Available online at http://ijim.srbiau.ac.ir/
Int. J. Industrial Mathematics (ISSN 2008-5621)
Vol. 5, No. 4, 2013 Article ID IJIM-00273, 9 pages
Research Article

Utilizing a new feed-back fuzzy neural network for solving a system of fuzzy equations

A. Jafarian * † , S. Measoomy Nia ‡

Abstract

This paper intends to offer a new iterative method based on artificial neural networks for finding solution of a fuzzy equations system. Our proposed fuzzified neural network is a five-layer feed-back neural network that corresponding connection weights to output layer are fuzzy numbers. This architecture of artificial neural networks, can get a real input vector and calculates its corresponding fuzzy output. In order to find the approximate solution of the fuzzy system that supposedly has a real solution, first a cost function is defined for the level sets of the fuzzy network and target output. Then a learning algorithm based on the gradient descent method is used to adjust the crisp input signals. The present method is illustrated by several examples with computer simulations.

Keywords : System of fuzzy equations; Fuzzy feed-back neural network (FFNN); Cost function; Learning algorithm.

1 Introduction

Fuzzy system are very useful for solving many problems in several applied fields like economics, finance, engineering and physics, because these problems often boil down to the solution of a fuzzy system. In recent years, various approaches for solving fuzzy systems have been reported. For example, Friedman et al. [12] used the embedding method and suggested a general model for solving a fuzzy linear system. Allahviranloo [6, 7, 8] used the iterative Jacobi and Gauss Siedel methods, the adomian method and the successive over-relaxation method, respectively. Theoretical aspects of a fuzzy linear system were discussed by Dubois and Prade [11]. Dehgan et al. [10] employed iterative techniques for solving fully fuzzy linear system. Linear and nonlinear fuzzy systems were solved by [3, 1, 9]. Moreover, recently some different valid methods for solving fuzzy systems have been developed. One of the well known approaches is artificial neural networks approach in which has been applied to approximate solution of these kinds of problems. Ishibuchi et al. [15] 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

*Corresponding author. jafarian5594@yahoo.com
†Department of Mathematics, Urmia Branch, Islamic Azad University, Urmia, Iran.
‡Department of Mathematics, Urmia Branch, Islamic Azad University, Urmia, Iran.

weights. Hayashi et al. [14] used the fuzzy delta learning rule such that the input-output relation of each unit was defined by the Zadeh extension principle [18]. Abbasbandy et al. [4, 2] proffered a structure of feed-forward fuzzy neural networks to find a real solution of a fuzzy polynomials system.

The main aim of this paper is to construct a new algorithm with the use of feed-back neural networks to obtain an approximate real solution of fuzzy equations system (if exist). It is mentioned that in the purposed fuzzy feed-back neural network, the connection weights and the fuzzy output are fuzzy numbers, and also input signals are crisp. So, we say to this neural network FFNN4. The given algorithm enables the FFNN4 to approximate crisp solution of the fuzzy system 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 system using FFNN4. Finally, some examples are collected in Section 4.

2 Basic definitions and notations

In this section an overview of general concepts and definitions is given that will be used in next sections, repeatedly. The most basic used notations in fuzzy calculus and artificial neural networks, are briefly introduced here.

2.1 Operation on fuzzy numbers

Definition 2.1 *A fuzzy number is a fuzzy set* $u : \mathbb{R}^1 \to I = [0, 1]$ *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:*

a) $u(x)$ *is monotonic increasing on* $[a, b]$,

b) $u(x)$ *is monotonic decreasing on* $[c, d]$,

c) $u(x) = 1, b \leq x \leq c$.

The set of all fuzzy numbers (as given by Definition 2.1) is denoted by E^1 [13, 16]. An alternative definition which yields the same E^1 is given by Kandel [19].

