

Neuro-fuzzy networks with hybrid learning

Hamid Beigy⁺ & Ali Mohammadzadeh Eydgahi^{*}

^{}Department of Electrical and Computer Eng., College of Eng.,
University of Tehran, Tehran, Iran*

*⁺Department of Computer Eng., Teacher Training University,
Tehran, Iran*

EMail: eydgahi@ece.ut.ac.ir

Abstract

In this paper, two neuro-fuzzy networks based on fuzzy logic and elastic fuzzy logic with their learning algorithms have been proposed. The learning process is based on the back propagation algorithm. To speed-up the learning process three learning algorithms using a momentum term are developed. An example of two link planar robot is used to show usefulness of the proposed networks and their learning algorithms.

1 Introduction

Fuzzy logic systems could be considered as one of the best tool for modeling the human reasoning that is approximate and partial in essence. Neural networks are computational paradigm which are suitable to handle partial and approximate reasoning [1]. There are increasing interests in combining these two techniques to create intelligent systems. One of the idea is to replace the classical neuron that is used in neural networks with a logical and/or neuron called fuzzy neuron [2] and take advantage of the learning capability of neural networks to tune the membership functions used in fuzzy logic systems. This combination creates the concept of fuzzy neural network [1],[3]. The fuzzy neural networks use the fuzzy logic framework and one of the neural network learning methods. Another interest is to translate the fuzzy logic system to neural network framework and use a neural network learning method.

In most of the proposed networks the membership functions are adapted. This adaptation may change the means of a linguistic word which is defined based on the human experiences. To overcome this problem the elastic fuzzy

logic has been proposed [4]. In elastic fuzzy logic, rules and input variables have weights called elasticities. These elasticity parameters represent the validity of a rule or importance of an input variable. The fuzzy logic network and elastic fuzzy logic network are capable of using both linguistic description and numerical information of the system under control.

In the fuzzy logic system an expert provides a set of N_R rules in a simple *IF-THEN* form. Each rule could have one of the following forms [5]:

$$1- \text{ If } X_1 \text{ is } A_1 \text{ and } \dots \text{ and } X_n \text{ is } A_n, \text{ Then } U = B \quad (1)$$

Where $A_1, A_2, \dots$, and B are linguistic labels .

$$2- \text{ If } X_1 \text{ is } A_1 \text{ and } \dots \text{ and } X_n \text{ is } A_n, \text{ Then } U = g(X) \quad (2)$$

Where g is a linear or nonlinear function of $X_1, X_2, \dots$, and X_n . In most cases g is a linear function of $X_1, X_2, \dots, X_n$ in the form of :

$$g(X_1, X_2, \dots, X_n) = P_0 + P_1 * X_1 + \dots + P_n * X_n$$

$$3- \text{ If } X_1 \text{ is } A_1 \text{ and } \dots \text{ and } X_n \text{ is } A_n, \text{ Then } U = P_0 \quad (3)$$

This case is a simplified form of equation (2) with $P_1 = P_2 = \dots = P_n = 0$.

The fuzzy system designer receives these rules and membership functions from an expert and then implements the following relations in the actual network:

$$X_i = \mu_{A_i}(X) \quad i = 1, \dots, N_A \quad (4)$$

$$R_r = X_{r,1} * X_{r,2} * \dots * X_{r,N_r} \quad r = 1, \dots, N_R \quad (5)$$

$$R = \sum_{r=1}^{N_R} R_r \quad (6)$$

$$U = \frac{\sum_{r=1}^{N_A} (R_r U_r)}{R} \quad (7)$$

Where X is an input vector, N_A shows the number of input linguistic labels, N_R represents the number of rules, N_r is the number of input linguistic labels in r th rule, $X_{r,j}$ gives j th input variable in the r th rule, R_r shows firing

degree of r th rule to input vector, U_r gives the output of r th rule, and $*$ denotes T-norm operator.

In the elastic fuzzy logic, equation (5) is replaced by

$$R_r = \gamma_{r,0} * X_{r,1}^{\gamma_{r,1}} * \dots * X_{r,N_r}^{\gamma_{r,N_r}} \quad r = 1, \dots, N_R \quad (8)$$

Where gamma parameters are called *elasticities* and the $\gamma_{r,0}$ represents the degree of strength of validity of r th rule. The $\gamma_{r,i}$ represents the degree of importance of j th input condition in r th rule.

Elastic fuzzy logic enables the user to use different neurocontrol techniques to add and delete weights and units (rules) because in such a network weights are adapted instead of parameters of a node. There is no method which enables the designer to choose the number of rules. But in the elastic fuzzy logic by changing the elasticity parameters this can be done. Thus, if an elasticity in adaptation phase goes to zero the corresponding weight or rule can be deleted.

2 The proposed network based on fuzzy logic

We considered a network based on fuzzy IF-THEN rules as given by equation (1). The proposed network is shown in figure 1.

