

# Utilizing a Novel Architecture of Feed-back Fuzzy Neural Networks for Solving a Fuzzy Polynomial

A. Jafarian, Z. Ali nia and S. Measoomy nia

Department of Mathematics, Ourmia Branch  
Islamic Azad University, Ourmia, Iran  
jafarian5594@yahoo.com  
zeinab\_alinia@yahoo.com  
measoomy@yahoo.com

## Abstract

In this paper, we intend to offer a new method that based on fuzzy neural nets for finding a solution of a fuzzy polynomial that supposedly has a real solution. Our proposed neural network is a three-layer feed-back fuzzy neural network that corresponding connection weights to output layer are fuzzy numbers (FFNN4). This neural network can get a crisp input signal and then calculates it's corresponding fuzzy output. Presented method can give a real approximate solution for given polynomial with using of a cost function which is defined for the level sets of fuzzy output and target output. For this scope, a learning algorithm based on the gradient descent method will be introduced that can adjust the crisp input signal. The proposed method is illustrated by several examples with computer simulations.

**Keywords:** Fuzzy polynomials; Fuzzy feed-back neural networks; Cost function

## 1 Introduction

The fuzzy equation method is used for solving many problems in several applied fields like economics, finance, engineering and physics. These problems often boil down to the solution of a fuzzy equation. Therefore, various approaches for solving these problems have been reported in the last years. One of the valid methods is artificial neural network approach which have been developed for solving these kind of problems. First time, Buckley in [4] applied a structure of fuzzy neural networks for solving fuzzy equations and then extend this approach in [5, 6, 7]. In continuance Ishibuchi et al. [3] defined a cost function for every pair of fuzzy output vector and its corresponding fuzzy

target vector and then proposed a learning algorithm of fuzzy neural networks with triangular and trapezoidal fuzzy weights. Hayashi et al. [12] used the fuzzy delta learning rule. The input-output relation of each unit was defined by the Zadeh extension principle [9]. S. Abbasbandy et al. [11] proffered a structure of feed-forward fuzzy neural networks to find a real solution of a fuzzy polynomials system by introducing a learning algorithm. The main aim of this paper is to construct a new algorithm with the use of feed-back neural networks to obtain a approximate real solution fuzzy polynomials (if exist). This algorithm enables the FFNN4 to approximate crisp solution of the fuzzy polynomial that has been described in above for any desired degree of accuracy. This paper is organized as follows. First in section 2, the basic notations and definitions of fuzzy numbers and fuzzy derivative are briefly presented. Section 3 describes how to find a real solution of given fuzzy polynomial using FFNN4. Finally, some examples are collected in Section 4.

## 2 preliminaries

This section briefly deals with the foundation of fuzzy numbers. We first presented the notation of fuzzy sets and then we introduced some semantical aspects of fuzzy numbers; In addition, the input-output relation of the employed fuzzy neural network presented in this paper. We started by defining the fuzzy number.

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

- i**  $u$  is upper semicontinuous,
- ii**  $u(x) = 0$  outside some interval  $[a, d]$ ,
- iii** There is a real numbers  $b, c : a \leq b \leq c \leq d$  for which:
  1.  $u(x)$  is monotonic increasing on  $[a, b]$ ,
  2.  $u(x)$  is monotonic decreasing on  $[c, d]$ ,
  3.  $u(x) = 1, b \leq x \leq c$ .

The set of all fuzzy numbers (as given by definition 1 ) is denoted by  $E^1$  [10, 2].

**Definition 2.** A fuzzy number  $u$  is a pair  $(\underline{u}, \bar{u})$  functions  $\underline{u}(r), \bar{u}(r) : 0 \leq r \leq 1$ . which satisfy the following requirements:

- i**  $\underline{u}(r)$  is a bounded monotonic increasing left continuous function,
- ii**  $\bar{u}(r)$  is a bounded monotonic decreasing left continuous function,
- iii**  $\underline{u}(r) \leq \bar{u}(r) : 0 \leq r \leq 1$ .