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

i) $\underline{u}(r)$ *is a bounded monotonic increasing left continuous function,*

ii) $\overline{u}(r)$ *is a bounded monotonic decreasing left continuous function,*

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

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

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

where $a \leq b \leq c$. It,s parametric form is: $\underline{u}(r) = a+(b-a)\alpha$ and $\overline{u}(r) = c-(c-b)\alpha$, $0 \leq r \leq 1$. We briefly mention fuzzy number operations defined by the Zadeh extension principle [18, 17], as:

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

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

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

where $[A]^r_l$ and $[A]^r_u$ are the lower and the upper limits of the r-level set $[A]^r$, respectively.

For arbitrary $u = (\underline{u}, \overline{u})$ and $v = (\underline{v}, \overline{v})$ we define addition $(u + v)$ and multiplication by k as [13, 16]:

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

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

2.2 Calculation of fuzzy output in FFNN4

In this part, we aim to give a short review on learning of fuzzified feed-back neural networks. First consider a five-layer FFNN4 with two input units, n neuron in each hidden layer and one output units. In given structure, we assume that the corresponding connection weights to output layer and target output are triangular fuzzy numbers. In Fig. 1 the input vector (x_0, y_0), the input-output relation of each unit in hidden layers and the output Y_p have been presented. These relations can be written as following:

Input units:
The input neurons make no change in their inputs, so:

$$o_1 = x_0\ and\ o_2 = y_0. \tag{2.1}$$

First hidden units:

$$O_{pj} = f_{pj}(o_1),\ j = 1, ..., n.$$

Second hidden units:

$$O'_{pj} = g_{pj}(o_2).$$

Third hidden units:

$$U_{pj} = net_{pj},\quad net_{pj} = O_{pj}\ .\ O'_{pj}.$$

Output unit:

$$Y_p = Net_p,\quad Net_p = \sum_{j=1}^{n} U_{pj}\ .\ A_{pj}.$$

Using above equations the α-level sets of the fuzzy output Y_p can be written as following [5]:

$$[Y_p]^\alpha = [Net_p]^\alpha, \tag{2.2}$$

$$[Net_p]^\alpha = [[Net_p]^\alpha_l\ ,\ [Net_p]^\alpha_u],$$

$$[Net_p]^\alpha_l = \sum_{j\epsilon M} U_{pj}\ .\ [A_{pj}]^\alpha_l + \sum_{j\epsilon C} U_{pj}\ .\ [A_{pj}]^\alpha_u,$$

$$[Net_p]^\alpha_u = \sum_{j\epsilon M} U_{pj}\ .\ [A_{pj}]^\alpha_u + \sum_{j\epsilon C} U_{pj}\ .\ [A_{pj}]^\alpha_l,$$

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

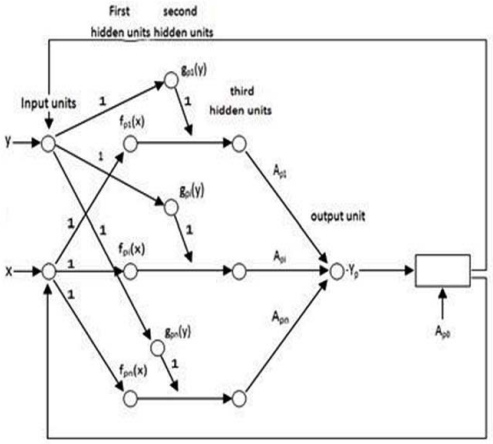

Figure 1: Schematic diagram of the proposed FFNN4.

