INFORMATICA, 2004, Vol. 15, No. 3, 337–362

Representations of Multi-Model Based Controllers by Using Artificial Intelligence Tools

Asier IBEAS, Manuel de la SEN
Dpto. de Ingeniería de Sistemas y Automática, Facultad de Ciencia y Tecnología
Universidad del País Vasco, Campus de Leioa
Apdo. 644, 48080 Bilbao, Spain
e-mail: iebibhea@ehu.es

Received: April 2004

Abstract. This paper develops a representation of multi-model based controllers by using artificial intelligence typical structures. These structures will be neural networks, genetic algorithms and fuzzy logic. The interpretation of multimodel controllers in an artificial intelligence frame will allow the application of each specific technique to the design of improved multimodel based controllers. *The obtained artificial intelligence based multimodel controllers are compared with classical single model based ones. It is shown through simulation examples that a transient response improvement can be achieved by using multiestimation based techniques.* Furthermore, a method for synthesizing multimodel based neural network controllers from already designed single model based ones is presented. *The proposed methodology allows to extend the existing single model based neural controllers to multimodel based ones, extending the applicability of this kind of techniques to a more general type of controllers.* Also, some applications of genetic algorithms and fuzzy logic to multimodel controller design are proposed. Thus, the mutation operation from genetic algorithms inspires a robustness test which consists of a random modification of the estimates which is used to select the estimates leading to the better identification performance towards parameterizing online the adaptive controller. Such a test is useful for plants operating in a noisy environment. *The proposed robustness test improves the selection of the plant model used to parameterize the adaptive controller in comparison to classical multimodel schemes where the controller parameterization choice is basically taken based on the identification accuracy of each model.* Moreover, the fuzzy logic approach suggests new ideas to the design of multiestimation structures which can be applied to a broad variety of adaptive controllers such as robotic manipulator controller design.

Key words: multimodel control, artificial intelligence, neural networks, genetic algorithms, fuzzy logic, switching.

1. Introduction

Adaptive controllers have been broadly studied during the last years since they allow to design control systems able to modify their behavior according to the characteristics of a changing environment or operation point (VanDoren (Ed.), 2003). Nevertheless, adaptive controllers may lead to a poor transient response in terms of plant output deviation from a reference plant one, due to an inadequate estimated plant parameter initialisation. *Thus, multi-model based controllers appeared as a way to improve the*

transient response of adaptive systems (Narendra and Balakrishnan, 1994; Gregorcic *et al.*, 2001; Chang and Davison, 1999; Narendra and Balakrishnan, 1997; Mosca and Agnoloni, 2001; Hocherman-Frommer *et al.*, 1998; Ibeas *et al.*, 2003) *by considering a set of parameter estimation algorithms running in parallel each one being initialised by a different set of estimated plant parameter values.* This mechanism allows to better fit a bad choice of the initial values of the estimates to start to run the estimation algorithm due to a convenient parameterization of the adaptive controller by adequate switching between estimators. A general multimodel based control scheme is composed by a set of different plant models running in parallel. These models, which may be fixed (Narendra and Balakrishnan, 1994; Gregorcic *et al.*, 2001; Chang and Davison, 1999) or adaptive (Narendra and Balakrishnan, 1997; Mosca and Agnoloni, 2001; Hocherman-Frommer *et al.*, 1998; Ibeas *et al.*, 2003), are different one from each other in what it is concerned with its structure and/or its parameter values. Thus, each one contains different characteristics of the controlled process. Then, a higher level switching structure between the various plant models chooses, at each time instant, which model is used to calculate the control law at that time instant according to a corresponding closed-loop performance index built for each estimation algorithm. In (Mosca and Agnoloni, 2001; Hocherman-Frommer *et al.*, 1998), several performance indexes for that purpose are discussed. Although the specific form of the performance index is different from one work to another, it is common to define it based on the difference between the real plant output and each model one, which is a natural choice since that index reflects how far a specific model is from the real plant behavior. Thus, the switching law acts as a supervisor of the system behavior deciding the estimator which will parameterize the adaptive controller in real time. This kind of control architecture allows to develop control system schemes capable to achieve a good performance in terms of speed, accuracy and stability for increasingly complex systems.