A popular fuzzy number is the triangular fuzzy number  $u = (a, b, c)$  with membership function,

$$\mu_u(x) = \begin{cases} \frac{x-a}{b-a}, & a \leq x \leq b \\ \frac{c-x}{c-b}, & b \leq x \leq c \\ 0, & \text{otherwise,} \end{cases}$$

where  $a \leq b \leq c$ . It's parametric form is:

$$\underline{u}(r) = a + (b - a)\alpha \text{ and } \bar{u}(r) = c - (c - b)\alpha, \quad 0 \leq \alpha \leq 1.$$

## 2.1 Operation on fuzzy numbers

We briefly mention fuzzy number operations defined by the Zadeh extension principle [9, 8].

$$\begin{aligned} \mu_{A+B}(z) &= \max\{\mu_A(x) \wedge \mu_B(y) \mid z = x + y\}, \\ \mu_{f(Net)}(z) &= \max\{\mu_A(x) \wedge \mu_B(y) \mid z = xy\}, \end{aligned}$$

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 the activation function of output unit of our fuzzy neural network.

The above operations on fuzzy numbers are numerically performed on level sets (i.e.  $\alpha$ -cuts).

For  $0 \leq \alpha \leq 1$ , a  $\alpha$ -level set of a fuzzy number A is defined as:

$$\begin{aligned} [A]^\alpha &= \{x \mid \mu_A(x) \geq \alpha, x \in \mathbb{R}\}, \\ [A]^\alpha &= [[A]_l^\alpha, [A]_u^\alpha], \end{aligned}$$

where  $[A]_l^\alpha$  and  $[A]_u^\alpha$  are the lower and the upper limits of the  $\alpha$ -level set  $[A]^\alpha$ , respectively.

From interval arithmetic [1], the above operations on fuzzy numbers are written for the  $\alpha$ -level sets as follows:

$$[A]^\alpha + [B]^\alpha = [[A]_l^\alpha, [A]_u^\alpha] + [[B]_l^\alpha, [B]_u^\alpha] = [[A]_l^\alpha + [B]_l^\alpha, [A]_u^\alpha + [B]_u^\alpha], \quad (1)$$

$$f([Net]^\alpha) = f([Net]_l^\alpha, [Net]_u^\alpha) = [f([Net]_l^\alpha), f([Net]_u^\alpha)],$$

$$k[A]^\alpha = k[[A]_l^\alpha, [A]_u^\alpha] = [k[A]_l^\alpha, k[A]_u^\alpha], \quad \text{if } k \geq 0, \quad (2)$$

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

For arbitrary  $u = (\underline{u}, \bar{u})$  and  $v = (\underline{v}, \bar{v})$  we define addition ( $u + v$ ) and multiplication by k as [10, 2]:

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

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

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

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

## 2.2 Calculation of fuzzy output in FFNN4

In this subsection, we have given a short review on learning of fuzzified feedback neural networks. First consider a three-layer FFNN4 with one input unit,  $n$  neuron in hidden layer and one output unit. In given structure, we assume that the corresponding connection weights to output layer and target output are triangular fuzzy numbers. Fig. 1 has been showed the input signal  $x_0$ , the input-output relation of each unit and fuzzy output  $Y$ . These relations can be written as following:

Input unit:

The input neuron makes no change in it's input, so:

$$o = x_0. \quad (3)$$

The hidden units:

$$O_i = f(net_i),$$

$$net_i = o \cdot w_i, \quad i = 1, \dots, n.$$

The output unit:

$$Y = f(Net), \quad Net = \sum_{i=1}^n (O_i \cdot A_i),$$

where  $f$  is identical activation function. With using above equations, we can calculate the  $\alpha$ -level sets of the fuzzy output  $Y$ . Therefore, from Eqs. (1)-(2) we can write [1]:

$$[Y]^\alpha = [Net]^\alpha, \quad (4)$$

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

$$[Net]_l^\alpha = \sum_{j \in M} (O_j \cdot [A_j]_l^\alpha) + \sum_{j \in C} (O_j \cdot [A_j]_u^\alpha),$$

$$[Net]_u^\alpha = \sum_{j \in M} (O_j \cdot [A_j]_u^\alpha) + \sum_{j \in C} (O_j \cdot [A_j]_l^\alpha),$$

where  $M = \{j | O_j \geq 0\}$ ,  $C = \{j | O_j < 0\}$  and  $M \cup C = \{1, \dots, n\}$ .

