



## An online learning algorithm for neurocontroller based on a fuzzy estimate of the control error

B.Boutamina, K.Belarbi, S. Filali  
*Laboratoire d'automatique et de robotique*  
*Faculty of engineering, University of Constantine*  
 Route de Ain el Bey, Constantine, 25000, Algeria  
 brahbout@yahoo.fr  
 Tel, Fax: 21331819050

### Abstract

An online learning strategy for neurocontrollers based on minimisation of the control error is proposed. Since this latter is unknown a fuzzy system infers its estimate from the plant output error and its variation. This estimate is then used for weight updating in an online backpropagation algorithm. This procedure affects only the step size of the updating law if the estimate has the correct sign. The rule basis is derived from heuristic analysis. The algorithm is applied in simulation for the control two nonlinear systems. The results show that learning is fast and depends on the learning rate.

Keywords: Neural network control, fuzzy systems, online learning, backpropagation

*Keywords : Neural network control, fuzzy systems, online learning, backpropagation*

### 1. Introduction

A variety of control schemes involving the multilayered perceptron, MLP, have been proposed in the literature [1]. Most MLP controllers are based on supervised learning and backpropagation. We can identify two basic methodologies: intuitive, heuristic methods and methods with mathematically proven stability.

A variety of algorithms have been proposed based on the first methodology. In inverse control [12] the network is trained off line to learn the inverse dynamic of the plant to be controlled. After convergence, the network is used as a controller for the plant. In specialised learning, also described in [12], the network is trained on line to find the control input that minimizes the error between the desired output and plant output. The error must thus be backpropagated through the plant to the neurocontroller. This implies the knowledge of the jacobian of the plant. A number of authors [4, 6, 8, 12] use an off line trained neural network to emulate the plant. It is then used online to backpropagate the error signal to the neurocontroller. In [13] the jacobian is approximated by its sign. Other related techniques use a parallel controller

to provide the correct control signal [3],[7] or an approximate inverse of the plant [17]

More recently, a number of adaptive control algorithms using neural networks have been developed with guaranteed performance and proved stability results [2, 5, 10, 11, 14, 16, 18, 19]. These works are based on Lyapunov stability theory and generally use the neural networks to model plant nonlinearities. Most of the techniques developed rely on some off line training at least for weights initialization.

In this work, a new on line learning algorithm for neurocontroller based on the first methodology is introduced. The objective is to minimize the control error. A fuzzy inference system is used to provide an estimate of the control error which is then backpropagated online for updating the network weights. The fuzzy inference system has the property of universal approximation. In the scheme proposed in this paper, the fuzzy inference system approximates the control error from past output errors only, no pretraining of the network is carried out and no other controller than the neural network is used.

This paper is organized as follows: Section (2) gives the description of the algorithm, Section (3) discusses convergence and design issues and Section (4) presents simulation results of the applications of the algorithm to two non linear systems.

### 2. Algorithm Description:

#### *Basic idea*

The supervised learning algorithm developed in this work is based on the error between the actual output  $u(t)$  of the neurocontroller and the ideal unknown control signal  $u_d(t)$  that should produce the neurocontroller to ensure perfect tracking.

Define the control error

$$e_u^* = u(t) - u_d(t) \quad (1)$$

The neurocontroller is trained online to minimize the following objective function:



$$J = \frac{1}{2} (e_u^*)^2 \quad (2)$$

The network weights are updated according to the gradient descent rule:

$$w(t+1) = w(t) - \alpha \frac{\partial J}{\partial w} \quad (3)$$

where  $\alpha$  is the learning rate. The gradient is given by:

$$\frac{\partial J}{\partial w} = -(u_d(t) - u(t)) \frac{\partial u(t)}{\partial w} = -e_u^* \frac{\partial u(t)}{\partial w} \quad (4)$$