The structure and operation of the switching law between the different plant models have been studied from an artificial intelligence point of view in an expert systems context (de la Sen *et al.*, 2004; de la Sen and Almansa, 2002). However, multimodel structures themselves have always been modeled in a classical control theory frame (Narendra and Balakrishnan, 1994; Narendra and Balakrishnan, 1997; Ibeas *et al.*, 2003). *This paper proposes a possible interpretation of multimodel structures in an artificial intelligence frame. The artificial intelligence structures chosen for such goal have been, artificial neural networks (ANN), genetic algorithms (GA) and fuzzy logic* (Da Ruan (Ed.), 1997; Fausett, 1998; Etxebarria, 1994; Beyers, 1998; Tilli, 1992; Ibeas and de la Sen, 2004). *This interpretation will allow the use of specific characteristics of each one to the design of improved multimodel control schemes. Thus, a method for synthesizing multimodel-based neural network controllers from pre-designed single model ones is proposed*. This methodology will allow the development of multimodel neural controllers from the original single model based ones in a very easy way extending the applicability of multimodel controllers. *Also, some applications of genetic algorithms and fuzzy logic to multimodel control design are presented. The genetic algorithm approach will inspire the incorporation of a robustness measure of each model in the performance index used*

for evaluating the closed-loop behavior of the set of estimation algorithms running in parallel. Thus, the resulting scheme will be able to improve its behavior in comparison to the classical multimodel schemes, where this robustness measure is not included, when the plant is operating in a noisy environment as computer simulations have shown. Moreover a fuzzy logic approach is presented to interpret the architecture of the multiestimation based controllers. This focus can be applied to the design of multimodel controllers as recent works (Ibeas and de la Sen, 2004; Ibeas *et al.*, 2004) apply to the control of uncertain robotic manipulators. An adaptive, being more general than that related to the use of fixed models, formalism is used for making the interpretation. Each representation will be compared with the traditional single model based adaptive controller in order to illustrate the usefulness of the proposed schemes.

The paper is organized as follows. In Section 2, a brief description of the multiestimation scheme architecture is given. Section 3 deals with the artificial neural network representation while Sections 4 and 5 deal with the genetic algorithms and fuzzy logic ones respectively. Finally, conclusions end the paper.

2. Basic Multiestimation Scheme

In this section, a brief description of the multiestimation scheme used for subsequent discussion is presented. It has been considered the adaptive case since the fixed one is included in that as a particular case. Our objective is to design a model reference following pole placement based multimodel control for the discrete (the continuous case can be treated in the same way) time invariant linear SISO plant described by the difference equation:

$$A(q^{-1})y_k = B(q^{-1})u_k, \tag{1}$$

where u_k and y_k are the input and the output sequences respectively, q^{-1} is the one-step delay operator and

$$A(q^{-1}) = 1 + a_{n-1}q^{-1} + a_{n-2}q^{-2} + \cdots + a_0 q^{-n}, \tag{2.1}$$

$$B(q^{-1}) = b_m q^{-(n-m)} + b_{m-1}q^{-(n-m+1)} + \cdots + b_0 q^{-n} \quad (b_m \neq 0), \tag{2.2}$$

provided that $n \geqslant \max(m, 1)$ or $m \geqslant n = 0$ ($\Rightarrow A(q^{-1}) = 1$ in (2.1)) with (2.2) changed to $B(q^{-1}) = b_m + b_{m-1}q^{-1} + \cdots + b_0 q^{-m}$ ($b_m \neq 0$). Note that in this last case, the plant is realizable since it can be re-written in terms of the reciprocal polynomials $A^*(q) = q^m A(q^{-1}) = q^m$ and $B^*(q) = q^m B(q^{-1}) = b_m q^m + b_{m-1}q^{m-1} + \cdots + b_0$ leading to the realizable transfer function:

$$\frac{B^*(q)}{A^*(q)} = \frac{b_m q^m + b_{m-1}q^{m-1} + \cdots + b_0}{q^m}.$$

Furthermore, note that the Eqs. 1, 2 can be obtained from the more general equation

$$A(q^{-1})y_k = B(q^{-1})u_k,$$

$$A(q^{-1}) = 1 + a_{n_a-1}q^{-1} + a_{n_a-2}q^{-2} + \cdots + a_0 q^{-n_a},$$
$$B(q^{-1}) = b_{n_b} + b_{n_b-1}q^{-1} + \cdots + b_0 q^{-n_b},$$

