# On-line Learning in Self Organising Fuzzy Controller With Application

**Joana Matos Dias and António Dourado Correia**
CISUC-Centro de Informática e Sistemas da Universidade de Coimbra
Departamento de Engenharia Informática
PÓLO II da Universidade de Coimbra
Pinhal de Marrocos
P 3030 Coimbra
PORTUGAL
e-mails: joana@student.dei.uc.pt , dourado@dei.uc.pt

**Abstract:** A Self Organising Fuzzy Controller for linear or non-linear processes with minimum knowledge about the process is developed with the only assumption that the output of the process is monotonic with respect to the input (as most industrial processes are). Beginning with an empty rule-base a fuzzy process model is progressively on-line built. In this way the controller gradually learns how to control the process. The inference and defuzzification mechanisms have their background on the Fuzzy Equality Relations Theory. A simple technique for on-line adaptation of a similarity factor is proposed. Feedforward effect is introduced in the definition of this similarity factor and in the defined *error_in_advance* value . This controller was successfully applied in simulation for both the regulation and the tracking problems. Practical essays were made on a real non-linear thermal process and the developed controller was able to provide satisfactory performance.

**Keywords:** Fuzzy control, Self Organising Control, Non-linear Control, Adaptive On-line Learning.

**Joana Matos Dias** was born in 1973 and got the Informatics Engineer degree from the University of Coimbra, Portugal, in 1996. She is actually developing a Ph.D research program on Intelligent Control Systems, with special interests in on-line learning in fuzzy and neurofuzzy controllers.

**António Dourado Correia** was born in 1954. He is Electrical Engineer (1977) from the University of Coimbra, DEA in 1981 and Docteur Ingénieur in 1983 by Université Paul Sabatier and LAAS du CNRS, Toulouse, France. Doctor by University of Coimbra in 1984. Actually Associate Professor of the Informatics Engineering Department of the University of Coimbra). His main interests (teaching and research) are Automatic Control (particularly Intelligent Control ) and Optimization.

## 1. Introduction

Fuzzy controllers usually require the programmer to build the rule- base according to the knowledge he has about the system. If the system experiences several and deep changes in time, those rules will no longer be useful. If one does not know much about a process, it is difficult to guess which samples are important. It would be better if the controller did not need either *a priori* off-line training or *a priori* rules and it self-constructed the rule -base with the information it acquires from the process. This is the aim of the SOC (Self Organising Controllers) in a fuzzy framework.

In recent years great efforts have been made and a significant number of strategies has been developed for SOCs (for reviews see Driankov and *Coll.* (1993), Park and *Coll.* (1995), Lee and Wang(1996)) with the general objective of reducing the dependence on process operators and experts. Other authors tried to solve the problem in a neuro-fuzzy framework (for example Lin and *Coll.* (1995), Presti and Zaboni (1995)) or by genetic optimisation (Karr and Gantry(1995)) or then developed methods for special types of control problems (Dieulot and *Coll.* (1996)).

One of the known techniques is based on the discretisation of the input and output variables domains, and the construction of a matrix of cells (Chen and Tsao(1989), Park and *Coll.* (1995)). If we consider that each cell can accommodate one and only one rule, then we can construct the rule -base. The discretisation of the output variable domain is generally done according to the desired reference in order to achieve better performance. The problem with this approach is that it leads to an enormous increase in the number of rules needed when considering processes of order higher than the 1$^{st}$ and desired references of broad amplitudes. This may increase prohibitively the need for computational resources. Another problem that has to be solved is how to update the rules in the rule- base so that the controller is able to deal with sudden changes in the process dynamics.

The more traditional methods for inference and defuzzification do not consider in a convenient way the relative importance of the fired rules, particularly in an adaptive context, where for new situations new rules must be dynamically created. Some authors have worked to overcome this drawback. For example, Park and *Coll.* (1995) have proposed for the regulation problem a strategy based on a similarity factor. However they fixed *a priori* this similarity factor obliging to use trial and error approaches to find a good value for each particular problem. In this paper we develop an adaptive strategy, simple but effective, for the similarity factor based on a time-window evaluation of performance. If the controlled process is time-varying this proposed

real-time adaptation of the similarity factor allows a significative increase of performance.

This paper is organised as follows: Section 2 states the assumptions made, Section 3 explains the fuzzification, inference and defuzzification mechanisms, Section 4 explains how is the adaptive degree of similarity modified, Section 5 addresses the question of dividing the rule base into layers, in Section 6 some results are shown and finally in Section 7 some conclusions are drawn.