3 The general method

In this Section, first the system of fuzzy equations is defined. Then we will concentrate on solving below fuzzy system. Here, we consider the system of fuzzy equations in following form:

$$\begin{cases} A_{11}f_{11}(x)g_{11}(y) + \cdots + A_{1n}f_{1n}(x)g_{1n}(y) = A_{10}, \\ \\ A_{21}f_{21}(x)g_{21}(y) + \cdots + A_{2n}f_{2n}(x)g_{2n}(y) = A_{20}, \end{cases} \tag{3.3}$$

where $A_{10}, ..., A_{1n}, A_{20}, ..., A_{2n}$ are fuzzy numbers. Moreover, we assume that f_{pj} and g_{pj} $(for\ p = 1, 2\ and\ j = 1, ..., n)$ are real functions of the crisp variables x and y (if exist), respectively. For simplify the fuzzy coefficient A_{pj} will always be considered real triangular fuzzy number. As observed in previous section to get an approximate solution, an architecture of FFNN4 for fuzzy system (3.3) has been given in Fig. 1. The modeling scheme is designed with the simple and versatile fuzzy neural network architecture.

3.1 Cost function

At first we assume that the real quantities x_0 and y_0 are initialized at random values for unknown variables x and y, respectively. In this subsection we intend to introduce how to deduce a learning algorithm to update the crisp parameters x_0 and y_0. Let A_{p0} be fuzzy target output corresponding to the fuzzy coefficient vector $(A_p = A_{p1}, ..., A_{pn})$. After presenting these parameters into the proposed network, we calculated the fuzzy output by using relations which were introduced in subsection (3.1). Next we defined a cost function for α-level sets of fuzzy output Y_p and its corresponding target output A_{p0} as follows:

$$e_p^\alpha = e_{pl}^\alpha + e_{pu}^\alpha, \tag{3.4}$$

where

$$e_{pl}^\alpha = \alpha.\frac{([A_{p0}]_l^\alpha - [Y_p]_l^\alpha)^2}{2}, \tag{3.5}$$

$$e_{pu}^\alpha = \alpha.\frac{([A_{p0}]_u^\alpha - [Y_p]_u^\alpha)^2}{2}, \tag{3.6}$$

In the cost function (3.4), e_{pl}^α and e_{pu}^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_p and target output A_{p0}, respectively. Then the total error of the given neural network is obtained as [15]:

$$e = \sum_{p=1}^{2}\sum_{\alpha} e_p^\alpha. \tag{3.7}$$

3.1.1 Learning algorithm of the FFNN4

Our main aim is adjusting the parameters x_0 and y_0 with the using of learning algorithm which be introduced in below. For crisp parameter y_0 adjust rule can be written as follows:

$$y_0(t+1) = y_0(t) + \Delta y_0(t), \tag{3.8}$$

$$\Delta y_0(t) = -\eta\frac{\partial e_p^\alpha}{\partial y_0} + \gamma\Delta y_0(t-1), \tag{3.9}$$

where t is the number of adjustments, η is the learning rate and γ is the momentum term constant. The derivative $\frac{\partial e_p^\alpha}{\partial y_0}$ can be calculated from the cost function e_p^α using the input-output relation of our fuzzy neural network for the α-level sets in (2.1)-(2.2). We calculate $\frac{\partial e_p^\alpha}{\partial y_0}$ as follows:

$$\frac{\partial e_p^\alpha}{\partial y_0} = \frac{\partial e_{pl}^\alpha}{\partial y_0} + \frac{\partial e_{pu}^\alpha}{\partial y_0}. \tag{3.10}$$

Thus our problem is calculating of the derivatives $\frac{\partial e_{pl}^\alpha}{\partial y_0}$ and $\frac{\partial e_{pu}^\alpha}{\partial y_0}$. Since they are calculated in the same manner, we only show the calculate $\frac{\partial e_{pl}^\alpha}{\partial y_0}$. So we have:

$$\begin{aligned}\frac{\partial e_{pl}^\alpha}{\partial y_0} &= \frac{\partial e_{pl}^\alpha}{\partial [Y_p]_l^\alpha}\cdot\frac{\partial [Y_p]_l^\alpha}{\partial [Net_p]_l^\alpha}\cdot\frac{\partial [Net_p]_l^\alpha}{\partial y_0}\\ &= -\alpha\ .\ ([A_{p0}]_l^\alpha - [Y_p]_l^\alpha)\ .\ \frac{\partial [Net_p]_l^\alpha}{\partial y_0},\end{aligned}$$

where

$$\frac{\partial [Net_p]_l^\alpha}{\partial y_0} = \sum_{j=1}^{n}(\frac{\partial [Net_p]_l^\alpha}{\partial g_{pj}(y)}\ .\ g'_{pj}(y)|_{y=y_0}).$$