with $n_a, n_b \geqslant 0$ by taking $n_a = n_b = n$ and making zero an adequate set of coefficients of $B(q^{-1})$. The above Eqs. 1, 2 define a linear difference equation which is usually written in adaptive control as the inner product of two vectors:

$$y_k = \varphi_k^T \theta = \theta^T \varphi_k, \tag{3}$$

where

$$\varphi_k^T = \left[-y_{k-1} \;\; -y_{k-2} \;\; \cdots \;\; -y_{k-n} \;\; u_{k-n+m} \;\; u_{k-n+m-1} \;\; \cdots \;\; u_{k-n} \right]$$

is the so called *regressor* and

$$\theta^T = \left[a_{n-1} \;\; a_{n-2} \;\; \cdots \;\; a_0 \;\; b_m \;\; b_{m-1} \;\; \cdots \;\; b_0 \right]$$

symbolises the true plant parameter vector (Ibeas *et al.*, 2003). This plant is an example of how the presented methodology can be applied in a concrete problem. Furthermore, in (Ibeas and de la Sen, 2004; de la Sen *et al.*, 2003), you can find the development of multimodel controllers to more general types of uncertain dynamical systems. However, in order to illustrate the application of the artificial intelligence representations to a concrete problem in an easy way, the above linear difference plant has been chosen (Ibeas *et al.*, 2004). If the true plant parameter vector is unknown, parameter estimation has to be used. Thus, an estimated parameter vector $\hat{\theta}_k$ is considered at each sample k provided by any parameter estimation algorithm. The estimated parameter vector generates an estimated plant output through an equation analogue to (3):

$$\hat{y}_k = \varphi_k^T \hat{\theta}_k. \tag{4}$$

Then, this estimated parameter vector is used for control law calculations at each sample. *If this estimated vector is initialised far from the real plant parameter vector, then the transient response will have large deviations from the desired output resulting in a bad performance. This fact motivates the consideration of a set of estimation algorithms running in parallel, each one with its own estimated parameter vector*:

$$\hat{\theta}_k^{(1)}, \hat{\theta}_k^{(2)}, \ldots, \hat{\theta}_k^{(N_e)}, \quad \hat{y}_k^{(i)} = \varphi_k^T \hat{\theta}_k^{(i)}, \quad 1 \leqslant i \leqslant N_e,$$

where N_e is the number of total estimators. Each estimated vector is updated at each sample according to a parameter estimation algorithm (which may be different for each one) driven by the input and output measurements of the plant. The multiestimation scheme block diagram is displayed in Fig. 1.

There exist N_e estimation algorithms running in parallel (i.e., at each sampling time $t_k = k\tau, \tau$ being the sampling period, every algorithm gives the estimates parameter

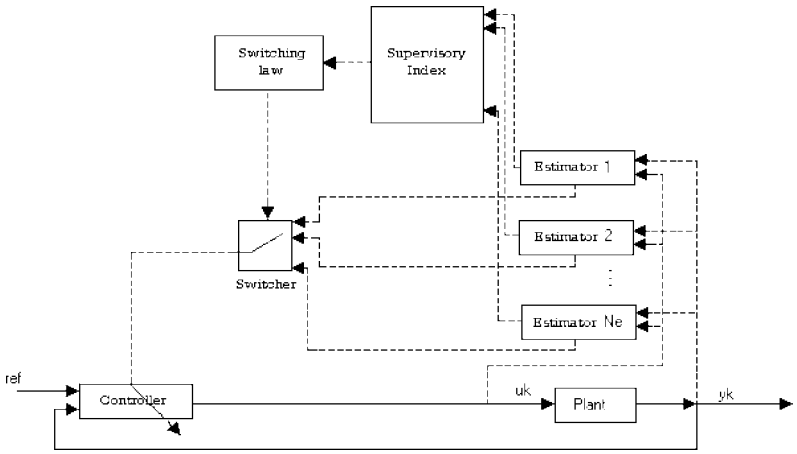

Fig. 1. Basic multiestimation scheme architecture.

