Menemui Matematik (Discovering Mathematics)
Vol. 40, No. 1: 1-10 (2018)

Learning Algorithm of Fuzzy Neural Network for Solving Trapezoidal Fuzzy Polynomial Equation

Nurhakimah Ab.Rahman[1] , Lazim Abdullah[2] and Ahmad Termimi Ab Ghani[3]
[1,2,3] *School of Informatics and Applied Mathematics, Universiti Malaysia Terengganu, 21030 Kuala Terengganu, Terengganu*
[1]hamikahrun@yahoo.com, [2]lazim_m@umt.edu.my, [3]termimi@umt.edu.my

ABSTRACT

Fuzzy polynomial equations have been solved by using many computational methods, including the fuzzy neural network. However, it is limited to triangular membership function. While, trapezoidal fuzzy numbers can handle uncertainty and vagueness, similar to the triangular fuzzy number. Hence, this study wishes to introduce a learning algorithm of fuzzy neural network to solve the trapezoidal fuzzy polynomial equation in the form of $A_1x + A_2x^2 + ... + A_nx^n = A_0$. The learning algorithm comprises fuzzy coefficients $A_i\left(i = 1,..., n \right)$ and fuzzy target output A_0 that based on the use of a fuzzified neural network. To illustrate the efficiency of the learning algorithm, two numerical examples of trapezoidal fuzzy polynomial equation are presented. As a result, the trapezoidal fuzzy polynomial equation is efficiently produces the approximate solution by using a learning algorithm of fuzzy neural network.

Keywords: Fuzzy neural network, Fuzzy polynomial equation, Fuzzy numbers

INTRODUCTION

Fuzzy polynomials (FPs) have been used by many researchers for solving many problems in economics, engineering, physics, and finance. Since then, various approaches for solving FPs problems have been introduced and one of the recent approaches is fuzzy neural networks (FNN). The FNN model has been rapidly developed not only for solving various types of mathematical equations but also have been used in various fields (Tahavvor and Yaghoubi, 2012). In FNN, a cost function was defined for every pair of fuzzy output vector and fuzzy target vector (Ishibuchi *et al*., 1995). Therefore for the first time, Buckley and Qu (1991) have applied a structure of FNN in solving fuzzy equations. Then, this approach was extended further by the same researcher (Buckley and Eslami, 1997; Buckley *et al*., 2002). Besides that, the applications of FNN have been summarized in the fuzzy expert systems, fuzzy hierarchical analysis, and fuzzy systems modeling (Hayashi et *al.,* 1993). Years after this breakthrough, linear and nonlinear fuzzy equations have been solved numerically by Abbasbandy and Asady (2004), Abbasbandy and Alavi (2005), Abbasbandy and Ezzati (2006), Asady *et al.* (2005) and Abbasbandy and Otadi (2006a). After a few years, FNN has been used to obtain the approximate solution for dual FPs (Jafarian and Jafari, 2012).

In this study, an architecture of FNN equivalent to a fuzzy equation of the form $A_1x + A_2x^2 + ... + A_nx^n = A_0$ was built, where $A_0, A_1, ..., A_n$ are FNs. The proposed neural network had two layers, in which the input-output relation of each unit was

defined by the extension principle (Zadeh, 1975). The coefficients of the fuzzy equation were considered as input signals, while the right-hand FNs were considered as a target output. The output from the neural network was numerically compared with the target output. Learning algorithm of FNN was employed to find a real root for trapezoidal fuzzy polynomial equations (FPE). The trapezoidal fuzzy number that characterized by interval defuzzier, left fuzziness and right fuzziness has a great potential for obtaining approximate solutions using learning algorithm of FNN. It is anticipated that polynomials with trapezoidal fuzzy numbers would yield a new approximate solution for FPs. Taking the characteristics of trapezoidal fuzzy numbers and the vagueness requirements of FNN learning algorithm, this paper seeks to propose approximate solutions for FPE using a learning algorithm where the coefficients are trapezoidal fuzzy numbers.

