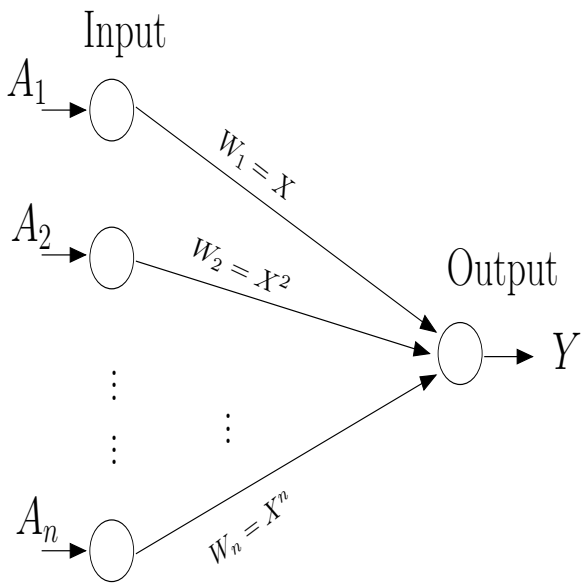

Fig. 1. Fuzzy neural network for solving fully fuzzy polynomial equations.

All signals and weights are fuzzy numbers. The input neurons make no change in their inputs and the signal A_i interacts with the weight W_i, so the input to the output neuron is

$$A_1\hat{*}W_1 + A_2\hat{*}W_2 + \ldots + A_n\hat{*}W_n$$

and the output, in the output neuron, equals its input, so

$$Y = A_1\hat{*}W_1 + A_2\hat{*}W_2 + \ldots + A_n\hat{*}W_n.$$

How does the FNN_3 solve the fuzzy polynomial equations? The training data is $(A_1, \ldots, A_n)$ for input and target (desired) output is A_0. We proposed a learning algorithm from the cost function for adjusting weights.

Following Section 4, we proposed a learning algorithm such that the network can approximate the fuzzy solution of Eq. (3.20) to any degree of accuracy.

4 Learning fuzzy neural network

A cost function to be minimized is defined for each h-level sets as follows:

$$[E(W_1, \ldots, W_n)]_h = [[E(W_1, \ldots, W_n)]_h^L, [E(W_1, \ldots, W_n)]_h^U], \tag{4.24}$$

where

$$[E(W_1, \ldots, W_n)]_h^L = \frac{1}{2}([Y]_h^L - [A_0]_h^L)^2,$$

$$[E(W_1, \ldots, W_n)]_h^U = \frac{1}{2}([Y]_h^U - [A_0]_h^U)^2.$$

Hence $[E(W_1, \ldots, W_n)]_h^L$ denotes the error between the left-hand sides of the h-level sets of the desired and the computed output, and $[E(W_1, \ldots, W_n)]_h^U$ denotes the error between the right-hand sides of the h-level sets of the desired and the computed output. Then the error function for the training pattern is

$$[E(W_1,\ldots,W_n)]_h^L = \frac{1}{2}(\sum_{j=1}^{n}[A_jW_j]_h^L - [A_0]_h^L)^2, \tag{4.25}$$

$$[E(W_1,\ldots,W_n)]_h^U = \frac{1}{2}(\sum_{j=1}^{n}[A_iW_j]_h^U - [A_0]_h^U)^2.$$

Clearly, this is a problem of optimization of quadratic functions without constrains that can usually be solved by gradient descent algorithm. In fact, denoting

$$[\nabla E(W)]_h^L = ([\frac{\partial E(W)}{\partial W_1}]_h^L,\ldots,[\frac{\partial E(W)}{\partial W_n}]_h^L)^T,$$

$$[\nabla E(W)]_h^U = ([\frac{\partial E(W)}{\partial W_1}]_h^U,\ldots,[\frac{\partial E(W)}{\partial W_n}]_h^U)^T,$$

In order to solve Eq. (3.22), assume k iterations to have been done and get the k^{th} iteration point $W_{(k)}$.