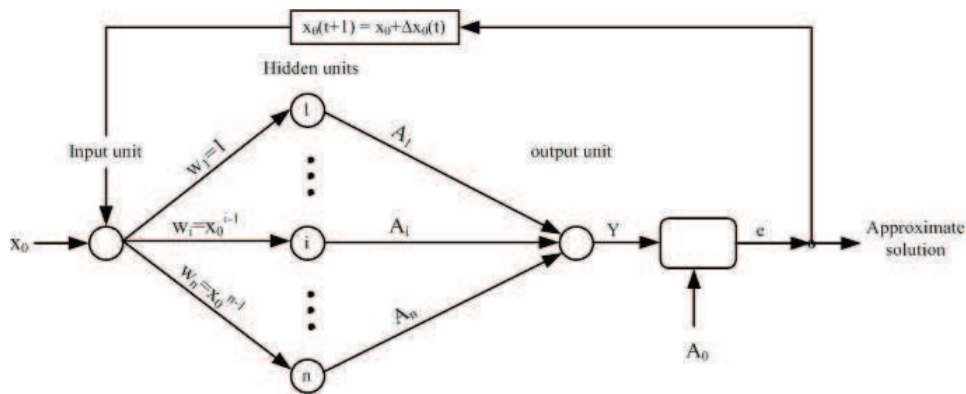


Fig. 1. The proposed FFNN4

### 3 Fuzzy polynomial

In this section, first a fuzzy polynomial is defined. Then we will concentrate on solving this fuzzy problem. Here, we consider the fuzzy polynomial in following form:

$$A_1x + \dots + A_nx^n = A_0, \tag{5}$$

for  $x \in \mathbb{R}$  (if exist), where  $A_j$  (for  $j = 0, \dots, n$ ) are fuzzy numbers. For simplify the fuzzy coefficient  $A_j$  will always be considered real triangular fuzzy number. As observed in previous section to get an approximate solution, an architecture of FFNN4 for fuzzy polynomial (5) has been given in Fig. 1. The modeling scheme is designed with the simple and versatile fuzzy neural network architecture.

#### 3.1 Cost function

In this subsection we intend to introduce how to deduce a learning algorithm to adjust the crisp parameter  $x_0$ . Let  $A_0$  be fuzzy target output corresponding to the fuzzy coefficient vector  $A = (A_1, \dots, A_n)$ . At first, we defined a cost function for  $\alpha$ -level sets of fuzzy output  $Y$  and it's corresponding target output  $A_0$  as follows:

$$e^\alpha = e_l^\alpha + e_u^\alpha, \tag{6}$$

where

$$e_l^\alpha = \alpha \cdot \frac{([A_0]_l^\alpha - [Y]_l^\alpha)^2}{2}, \tag{7}$$

$$e_u^\alpha = \alpha \cdot \frac{([A_0]_u^\alpha - [Y]_u^\alpha)^2}{2}, \tag{8}$$

In the cost function (6),  $e_l^\alpha$  and  $e_u^\alpha$  can be viewed as the squared errors for the lower limits and the upper limits of the  $\alpha$ -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 [3]:

$$e = \sum_{\alpha} e^{\alpha}. \quad (9)$$

### 3.1.1 Learning algorithm of the FFNN4

Let the real quantity  $x_0$  is initialized at random value for variable  $x$ . Our main aim is adjusting the parameter  $x_0$  with the using of learning algorithm which be introduced in below. For crisp parameter  $x_0$  adjustment rule can be written as follows:

$$x_0(t+1) = x_0(t) + \Delta x_0(t), \quad (10)$$

$$\Delta x_0(t) = -\eta \frac{\partial e^{\alpha}}{\partial x_0} + \gamma \Delta x_0(t-1), \quad (11)$$