The organization of this study is presented as follows. In Section 2, the necessary preliminaries for defining fuzzy numbers is introduced. Then, Section 3 presents the learning algorithm of FNN for solving FPE. To illustrate the efficiency of the proposed method, two numerical examples are considered in Section 4. Finally, conclusions come in Section 5.

PRELIMINARIES

Definition 1: (Goetschel and Voxman, 1986) A fuzzy number is defined by fuzzy set $u : R \to [0,1]$ which satisfies the following:

I. u is upper semicontinuous.
II. $u(x) = 0$ outside some interval $[a, d]$.
III. There are real numbers $b, c : a \le b \le c \le d$, for which
  i. $u(x)$ is monotonic increasing on $[a, b]$.
  ii. $u(x)$ is monotonic decreasing on $[c, d]$.
  iii. $u(x) = 1,\ b \le x \le c$

Definition 2: (Ma *et al*., 1999; Kaleva, 1987; Jafarian and Nia, 2011) A pair of function $(\underline{u}, \overline{u})$ with $r \in R$ is called a FN if satisfying the following conditions.

I. $\underline{u}(r)$ is a bounded monotonic increasing left continuous function.
I. $\overline{u}(r)$ is a bounded monotonic decreasing left continuous function.
II. $\underline{u}(r) \le \overline{u}(r),\ 0 \le r \le 1$

Definition 3: (Ma *et al*., 1999) The trapezoidal FNs $\mu_{trap} = (x_0, y_0, \alpha, \beta)$ with two defuzzifier, x_0, y_0 and left fuzziness $\alpha > 0$ and right fuzziness $\beta > 0$ is a fuzzy set where the membership function is as

$$\mu_{trap}(x)=\begin{cases}\frac{1}{\alpha}(x-x_0+\alpha) & if \quad x_0-\alpha\le x\le x_0\\ 1 & if \quad x\in[x_0,y_0]\\ \frac{1}{\beta}(y_0-x+\beta) & if \quad y_0\le x\le y_0+\beta\\ 0 & if \quad otherwise\end{cases}$$

The parametric form for trapezoidal FNs can be obtained as below.

$$\underline{\mu}_{trap}(x)=x_0-\alpha+\alpha x \quad , \quad \overline{\mu}_{trap}(x)=y_0+\beta-\beta x$$

Operation on FNN

FNs operations are defined by the extension principle (Zadeh, 1975; 2005)

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

where A and B are FNs, $\mu_*(\cdot)$ denotes the membership function of each FNs, $\wedge$ is the minimum operator, and f is a continuous activation function (such as $f(x)=x$) of FNN 's output unit .

The above operations on FNs were numerically performed on level sets ($\alpha-$cuts). For $0<\alpha\le 1$, an $\alpha-$level set of a FNs A is defined as $[A]^\alpha=\{x|\mu_A(x)\ge\alpha,\ x\in R\}$ and $[A]^0=\overline{\mathrm{Y}_{\alpha\in(0,1]}[A]^\alpha}$. Since level sets of FNs become closed intervals, $[A]^\alpha$ is denoted by $[A]^\alpha=\left[[A]_l^\alpha,[A]_u^\alpha\right]$ where $[A]_l^\alpha$ and $[A]_u^\alpha$ are the lower and the upper limits of the $\alpha-$level set $[A]^\alpha$, respectively. Based on interval arithmetic, the operations on FNs are written for the $\alpha-$level sets as follows (Alefeld and Herzberger, 1983):

$$[A]^\alpha+[B]^\alpha=[[A]_l^\alpha,[A]_u^\alpha]+[[B]_l^\alpha,[B]_u^\alpha]=[[A]_l^\alpha+[B]_l^\alpha,[A]_u^\alpha+[B]_u^\alpha],$$
$$f([Net]^\alpha)=f([Net]_l^\alpha,[Net]_u^\alpha)=[f([Net]_l^\alpha),f([Net]_u^\alpha)]$$
(1)
$$k[A]^\alpha=k[[A]_l^\alpha,[A]_u^\alpha]=[k[[A]_l^\alpha,k[A]_u^\alpha]],\ \ if\ \ k\ge 0,$$
$$k[A]^\alpha=k[[A]_l^\alpha,[A]_u^\alpha]=[k[[A]_u^\alpha,k[A]_l^\alpha]],\ \ if\ \ k<0,$$
(2)