## 2. Generality of the Proposed Self Organising Controller

The SOC proposed is applicable to processes where the following *a priori* knowledge exists:

i) The process can be modelled by an NARX model:

$$y_k = f(y_{k-1}, y_{k-2},\ldots,u_{k-T}, u_{k-T-1},\ldots, e_k) \quad (1)$$

with $f$ a linear or non-linear function, time variant or not, $y_k$, $y_{k-1}$ the outputs of the system and $u_k, u_{k-1},\ldots$ the inputs to the system at the $k,k-1,\ldots$ sampling instants. $T$ represents the discrete pure time delay. $e_k$ is a stochastic disturbance.

ii) The pure time delay $T$ is known or, at least, its upper bound.

iii) The reference is known at least $T+1$ sampling instants in advance.

iv) The order (memory) of the system is known (at least its upper bound).

v) The relation, one out of two possible, between the change in the input signal and the change in the output signal is known:

1. the output increases/decreases with the increase/ /decrease of the input.
2. the output decreases/increases with the increase/ /decrease of the input.

The assumption iii) tries to give the controller an anticipative characteristic so that it can deal with the pure time delay of the process. When considering the regulation problem, this assumption is readily fulfilled. With the tracking problem this is still possible if the reference is known *a priori*, as the case is with most of the problems of practical relevance.

## 3. The Rule -base and the Fuzzification, Inference and Defuzzification Mechanisms

### 3.1 The Rule -base

The rule- base is initially empty. It is constructed on-line. Considering (1), solving for $u_{k-T}$, (2) is obtained:

$$u_{k-T} = g(y_k, y_{k-1},\ldots,u_{k-T-1},\ldots,e_k) \quad (2)$$

Then the rules format is derived from the linguistic transformation of (2), in the form of a FARX-Fuzzy Auto Regressive with Exogenous Input model , as shown in (3):

*If* $y_k$ *is* $A_1$ *and* $y_{k-1}$ *is* $A_2$ *and ... and* $u_{k-T-1}$ *is* $B_1$ *and ... then* $u_{k-T}$ *is* $C$. (3)

$A_1, A_2, \ldots, B_1,\ldots,C$ are linguistic values.
If we add $T+1$ to each and every index, each rule gives us a value for $u_{k-1}$, if the antecedent is true (in a fuzzy way). We consider the future values of the process output the same as the future (known) values of the reference: implicit in this is the assumption that the controller will be capable of making the output follow the reference. At every sampling time an input signal is sent to the process and an output value is received from it. Based on these values and on the memorised ones, the rules are constructed or updated. At the $i^{th}$ sampling instant $A_1$ is substituted for *"approximately* $y_i$*"*, $A_2$ for *"approximately* $y_{i-1}$*"*,..., $B_1$ by *"approximately* $u_{i-T-1}$*"* and so on.

The input and output domains are continuous, so they must be discretised. This discretisation determines the number of rules that constitutes the rule- base. The rule- base can then be considered as a matrix with each of its rules occupying a cell in that matrix. Considering a $1^{st}$ order process, for the sake of simplicity, we can see that if we want the rule base to be consistent then the rules with the same antecedents cannot have different consequents. Therefore, each cell in the matrix can contain one and only one rule. For a $1^{st}$ order process we only need to perform the discretisation of the output variable domain (Y). We will have the space Y*Y $(y_k, y_{k-1})$ split into cells.

When we need insert a rule in the rule base belonging to a cell that has already got a rule, then the older rule is forgotten and the new rule replaces it. When considering processes of order higher than the $1^{st}$ it is also necessary to perform the discretisation of the input variable. If we want to achieve better results then this operation

should be more accurate near the desired operating point. If this is not a serious drawback for the regulation problem, it is a very serious one when considering the tracking problem and references of broad amplitudes, because this leads to an enormous increase in the number of possible rules. The number of cells that covers the region corresponding to the desired reference increases drastically, and when the process output begins to follow the reference then the increase in the number of rules is notorious with its implication in the sampling interval. The discretisation of the input variable can be done uniformly because we do not know *a priori* where we can find the correct values for the input.