vector $\hat{\theta}_k^{(i)}$ and the estimated plant output $\hat{y}_k^{(i)}, 1 \leqslant i \leqslant N_e$, based on past plant input and output measurements). Each algorithm is different from each other in what is concerned with the estimated parameter vector initialisation and/or the kind of the estimation algorithm and integrate the so-called multiestimation scheme. Thus, every identification algorithm is indexed with only one integer $1 \leqslant i \leqslant N_e$. Denote by c_k the identification algorithm which parameterises the adaptive controller at time t_k. A switching rule based on the identification errors $e_k^{(i)} = y_k - \hat{y}_k^{(i)} = \varphi_k^T \tilde{\theta}_k^{(i)}, \tilde{\theta}_k^{(i)} = \theta - \hat{\theta}_k^{(i)}$, of the N_e estimation algorithms chooses at each sampling time t_k the individual estimation scheme (identifier) c_k which parameterises the active controller at time t_k. The proposed identification performance index for each estimation algorithm is

$$J_k^{(i)} = \sum_{\ell=k-M}^{k} \lambda_\ell^{k-\ell} \Big[\beta_1 \big(y_\ell - \hat{y}_\ell^{(i)}\big)^2 + \beta_2 \big(\hat{u}_\ell^{(c_{k-1})} - \hat{u}_\ell^{(i)}\big)^2 \Big], \quad \beta_1 + \beta_2 = 1, \quad \beta_1, \beta_2 \geqslant 0, \tag{5}$$

where M is an integer number large enough to give sense to the performance evaluation. The first addend of the index (5), can be interpreted as a measure of the long-term accuracy of each identification algorithm, where the forgetting factor $\lambda_\ell \in (0, 1)$ (which, in general, can be sample-dependent) establishes the effective memory of the index in rapidly changing environments. The second part of the index (5) can be interpreted as the possible jump in the control law associated to switching between controller parameterisations. The supervisory index is built through a convex linear combination of the previous terms as it is highlighted by Eq. 5. Moreover, *the real plant parameter vector is not used in any calculation since the identification error may be equivalently written as* $e_k^{(i)} = y_k - \varphi_k^T \hat{\theta}_k^{(i)}$ *being a measurable signal since the real plant output is measured*. The switching law must respect a minimum *dwell* or *residence time* between consecutive switchings between estimators in order to guarantee closed-loop stability. Furthermore,

once the supervisory index (5) has been stated, the switching rule for the adaptive controller reparameterisation is obtained from the performance index (5) as follows. If the elapsed time from the last switching is smaller than the residence time, then switching is not allowed and the current controller is maintained in operation. On the other hand, if the elapsed time from the last switching is larger than or equal to the residence time, then switching is allowed. In this case, the controller chosen from the multiestimation scheme is such that it has the minimum performance index (5) from all pairs estimator-controller parameterisation available. If two estimators have the same performance index, then the selected controller is the one with the minimum index in the ordered set $\{1, 2, \ldots, N_e\}$. If π denotes the last sample in which switching between estimators happened, then the switching map c_k at a sample $k, (k > \pi)$ can be stated as

$$c_k = \begin{cases} c_{k-1}, & k - \pi < N_D, \\ j \,|\, j = \min\left\{\zeta \,|\, J_k^{(\zeta)} = \min\{J_k^{(r)} \,|\, r \in K\}\right\}, & k - \pi \geqslant N_D, \end{cases} \tag{6}$$

where N_D (positive integer) is the residence number of samples. Then, the selected estimated parameter vector $\hat{\theta}_k^{(c_k)}$ is used by the control algorithm, as an estimation of the real plant parameters, to calculate the control compensators and to generate the control signal. *Thus, the parallel multiestimation scheme is used to improve the transient response of single model based adaptive controllers through appropriate switching to a more convenient parameterization of the controller. Furthermore, the proposed control methodology has been applied to discrete plants with unmodeled dynamics (de la Sen et al., 2003), and even to continuous nonlinear dynamical systems such as robot manipulators (Ibeas and de la Sen, 2004). Nevertheless, the artificial intelligence representations are developed over a linear discrete SISO plant in order to show the representation with clarity and the application of the representations to the standard multiestimation scheme. The artificial intelligence structures are proposed as models of representation of the multiestimation scheme and they will allow the incorporation of ideas inspired in them to the design of this kind of control schemes.* Furthermore, note that since a set of N_e estimation algorithms integrate the parallel multiestimation scheme then, it is obvious the usefulness of selecting a subset of them that better fit the identification performance (genetic approach). Also, a convex linear combination of each estimated parameter vector from all the set of estimators may be formulated by choosing the weights of such a combination from fuzzy logic rules instead of selecting any of the individual estimates via switches through time as usual in standard parallel multiestimation schemes. For example, the fuzzy logic approach has been applied to the design of multimodel based controllers for nonlinear robotic manipulators (Ibeas and de la Sen, 2004). Since the objective of this paper is to develop an artificial intelligence representation of multi model structures rather than describing the control scheme, the reader is referred to reference (Ibeas *et al.*, 2003), where a complete discussion of the stability issues and the pole placement adaptive control design is available. *The switching rule design has been focused from an expert systems formalism in previous works (de la Sen et al., 2004; de la Sen and Almansa, 2002). However, the multiple models that compose the multiestimation scheme have been always modeled by classical control theory structures.* Thus, the structure of each model is set independently