The neucontroller is used to provide the control signal  $u(t)$ . In this work we consider a multilayered feedforward network with a single hidden layer. It has  $N$  neurons in the input layer,  $N_H$  neurons in the hidden layer and one neuron in the output layer. The output is of the form:

$$u(t) = \sum_{i=1}^{N_H} w_{Hi}(t) z_i(t) \quad (5)$$

with

$$z_i(t) = F(x_i(t)) \quad (6)$$

and

$$x_i(t) = \sum_{j=1}^N w_{e_{i,j}}(t) e_j(t) \quad (7)$$

where

$e(t) = [e_1(t) \dots e_N(t)]^T$ : input vector

$x_i(t)$  output of hidden node  $i$  before activation function,

$z_i(t)$  output of hidden node after activation function  $F$ .

$w_{Hi}$  are the weights from hidden to output layer

$w_{e_{i,j}}$  are the weights from input to hidden layer

The online weight update is carried out with the estimated control error  $\hat{e}_u$ :

$$w(t+1) = w(t) + \alpha(t) \hat{e}_u \frac{\partial u}{\partial w} \quad (8)$$

Applying the chain rule we obtain:

$$w_{H_i}(t+1) = w_{H_i}(t) + \alpha(t) e_u^* z_i(t) \quad (9)$$

and

$$w_{e_{i,j}}(t+1) = w_{e_{i,j}}(t) + \alpha(t) e_u^* w_{H_i} \frac{\partial F}{\partial x_i} p e_j \quad (10)$$

The term  $\frac{\partial u}{\partial w}$  can be computed exactly through the chain rule for all weights of the networks whereas the control error  $e_u^*$  is unknown. A fuzzy inference system, FIS, is used to provide an estimate of  $e_u^*$  noted  $\hat{e}_u$ .

#### The fuzzy estimator:

The control error being unknown, a fuzzy inference system, FIS, is used to provide an estimate of  $e_u^*$  noted  $\hat{e}_u$ . It is well known that a FIS has smoothing properties and thus it is capable of transforming information related to the history of the output tracking error  $e_y(t) = y_d(t) - y(t)$  into a reliable estimate  $\hat{e}_u$  which is then used in (9) and (10) in place of  $e_u^*$ . Since both  $e_u^*$  and  $\hat{e}_u$  are scalars, if the latter has the correct sign, this procedure will only

affect the step size in the updating laws. The FIS transforms information related to the history of the output tracking error  $e_y(t) = y_d(t) - y(t)$  into an estimate  $\hat{e}_u$ .

The main component of the FIS is the rule base composed of rules of the form [9]:

If  $X_1$  is NEGATIVE and  $X_2$  is POSITIVE... then  $E_u$  is ZERO

Where  $X_1, X_2$  are linguistic variables associated with the output tracking error and its derivatives,  $E_u$  is a linguistic variable associated with the estimated control error and NEGATIVE, POSITIVE, ZERO are linguistic values associated with these variables.

We can distinguish between two rule bases depending on the relative direction of the control and output signals. Rule base, RB1, for the case where the control and output have the same direction, that is an increase (decrease) in the control variable produces an increase (decrease) in the output variable and rule base RB2, for the other case. For instance, if the fuzzy estimator has two crisp, numerical input variables: the output tracking error  $e_y(t)$  at time  $t$  and its change  $de_y(t) = e_y(t) - e_y(t-1)$ , the heuristic reasoning behind RB1 is as follows:

- The control error is zero ( $u_s^*(t) = u(t)$ ) when both  $e(t)$  and  $de(t)$  are zero or when the error is decreasing and keeping the same sign (output approaching the reference from above or from below).

- The control error is negative ( $u_s^*(t) < u(t)$ ) when the output is greater than the reference and  $de(t) = 0$ , (output remaining at a point above reference) or  $de(t) < 0$  (output is drifting away from the reference from above).

- The control error is positive ( $u_s^*(t) > u(t)$ ) when the output is less than the reference and  $de(t) = 0$ , (output remaining at a point below reference) or  $de(t) > 0$  (output is drifting away from reference from below).