If $U_{pj} \geq 0$

$$\frac{\partial [Net_p]_l^\alpha}{\partial g_{pj}(y)} = ([A_{pj}]_l^\alpha\ .\ f_{pj}(x_0)),$$

otherwise

$$\frac{\partial [Net_p]_l^\alpha}{\partial g_{pj}(y)} = ([A_{pj}]_u^\alpha\ .\ f_{pj}(x_0)).$$

Now we have:

$$\begin{aligned}\frac{\partial e_p^\alpha}{\partial y_0} = -\alpha.\sum_{j\ \epsilon\ M}\{([A_{p0}]_l^\alpha\\ -\ [Y_p]_l^\alpha)\ .\ ([A_{pj}]_l^\alpha\ .\ f_{pj}(x_0).\ g'_{pj}(y)|_{y=y_0})\}\end{aligned}$$

$$\begin{aligned}-\alpha.\sum_{j\ \epsilon\ M}\{([A_{p0}]_u^\alpha\\ -\ [Y_p]_u^\alpha)\ .\ ([A_{pj}]_u^\alpha\ .\ f_{pj}(x_0).\ g'_{pj}(y)|_{y=y_0})\}\end{aligned}$$

$$\begin{aligned}-\alpha.\sum_{j\ \epsilon\ C}\{([A_{p0}]_l^\alpha\\ -\ [Y_p]_l^\alpha)\ .\ ([A_{pj}]_u^\alpha\ .\ f_{pj}(x_0).\ g'_{pj}(y)|_{y=y_0})\}\end{aligned}$$

$$-\alpha.\sum_{j\ \epsilon\ C}\{([A_{p0}]_u^\alpha - [Y_p]_u^\alpha).([A_{pj}]_l^\alpha . f_{pj}(x_0).g'_{pj}(y)|_{y=y_0})\},$$

where $M=\{j|\ U_{pj}\geq 0\}$ and $C=\{j|\ U_{pj}<0\}$. Similarly, we can update the parameter x_0 with using of relations which applied for updating the input signal y_0. Now the learning algorithm can be summarized as follows:

Learning process
Step 1: $\eta>0$, $\gamma>0$, $Emax>0$ are chosen. Then crisp quantities x_0 and y_0 are initialized at random values.
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 $\alpha=\alpha_1,...,\alpha_m$.
Step 4: Repeat the following procedures for $p=1,2$:

[i] Forward calculation: Calculate the α-level set of the fuzzy output Y_p by presenting the α-level set of the fuzzy coefficients vector A_p.

[ii] Back-propagation: Adjust crisp parameters x_0 and y_0 using the cost function (3.4) for the α-level sets of the fuzzy output Y_p and the target output A_{p0}.

Step 5: Cumulative cycle error is computed by adding the present error to E.
Step 6: 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 3*.

4 Numerical examples

In this Section we apply this method on three examples. For each example, the computed values of the approximate solution are calculated over a number of iterations and then the cost function is plotted. In the following simulations, we use the specifications as follows:

1. Values of $\alpha=0,0.1,...,1$.
2. Learning constant $\eta=0.001$.
3. Momentum constant $\gamma=0.001$.
4. Stoping conditions: $Emax=0.001$.