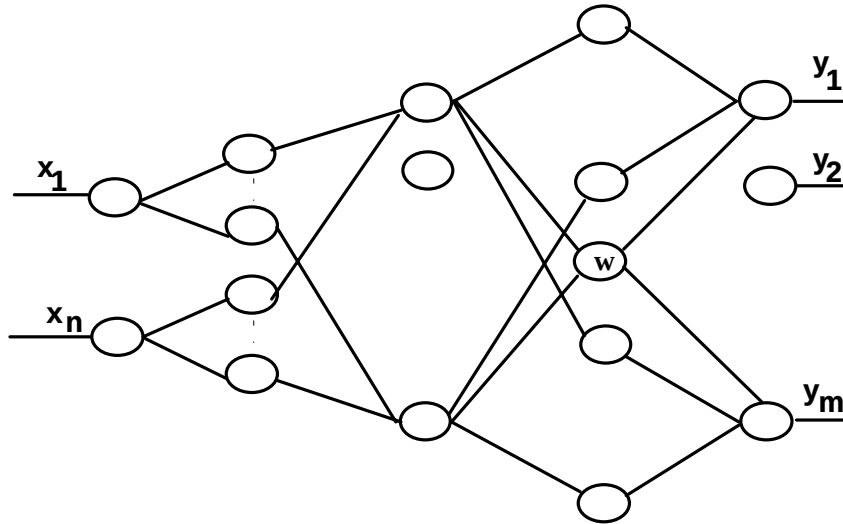


Figure 1: The proposed network based on fuzzy logic

In this network each layer's node has the following function:

- 1- Layer 1: Nodes in this layer are input nodes.
- 2- Layer 2: Nodes in this layer are input variable membership functions. This layer's nodes calculate equation (4).

3- Layer 3: Nodes in this layer are rule nodes. This layer's nodes calculate equation (5).

4- Layer 4: Nodes in this layer are consequent part of rules that calculate localized response of each rule, except the node labeled with '**W**' that calculates the equation (6). For simplicity we used local mean of maximum (LMOM) defuzzification [6]. The local mean of maximum uses the center of the line segment at height of the firing degree of i th rule. Figure 2 shows LMOM operator for trapezoid membership functions which is equal to:

$$U_i = \frac{w_i}{2} [w_i(LH_i + RH_i) + (1 - w_i)(LZ_i + RZ_i)] \quad (9)$$

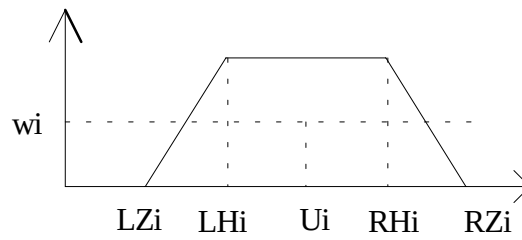


Figure 2: LMOM of trapezoid membership function

5- Layer 5: Nodes in this layer are outputs nodes. The number of nodes in this layer are equal to the number of outputs of fuzzy logic controller and is calculated by equation (7).

Learning Mechanism: We adapted the output membership functions by minimizing the error function E_D which is given by

$$E_p = \frac{1}{2} \sum_j (y_d^j - y^j)^2 \quad (10)$$

Where y_d^j is j th desired output and y^j is j th output of the network. Therefore, change in output membership functions equals to:

$$RH_{ij}(k+1) = RH_{ij}(k) + \frac{\eta}{2} e_j \frac{w_i}{\sum_k w_k} w_i \quad (11)$$

$$LH_{ij} (k + 1) = LH_{ij} (k) + \frac{\eta}{2} e_j \frac{w_i}{\sum_k w_k} w_i \quad (12)$$

$$RZ_{ij} (k + 1) = RZ_{ij} (k) + \frac{\eta}{2} e_j \frac{w_i}{\sum_k w_k} w_i \quad (13)$$

$$LZ_{ij} (k + 1) = LZ_{ij} (k) + \frac{\eta}{2} e_j \frac{w_i}{\sum_k w_k} w_i \quad (14)$$

Where η is learning rate, e_j shows the j th output error, and w_k represents the firing degree of the k th rule.

In the equation (10) it is assumed that the error surface is in quadratic form and equations (11) through (14) attempt to move in opposite direction of gradient to reach the global minimum of the error surface. If the actual error surface is not in the quadratic form, it may have several local minimums and the proposed learning algorithm does not guarantee to reach the global minimum. Using back propagation learning algorithm for neural networks could remove this deficiency by adding a momentum term to equations (11) through (14).

We could apply the momentum term to equations (11) through (14) in the following three different ways.

a) Apply the momentum term to all rules independent of which rules are fired. Therefore, output membership functions are changed to:

$$RH_{ij} (k+1) = RH_{ij} (k) + \frac{\eta}{2} e_j \frac{w_i}{\sum_k w_k} w_i + \beta (RH_{ij} (k) - RH_{ij} (k-1)) \quad (15)$$

$$LH_{ij} (k+1) = LH_{ij} (k) + \frac{\eta}{2} e_j \frac{w_i}{\sum_k w_k} w_i + \beta (LH_{ij} (k) - LH_{ij} (k-1)) \quad (16)$$

$$RZ_{ij} (k+1) = RZ_{ij} (k) + \frac{\eta}{2} e_j \frac{w_i}{\sum_k w_k} w_i + \beta (RZ_{ij} (k) - RZ_{ij} (k-1)) \quad (17)$$

$$LZ_{ij} (k+1) = LZ_{ij} (k) + \frac{\eta}{2} e_j \frac{w_i}{\sum_k w_k} w_i + \beta (LZ_{ij} (k) - LZ_{ij} (k-1)) \quad (18)$$

Where β shows the momentum factor and k is the iteration number.

The deficiency of this algorithm is that in each iteration all rules are not fired and those rules that are not fired do not have any effect on the output of that iteration. Thus, their parameters can not be adjusted. If we adjust all output nodes (memberships) the previous modifications are cancelled by this adjustment. So, for this algorithm it is possible to obtain unexpected results.