## 3.2 The Fuzzification Stage

The membership functions used in fuzzification of the output and of the control values are triangular (4):

$$\mu_{ui} = \begin{cases} 1 + \dfrac{u_k - u_i}{b - a} & a \le u_k \le u_i \\ 1 - \dfrac{u_k - u_i}{b - a} & u_i \le u_k \le b \\ 0 & \textit{otherwise} \end{cases} \qquad (4)$$

with $u_k \in [a,b]$. This interval includes all possible values for the respective variable. These functions have been proposed by Park and *Coll.* (1995). Each crisp $u_k$ value is considered as a degenerated fuzzy singleton and defines the unity vertex of a fuzzy set. In this way the only information needed to define the membership function is the crisp value and the rule- base is always complete.

## 3.3 The Inference Stage

The inference mechanism aims at compounding the influence of all the fired rules. The compound function may take several forms. For example Park and *Coll.* (1995) used the Euclidean distance between the newly measured crisp values at the controller input and the vertices of the existent membership functions. In the present work, and in order to introduce a *feedforward* effect in the control signal, it is used the Euclidean distance between the newly measured values and the future values of the reference taking into account the time delay of the process.

Considering, for instance, a 1st order process each rule has the format: "*If* $y_k$ *is approximately* $y_{1i}$ *and* $y_{k-1}$ *is approximately* $y_{2i}$ *then* $u_{k-1}$ *is approximately* $u_i$". If the future values of the reference are R*1* and R*2* (respectively in $k+T+1$ and $k+T$ sampling times), the degree of equality between them and each rule antecedent is calculated as (5):

$$D_i = \sqrt{(y_{1i} - R1)^2 + (y_{2i} - R2)^2} \qquad (5)$$

If $D_i$ is high, this means that the average (in the sense of the Euclidean distance) of truth in the rule is low, meaning that the rule is not *similar* to the real situation. If $D_i$ is low, then the rule is *similar*. If $D_i$ is zero, then the rule is completely truth. A *degree of similarity* can then be defined as (6)

$$\omega_i = -\frac{1}{Dmax} * D_i + 1 \qquad (6)$$

The consequent fuzzy set for each rule is an interval of values [*p1,p2*], where

$$\mu_{ui}(u_k) \ge \omega_i, \forall u_k \in [p1, p2]. \qquad (7)$$

The total output is taken as the intersection of all the intervals (one for each rule). So, at the end of the inference mechanism, we have an interval of possible values for $u_{k-1}$ that we name *learned interval*.

$$\Delta u = [\min, \max] \qquad (8)$$

The justification for the choice of the intersection operator can be found in the Theory of Fuzzy Equality Relations (Klawon and *Coll.*(1995)). In the inference mechanism, one has to take the intersection of all the intervals, one for each rule. It is possible that this intersection results in an empty set. In this case, the best thing to do is just to ignore the rule producing an empty set. The conclusion reached by the experiences made was that this rule is insignificant, illustrating a transitory state of the system and should not be taken into account.

## 3.4 The Defuzzification Stage

The output of the inference mechanism is an interval [min,max] (8). This needed crisp value for $u_{k-1}$ belongs to the interval and, in the $k^{th}$ iteration, is calculated in the present work as a function of the *error_in_advance* as follows:

i) *error_in_advance = reference(k+T+1) - yk.*

This fact also introduces some feedforward effect in the control signal.

ii) Considering the relation 1) of the assumption v):

if *error_in_advance* > 0 then

$$u_{k+1} = \frac{\max + u_{k-T}}{2} \qquad (9)$$

else

$$u_{k+1} = \frac{\min + u_{k-T}}{2}. \qquad (10)$$

Considering the relation 2 of the assumption v):

if *error_in_advance* > 0 then

$$u_{k+1} = \frac{\min + u_{k-T}}{2} \qquad (11)$$

else

$$u_{k+1} = \frac{\max + u_{k-T}}{2}. \qquad (12)$$

## 4. The Adaptive Degree of Similarity

The *Dmax* parameter represents the maximum distance considered that allows $\omega_i$ to be positive. This parameter not only determines which rules will contribute with an interval for the final outcome, but also influences the calculation of the interval.

If the *Dmax* has a value that is too low, then the system output will oscillate. If its value is too high, the system output will have permanent error.

A time window is now considered to observe the behaviour of the output. If it oscillates the *Dmax* parameter is increased; if there is a permanent error it decreases. This way it is possible to make *Dmax* adaptive (on -line) as follows (13) (14):

$$Dmax = Dmax * (1 + percentage\ of\ oscillation) \qquad (13)$$

if the *Dmax* has to be increased;

$$Dmax = Dmax * (1 - percentage\ of\ error) \qquad (14)$$

if the *Dmax* has to be decreased.