Example 4.1 *Consider the following system of fuzzy equations:*

$$\begin{cases}(-1,0,-1)e^{x-1}(y+1)^2 \\ \quad +(-3,0,3)sin(x-1).cos(y)=(-1,0,1), \\ (1,2,4)e^{x-1}(y+1)^2 \\ \quad +(2,5,6)sin(x-1)cos(y)=(1,2,4).\end{cases}$$

where $x,y\ \epsilon\ \mathbb{R}$ and the exact solution is $x=1$ and $y=0$. In this example, we apply the proposed method to approximate the solution of this fuzzy system. The training starts with $x_0=0.5$ and $y_0=0.5$. Table 1 shows the approximated solutions over number of iterations and Figs. 2 and 4 show the accuracy of the solutions $x_0(t)$ and $y_0(t)$ where t is the number of iterations.

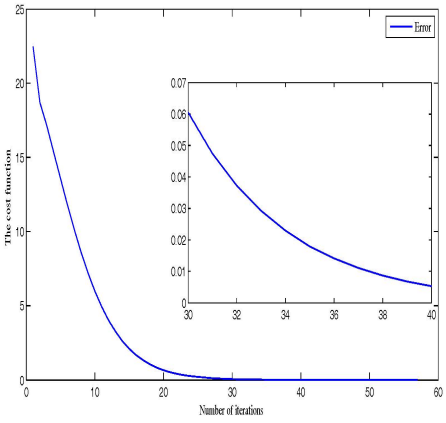

Figure 2: The cost function for Example 4.1.

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

t	$x_0(t)$	$y_0(t)$	e	t	$x_0(t)$	$y_0(t)$	e
1	0.60601	0.48232	22.4739	42	0.99626	0.0057441	0.0032690
2	0.64331	0.44822	18.6951	43	0.99669	0.0050850	0.0025620
3	0.67040	0.41402	17.1529	44	0.99707	0.0045014	0.0020079
4	0.69603	0.38032	15.3804	45	0.99740	0.0039848	0.0015735
5	0.72100	0.34757	13.5776	46	0.99770	0.0035274	0.0012331
6	0.74515	0.31616	11.8196	47	0.99797	0.0031226	0.0009663

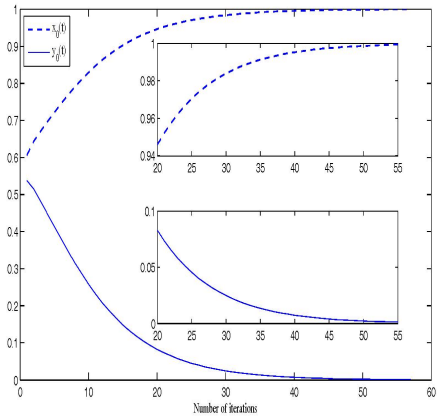

Figure 3: Convergence of the calculated solutions for Example 4.1.

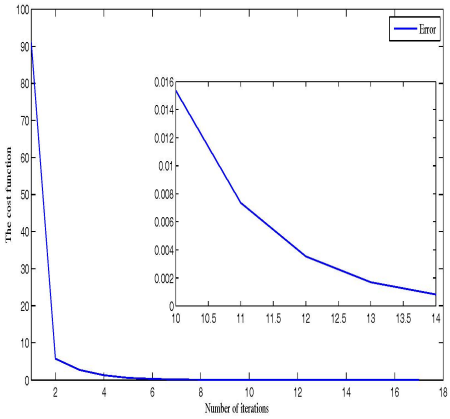

Figure 4: The cost function for Example 4.2.

Example 4.2 *Let fuzzy system*

$$\begin{cases} (1,2,3)xy \\ \quad +(2,3,4)e^{x+1}cos(y-1) = (-1,1,3), \\ (2,3,4)xy^2 \\ \quad +(3,4,5)x^3sin(y-1) = (-4,-3,-2). \end{cases}$$

with the exact solutions $x = -1$ and $y = 1$. We trained the fuzzy neural network as described in last example. Before starting calculations, we assume that $x_0 = -1.5$ and $y_0 = 0.5$. Numerical result can be found in Table 2. Figs. 5 and show the accuracy of the solutions $x_0(t)$ and $y_0(t)$ where t is the number of iterations.