But, it is better to have a momentum term than not having it. To overcome this deficiency, we should adjust only those rules that are fired.

b) Apply the momentum term only to those rules that are fired independent of their firing degree. Then, adjust those parameters that effect the output of the fuzzy logic network. Therefore, output membership functions are change to:

$$RH_{ij}(k+1) = RH_{ij}(k) + \frac{\eta}{2} e_j \frac{w_i}{\sum_k w_k} w_i + \beta V_i (RH_{ij}(k) - RH_{ij}(k-1)) \quad (19)$$

$$LH_{ij}(k+1) = LH_{ij}(k) + \frac{\eta}{2} e_j \frac{w_i}{\sum_k w_k} w_i + \beta V_i (LH_{ij}(k) - LH_{ij}(k-1)) \quad (20)$$

$$RZ_{ij}(k+1) = RZ_{ij}(k) + \frac{\eta}{2} e_j \frac{w_i}{\sum_k w_k} w_i + \beta V_i (RZ_{ij}(k) - RZ_{ij}(k-1)) \quad (21)$$

$$LZ_{ij}(k+1) = LZ_{ij}(k) + \frac{\eta}{2} e_j \frac{w_i}{\sum_k w_k} w_i + \beta V_i (LZ_{ij}(k) - LZ_{ij}(k-1)) \quad (22)$$

Where $V_i = \begin{cases} 0 & \text{if } i\text{th rule is not fired} \\ 1 & \text{if } i\text{th rule is fired} \end{cases} \quad (23)$

The deficiency with this algorithm is that all rules do not fire with equal degree i.e. they do not have equal effect on the output. So, the momentum term should not be applied equally to all fired rules. To overcome this deficiency we must consider the firing degree of each rule for momentum term.

c) Apply the momentum term to fired rules based on their firing degree. Thus, changes in output membership functions become:

$$RH_{ij}(k+1) = RH_{ij}(k) + \frac{\eta}{2} e_j \frac{w_i}{\sum_k w_k} w_i + \beta w_i (RH_{ij}(k) - RH_{ij}(k-1)) \quad (24)$$

$$LH_{ij}(k+1) = LH_{ij}(k) + \frac{\eta}{2} e_j \frac{w_i}{\sum_k w_k} w_i + \beta w_i (LH_{ij}(k) - LH_{ij}(k-1)) \quad (25)$$

$$RZ_{ij}(k+1) = RZ_{ij}(k) + \frac{\eta}{2} e_j \frac{w_i}{\sum_k w_k} w_i + \beta w_i (RZ_{ij}(k) - RZ_{ij}(k-1)) \quad (26)$$

$$LZ_{ij}(k+1) = LZ_{ij}(k) + \frac{\eta}{2} e_j \frac{w_i}{\sum_k w_k} w_i + \beta w_i (LZ_{ij}(k) - LZ_{ij}(k-1)) \quad (27)$$

Where w_i represents the firing degree of i th rule.

3 The proposed network based on elastic fuzzy logic

We considered an elastic network based on fuzzy logic IF-THEN rules as given by equations (1) and (8). The proposed network is shown in figure 3.

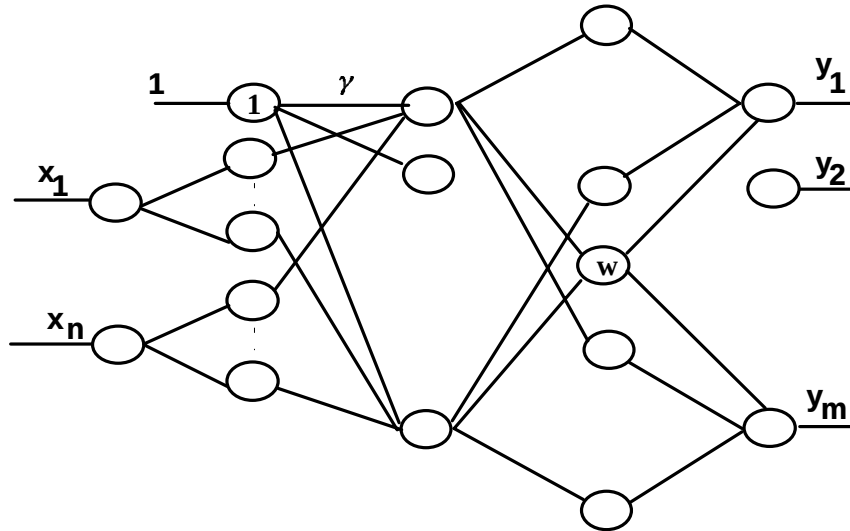


Figure 3: The proposed network based on elastic fuzzy logic

The function of nodes in all layers except layers 2 and 3 in this network are the same as those in the proposed network based on fuzzy logic.

In Layer 2: The node labeled with '1' is used for checking the validity of rules in the rule base. Other nodes in this layer are input membership functions for which we used correlation-product inference engine. We also used natural logarithm of input membership functions by following relation.

$$\ln(R_r) = \ln(\gamma_{r,0}) + \gamma_{r,1} * \ln(x_{r,1}) + \dots + \gamma_{r,N_r} * \ln(x_{r,N_r}) \quad (28)$$

where $\ln$ stands for natural logarithm.

In Layer 3: Each node is a weighted summer with an exponential function after it to implement equation (8).