## 5. Dividing the Rule -base Into Layers

If we consider a time invariant process, then it is easily concluded that several rules in the rule-base, mainly those which are in cells that do not belong to the region related with the desired reference, are not needed once the system output is following the reference. We could then structure the rule -base into two layers: an *active layer* and an *inactive layer*.

The active layer has the rules that are currently used. It would be good if this layer had always the same number of rules so that the sampling interval had not to be increased. If a rule belongs to a cell that is far away from the space currently occupied by the system's output/input then this rule could be moved to an inactive layer. This layer has all the rules that are not currently used, but that are kept in the rule -base because they may be needed in the future if some change in the desired reference occurs. If this is easy to implement when considering time invariant processes and the regulation problem, many problems are faced with, considering time variant processes and tracking problems.

It would be a good step to have a *vigilant layer* whose role would be to observe the performance of the output variable and to decide when to move a rule from the active to the inactive layer and vice versa and also when to erase a rule from the rule -base.

What sometimes happens is that the algorithm gets stuck to a rule that represents exactly the desired situation. If the process suffers any changes then this rule is no longer a valid rule but will continue in the rule- base until another rule is placed on its cell. As the process output will no longer be covering that region of the rule- base matrix, because the process changed its behaviour, and so the same input will lead to a different output, a new rule will not be created to replace the older and inaccurate one. This rule should not go to the inactive layer because it does not illustrate the behaviour of the new process characteristics. It should simply be erased.

The vigilant layer must have rules created by the human operators, and this may constitute a problem. It is very easy to construct these rules for the regulation problem, because the reference is constant and we can observe the relation between the change in the input and the change in the output signals.

If the output signal changes without a corresponding change in the input signal then it is easy to detect changes in systems parameters or disturbances and we can determine a corrective

action and forget the rules responsible for the input signal. But with non-constant references another kind of approaches is needed.

Further research is needed to determine what kinds of rules should constitute this vigilant layer and how to determine when to move a rule from the active to the inactive layer or simply to erase it.

# 6. Results and Discussion

## 6.1 1st Order Linear Process

Let's consider a 1st order linear process just to illustrate some basic characteristics of the controller. The process is defined by a difference equation given in (15):

$$y_k=0.8553*y_{k-1} + 0.1543*u_{k-2} \qquad (15)$$

In Figures 1, 2 and 3 we can see how the controller learns to control the process, the input given by the controller and finally the evolution of the interval that is the output of the inference mechanism. In this example we are considering as the domain of the input variable the interval [10,10].

The choice of the input variable domain influences the time needed for the learning phase. As an example, if we now consider the interval [2,2] as the input variable domain, we see in figure 4 that the controller learns much more rapidly and smoothly to control the process.

## 6.2 2nd Order Linear System

Considering the 2nd order linear system given by (16) with $\xi$=1 and $\omega_n$=4.4,

$$G(s) = \frac{\omega_n^2}{s^2 + 2\xi\omega_n + \omega_n^2} \qquad (16)$$

and a sampling interval of *200ms*, with a step reference the results are shown in Figures 5 and 6.

In Figure 6 we can observe how the value of the *Dmax* parameter is adaptively (on-line) modified according to the performance of the output of the system. After the learning phase it tends to remain constant.

## 6.3 Application To A Non-linear Real Process

The process is a small- scale laboratory one intended to produce a certain quantity of heated air. A simplified representation of the real process PT326 is shown in Figures 7 and 8. The air enters the tube due to a blower. Inside the tube there is an energised electrical resistance that heats the air (the actuator). The goal is to control the air temperature along the tube, which can be read by a sensor placed in one of three possible positions.

The results obtained for the regulation problem are illustrated in Figure 9.

Considering now a sinusoidal reference, we have to consider a sin with not a broad amplitude of values, so that we can have manageable sampling intervals (*150ms*). The result is shown in Figure 10. For the tracking problem we also consider a triangular and a square reference (Figures 11 and 12). As can be seen, the performance increases substantially after the relatively short learning phase.

**Note**: The computer configuration used was the following: Pentium 100 Mhz µcomputer, 16MB Ram, Matlab Programming Language for simulations, Dos LabWindows for real-time control.