Clearly, this procedure provides the correct sign of  $\hat{e}_u$ .

Rule bases RB1 and RB2 are shown table 1 and 2 respectively for the case where all variables are fuzzified with three fuzzy sets, N: NEGATIVE, Z: ZERO, and P: POSITIVE.

The crisp output  $\hat{e}_u$  of the fuzzy estimator is computed with the centre of gravity defuzzification formula:

$$\hat{e}_u = \frac{\sum_{k=1}^m c_k \mu_{B_k}(\hat{e}_u)}{\sum_{k=1}^m \mu_{B_k}(\hat{e}_u)} \quad (11)$$

where  $c_k$  are the centers of fuzzy set  $B_k$  associated with fuzzy variable  $E_u$  and the grade of membership  $\mu_{B_k}(\hat{e}_u)$  is such that:

$$0 \leq \mu_{B_k}(\hat{e}_u) \leq 1 \quad (12)$$

The value of  $\hat{e}_u$  can thus be controlled by the choice of the coefficients  $c_k$ .

This is the basic setting for the FIS, namely rule base RB1 or RB2 and three fuzzy sets for all variables that will be used throughout.



Table 1 : Rule base RB1

e \ de	N	Z	P
N	N	N	Z
Z	N	Z	P
P	Z	P	P

Table 2 : Rule base RB2

e \ de	N	Z	P
N	P	P	Z
Z	P	Z	N
P	Z	N	N

### 3. Convergence analysis and design consideration:

#### Convergence analysis

In equation (4) giving the gradient formula, the term  $\frac{\partial u}{\partial w}$  determines the direction of the gradient whereas the term  $e_u^*$  is scalar and scales the step size. Thus as long as the estimated error  $\hat{e}_u$  used in (12) have the correct sign, the introduction of this latter in place of  $e_u^*$  will only affect the step size. The algorithm will converge if the estimated control error has the correct sign that is the sign of  $e_u^*$ . The FIS introduced above clearly provides an estimated control error with the correct sign.

#### Design considerations

There are two sets of design parameters, those associated with the fuzzy estimator and those associated with the neural network. The FIS can be constructed so as to provide a bounded estimated control error  $\hat{e}_u(t)$ . Considering the basic setting introduced above and since the control error affects only the step size, we can impose the remaining parameters of the FIS so as to obtain a normalised control error defined as:

$$-1 < \hat{e}_u < +1 \quad (13)$$

This can be achieved by a proper choice of the parameters  $c_k$  of equation (5). If the universe of discourse  $U_{\hat{e}_u}$  of the control error is constrained as :

$$-1 < U_{\hat{e}_u} < +1 \quad (14)$$

and with three fuzzy sets, as in the basic setting, the following choice:

$$c_{NEGATIVE} = -1, c_{ZERO} = 0, c_{POSITIVE} = +1$$

will ensure that (13) is satisfied.

Likewise we normalise the universe of discourses of the input variables

$$-1 < U_e < +1 \quad -0.1 < U_{de} < +0.1 \quad (15)$$

Moreover all fuzzy sets are uniformly distributed in the respective universes of discourse. These choices define completely the fuzzy estimator. The design parameters for the controller will thus be reduced to the parameters of the neural network and the gain k.

### 4. Cases study

In this section we apply in simulation the online learning algorithm for the control of two non linear systems. For all the simulations carried out, we use the basic setting introduced above. We are thus left only with the neural network design parameters: the learning rate, the number of hidden neurons and the activation function.

#### Case study 1: Concentration control in a CSTR:

The continuously stirred tank reactor, CSTR considered here is described by the following two non linear equations [15]:

$$\dot{x}_1 = -x_1 + D_a(1-x_1)e^{\frac{x_2}{1+x_2/\gamma}} \quad (16)$$

$$\dot{x}_2 = -x_2 + BD_a(1-x_1)e^{\frac{x_2}{1+x_2/\gamma}} - \beta(x_2 - q_c) \quad (17)$$

With  $B=21.5$ ,  $\gamma=28.5$ ,  $D_a=0.036$  and  $\beta=25.2$ . The nominal operating point is  $x_1=0.4126$ ,  $x_2=3.28$  and  $q_c=3.04$ . All variables dimensionless and normalised [15]. The objective is to control the concentration  $x_1$  by manipulating the coolant flow rate  $q_c(t)$ .