Learning Mechanism: We adapted the gamma parameters by minimizing the equation (10). The changes of gamma parameters are equal to:

$$\gamma_{ij}^{(k+1)} = \gamma_{ij}^{(k)} - \frac{\eta}{\sum_i w_i} * w_i * \ln(x_j) \sum_k e_k (y_k - U_{ki}) \quad (29)$$

The reasons for using the momentum term for the proposed network based on fuzzy logic are also valid for this network. Again, we could use the momentum term in three different ways as follows:

a) Apply the momentum term to all rules. Therefore, the gamma parameters are changed to:

$$\gamma_{ij}(k+1) = \gamma_{ij}(k) - \frac{\eta}{\sum_i w_i} * w_i * \text{Ln}(X_j) \sum_k e_k (y_k - U_{ki}) + \beta (\gamma_{ij}(k) - \gamma_{ij}(k-1)) \quad (30)$$

b) Apply the momentum term only to the fired rules. Thus, the changes of gamma parameters are equal to:

$$\gamma_{ij}(k+1) = \gamma_{ij}(k) - \frac{\eta}{\sum_i w_i} * w_i * \text{Ln}(X_j) \sum_k e_k (y_k - U_{ki}) + \beta v_i (\gamma_{ij}(k) - \gamma_{ij}(k-1)) \quad (31)$$

c) Apply the momentum term to fired rules based on their firing degree. Therefore, the gamma parameters are changed to:

$$\gamma_{ij}(k+1) = \gamma_{ij}(k) - \frac{\eta}{\sum_i w_i} * w_i * \text{Ln}(X_j) \sum_k e_k (y_k - U_{ki}) + \beta w_i (\gamma_{ij}(k) - \gamma_{ij}(k-1)) \quad (32)$$

4 Example

A two link planar robot with following dynamic equation was considered to demonstrate the effectiveness of the proposed networks and algorithms.

$$\begin{aligned} \tau_1 = & \left[\left(\frac{m_1}{3} + m_2 \right) l_1^2 + m_2 l_1 l_2 \cos(\theta_2) + \frac{m_2 l_2^2}{3} \right] \ddot{\theta}_1 \\ & + \left(\frac{m_2 l_1 l_2 \cos(\theta_2)}{2} + \frac{m_2 l_2^2}{3} \right) \ddot{\theta}_2 - m_2 l_1 l_2 \sin(\theta_2) \left(\dot{\theta}_1 \dot{\theta}_2 + \frac{\dot{\theta}_2^2}{2} \right) \\ & + g_0 \left[\left(\frac{m_1}{2} + m_2 \right) l_1 \cos(\theta_1) + \frac{m_2 l_2 \cos(\theta_1 + \theta_2)}{2} \right] \\ & \tau_2 = \left(\frac{m_2 l_1 l_2 \cos(\theta_2)}{2} + \frac{m_2 l_2^2}{3} \right) \ddot{\theta}_1 + \frac{m_2 l_2^2}{3} \ddot{\theta}_2 \\ & + \frac{m_2 l_1 l_2 \sin(\theta_2)}{2} \dot{\theta}_1 + g_0 \frac{m_2 l_2 \cos(\theta_1 + \theta_2)}{2} \end{aligned} \quad (33)$$

Where g_0 denotes gravitational acceleration and equals to 9.81 m / Sec^2 . The problem is to move robot joints from any initial points in robot work space to their target points when the robot parameters and load are changing.

The control strategy used is based on the method given by the authors in [7]. The robot parameters were selected as $l_1 = l_2 = 0.432\text{m}$, $m_1 = 15.91\text{ kg}$ and $m_2 = 11.36\text{ kg}$.

The membership functions parameters for this controller are shown in table 1. The step responses of the robot for both networks are shown in figures 4 and 5.

Table 1: Parameters of membership functions before adaptation

a) Membership functions parameters for T_1

Linguistic Term	LZ	LH	RH	RZ
NB	-10000	-10000	-5000	-4000
PM	-8000	-4000	-4000	-200
NS	-4000	-200	-200	0
ZE	-200	0	0	200
PS	0	200	200	4000
PM	200	4000	4000	8000
PB	4000	5000	10000	10000

b) Membership functions parameters for T_2

Linguistic Term	LZ	LH	RH	RZ
NB	-6000	-6000	-2000	-500
PM	-4000	-1000	-1000	-20
NS	-1000	-200	-200	0
ZE	-20	0	0	20
PS	0	200	200	1000
PM	20	1000	1000	4000
PB	500	2000	6000	6000

Where T_1 and T_2 represent torques for joint 1 and joint 2.