of the rest. The objective of this paper is to propose some methods to represent the set of multiple models in an artificial intelligence frame. This focus will allow to extend the existing designs of single model based controllers to the design of multimodel based ones in an easy way as it is shown in the sequel. Therefore, in the next sections an artificial intelligence representation of the above multiestimation scheme is given for various typical artificial intelligence structures (Da Ruan (Ed.), 1997).

3. Artificial Neural Networks

In this section, an artificial neural network (ANN) representation for the multiestimation based control scheme described in Section 2 is developed, (Fausett, 1998; Wang and Qing, 2004; Etxebarria and de la Sen, 1996; Melin and Castillo, 2003).

3.1. *ANN for Representing Single Estimation Based Schemes*

The following two layered ANN is presented as a model for a discrete time single model based adaptive control in (Etxebarria, 1994). The difference Eqs. 1, 2 is implemented, for estimation purposes, by the ANN displayed in Fig. 2, where the activation functions are linear for all neurons.

The ANN output (which plays the role of the estimated plant output) can be written as

$$\hat{y}_k = \sum_{i=1}^{n} w_{i,k} y_{k-i} + \sum_{j=0}^{m} w_{n+j+1,k} u_{k-j} = w_k^T \varphi_k, \tag{7}$$

where

$$w_k^T = \begin{bmatrix} w_{1,k} & w_{2,k} & \dots & w_{n+m,k} \end{bmatrix}$$

is the network weights vector and φ is the above defined regressor. Comparing Eq. 6 with Eq. 4, it can be observed that the network weight vector w_k plays the same role as the

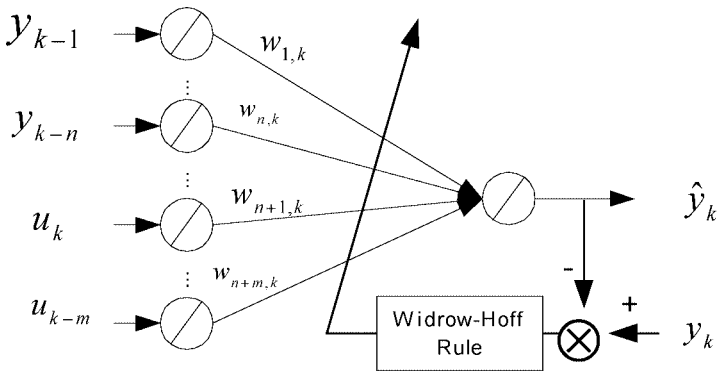

Fig. 2. Single neural network estimator.

estimated plant parameter vector $\hat{\theta}_k$. Network weights (or equivalently plant parameters) are updated by using the well known *Widrow-Hoff* rule for multiple-input single-output ANN:

$$w_k = w_{k-1} + \frac{\alpha\,(y_k - \hat{y}_k)\,\varphi_{k-1}}{\varepsilon + \varphi_{k-1}^T \varphi_{k-1}}, \quad \varepsilon > 0, \quad \alpha \in (0,2), \tag{8}$$

where $\hat{y}_k$ denotes the ANN estimated plant output while y_k denotes the real measured plant output (Etxebarria, 1994). Thus, network weights are updated by comparing the network output with the real plant output (which it is the target value). Then, the weights vector (whose role is represented by the estimated plant parameters vector) is used for controller design purposes. This network is extended to represent multiestimation structures in the following section.