Remark 4.1. *Since the equations (4.25) are quadratic functions, supposing* $0 \leq [W_1]_h^L \leq [W_1]_h^U$ *and* $0 \leq [A_i]_h^L \leq [A_i]_h^U$ *for* $i = 1, 2, ..., n$. *We rewrite these as follows:*

$$\begin{aligned}[E(W)]_h^L &= \tfrac{1}{2}(\textstyle\sum_{j=1}^{n}[A_jW_j]_h^L - [A_0]_h^L)^2 \\ &= \tfrac{1}{2}([W]_h^L)^T[Q]_h^L[W]_h^L + ([B]_h^L)^T[W]_h^L + [C]_h^L,\end{aligned}$$

where

$$[W]_h^L = ([W_1]_h^L, [W_2]_h^L, \ldots, [W_n]_h^L)^T,$$

$$[Q]_h^L = [(q_{ij})_{n\times n}]_h^L,$$

$$[B]_h^L = ([b_1]_h^L, [b_2]_h^L, \ldots, [b_n]_h^L)^T,$$

$$[C]_h^L = \tfrac{1}{2}([A_0]_h^L)^2,$$

$$[q_{ij}]_h^L = [A_i]_h^L[A_j]_h^L,$$

with $[q_{ij}]_h^L = [q_{ji}]_h^L$ *and* $[b_i]_h^L = -[A_i]_h^L[A_0]_h^L$. *Therefore we have*

$$[\nabla E(W)]_h^L = [Q]_h^L[W]_h^L + [B]_h^L \tag{4.26}$$

and

$$\begin{aligned}[E(W)]_h^U &= \tfrac{1}{2}(\textstyle\sum_{j=1}^{n}[A_jW_j]_h^U - [A_0]_h^U)^2 \\ &= \tfrac{1}{2}([W]_h^U)^T[Q]_h^U[W]_h^U + ([B]_h^U)^T[W]_h^U + [C]_h^U,\end{aligned}$$

where

$$[W]_h^U = ([W_1]_h^U, [W_2]_h^U, \ldots, [W_n]_h^U)^T,$$

$$[Q]_h^U = [(q_{ij})_{n\times n}]_h^U,$$

$$[B]_h^U = ([b_1]_h^U, [b_2]_h^U, \ldots, [b_n]_h^U)^T,$$

$$[C]_h^U = \tfrac{1}{2}([A_0]_h^U)^2,$$

$$[q_{ij}]_h^U = [A_i]_h^U [A_j]_h^U,$$

with $[q_{ij}]_h^U = [q_{ji}]_h^U$ *and* $[b_i]_h^U = -[A_i]_h^U [A_0]_h^U$. *Also*

$$[\nabla E(W)]_h^U = [Q]_h^U [W]_h^U + [B]_h^U. \tag{4.27}$$

To find the stationary point of $[E(W)]_h = ([E(W)]_h^L, [E(W)]_h^U)$, we should put $[\nabla E(W)]_h^L = [\nabla E(W)]_h^U = 0 \triangleq (0, 0, \ldots, 0)^T$. When $[Q]_h^L$ and $[Q]_h^U$ are positive definite matrices, the stationary point can be obtained as follows:

$$[\hat{W}]_h^L = -([Q]_h^L)^{-1}[B]_h^L, \tag{4.28}$$

$$[\hat{W}]_h^U = -([Q]_h^U)^{-1}[B]_h^U.$$

The Hessian matrices at this point are

$$[\nabla^2 E(\hat{W})]_h^L = [\nabla(\nabla E(\hat{W}))]_h^L =$$

$$\begin{bmatrix} [\frac{\partial^2 E(W)}{\partial W_1^2}]_h^L & [\frac{\partial^2 E(W)}{\partial W_2 \partial W_1}]_h^L & \cdots & [\frac{\partial^2 E(W)}{\partial W_n \partial W_1}]_h^L \\ [\frac{\partial^2 E(W)}{\partial W_1 \partial W_2}]_h^L & [\frac{\partial^2 E(W)}{\partial W_2^2}]_h^L & \cdots & [\frac{\partial^2 E(W)}{\partial W_n \partial W_2}]_h^L \\ \cdots & \cdots & \cdots & \cdots \\ [\frac{\partial^2 E(W)}{\partial W_1 \partial W_n}]_h^L & [\frac{\partial^2 E(W)}{\partial W_2 \partial W_n}]_h^L & \cdots & [\frac{\partial^2 E(W)}{\partial W_n^2}]_h^L \end{bmatrix}_{W=\hat{W}} = [Q]_h^L,$$