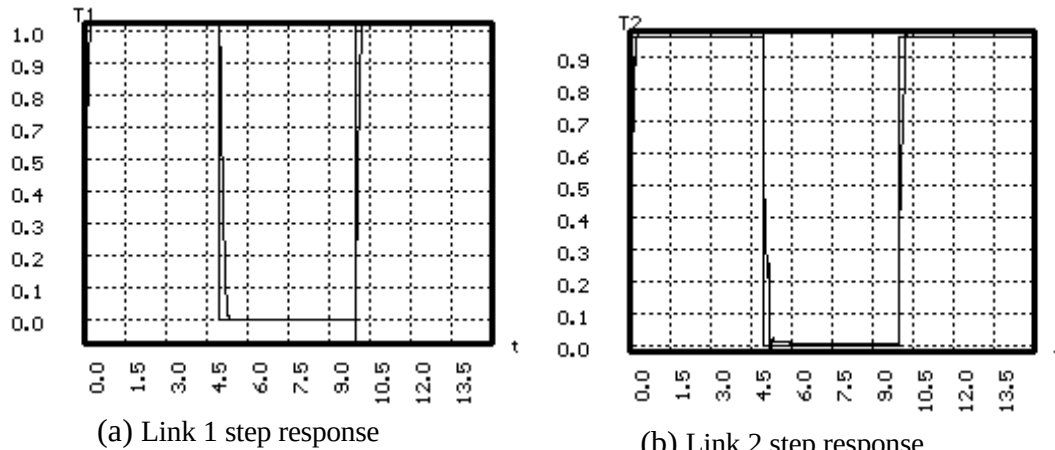
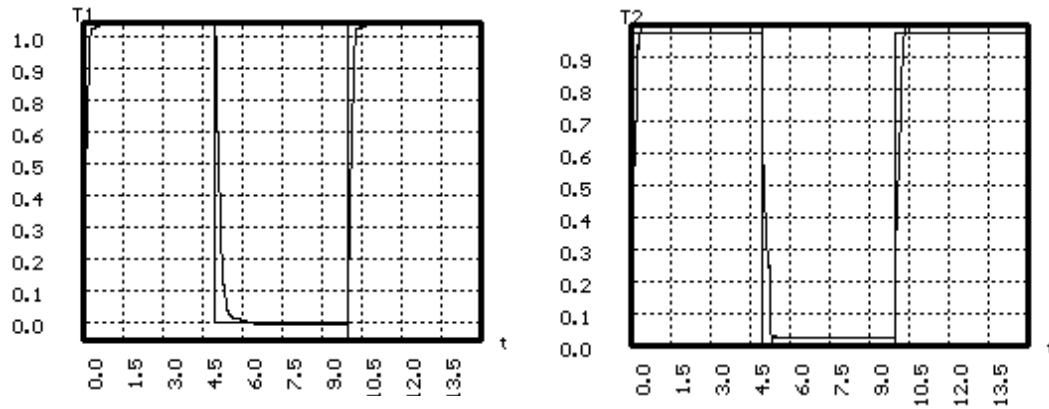


Figure 4: Step response of robot for network based on fuzzy logic

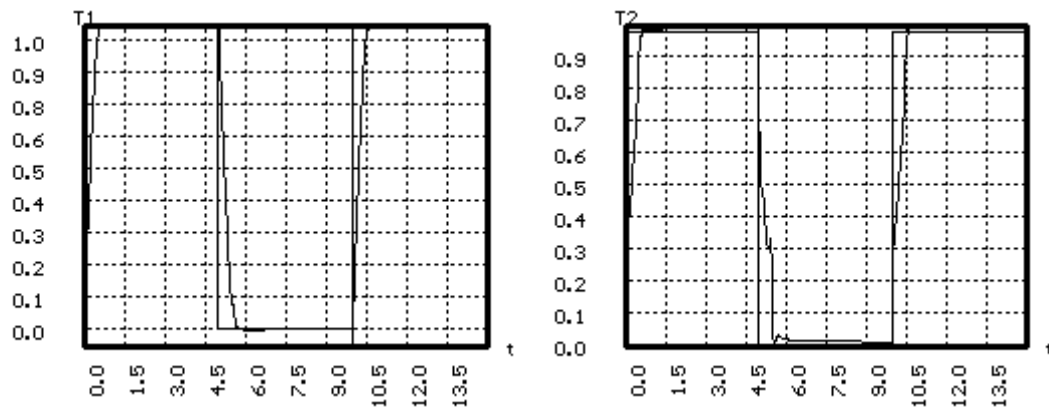


(a) Link 1 step response

(b) Link 2 step response

Figure 5: Step response of robot for network based on elastic fuzzy logic

We changed the robot parameters to $l_1 = l_2 = 1.0\text{m}$, $m_1 = 3\text{ kg}$ and $m_2 = 5\text{ kg}$ [8] to show that the controller can adapt the new parameters to control the new robot. The step responses of the new robot for network based on fuzzy logic without the momentum term is shown in figure 6. Tables 2 shows the membership functions parameters of the controller after adaptation.



(a) Link 1 step response

(b) Link 2 step response

Figure 6: Step response of robot for network based on fuzzy logic

Table 2: Parameters of membership functions before adaptation

a) Membership functions parameters for T_1

Linguistic Term	LZ	LH	RH	RZ
NB	-10000	-10000	-5000	-4000
PM	-8012	-4000	-4000	-212
NS	-4027	-203	-203	-27
ZE	-109	82	82	291
PS	0	200	200	3999
PM	295	4009	4009	8095
PB	4000	5000	10000	10000

b) Membership functions parameters for T_2

Linguistic Term	LZ	LH	RH	RZ
NB	-6000	-6000	-2000	-500
PM	-4000	-1000	-1000	-20
NS	-1002	-201	-201	-20
ZE	-14	5	5	26
PS	1	200	200	1001
PM	21	1000	1000	4001
PB	500	2000	6000	6000

Figure 7 shows the square of error for link 2 without momentum term.

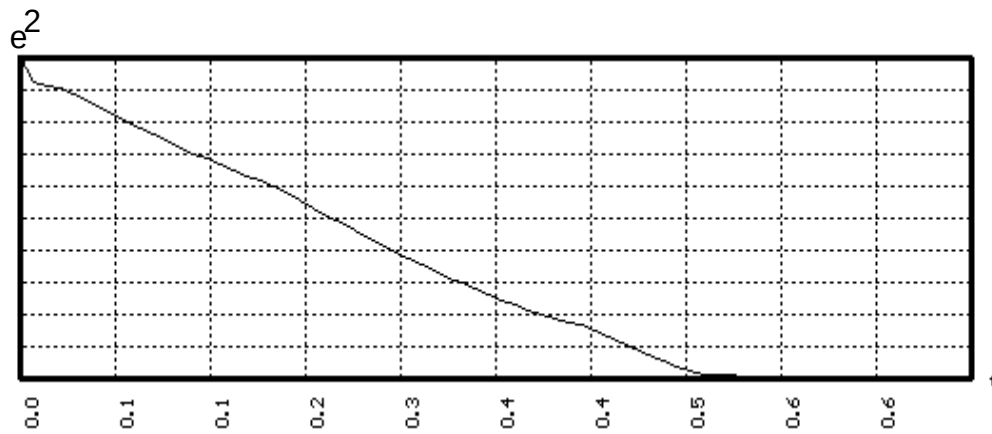


Figure 7: The square of error for link 2 of robot without momentum term

Figures 8 through 10 show the square of error for link 2 with momentum term applied to second output of the network.