According to Nguyen (1978) and Goetschel and Voxman (1986), addition $(u+v)$ and multiplication by k for arbitrary $u=(\underline{u},\overline{u})$ and $v=(\underline{v},\overline{v})$ are defined as:

$$\overline{(u+v)}(r)=\overline{u}(r)+\overline{v}(r)$$
$$\underline{(u+v)}(r)=\underline{u}(r)+\underline{v}(r)$$
$$\overline{(ku)}(r)=k\cdot\overline{u}(r),\ \ \underline{(kv)}(r)=k\cdot\underline{u}(r),\ \ if\ \ k\ge 0,$$
$$\overline{(ku)}(r)=k\cdot u(r),\ \ \underline{(kv)}(r)=k\cdot\overline{u}(r),\ if\ \ k<0.$$

Then, a real solution of the FPE (if exist) in general form can be found by

$$A_1 f_1(x) + ... + A_n f_n(x) = A_0 \quad (3)$$

where $A_i \in E^1$ and $f_i(x)(i = 1,...,n)$ are real functions.

In order to obtain an approximate solution, an architecture of FNN equivalent to Equation (3) is built. The network is shown in Figure 1.

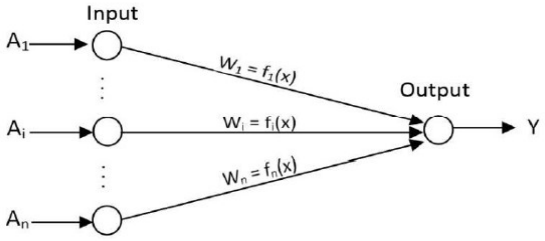

Figure 1: The proposed neural network (Abbasbandy and Otadi, 2006b)

Input Output Relation of Each Unit

A two layer FNN with *n* input neurons and one output neuron were considered. It is clear that the input vector, the target output of the triangular FNs and the connection weights are crisp numbers. When a fuzzy input vector $A = (A_1, A_2,..., A_n)$ is presented to the FNN, then the input-output relation of each unit can be written as follows.

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

$$O_i = A_i \,,\ i = 1,2,...,n \quad (4)$$

Output units:

$$Y = f(Net)$$
$$Net = \sum_{i=1}^{n} (W_i . O_i) \quad (5)$$

where A_i was a trapezoidal FNs and W_i was a crisp connection weight. The relations between the input neurons and the output neurons in Equation (4)-(5) are defined by the extension principle by Zadeh (1975) as in (Hayashi *et al*.; Ishibuchi *et al*., 1993).

Calculation of Fuzzy Output

The fuzzy output from a neuron in the second layer is numerically calculated for crisp weights and level sets of fuzzy inputs. The input-output relations of the FNN as shown in Figure 1 can be written for the $\alpha-$level sets as follows:

Input units:

$$[O_i]^{\alpha} = [A_i]^{\alpha}, \ i = 1,...,n \tag{6}$$

Output unit:

Let f be a one-to-one activation function. Then,

$$[Y]^{\alpha} = f([Net]^{\alpha}),$$
$$[Net]^{\alpha} = \sum_{i=1}^{n} (W_i \cdot [O_i]^{\alpha}) \tag{7}$$

From Equations (6)-(7), $\alpha-$level sets of the fuzzy output Y are calculated from those of the fuzzy inputs and crisp weights. From Equations (1)-(2), the above relations are transformed to the following form:

Input units:

$$[O_i]^{\alpha} = [[O_i]_l^{\alpha}, [O_i]_u^{\alpha}] = [[A_i]_l^{\alpha}, [A_i]_u^{\alpha}], \ i = 1,...,n$$

Output unit:

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

where

$$[Net]^{\alpha} = [[Net]_l^{\alpha}, [Net]_u^{\alpha}] = [\sum_{i \in M} (W_i \cdot [O_i]_l^{\alpha}) + \sum_{i \in C} (W_i \cdot [O_i]_u^{\alpha}), \sum_{i \in M} (W_i \cdot [O_i]_u^{\alpha}) + \sum_{i \in C} (W_i \cdot [O_i]_l^{\alpha})]$$

and

$$M = \{i | W_i \geq 0\}, \ C = \{i | W_i < 0\}, M \bigcup C = \{1,...,n\}$$

The architecture of solution FNN to Equation (3) are given in Figure 1.

LEARNING ALGORITHM OF FUZZY NEURAL NETWORK

Let a real quantity x_0 be initialized at random value for a variable x. Then update the crisp weights W_j ($for\ j = 1,...,n$) so that $W_j = x^j$. Adjust the parameter x_0 and then update weights W_j ($for\ j = 1,...,n$) using x_0. For crisp parameter x_0 the adjusted rule can be written as follows:

$$x_0(t+1) = x_0(t) + \Delta x_0(t) \tag{8}$$

$$\Delta x_0(t) = -\eta \cdot \frac{\partial e^{\alpha}}{\partial x_0} + \gamma \cdot \Delta x_0(t-1) \tag{9}$$

where t is the number of adjustments, η is the learning rate and γ is the momentum term constant. Thus, the problem would be to calculate the derivative $\frac{\partial e^{\alpha}}{\partial x_0}$ in (9). The derivative $\frac{\partial e^{\alpha}}{\partial x_0}$ can be calculated from the cost function e^{α} using the input-output relation of the FNN. The $\frac{\partial e^{\alpha}}{\partial x_0}$ is calculated as follows:

$$\frac{\partial e^{\alpha}}{\partial x_0} = \frac{\partial e_l^{\alpha}}{\partial x_0} + \frac{\partial e_u^{\alpha}}{\partial x_0} \tag{10}$$

where

$$\frac{\partial e_l^{\alpha}}{\partial x_0} = \left(\frac{\partial e_l^{\alpha}}{\partial [Y]_l^{\alpha}} \cdot \frac{\partial [Y]_l^{\alpha}}{\partial [Net]_l^{\alpha}} \cdot \frac{\partial [Net]_l^{\alpha}}{\partial W_1} \cdot \frac{\partial W_1}{\partial x_0} \right) + \ldots + \left(\frac{\partial e_l^{\alpha}}{\partial [Y]_l^{\alpha}} \cdot \frac{\partial [Y]_l^{\alpha}}{\partial [Net]_l^{\alpha}} \cdot \frac{\partial [Net]_l^{\alpha}}{\partial W_j} \cdot \frac{\partial W_j}{\partial x_0} \right)$$
$$+ \ldots + \left(\frac{\partial e_l^{\alpha}}{\partial [Y]_l^{\alpha}} \cdot \frac{\partial [Y]_l^{\alpha}}{\partial [Net]_l^{\alpha}} \cdot \frac{\partial [Net]_l^{\alpha}}{\partial W_n} \cdot \frac{\partial W_n}{\partial x_0} \right)$$

and

$$\frac{\partial e_u^{\alpha}}{\partial x_0} = \left(\frac{\partial e_u^{\alpha}}{\partial [Y]_u^{\alpha}} \cdot \frac{\partial [Y]_u^{\alpha}}{\partial [Net]_u^{\alpha}} \cdot \frac{\partial [Net]_u^{\alpha}}{\partial W_1} \cdot \frac{\partial W_1}{\partial x_0} \right) + \ldots + \left(\frac{\partial e_u^{\alpha}}{\partial [Y]_u^{\alpha}} \cdot \frac{\partial [Y]_u^{\alpha}}{\partial [Net]_u^{\alpha}} \cdot \frac{\partial [Net]_u^{\alpha}}{\partial W_j} \cdot \frac{\partial W_j}{\partial x_0} \right)$$
$$+ \ldots + \left(\frac{\partial e_u^{\alpha}}{\partial [Y]_u^{\alpha}} \cdot \frac{\partial [Y]_u^{\alpha}}{\partial [Net]_u^{\alpha}} \cdot \frac{\partial [Net]_u^{\alpha}}{\partial W_n} \cdot \frac{\partial W_n}{\partial x_0} \right)$$