The reference is variable and consists of consecutive steps of different values. An analysis of the process shows that an increase (decrease) in coolant flow produces an increase (decrease) in the concentration, we thus choose rule base RB1.

In this example, we analyze the behavior of the algorithm with respect to two design parameters of the neural network: the learning rate and the number of neurons. The activation function is sigmoid:

$$z_i = \frac{1}{1+e^{-x_i}} \quad (18)$$

In order to impose the same initial conditions for all tests, the initial weights are always set to zero. In the first series of tests we set a constant learning rate  $\alpha(t)=1$ , and we vary the number of hidden neurons from one to five.

Figure 2 (a) shows the concentration for all tests and figure 2 (b) shows the coolant flow rate. A perturbation of +10% is introduced on the parameters  $B$ ,  $D_a$  and  $\beta$  at time  $t=15$  minutes. Tracking of the set point improves with the number of hidden neurons. However it starts saturating from three neurons. This is predictable since the number of neuron usually improves the approximation property of a network and there is limit beyond which there will be no improvement. Figure 2 (c) shows the evolution of the estimated control error: it has a smooth behaviour which confirms the smoothing properties of the fuzzy inference system.



**Case study 2: Single link manipulator**

In this last example we consider a discrete-time single-link manipulator [2]:

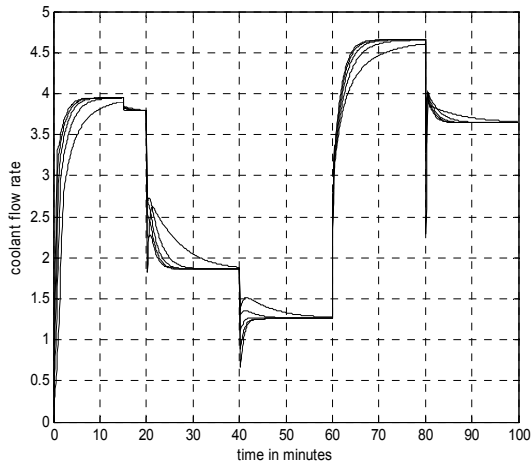
$$x_1(k+1) = x_2(k) \tag{19}$$

$$x_2(k+1) = (2T - 1)x_1(k) + (2 - 2T)x_2(k) + 10T^2 \sin x_1(k) + u(k) \tag{20}$$

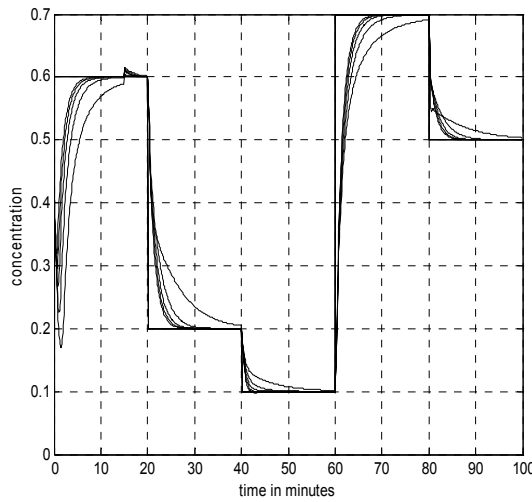
where  $x_2(k)$  is the angular position,  $T=0.01$ . The desired trajectory is  $x_{2d}(k) = (\pi/2)\sin(\pi kT/5)$ . In this case study rule base RB1 is used for the fuzzy estimator. There are two hidden neurons in the neurocontroller with the following activation function:

$$z_i = \frac{1 - e^{-x_i}}{1 + e^{-x_i}} \tag{21}$$

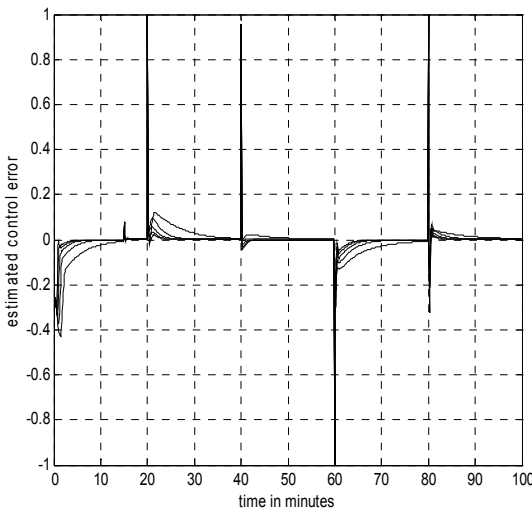
Figure 2 (a) shows the output trajectory for different values of the learning rate:  $\alpha = [0.001, 0.01, 0.1, 1, 10]$ ; It can be seen that with a very small number of neurons the output tracks the desired trajectory and the tracking error improves with the learning rate. Figure 2 (b) shows the estimated control error.



(a)

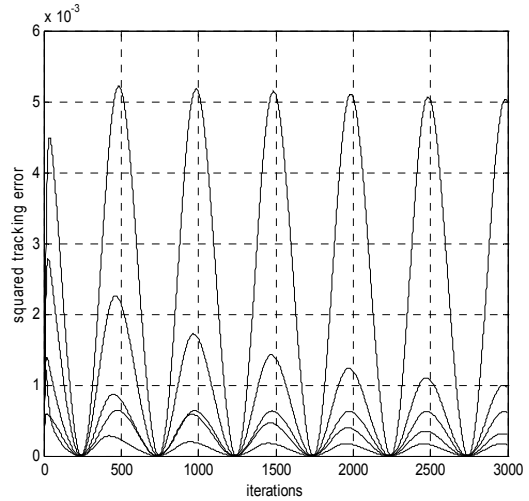


(b)

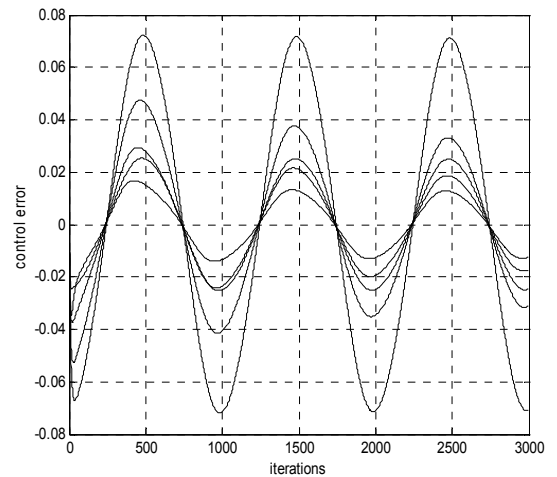


(c)

Figure 1 : Example 1: results of the control for different number of neurons: (a) The concentration, (b) The coolant flow rate, (c) The estimated control error



(a)



(b)



Figure 2 Results for example 2 (a) the squared tracking error, (b) the estimated control error

## 5. Conclusion

An online learning strategy for neurocontrollers based on minimization of the control error is proposed. The control error being unknown, a fuzzy inference system, FIS, is used to estimate it. The FIS transforms information related to the history of the plant output error into an estimate of the control error which is then used for weights updating in an online backpropagation algorithm. This procedure affects only the step size of the updating law if the estimate has the correct sign. The rule base of the FIS is derived heuristically from simple analysis. Normalization of the universe of discourses allowed us to reduce the design parameters to those of the neural networks. The strength of the method is its simplicity, its drawback is that for the moment there is no guarantee of closed loop stability. A study of the behaviour of the algorithm with respect to the number of neurons shows that the number of neurons has a direct effect on the speed of the response. When this number reaches a threshold value, this effect saturates and is no longer apparent. This value is usually around five neurons. The number of neurons could then be chosen as the value that gives the best response which could correspond to the threshold value.