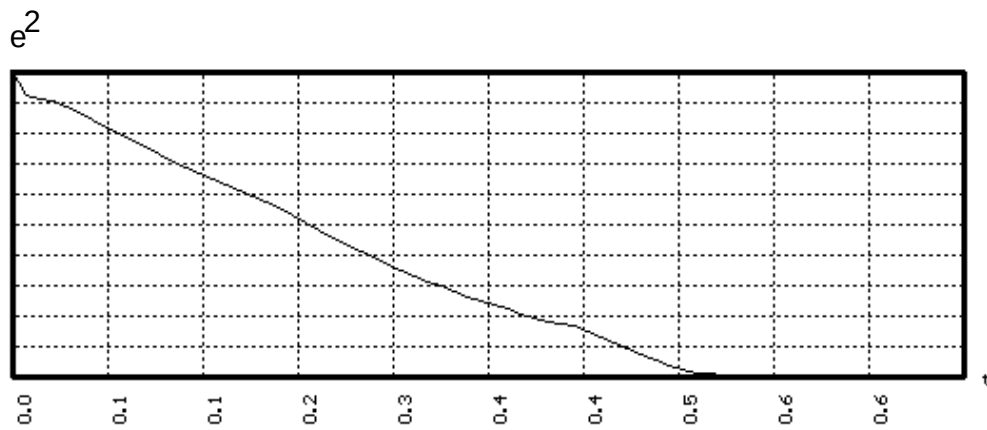


Figure 8: The square of error for link 2 of robot with momentum term applied to all rules

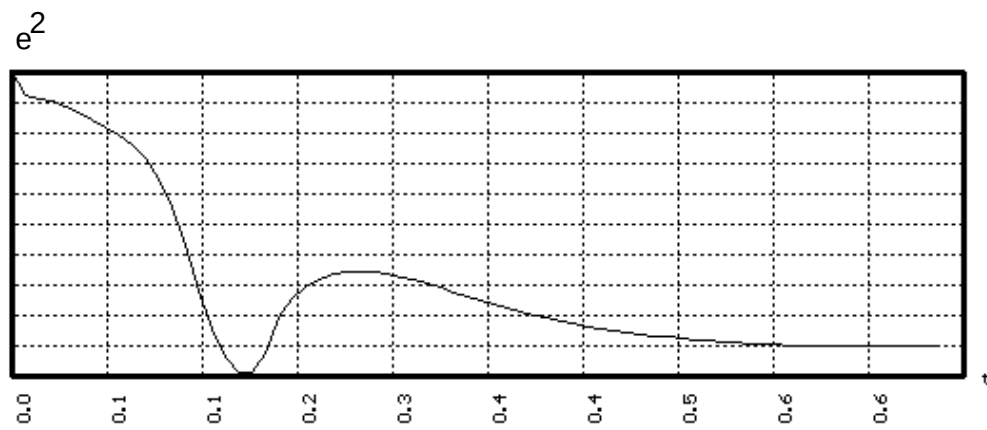


Figure 9: The square of error for link 2 of robot with momentum term applied equally to all fired rules

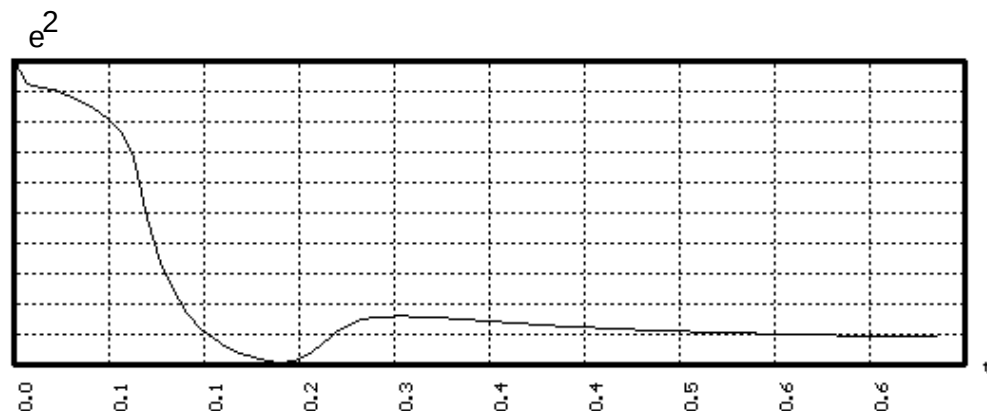
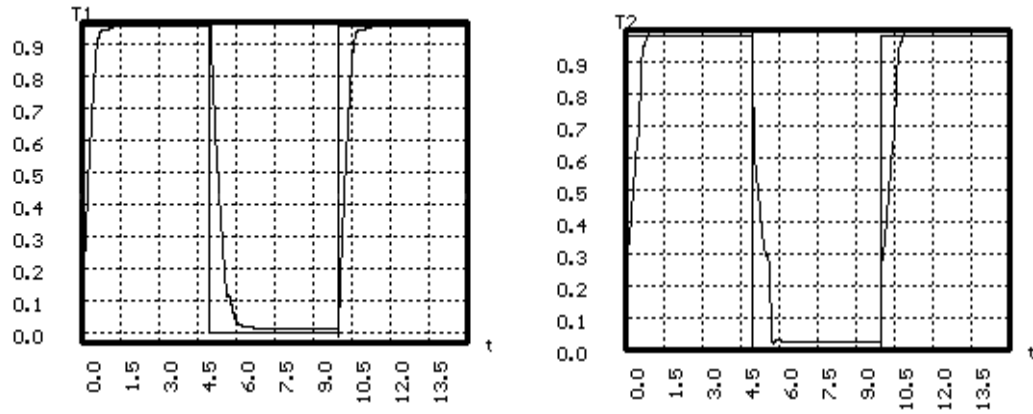