3.2. *ANN for Representing Multiestimation Schemes*

Now, *the multiestimation scheme introduced in Section 2 can be represented with an ANN by increasing the number of neurons in the output layer of the above ANN to a number of neurons equal to the number of different estimators used in the multiestimation scheme.* Since the output layer has one single neuron in the previous network, a multiestimation scheme with N_e estimators running in parallel will have N_e neurons in its output layer as Fig. 3 displays for the case of $N_e = 2$.

Hence, the number of connections between neurons and the number of weights are increased. Thus, the proposed ANN is a unique structure containing itself the N_e esti-

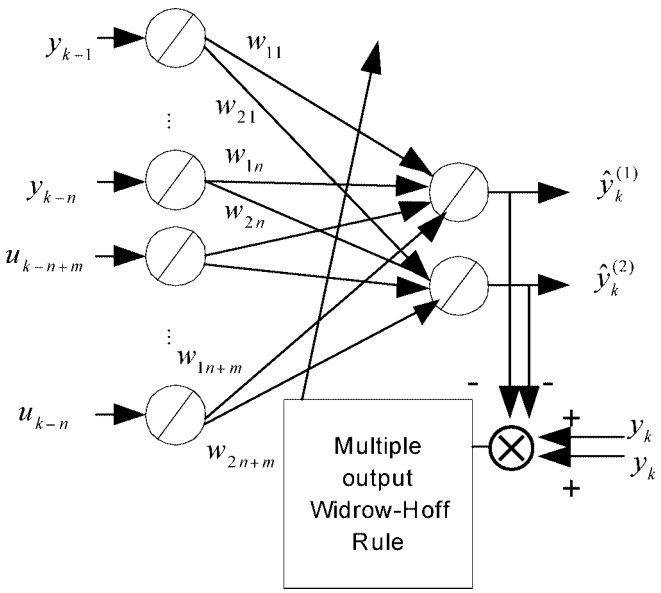

Fig. 3. Multiestimation neural network.

mated parameter vectors (which are represented by the corresponding weights vectors). Furthermore, the ANN plant estimated outputs can be written as

$$\hat{y}_k^{(i)} = \sum_{\ell=1}^{n} w_{i\ell,k} y_{k-\ell} + \sum_{\ell=0}^{m} w_{i,n+\ell+1,k} u_{k-\ell} = w_{i\cdot,k}^T \varphi_k, \quad 1 \leqslant i \leqslant N_e,$$

with

$$w_{i\cdot,k}^T = \left[\, w_{i1,k} \quad w_{i2,k} \quad \dots \quad w_{i(n+m),k} \, \right], \quad 1 \leqslant i \leqslant N_e.$$

Note that from the network weight values $w_{ij,k}$, two estimated plant parameter vectors may be defined at each sample k by

$$w_{1\cdot,k} = \hat{\theta}_k^{(1)}, \quad w_{2\cdot,k} = \hat{\theta}_k^{(2)}.$$

The target vector (with which the ANN is trained) is defined in this case by repeating the original target value as many times as the number of estimators used. Since the original target value was the real measured plant output, y_k then, in the case with two estimators, the new target vector is defined by

$$y_k^{*T} = \left[\, y_k \quad y_k, \, \right],$$

while in the general case with N_e estimators, it will be

$$y_k^{*T} = \Big[\underbrace{y_k \quad y_k \quad \dots \quad y_k}_{N_e} \Big].$$

Then, the switching logic compares the performance indexes (5) associated to each output of the ANN and chooses the set of weights (estimated parameter vector) associated with the best estimated output according to (6) in order to calculate the control signal. The training rule is the generalization of the above *Widrow-Hoff* single output training rule (8) to the multiple output case:

$$\begin{aligned} w_{ij,k} = \; & w_{ij,k-1} + \frac{\alpha\big(y_k - \hat{y}_k^{(i)}\big)\varphi_{j,k-1}}{\varepsilon + \varphi_{k-1}^T \varphi_{k-1}}, \\ & \varepsilon > 0, \quad \alpha \in (0,2), \quad i = 1,2,\dots,N_e, \quad j = 1,2,\dots,n+m+1, \end{aligned} \tag{9}$$

where $\varphi_{j,k-1}$ stands for the j-th component of the vector φ_{k-1}. Note that the updating law for the network weights (or equivalently, the estimated plant parameters vectors) is formulated for the multiple output ANN as a unique entity as well.