where  $t$  is the number of adjustments,  $\eta$  is the learning rate and  $\gamma$  is the momentum term constant. The derivative  $\frac{\partial e^{\alpha}}{\partial x_0}$  can be calculated from the cost function  $e^{\alpha}$  and using the input-output relation of our fuzzy neural network for the  $\alpha$ -level sets in (3)-(4). We calculate  $\frac{\partial e^{\alpha}}{\partial x_0}$  as follows:

$$\frac{\partial e^{\alpha}}{\partial x_0} = \frac{\partial e_l^{\alpha}}{\partial x_0} + \frac{\partial e_u^{\alpha}}{\partial x_0}. \quad (12)$$

Thus our problem is calculating of the derivatives  $\frac{\partial e_l^{\alpha}}{\partial x_0}$  and  $\frac{\partial e_u^{\alpha}}{\partial x_0}$ . Since they are calculated in the same manner, we only show the calculation  $\frac{\partial e_l^{\alpha}}{\partial x_0}$ . So we have:

$$\frac{\partial e_l^{\alpha}}{\partial x_0} = \frac{\partial e_l^{\alpha}}{\partial [Y]_l^{\alpha}} \cdot \frac{\partial [Y]_l^{\alpha}}{\partial [Net]_l^{\alpha}} \cdot \frac{\partial [Net]_l^{\alpha}}{\partial x_0} = -\alpha \cdot ([A_0]_l^{\alpha} - [Y]_l^{\alpha}) \cdot \frac{\partial [Net]_l^{\alpha}}{\partial x_0},$$

where

$$\frac{\partial [Net]_l^{\alpha}}{\partial x_0} = \sum_{j=1}^n \left( \frac{\partial [Net]_l^{\alpha}}{\partial O_j} \cdot \frac{\partial O_j}{\partial x_0} \right) = \sum_{j=1}^n \left( \frac{\partial [Net]_l^{\alpha}}{\partial O_j} \cdot j \cdot x_0^{j-1} \right).$$

If  $O_j \geq 0$

$$\frac{\partial [Net]_l^{\alpha}}{\partial O_j} = [A_j]_l^{\alpha},$$

otherwise

$$\frac{\partial [Net]_l^{\alpha}}{\partial O_j} = [A_j]_u^{\alpha}.$$

After calculation  $\frac{\partial e_u^{\alpha}}{\partial x_0}$  with a similar process which applied for derivative  $\frac{\partial e_l^{\alpha}}{\partial x_0}$ , we can write:

$$\frac{\partial e^{\alpha}}{\partial x_0} = -\alpha \cdot \sum_{j \in M} \{ ([A_0]_l^{\alpha} - [Y]_l^{\alpha}) \cdot [A_j]_l^{\alpha} \cdot j \cdot x_0^{j-1} \}$$

$$\begin{aligned}
& -\alpha \cdot \sum_{j \in M} \{[A_0]_u^\alpha - [Y]_u^\alpha\} \cdot ([A_j]_u^\alpha \cdot j \cdot x_0^{j-1}) \\
& -\alpha \cdot \sum_{j \in C} \{[A_0]_l^\alpha - [Y]_l^\alpha\} \cdot ([A_j]_l^\alpha \cdot j \cdot x_0^{j-1}) \\
& -\alpha \cdot \sum_{j \in C} \{[A_0]_u^\alpha - [Y]_u^\alpha\} \cdot ([A_j]_l^\alpha \cdot j \cdot x_0^{j-1}),
\end{aligned}$$

where  $M = \{j \mid O_j \geq 0\}$  and  $C = \{j \mid O_j < 0\}$ . Now the learning algorithm can be summarized as follows:

### Learning algorithm

*Step 1:*  $\eta > 0$ ,  $\gamma > 0$ ,  $E_{max} > 0$  are chosen. Then 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:* Let  $t := t + 1$ . Repeat *Step 5* for  $\alpha = \alpha_1, \dots, \alpha_m$ .