$$[\nabla^2 E(\hat{W})]_h^U = [\nabla(\nabla E(\hat{W}))]_h^U =$$

$$\begin{bmatrix} [\frac{\partial^2 E(W)}{\partial W_1^2}]_h^U & [\frac{\partial^2 E(W)}{\partial W_2 \partial W_1}]_h^U & \cdots & [\frac{\partial^2 E(W)}{\partial W_n \partial W_1}]_h^U \\ [\frac{\partial^2 E(W)}{\partial W_1 \partial W_2}]_h^U & [\frac{\partial^2 E(W)}{\partial W_2^2}]_h^U & \cdots & [\frac{\partial^2 E(W)}{\partial W_n \partial W_2}]_h^U \\ \cdots & \cdots & \cdots & \cdots \\ [\frac{\partial^2 E(W)}{\partial W_1 \partial W_n}]_h^U & [\frac{\partial^2 E(W)}{\partial W_2 \partial W_n}]_h^U & \cdots & [\frac{\partial^2 E(W)}{\partial W_n^2}]_h^U \end{bmatrix}_{W=\hat{W}} = [Q]_h^U,$$

which are positive definite matrices because $[Q]_h^L$ and $[Q]_h^U$ are positive definite. From optimization theory, we known that $[\hat{W}]_h = ([\hat{W}]_h^L, [\hat{W}]_h^U) = (-[Q^{-1}]_h^L [B]_h^L, -[Q^{-1}]_h^U [B]_h^U)$, is the unique solution of the problem.

Remark 4.2. *The above method is not very convenient in applications. Now we consider its explicit scheme. Since*

$$[\nabla E(W)]_h^L = [Q]_h^L[W]_h^L + [B]_h^L, \qquad [\nabla E(W)]_h^U = [Q]_h^U[W]_h^U + [B]_h^U$$

then,

$$[\nabla E(W_{(k)})]_h^L = [Q]_h^L[W_{(k)}]_h^L + [B]_h^L, \qquad [\nabla E(W_{(k)})]_h^U = [Q]_h^U[W_{(k)}]_h^U + [B]_h^U$$

We know that [17]

$$([\nabla E(W_{(k+1)})]_h^L)^T[\nabla E(W_{(k)})]_h^L = 0, \qquad ([\nabla E(W_{k+1})]_h^U)^T[\nabla E(W_{(k)})]_h^U = 0$$

therefore we have [14]

$$([Q]_h^L([W_{(k)}]_h^L - [\mu_{(k)}]_h^L[\nabla E(W_{(k)})]_h^L) + [B]_h^L)^T([Q]_h^L[W_{(k)}]_h^L + [B]_h^L) = 0$$

and

$$([Q]_h^U([W_{(k)}]_h^U - [\mu_{(k)}]_h^U[\nabla E(W_{(k)})]_h^U) + [B]_h^U)^T([Q]_h^U[W_{(k)}]_h^U + [B]_h^U) = 0.$$

From these equations, we can easily get an expression for $[\mu_{(k)}]_h^L$ *and* $[\mu_{(k)}]_h^U$*:*

$$[\mu_{(k)}]_h^L = \frac{([\nabla E(W_{(k)})]_h^L)^T[\nabla E(W_{(k)})]_h^L}{([\nabla E(W_{(k)})]_h^L)^T[Q]_h^L[\nabla E(W_{(k)})]_h^L} \tag{4.29}$$

and

$$[\mu_{(k)}]_h^U = \frac{([\nabla E(W_{(k)})]_h^U)^T[\nabla E(W_{(k)})]_h^U}{([\nabla E(W_{(k)})]_h^U)^T[Q]_h^U[\nabla E(W_{(k)})]_h^U}. \tag{4.30}$$

Substituting these into equations [14, 15, 22], we obtain

$$\begin{aligned} &W_{1(k+1)} = W_{1(k)} + \Delta W_{1(k)}, \\ &\Delta W_{1(k)} = -\mu_{(k)}\nabla E(W_{1(k)}) + \alpha\Delta W_{1(k-1)}, \end{aligned} \tag{4.31}$$

where k *indexes the number of adjustments,* $[\mu_{(k)}]_h^L$ *and* $[\mu_{(k)}]_h^U$ *are learning rates,* α *is a constant momentum term (a positive real number) and* $W_{(k)}^T = (W_{1(k)}, ..., W_{n(k)})^T$*. We have the explicit scheme*

$$\begin{aligned} &[W_{1(k+1)}]_h^L = [W_{1(k)}]_h^L - \frac{([\nabla E(W_{(k)})]_h^L)^T[\nabla E(W_{(k)})]_h^L}{([\nabla E(W_{(k)})]_h^L)^T[Q]_h^L[\nabla E(W_{(k)})]_h^L}[\nabla E(W_{1(k)})]_h^L \\ &+\alpha[\Delta W_{1(k-1)}]_h^L \end{aligned} \tag{4.32}$$

and

$$\begin{aligned} &[W_{1(k+1)}]_h^U = [W_{1(k)}]_h^U - \frac{([\nabla E(W_{(k)})]_h^U)^T[\nabla E(W_{(k)})]_h^U}{([\nabla E(W_{(k)})]_h^U)^T[Q]_h^U[\nabla E(W_{(k)})]_h^U}[\nabla E(W_{1(k)})]_h^U \\ &+\alpha[\Delta W_{1(k-1)}]_h^U. \end{aligned} \tag{4.33}$$