Figure 10: The square of error for link 2 of robot with momentum term applied to all fired rules based on their firing degree

The step responses of the new robot for network based on elastic fuzzy logic without the momentum term is shown in figure 11. Tables 3 shows the gamma parameters of the controller after adaptation.



(a) Link 1 step response

(b) Link 2 step response

Figure 11: Step response of robot for network based on elastic fuzzy logic

Table 3: Gamma parameters after adaptation

Rule No.	γ_{i0}	γ_{i1}	γ_{i2}
0	0.6268	2.4014	2.4014
1	0.5700	2.6864	2.3340
2	0.3350	4.2805	1.1460
3	0.5300	2.9048	2.5098
4	0.6268	2.4014	2.4014
5	0.6217	2.3758	2.4260
6	0.5564	2.6928	2.3233
7	0.3100	4.2608	1.2210
8	0.4983	2.9526	2.3933
9	0.6217	2.3759	2.4260
10	0.3481	1.5735	4.1654
11	0.1225	1.8876	4.4504
12	0.0002	6.0073	2.6304
13	0.0759	1.8622	5.8770
14	0.3482	1.5745	4.1653
15	0.5357	2.3695	2.8727
16	0.3621	3.1147	3.1710
17	0.1415	5.2122	1.2010
18	0.4756	2.7340	2.9566
19	0.5357	2.3696	2.8726
20	0.6268	2.4014	2.4014
21	0.5698	2.6874	2.3324
22	0.3350	4.2806	1.1415
23	0.5326	2.8900	2.5380
24	0.6268	2.4014	2.4014

In table 3, rules numbered 12 and 13 could be deleted because their elasticities goes to zero and therefore the controller can control the robot with 23 rules instead of 25 rules.

Figure 12 shows the square of error for link 2 without momentum term.

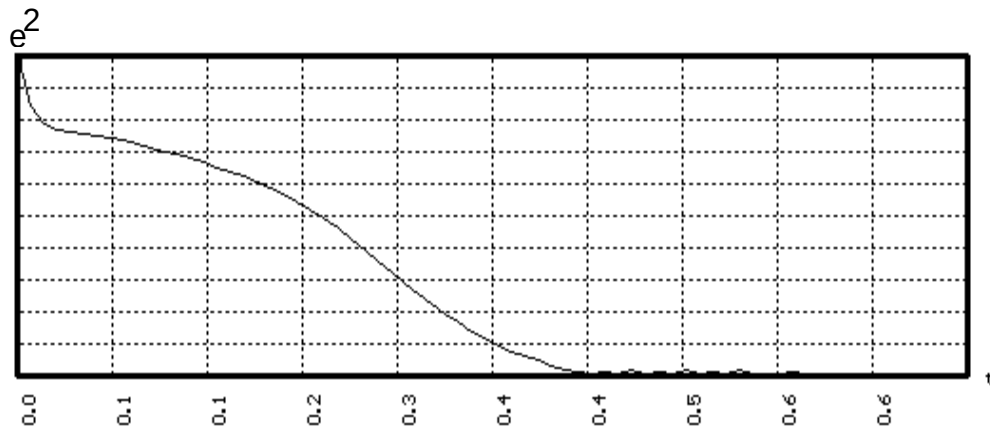


Figure 12: The square of error for link 2 of robot without momentum term

Figures 13 through 15 show the square of error for link 2 with momentum term applied to second output of the network.

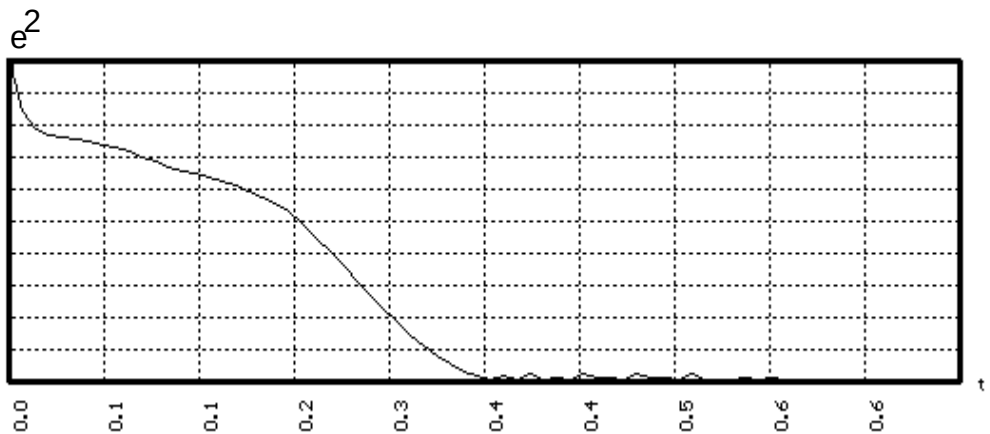


Figure 13: The square of error for link 2 of robot with momentum term applied to all rules