The above idea can be extended to a more general case in which the ANN has a number of layers greater than two and a number of neurons in the output layer greater than one with arbitrary (generally nonlinear continuous) activation functions. Thus, *the*

following rule is proposed in order to obtain multimodel based ANN controllers from a pre-designed ANN single model one. Suppose that the single model ANN has N_ℓ layers and N_o neurons in its output layer. *Then, if a new ANN is defined for the multimodel structure as an ANN with the same number of layers as the original one, and a number of neurons in the output layer equal to* $N_o' = N_e N_o$ where N_e is the number of estimators considered, then, the target vector is now built by repeating the original target vector (from the single model ANN) as many times as the number of estimators considered. The switching logic acts as an intelligent supervisor deciding the set of weights that will be used for control purposes at each time instant. *In such an easy way, the multimodel structure can be integrated with conventional neural network based controllers in order to obtain more general ANN based multimodel structures.* The training rule is the same as in the first ANN, extended to the new weights associated to new connections. The general multimodel neural network scheme is displayed in Fig. 4.

The Table 1 summarises the way in which a single model based ANN can be extended to a multiple model based one.

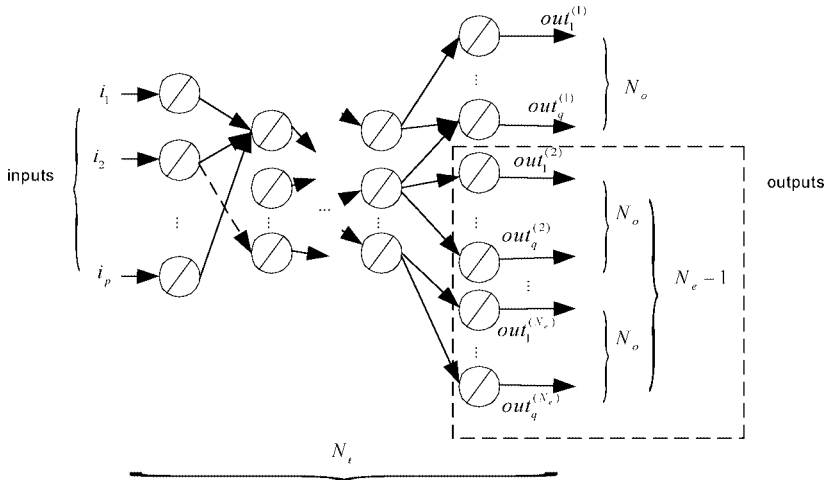

Fig. 4. General multimodel ANN scheme.

Table 1
Table extending the single based ANN to a multimodel based one

	Original Neural Network	***Multimodel Neural Network***
Number of models considered	1	N_e
Number of layers	N_ℓ	N_ℓ
Number of neurons in the output layer	N_o	$N_o' = N_e N_o$
Target vector	x	$x^* = \underbrace{\begin{bmatrix} x & x & \dots & x \end{bmatrix}}_{N_e}$

3.3. *Simulation Example for the ANN Representation*

In this Section, a simulation example containing two estimation algorithms and the above training rule (9) for the ANN displayed in Fig. 3 is presented. The switching logic is assumed to respect a minimum residence number of samples given by $N_D = 2$ between successive switchings between estimation algorithms in order to guarantee closed-loop stability (Ibeas *et al.*, 2003). The switching law and the controller design (based on a model following pole placement algorithm) are described with detail in (Ibeas *et al.*, 2003). The discrete plant has the real plant parameter vector

$$\theta^T = \begin{bmatrix} -1.9 & 0.73 & -0.195 & 1 & -0.6 & 0.0875 \end{bmatrix},$$

and the reference model is characterized by the following vector

$$\theta_m^T = \begin{bmatrix} -0.6 & 0.11 & -0.006 & 1 & -0.32 & 0.0255 \end{bmatrix},$$

while the estimators are initialised with the following estimated parameter vectors (or network weights):

$$w_{1\cdot,0}^T = \hat{\theta}_0^{(1)T} = \begin{bmatrix} -0.5 & 0.25 & -0.5 & 0.79 & -0.5 & 0.08 \end{bmatrix},$$
$$w_{2\cdot,0}^T = \hat{\theta}_0^{(2)T} = \begin{bmatrix} -1.5 & 0.7 & -0.2 & 0.9 & -0.5 & 0.08 \end{bmatrix}.$$