We can adjust other weights by

$$[W_{i(k)}]_h = [[W_{i(k)}]_h^L, [W_{i(k)}]_h^U] = [([W_{1(k)}]_h^L)^i, ([W_{1(k)}]_h^U)^i] \quad for \;\; i = 2, \ldots, n.$$

We can also obtain similar relations for $[A_i]_h^L \leq [A_i]_h^U \leq 0$ *and* $[W_i]_h^L \leq [W_i]_h^U \leq 0$*,* $i = 1, \ldots, n$ *and other cases.*

The fully fuzzy polynomial equations may have no solution. In this case there is no hope to make the error measure close to zero.

4.1 Algorithm

Step 1: Read α (momentum term constant and positive real number), K (number of iterations), W_j (fuzzy weights W_j are initialized values), A_i $0, 1, ..., n$ (fuzzy coefficients) and k (indexes the number of adjustments).
Step 2: Compute $[E(W_1, \ldots, W_n)]_h$.
Step 3: Compute $\mu_{(k)}$ is a learning rate.
Step 4: The fuzzy weight $W = (W_j)$ is updated by the Eq. (4.31).
Step 5: If $k < K$ then $k := k + 1$ and we continue the training by going back to step 2, otherwise we go to step 6.
Step 6: The training cycle is completed.

5 Numerical examples

In the following, we study some examples of fuzzy polynomial equations.

Example 5.1. *Consider the following fully fuzzy polynomial equation*

$$(3, 1, 1)X = X + (-4, 5, 3),$$

where the exact solution is $X = (-2, 1, 1)$ *in LR representation.*
In the computer simulation of this example by fuzzy neural network, we use $K = 15$ *iterations of the learning algorithm, also* $\alpha = 0.01$.
The training starts with $W_1(1) = (-1, 0.5, 0.25)$. *Applying the proposed method to the approximate solution. Figure* 2 *shows the convergence behavior.*

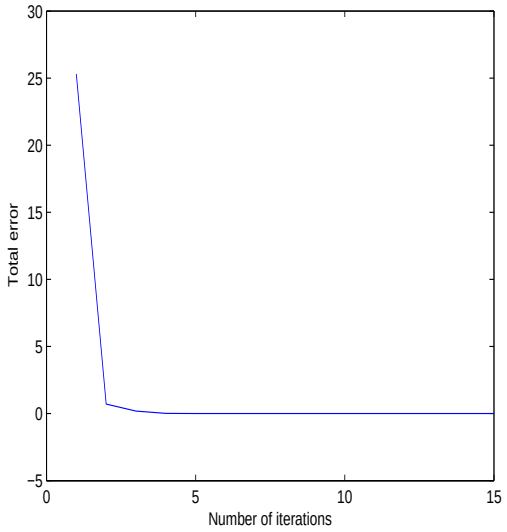

Fig.2. This figure shows the convergence behavior of numerical results obtained in example 1.

Example 5.2. *Consider the following fully fuzzy polynomial equation*

$$X^2 + 3X = (3, 1, 1)X + (4, 2, 2),$$

where the exact solution is $X = (2, 1, 1)$ *in LR representation.*
In the computer simulation of this example by fuzzy neural network, we use $K = 15$ *iterations of the learning algorithm, also* $\alpha = 0.01$.

The training starts with $W_1(1) = (1, 1, 1)$. *Applying the proposed method to the approximate solution. Figure* 3 *shows the convergence behavior.*

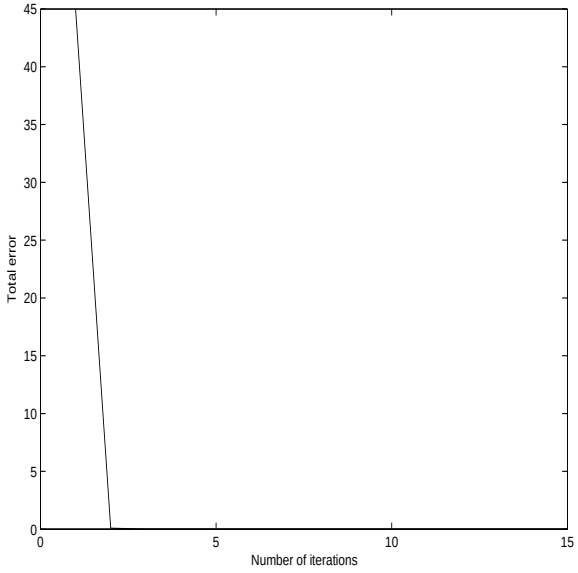

Fig. 3. This figure shows the convergence behavior of numerical results obtained in example 2.