If $W_j \geq 0$,

$$\frac{\partial e_l^{\alpha}}{\partial W_j} = -\alpha \cdot \left([A_0]_l^{\alpha} - [Y]_l^{\alpha} \right) \cdot [A_j]_l^{\alpha} \ , \ \& \ \frac{\partial e_u^{\alpha}}{\partial W_j} = -\alpha \cdot \left([A_0]_u^{\alpha} - [Y]_u^{\alpha} \right) \cdot [A_j]_u^{\alpha} \ , \ j = 1,\ldots,n$$

otherwise

$$\frac{\partial e_l^{\alpha}}{\partial W_j} = -\alpha \cdot \left([A_0]_l^{\alpha} - [Y]_l^{\alpha} \right) \cdot [A_j]_u^{\alpha} \ \& \ \frac{\partial e_u^{\alpha}}{\partial w_j} = -\alpha \cdot \left([A_0]_u^{\alpha} - [Y]_u^{\alpha} \right) \cdot [A_j]_l^{\alpha} \ .$$

Consequently,

$$\Delta x_0(t) = \eta \cdot \alpha \cdot \sum_{j \in M} \left\{ \left([A_0]_l^{\alpha} - [Y]_l^{\alpha} \right) \cdot [A_j]_l^{\alpha} + \left([A_0]_u^{\alpha} - [Y]_u^{\alpha} \right) \cdot [A_j]_u^{\alpha} \right] \cdot \left(j\left(x_0(t)^{j-1} \right) \right) \Big\} +$$
$$\eta \cdot \alpha \cdot \sum_{j \in C} \left\{ \left([A_0]_l^{\alpha} - [Y]_l^{\alpha} \right) \cdot [A_j]_u^{\alpha} + \left([A_0]_u^{\alpha} - [Y]_u^{\alpha} \right) \cdot [A_j]_l^{\alpha} \right] \cdot \left(j\left(x_0(t)^{j-1} \right) \right) \Big\} + \gamma \cdot \Delta x_0(t-1) \tag{11}$$

where $M = \{ j | W_j \geq 0 \}$ and $C = \{ j | W_j < 0 \}$.

After adjusting x_0 by (8) and (9), connection weights $W_i \left(for \ j = 1,\ldots,n \right)$ are updated with the FNN model as follows.

$$W_j(t+1) = f_j\left(x_0(t+1) \right), \ j = 1,\ldots,n \tag{12}$$

Let us assume that the input-output pair $A; A_0$ where $A = (A_1,..., A_n)$ are given as training data and also m values of α -level sets $(\alpha_1, \alpha_2,..., \alpha_m)$ are used for learning the FNN. Then the learning algorithm can be summarized as follows.

Step 1. Choose $\eta > 0, \gamma > 0$. Then initialized $x_0(0)$ at random value.
Step 2. Let $t := 0$ where t is the number of iterations of the learning algorithm.

Step 3. Calculate the crisp connection weights.
$W_i(t) = (x_0)^i$, $i = 1,...,n$
Step 4. Find $[Net\,L]$and $[Net\,U]$.
Step 5. Calculates $x_0(t+1)$.
Step 6. Let $t := t+1$. Repeat Steps 3 to 5 until the approximate solution is obtained.

NUMERICAL EXAMPLES

In the above section, a step-by-step presentation of an algorithm for the fuzzy neural network was outlined. Hence, in this section, the study will consider two numerical examples, which are FPE and DFPE.