# 7. Conclusions

The proposed SOC fuzzy controller shows good performance for regulation of normal (not very high) order processes. It is easily implementable and the computational time is low, allowing the use of microcomputers in real -time control. The adaptation of the similarity factor by a very simple technique yields a good performance. The feedforward effect introduced in the inference and deffuzification stages compensates for the pure time delay of the process.

There are however several characteristics of the controller that need further study and different approaches that will hopefully lead to an increase in its performance.

The direct relation between the accuracy of the discretisation of the input and output variables domains and the required sampling interval can be dealt with by structuring the rule- base into an active and an inactive layer. The active layer should always have the same number of rules, and this number should allow practical sampling intervals.

When considering time variant systems a vigilant layer should be incorporated such that older rules can be forgotten. The structure and construction of these algorithms require further study because it is difficult to find a set of rules that will apply in general cases. For the regulation problem, these rules could simply detect the occurrence of sudden changes in the process outputs without corresponding changes in its inputs. For the

tracking problem more complex rules have to be determined.

The Adaptation of the *Dmax* parameter is made by a rather simple algorithm but which releases the human operator from the boring task of finding a correct value for this parameter by trial and error.

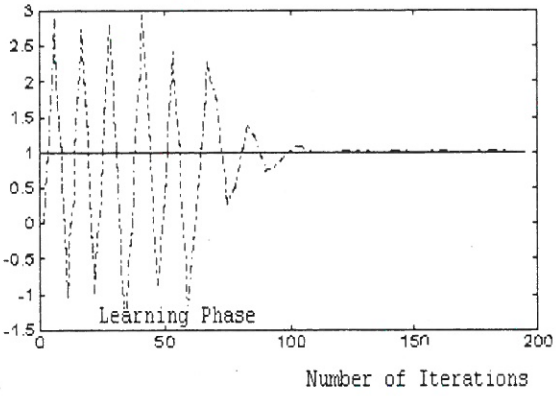


**Figure 1. Example 6.1 The Regulation Problem. Domain of Input Is Considered [-10, 10]**

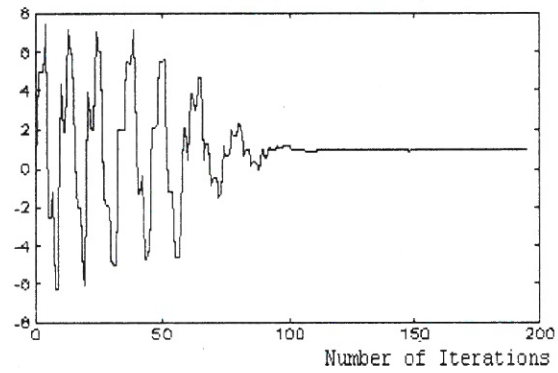


**Figure 2. Example 6.1Control Input Referring To Figure1**

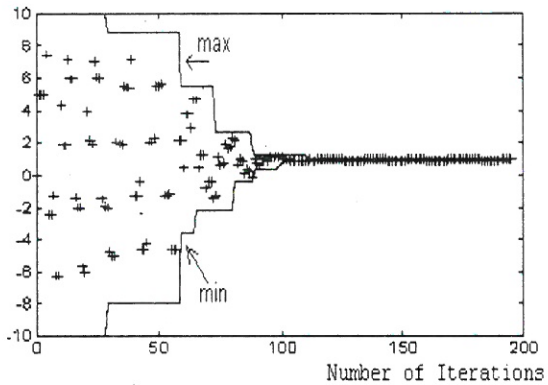


**Figure 3. The *Learned Interval* Given By (8). The Calculated Input Is Indicated By +**

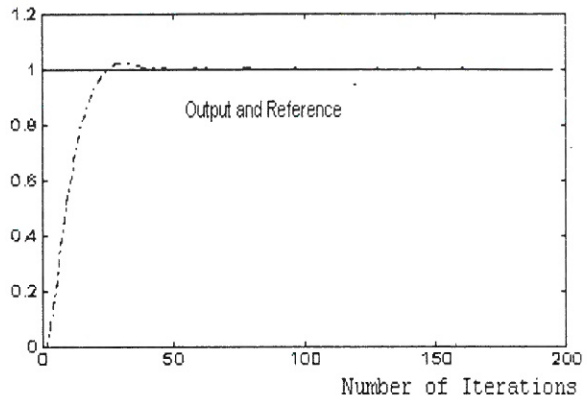


**Figure 4. Example 6.1 Regulation Problem. Influence of the Input Variable Domain Considered [-2,2]**

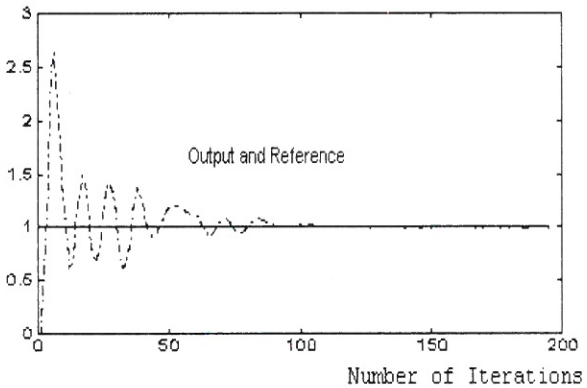


**Figure 5. Example 6.2 Regulation Problem**

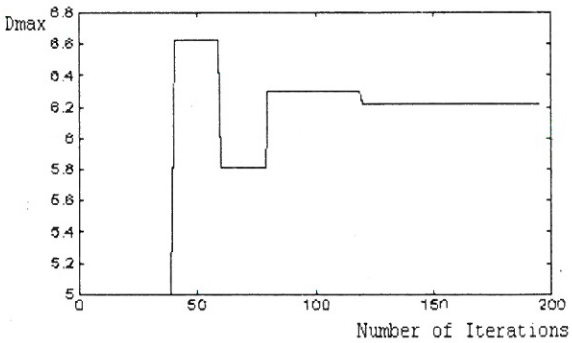


**Figure 6. The $D_{max}$ Parameter Adaptation in Example 6.2**

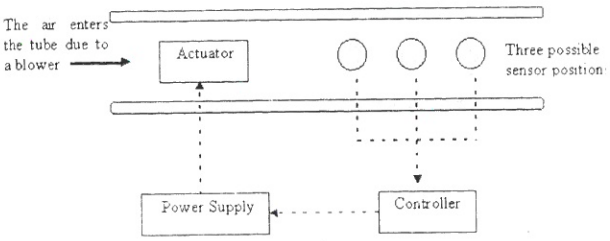


**Figure 7. Representation of PT326**

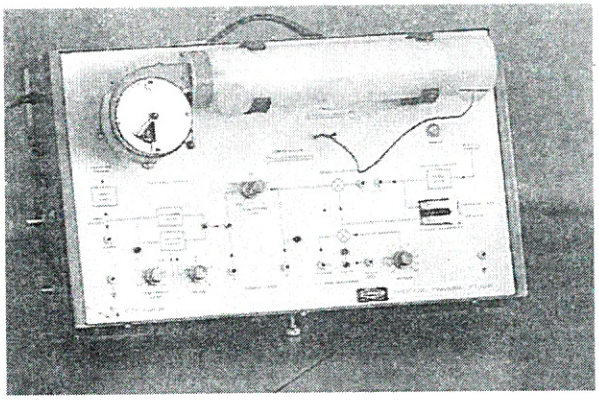

**Figure 8. Real Process PT 326 (picture)**

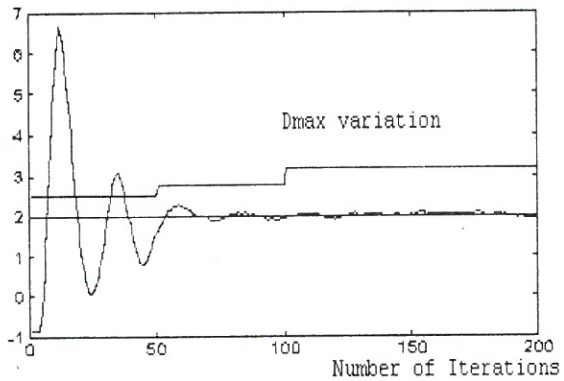


**Figure 9. Output of the Process and Variation of the *Dmax* Parameter**

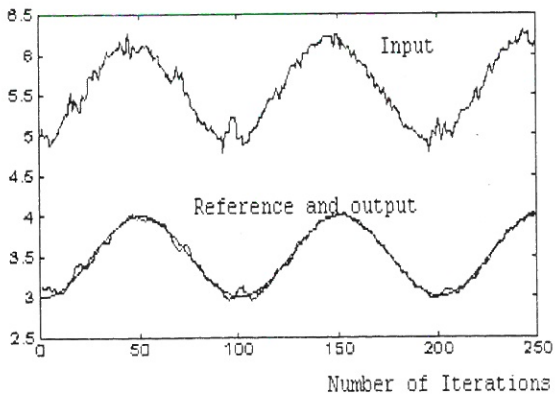


**Figure 10. The Tracking Problem With A Sinusoidal Reference. Note the Learning Phase**

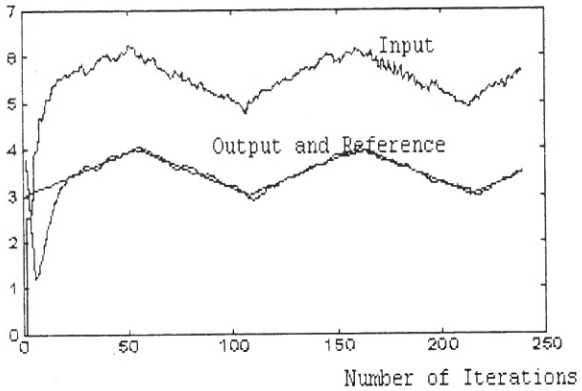


**Figure 11. The Tracking Problem With A Triangular Reference. Performance Increases With Learning**

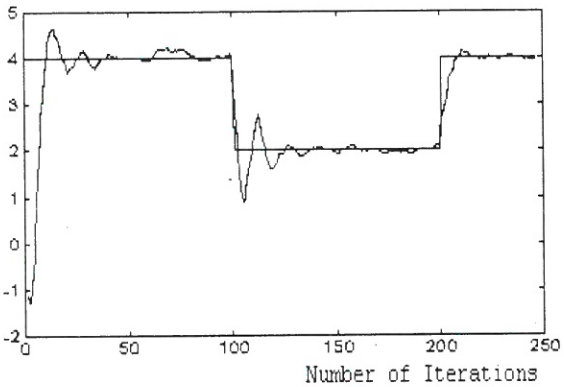