6 Summary and conclusions

Solving dual fully fuzzy polynomial equations (DFFPEs) by using universal approximators (UA), that is, FNN was presented in this paper. In this paper, we derived a learning algorithm of fuzzy weights of two-layer feedforward fuzzy neural networks whose input-output relations were defined by extension principle. The effectiveness of the derived learning algorithm was demonstrated by computer simulation of numerical examples.

References

[1] S. Abbasbandy and B. Asady, Newton's method for solving fuzzy nonlinear equations, Appl. Math. Comput. 159 (2004) 349-356.

[2] S. Abbasbandy and R. Ezzati, Newton's method for solving a system of fuzzy nonlinear equations, Appl. Math. Comput. 175, 2006, 1189-1199.

[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, M. Otadi and M. Mosleh, Iterative method for fuzzy equations, Soft Comput,12 (2007) 935-939.

[7] J.J. Buckley and E. Eslami, Neural net solutions to fuzzy problems: The quadratic equation, Fuzzy Sets and Systems 86 (1997) 289-298.

[8] J.J. Buckley and Y. QU, Solving linear and quadratic fuzzy equations, Fuzzy Sets and Systems 35 (1990) 43-59.

[9] J.J. Buckley, Y. Qu, Solving fuzzy equations: a new solutions concept, Fuzzy Sets Syst. 39 (1991) 291-301.

[10] P.A. Castillo, J.J. Merelo, M.G. Arenas and G. Romero, Comparing evolutionary hybrid systems for design and optimization of multilayer perceptron structure along training parameters, Inform. Sci. 177 (2007) 2884-2905.

[11] TH. Feuring, W.-M. Lippe, Fuzzy neural networks are universal approximators, IFSA World Congress 1995, Sao Paulo, Brasil, vol. 2, pp. 659-662, 1995.

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

[13] Y.C. Hu, Fuzzy integral-based perceptron for two-class pattern classification problems, Inform. Sci. 177 (2007) 1673-1686.

[14] 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.

[15] 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.

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

[17] H.X. Li, L.X. Li, J.Y. Wang, Interpolation functions of feedforward neural networks, Computers and mathematics with applications 46 (2003) 1861-1874.

[18] P. Liu, Analysis of approximation of continuous fuzzy functions by multivariate fuzzy polynomials, Fuzzy Sets and Systems, 127 (2002) 299-313.

[19] M. Ma, M. Friedman and A. Kandel, A new fuzzy arithmetic, Fuzzy Sets and Systems, 108 (1999) 83-90.

[20] P. Melin, O. Castillo, An intelligent hybrid approach for industrial quality control combining neural networks, fuzzy logic and fractal theory, Inform. Sci. 177 (2007) 1543-1557.

[21] M. Mosleh, M. Otadi,S. Abbasbandy, Evaluation of fuzzy regression models by fuzzy neural network, Journal of Computational and Applied Mathematics, (2010) In press.

[22] D.E. Rumelhart, J.L. McClelland and the PDP Research Group, Parallel Distributed Processing, Vol. 1, MIT Press, Cambridge, MA, 1986.

[23] L.M. San José-Revuelta, A new adaptive genetic algorithm for fixed channel assignment, Inform. Sci. 177 (2007) 2655-2678.

[24] C. C. Wang and C. F. Tsai, Fuzzy processing using polynomial bidirdectional hetero-associative network, Inform.Sci, 125 (2000) 167-179.

[25] S.J. Yoo, J.B. Park and Y.H. Choi, Indirect adaptive control of nonlinear dynamic systems using self recurrent wavelet neural networks via adaptive learning rates, Inform. Sci. 177 (2007) 3074-3098.

[26] L.A. Zadeh, The concept of a liguistic variable and its application to approximate reasoning: Parts 1-3, Inform. Sci. 8 (1975) 199-249, 301-357; 9 (1975) 43-80.

[27] L.A. Zadeh, Toward a generalized theory of uncertainty (GTU)an outline, Inform. Sci. 172 (2005) 1-40.