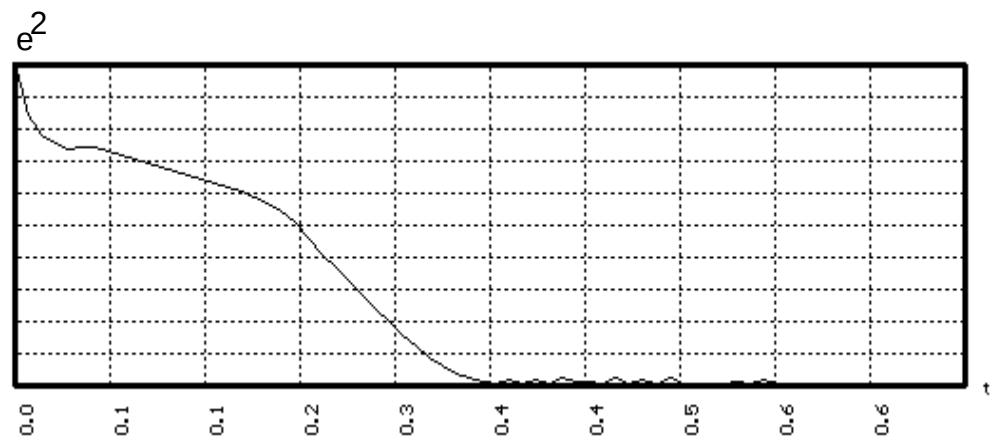


Figure 14: The square of error for link 2 of robot with momentum term applied equally to all fired rules

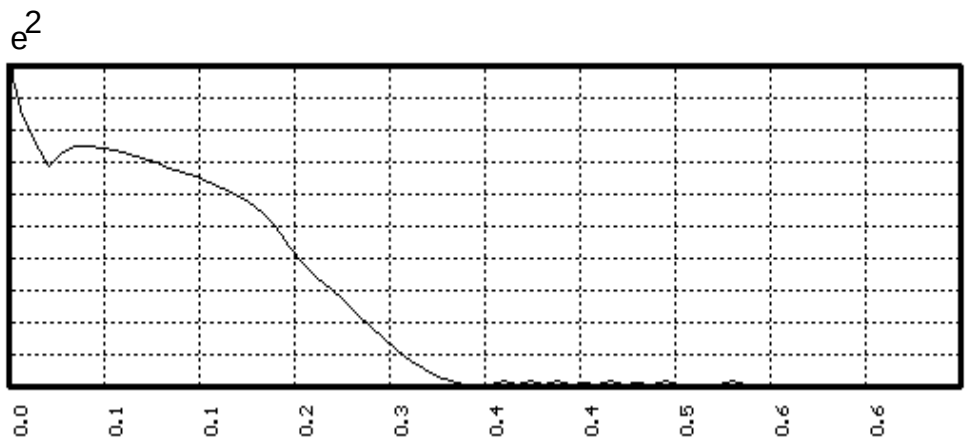


Figure 15: The square of error for link 2 of robot with momentum term applied to all fired rules based on their firing degree

5 Conclusions

In this paper, two adaptive networks have been proposed. The simulation results show that the momentum term speed up the learning process if momentum factor is selected correctly, but there is no procedure to select it. Another effect of momentum factor is on speed of convergence. If momentum factor is selected too large, it speeds up the learning too much and it may cause divergence and oscillations. If momentum factor is selected too low it slows down the learning too much and guarantees its convergence. It is better to select the momentum factor large in the first steps of training and then decrease it as the learning process grows. This method guarantees the speed and the convergence, but its range is dependent on the application.

6 Index

Neural networks, fuzzy logic, elastic fuzzy logic, back-propagation algorithm, momentum term.

7 References

1. Machado, R. J. and Rocha, A. F. D. Evaluative fuzzy neural networks, pp. 493-500, *Proceedings of IEEE Int. Conf. on Fuzzy Systems*, 1992.
2. Hirota, K. and Pedrycz, W. Fuzzy logic neural networks: Design and Computations, pp. 152-157, *Proceedings of IEEE Int. Joint Conf. on Neural Networks*, 1991.
3. Hayashi, Y., Czogala, E., and Buckley, J. J. Fuzzy neural controller, pp. 197-202, *Proceedings of IEEE Int. Conf. on Fuzzy Systems*, 1992.

4. Werbos, P. J. Neurocontrol and elastic fuzzy logic : Capabilities , Concepts, and Applications, *IEEE Trans. on Industrial Electronics*, 1993, 2, 170-180.
5. Jang, J. S. R. ANFIS: Adaptive-network-based fuzzy inference systems, *IEEE Trans. Syst. Man, Cybern.*, 1993, 3, 665-685.
6. H. R. Berenji, H. R. and Khedkar, P. Learning and tuning fuzzy logic controllers through reinforcements, *IEEE Trans. Neural Networks*, 1992, 5, 724-740.
7. Beigy, H and Eydgahi, A. M. An adaptive fuzzy model reference control for robot manipulators, pp. 59-62, *Proceedings of Int. Conf. on Automation*, 1995, Indore, India.
8. Lim, C. M. and Hiyama, C. M. Application of fuzzy logic control to a manipulator, *IEEE Trans. on Robotics and Automation*, 1991, 5, 688-691.