Example 1
In this example, the study will consider the trapezoidal membership function of FPE.
$(0.1,0.25,0.3,0.5)x + (-0.2,0,1,1.5)x^2 + (0.15,0.25,0.35,0.45)x^3 = (0.05,0.5,1.65,2.45)$

The study trains this example with three input unit and a single output. Let $x_0 = 0.5,\ \alpha = 0.5,\ \eta = 0.2,\ \gamma = 0.2$. Hence, details of calculation are shown as below.

Step 1. The parametric forms are as follows.

$[A_1]_l^\alpha = -0.05$ $[A_2]_l^\alpha = -0.7$ $[A_3]_l^\alpha = -0.025$ $[A_0]_l^\alpha = -0.775$

$[A_1]_u^\alpha = 0.5$ $[A_2]_u^\alpha = 0.75$ $[A_3]_u^\alpha = 0.475$ $[A_0]_u^\alpha = 1.725$

Step 2. Find $[Net\,L]$and $[Net\,U]$.

$$[Net\,L] = (0.5)^1(-0.05) + (0.5)^2(-0.7) + (0.5)^3(-0.025) = -0.203125$$
$$[Net\,U] = (0.5)^1(0.5) + (0.5)^2(0.75) + (0.5)^3(0.475) = 1.725$$

Step 3.

$$\Delta x_0(0) = (0.2)(0.5)\begin{bmatrix} ((-0.775+0.203125)(-0.05) + (1.725-0.496875)(0.5))\cdot(1)(0.5)^0 + \\ ((-0.775+0.203125)(-0.7) + (1.725-0.496875)(0.75))\cdot(2)(0.5)^1 + \\ ((-0.775+0.203125)(-0.025) + (1.725-0.496875)(0.475))\cdot(3)(0.5)^2 \end{bmatrix} + (0.2)(0)$$
$$= 0.2412304687$$

$$x_0(1) = x_0(0) + \Delta x_0(0)$$
$$= 0.5 + 0.2412304687$$
$$= 0.7412304687$$

Step 4. Repeat Step 2 and Step 3 until the approximate solution is obtained.

The following Table 1 shows the approximate solutions over a number of iterations.

Table 1: The approximate solutions for Example 1.

t	$x_0(t)$	e	t	$x_0(t)$	e
0	0.7412304687	0.258769531	7	0.9991514494	0.0008485506
1	0.9361450867	0.063854913	8	0.9998705948	0.0001294052
2	1.016096126	0.016096126	9	1.000100779	0.000100779
3	1.021237906	0.021237906	10	1.000079554	0.000079554
4	1.007907414	0.007907414	11	1.000022214	0.000022214
5	0.9999384970	0.000061503	12	0.9999959210	0.000004079
6	0.9983857575	0.0016142425	13	0.9999933847	0.0000066153

As can be seen from the Table 1, the trapezoidal fuzzy polynomial equation produces the approximate solution after 13 iterations, which is $x = 0.9999933847$.

Example 2

Let consider the following trapezoidal fuzzy polynomial equation,
$(0.1,0.3,0.3,0.5)x + (0.2,0.3,0.3,0.4)x^2 = (0.3,0.6,0.6,0.9)$
The study considers this example with two input unit and a single output. This example starts with $x_0 = 0.8,\ \alpha = 0.5,\ \eta = 0.5,\ \gamma = 0.5$. Hence, Table 2 presents the approximate solutions over a number of iterations.

Table 2: The approximate solutions for Example 2.

t	$x_0(t)$	e	t	$x_0(t)$	e
0	0.86894	0.13106	13	1.001254776	0.001254776
1	0.9496841681	0.050315831	14	1.001103482	0.001103482
2	1.008315195	0.008315195	15	1.000620502	0.000620502
3	1.034554001	0.034554001	16	1.00015	0.00015
4	1.034777909	0.034777909	17	0.9998593964	0.0001406036
5	1.021909860	0.02190986	18	0.9997659749	0.0002340251
6	1.007332772	0.007332772	19	0.9998056127	0.0001943873
7	0.9973319021	0.0026680979	20	0.9998971560	0.000102844
8	0.9933151304	0.0066848696	21	0.9999808758	0.0000191242
9	0.9937680318	0.0062319682	22	1.000029792	0.000029792
10	0.9962893601	0.0037106399	23	1.000043257	0.000043257
11	0.9989175776	0.0010824224	24	1.000034027	0.000034027
12	1.000630958	0.00630958			