*Step 4:*

- [i] Forward calculation: Calculate the  $\alpha$ -level set of the fuzzy output  $Y$  by presenting the  $\alpha$ -level set of the fuzzy coefficients vector  $A$ .
- [ii] Back-propagation: Adjust crisp parameter  $x_0$  using the cost function (6) for the  $\alpha$ -level sets of the fuzzy output  $Y$  and the target output  $A_0$ .

*Step 5:* Update the corresponding connection weights to the hidden layer as  $w_j(t+1) = (x_0(t+1))^{j-1}$ ,  $j = 1, \dots, n$ .

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

*Step 7:* The training cycle is completed. For  $E < E_{max}$  terminate the training session. If  $E > E_{max}$  then  $E$  is set to 0 and we initiate a new training cycle by going back to *Step 3*.

## 4 Numerical examples

To show the behavior and properties of this method, three examples are solved in this section. For each example, the computed values of the approximate solution are calculated over a number of iterations and also the cost function is plotted. In the following simulations, we use the specifications as follows:

1. Values of  $\alpha = 0, 0.1, \dots, 1$ .
2. Learning constant  $\eta = 0.0002$ .
3. Momentum constant  $\gamma = 0.0002$ .

4. Stopping conditions:  $Emax < 0.0001$ .

**Example 1.** Consider the following fuzzy polynomial:

$$(-5, -2, -1)x + (1, 2, 4)x^2 + (-2, 0, 2)x^3 = (-22, 4, 30),$$

where  $x \in \mathbb{R}$  and the exact solution is  $x = 2$ . In this example, we apply the proposed method to approximate the solution of this fuzzy polynomial. The training starts with  $x_0 = 3$ . Table 1 shows the approximated solution over a number of iterations and Figs. 2 and 3 show the accuracy of the solution  $x_0(t)$ .

$t$	$x_0(t)$	$e$	$t$	$x_0(t)$	$e$
1	2.3556	3826.5825	16	2.0010	0.018275746
2	2.1864	435.94162	17	2.0006	0.002925165
3	2.1073	114.85365	18	2.0004	0.001171592
4	2.0643	37.140442	19	2.0002	0.000469434
5	2.0394	13.179606	20	2.0001	0.000188140
6	2.0245	4.9139051	21	2.0000	0.000075415

Table 1

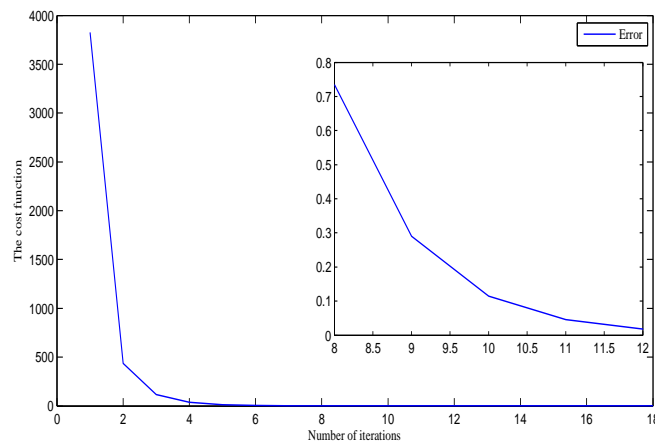


Fig. 2

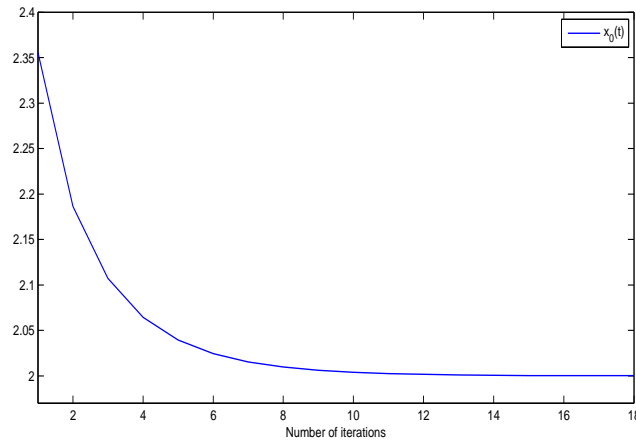


Fig. 3

**Example 2.** Let fuzzy polynomial

$$(1, 2, 3)x + (3, 4, 6)x^2 + (-1, 0, 1)x^3 = (3, 6, 10),$$

with the exact solution  $x = 1$ . First assume that  $x_0 = 0.1$ . Similarly Table 2 shows the approximated solution over a number of iterations and Figs. 4 and 5 show the accuracy of the solution  $x_0(t)$ .