The algorithm is applied in simulation for the control two non linear systems, a CSTR and a single link manipulator. The results show that learning is fast. The method is reproducible and its critical design parameter is the learning rate. The main drawback of the method is that it is difficult to establish the global stability of the closed loop system and it thus remains a heuristic control algorithm. This is a common feature to algorithm based on the gradient descent.

## 6. References

- [1] M. Agarwal, A systematic classification of neural network based control, IEEE Control Systems Magazine 17, pp. 75-93, 1997.
- [2] F.C. Chen and H.K. Khalil, Adaptive control of a class of non linear discrete-time systems using neural networks, IEEE Trans. Automatic Control 40, pp. 791-801, 1995.
- [3] H. Gomi and M. Kawato, Neural network control for a closed loop system using feedback error learning, Neural Networks 6, pp.993-946, 1993.
- [4] M. I. Jordan, Generic constraints on underspecified target trajectories, Proc. IJCNN'89, Washington DC, pp. 217-225, 1989.
- [5] F. L. Lewis , A. Yesildirek, and K. Liu, Multilayer neural net robot controller with guaranteed tracking performance, IEEE Trans. Neural Network, vol. 7, pp. 388-399, 1996.
- [6] K. S. Narendra and K. Parthasarathy, Identification and control of dynamical systems using neural networks, IEEE Trans. Neural Network 1, pp. 4-27, 1990.
- [7] R.T. Newton and Y. Xu, Neural network control of a space manipulator, IEEE Control Systems Magazine 13, pp.14-22, 1993.
- [8] D.H. Nguyen and B. Widrow, Neural networks for self learning control systems, IEEE Control Systems Magazine 10, pp. 18-23, 1990.
- [9] K.M. Passino and S. Yurkovich, Fuzzy Control, Reading, MA:Addison-Wesley, 1998.
- [10] M.M. Polycarpou, Stable adaptive neural control scheme for nonlinear systems, IEEE Trans. Automatic Control 41, pp. 447-451, 1996.
- [11]M. M. Polycarpou and M. Mears, Stable adaptive tracking of uncertain systems using nonlinearly parameterized online approximators, International Journal of Control 70, pp. 363-384, 1998.
- [12] D. Psaltis, A. Sideris and A. Yamamura, A multilayered neural network controller, IEEE Control Systems Magazine 8, pp.17-21, 1998.
- [13] M. Saerens and A. Soquet, A neural controller, Proc. 1<sup>st</sup> Int. Conf. Artificial Neural Networks, London, pp. 211-215, 1989.
- [14] R. M. Sanner and J. E. Slotine, Gaussian networks for direct adaptive control, IEEE Transactions on Neural Networks 3, pp. 837-863, 1992.
- [15] M. Shacham, N. Brauner, M. Cultip, Exothermic CSTRs: just how stable are the multiple steady states, Chemical Engineering Education, Winter, 1994
- [16] A. Yesildirek and F. Lewis, Feedback linearisation using neural networks, Automatica 31, pp. 1659-1664, 1995.
- [17] A.M.S. Zalzal and A.S. Morris, A neural network approach to adaptive robot control, Int. J. Neural Networks 2, pp. 17-35, 1991.
- [18] T. Zhang, S. S. Ge and C. C. Hang, Design and performance analysis of a direct adaptive controller for nonlinear systems, Automatica 35, pp. 1809-1817, 1999.
- [19] Y. Zhang, P. Peng and Z. Jiang, Stable neural controller design for unknown nonlinear systems using backstepping, IEEE Trans. Neural Network 11, pp. 1347-1360, 2000.