Example 4.3 *We consider the following fuzzy system*

$$\begin{cases} 2\ sin(\frac{\Pi}{2}(x+1))\ cos^3(y)+ \\ \quad (1,3,7)\ tan(\frac{\Pi}{4}(x+1))\ ln^{e(y+1)^2} = (3,5,9), \\ (3,7,8)\ sin(\frac{\Pi}{2}(x+1))\ cos^3(y)+ \\ \quad 3\ tan(\frac{\Pi}{4}(x+1))\ ln^{e(y+1)^2} = (6,10,11). \end{cases}$$

with the exact solutions $x = 0$ and $y = 0$. We trained the fuzzy neural network as described in previous examples. Before starting calculations, we assume that $x_0 = 0.5$, $y_0 = -0.5$, $\eta = 0.001$ and $\gamma = 0.001$. Numerical result can be found in Table 3. Figs. 6 and 7 show the accuracy of the solutions $x_0(t)$ and $y_0(t)$ where t is the number of iterations.

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

t	$x_0(t)$	$y_0(t)$	e	t	$x_0(t)$	$y_0(t)$	e
1	-1.1990	0.94837	91.1527	9	-1.0099	0.99594	0.03206490
2	-1.1365	0.95968	5.76155	10	-1.0068	0.99717	0.01535220
3	-1.0933	0.96933	2.75913	11	-1.0047	0.99803	0.00735863
4	-1.0638	0.97736	1.31074	12	-1.0033	0.99863	0.00353004
5	-1.0437	0.98362	0.62170	13	-1.0023	0.99905	0.00169440
6	-1.0301	0.98831	0.29528	14	-1.0016	0.99934	0.00081363

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

t	$x_0(t)$	$y_0(t)$	e	t	$x_0(t)$	$y_0(t)$	e
1	0.06917	-0.04241	111.4537	160	0.03242	-0.02320	0.0010458
2	0.06731	-0.04392	0.022291	161	0.03034	-0.02115	0.0010358
3	0.06663	-0.04367	0.016967	162	0.02826	-0.02280	0.0010259
4	0.06598	-0.04322	0.016318	163	0.02600	-0.02225	0.0010162
5	0.06535	-0.04287	0.015722	164	0.02510	-0.02159	0.0010067
6	0.06473	-0.04233	0.015159	165	0.02453	-0.02094	0.00099727

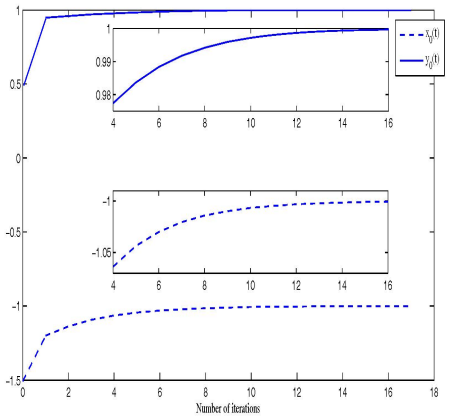

Figure 5: Convergence of the calculated solutions for Example 4.2.

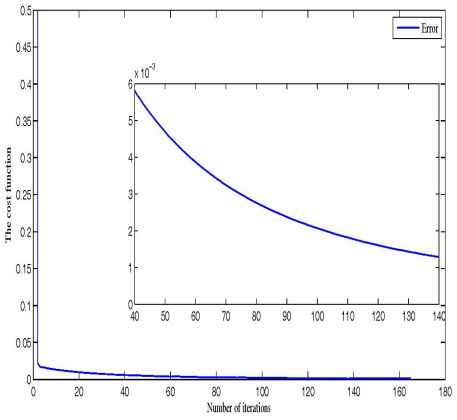

Figure 6: The cost function for Example 4.3.

5 Conclusion

In this paper, an architecture of feed-back neural networks has been proposed for approximating solution of a fuzzy equations system. we suggested a FFNN4 model equivalent to the system of fuzzy equations. The analyzed examples illustrated the ability and reliability of the present method. The obtained solutions, in comparison with exact solutions admitted 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 of computations.