$t$	$x_0(t)$	$e$	$t$	$x_0(t)$	$e$
1	0.18986	424.0374	0.99424	14	0.09092006
2	0.29646	375.7757	0.99675	15	0.02909691
3	0.41689	310.9503	0.99817	16	0.00925724
4	0.54379	233.4504	0.99897	17	0.00293540
5	0.66577	154.3744	0.99942	18	0.00092904
6	0.77105	88.01507	0.99968	19	0.000293727

Table 2

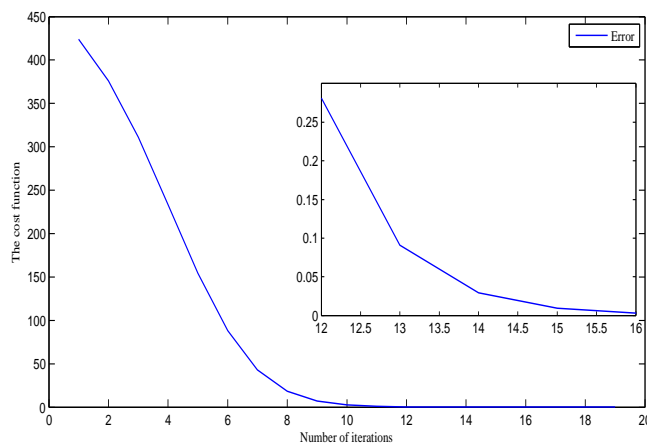


Fig. 4

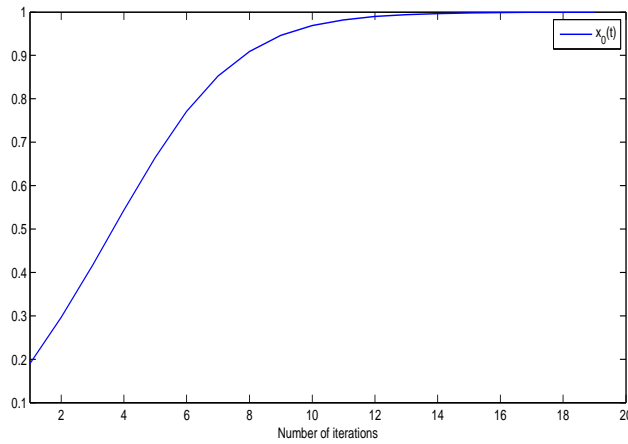


Fig. 5

**Example 3.** We consider the fuzzy polynomial:

$$(-1, 1, 3)x + (3, 5, 7)x^2 + (-1, 0, 1)x^3 = (-1, 4, 9),$$

with the exact solution  $x_0 = -1$ . In this example, Table 3 shows the approximated solution over a number of iterations and Figs. 6 and 7 show the accuracy of the solution  $x_0(t)$ .

$t$	$x_0(t)$	$e$	$t$	$x_0(t)$	$e$
1	-1.3084	929.8430	10	-1.0023	0.011664510
2	-1.1521	83.15148	11	-1.0014	0.004328644
3	-1.0837	19.51537	12	-1.0009	0.001609957
4	-1.0483	5.787343	13	-1.0005	0.000599612
5	-1.0285	1.903216	14	-1.0003	0.000223506
6	-1.0171	0.660340	15	-1.0002	0.000083350