**Figure 12. The Tracking Problem With A Square Reference. Performance Increases With Learning**

# REFERENCES

CHEN , Y.Y and TSAO, T.C., **A Description of the Dynamical Behaviour of Fuzzy Systems,** IEEE TRANS. ON SYSTEMS MAN AND CYBERNETICS, Vol. 19, No. 4, 1989, pp. 745-755.

DIEULOT, J.-Y., HAJRI, S. and BORNE, P., **A Fuzzy Control Synthesis Using the Variable Structure Approach,** STUDIES IN INFORMATICS AND CONTROL, Vol. 5, No. 1, 1996, pp. 5-14, I.C.I. Publications, Bucharest.

DRYANKOV, D., HELLENDOORN, H. and REIFRANK, M., **An Introduction To Fuzzy Control**, SPRINGER VERLAG, 1993.

KARR, C.A. and GENTRY, E.J., **Fuzzy Control of PH Using Genetic Algorithms,** IEEE TRANS. ON FUZZY SYSTEMS, Vol. 1, No. 1, 1993, pp 46-53.

KLAWONN, F., GEBHARDT, J. and KRUSE, R., **Fuzzy Control On the Basis of Equality Relations With An Example From Idle Speed**

**Control**, IEEE TRANSACTIONS ON FUZZY SYSTEMS, Vol. 3, No. 3, 1995, pp.336-349.

LEE, C.H. and WANG, S.D., **A Self-organizing Adaptive Fuzzy Controller,** FUZZY SETS AND SYSTEMS, Vol. 80, Elsevier, 1996, pp. 295-313.

LIN, C.-T., LIN, C.-J. and LEE, C.S.G., **Fuzzy Adaptive Learning Control Network With On-line Neural Learning,** FUZZY SETS AND SYSTEMS, Vol. 71, 1995, pp. 25-45.

PARK, Y.-M., MOON, U.-C. and LEE, K.Y., **A Self-Organizing Fuzzy Logic Controller for Dynamic Systems Using A Fuzzy Auto-Regressive Moving Average Model**, IEEE TRANSACTIONS ON FUZZY SYSTEMS, Vol. 3, No.1, 1995, pp. 75-82.

PRESTI, M. L., POLUZZI, R. and ZANABON, A.M., **Synthesis of Fuzzy Controllers Through Neural Networks,** FUZZY SETS AND SYSTEMS, Vol. 71, 1995, pp. 47-70.