References

[1] S. Abbasbandy, M. Alavi, *A method for solving fuzzy linear systems*, Iranian Journal of Fuzzy Systems 2 (2005) 37-43.

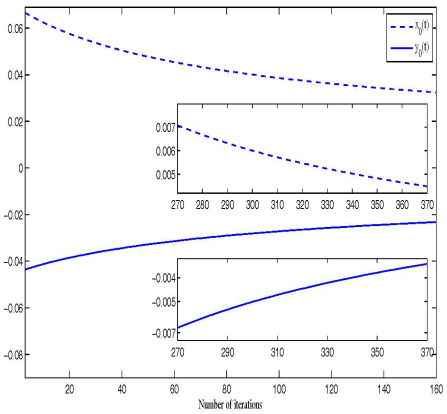

Figure 7: Convergence of the calculated solutions for Example 4.3.

[2] S. Abbasbandy, M. Otadi, *Numerical solution of fuzzy polynomials by fuzzy neural network*, Appllied Mathematical and Computation 181 (2006) 1084-1089.

[3] S. Abbasbandy, R. Ezzati, *Newton's method for solving a system of fuzzy nonlinear equations*, Appllied Mathematical and Computation 175 (2006) 1189-1199.

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

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

[6] T. Allahviranloo, *Numerical methods for fuzzy system of linear equations*, Appllied Mathematical and Computation 155 (2004) 493-502.

[7] T. Allahviranloo, *The Adomian decomposition method for fuzzy system of linear equations*, Appllied Mathematical and Computation 163 (2005) 553-563.

[8] T. Allahviranloo, *Successive over relaxation iterative method for fuzzy system of linear equations*, Appllied Mathematical and Computation 162 (2005)189-196.

[9] B. Asady, S. Abbasbandy, M. Alavi, *Fuzzy general linear systems*, Appllied Mathematical and Computation 169 (2005) 34-40.

[10] M. Dehgan, B. Hashemi, M. Ghatee, *Solution of the fully fuzzy linear systems using iterative techniques*, Chaos Solitons and Fractals 26 (2005) 835-843.

[11] D. Dubois, H. Prade, *Systems of linear fuzzy constraints*, Fuzzy Sets and Systems 3 (1980) 37-48.

[12] M. Friedman, M. Ming, A. Kandel, *Fuzzy linear systems*, Fuzzy Sets and Systems 96 (1998) 201-209.

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

[14] Y. Hayashi, J. J. Buckley, E. Czogala, *Fuzzy neural network with fuzzy signals and weights*, International Journal of Nonlinear Sciences and Numerical Simulation 8 (1993) 527-537.

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

[16] H. T. Nguyen, *A note on the extension principle for fuzzy sets*, Journal of Mathematical Analysis and Applications 64 (1978) 369-380.

[17] L. A. Zadeh, *Toward a generalized theory of uncertainty (GTU) an outline*, Information Sciences 172 (2005) 1-40.

[18] L. A. Zadeh, *The concept of a liguistic variable and its application to approximate reasoning: Parts 1-3*, Information Sciences 8 (1975) 199-249.

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

Ahmad Jafarian was born in 1978 in Tehran, Iran. He received B. Sc (1997) in Applied mathematics and M.Sc. in applied mathematics from Islamic Azad University Lahi-jan Branh to Lahijan, Ferdowsi University of Mashhad respectively. He is a Assistant Prof. in the department of mathematics at Islamic Azad University, Urmia Branch, roumieh, in Iran. His current interest is in artificial intelligence, solving nonlinear problem and fuzzy mathematics.

Safa Measoomy nia received the M.Sc. degree in Numerical Analysis from The Islamic Azad University, Urmia, Iran. At present, his research interests are in the fields of fuzzy mathematics, integral equations and artificial neural networks.