The approximate solution is obtained after 24 iterations, which is $x = 1.000034027$

CONCLUSIONS

This study has proposed a learning algorithm of FNN for solving trapezoidal FPE. A step-by-step of the proposed algorithm has been shown. Hence, to illustrate the efficiency of the learning algorithm, two numerical examples were provided. Numerical results indicate that the number of iterations is influenced by the initial value $x_0(0)$, α , η and γ . The learning algorithm FNN which imitates the trapezoidal FPE has produced a series of approximate solutions. The trapezoidal fuzzy numbers used in the input of FNN have successfully shown a remarkable accuracy. An exploration related to other types of fuzzy numbers and learning algorithms are among the potential research for the future.

ACKNOWLEDGEMENTS

We would like to thank the School of Informatics and Applied Mathematics, Universiti Malaysia Terengganu and also the Ministry of Higher Education Malaysia under MyBrain15 program for partially financing this research.

REFERENCES

Abbasbandy, S. and Alavi, M. (2005), A method for solving fuzzy linear systems. *Iranian J. Fuzzy Set*, **2**: 37-43.

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

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

Abbasbandy, S. and Otadi, M. (2006a), Fuzzy neural network approach to fuzzy polynomials. *Mathware Soft Comput.*, **13**: 127-134.

Abbasbandy, S. and Otadi, M. (2006b), Numerical solution of fuzzy polynomials by fuzzy neural network. *Appl. Math. Comput.*, **181**: 1084-1089.

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

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

Buckley, J.J. and Eslami, E. (1997), Neural net solutions to fuzzy problems: the quadratic equation. *Fuzzy Set Syst.*, **86**: 289-298.

Buckley, J.J. and Qu, Y. (1991), Solving fuzzy equations: A new solution concept. *Fuzzy Set Syst.*, **39(3)**: 291-301.

Buckley, J.J., Feuring, T. and Hayashi, Y. (2002), Solving fuzzy equations using evolutionary algorithms and neural nets. *Soft Comput.,* **6(2)**: 116-123.

Goetschel, R. and Voxman, W. (1986), Elementary fuzzy calculus. *Fuzzy Set Syst.*, **18(1)**: 31-43.

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

Ishibuchi, H., Kwon, K. and Tanaka, H. (1995), A learning algorithm of fuzzy neural networks with triangular fuzzy weights. *Fuzzy Set Syst.,* **71(3)**: 277-293.

Ishibuchi, H., Okada, H. and Tanaka, H. (1993b), Fuzzy neural networks with fuzzy weights and fuzzy biases. In *Proceeding of IEEE International Conference on Neural Networks*, pp. 1650-1655.

Jafarian, A. and Jafari, R. (2012), Approximate solutions of dual fuzzy polynomials by feed-back neural networks. *J. Soft Comput. Appl.*, doi:10.5899/2012/jsca-00005.

Jafarian, A. and Nia, S.M. (2011), Solving fuzzy polynomials using neural nets with a new learning algorithm. *Australian J. Basic and Appl. Sci.*, **5(9)**: 2295-2301.

Kaleva, O. (1987), Fuzzy differential equations. *Fuzzy Set Syst.*, **24**: 301-317.

Ma, M., Friedman, M, and Kandel, A. (1999), A new fuzzy arithmetic. *Fuzzy Set Syst.*, **108**: 83-90.

Nguyen, H.T. (1978), A note on the extension principle for fuzzy set. *J Math. Anal. Appl.*, **64(2)**: 369-380.

Tahavvor, A.R and Yaghoubi, M. (2012), Analysis of natural convection from a column of cold horizontal cylinders using artificial neural network. *Appl. Math. Model.,* **36(7)**: 3176-3188.

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

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