Table 3

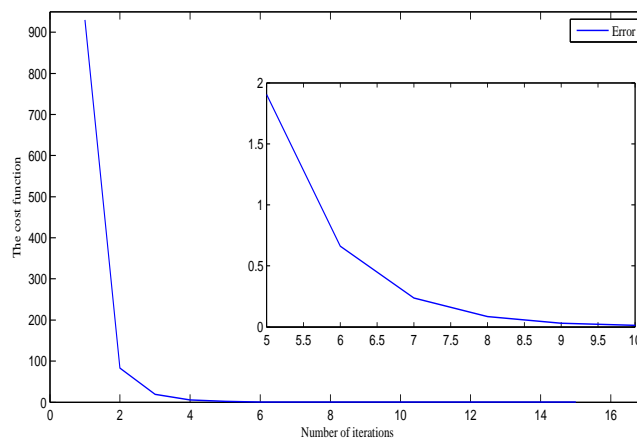


Fig. 6

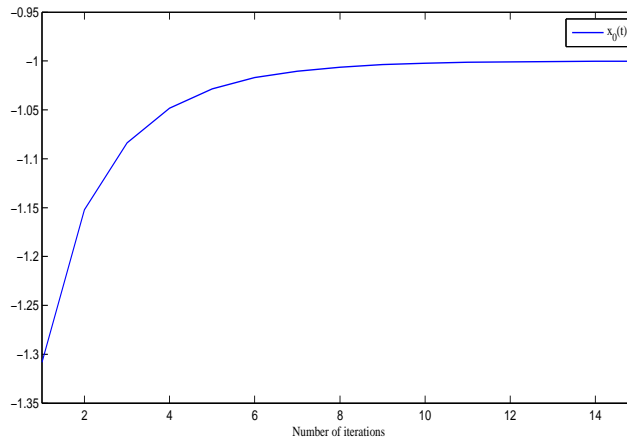


Fig. 7

## 5 Conclusions

In this paper, an architecture of feed-back neural networks has been proposed to approximate solution of a fuzzy polynomial. we suggested a FFNN4 model equivalent to the fuzzy polynomial. The analyzed examples illustrate the ability and reliability of the present method. The obtained solutions, in comparison with exact solutions admit a remarkable accuracy. Finally, we can conclude that for this kind of problems the introduced model presents good performance than feed-forward samples. Because the speed of convergence is increased which depends on number and widths of computations.

## References

- [1] G. Alefeld, J. Herzberger, Introduction to Interval Computations, Academic Press, New York, 1983.
- [2] H.T. Nguyen, A note on the extension principle for fuzzy sets, J. Math. Anal. Appl. 64 (1978) 369-380.
- [3] H. Ishibuchi, K.Kwon, H.Tanaka, A learning of fuzzy neural networks with triangular fuzzy weghts, Fuzzy Sets Syst. 71 (1995) 277-293.
- [4] J.J. Buckley, Y. Qu, Solving linear and quadratic fuzzy equations, Fuzzy Sets Syst. 35 (1990) 4359.
- [5] J.J. Buckley, E. Eslami, Neural net solutions to fuzzy problems: The quadratic equation, Fuzzy Sets Syst. 86 (1997) 289298.

- [6] J.J. Buckley, Y. Qu, Solving fuzzy equations: A new solution concept, *Fuzzy Sets Syst.* 39 (1991) 291301.
- [7] J.J. Buckley, T. Feuring, Y. Hayashi, Solving fuzzy equations using evolutionary algorithms and neural nets, *soft comput.* 6 (2002) 116-123.
- [8] L.A. Zadeh, Toward a generalized theory of uncertainty (GTU) an outline, *Inform. Sci.* 172 (2005) 1-40.
- [9] L.A. Zadeh, The concept of a linguistic variable and its application to approximate reasoning: Parts 1-3, *Inform. Sci.* 8 (1975) 199-249, 301-357; 9 (1975) 43-80.
- [10] R. Goetschel, W. Voxman, Elementary calculus, *Fuzzy Sets Syst.* 18 (1986) 31-43.
- [11] S. Abbasbandy, M. Otadi, Numerical solution of fuzzy polynomials by fuzzy neural network, *Appl. Math. Comput.* 181 (2006) 1084-1089.
- [12] Y. Hayashi, J.J. Buckley, E. Czogala, Fuzzy neural network with fuzzy signals and weights, *Int. J. Intell. Syst.* 8 (1993) 527-537.

**Received: July, 2011**