It is taken $\varepsilon = 0.001$ and $\alpha = 1$. The input signal is a unity square wave with a 20 samples period. The performance index to decide switches is given by Eq. 5 with $\lambda = 0.95, \beta_2 = 1$; $\beta_3 = 0$. This choice for the parameters β indicates that the switching process only takes care of the identification error in order to choose the estimator which will parameterise the adaptive controller. The single adaptive control scheme is initialised with the first estimator. The following simulations comparing both, the single estimation ANN and the multiestimation one, are obtained:

From Figs. 5 and 6, it can be concluded that a transient response improvement can be achieved by a using a multiestimation scheme. The transient overshoot is less in the multiestimation scheme than in the single based one while the real plant output tends to the desired output smoother in the multiestimation case than in the single based one as well. It is showed through Figs. 7 and 8 that the system improves its behavior by using the best weight set at each time (respecting the residence time constraint). The switching map c_k illustrating the switching process between both set of weights (estimated parameter vectors) is showed in Fig. 7. Fig. 8 displays the real and estimated outputs of the ANN as well as the combined estimated output defined by $\hat{y}_k^{(c_k)}$ where c_k denotes the switching map. Moreover, Fig. 9 displays the evolution of the combined estimated plant parameter vector $\hat{\theta}_k^{(c_k)} = (\hat{\theta}_{k,j}^{(c_k)})$ through time. The proposed ANN integrates in a unique structure the complete set of estimation models running in parallel and the updating equations for the weights, whose role is represented by the estimated parameter vectors of a classical multiestimation scheme. Thus, from a designed single model-based ANN, a multimodel ANN has been designed in an easy way and applied to a concrete problem.

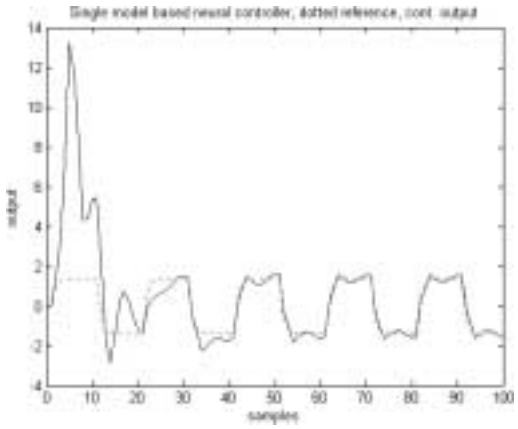

Fig. 5. Single model based output.

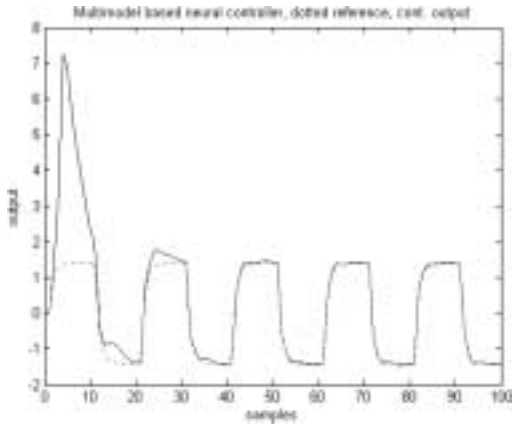

Fig. 6. Multimodel based neural controller.

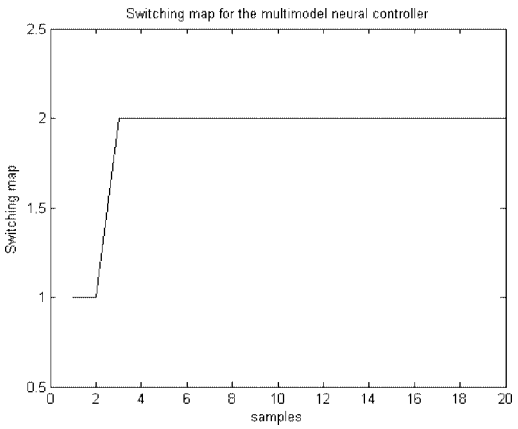

Fig. 7. Switching map for the multimodel ANN.

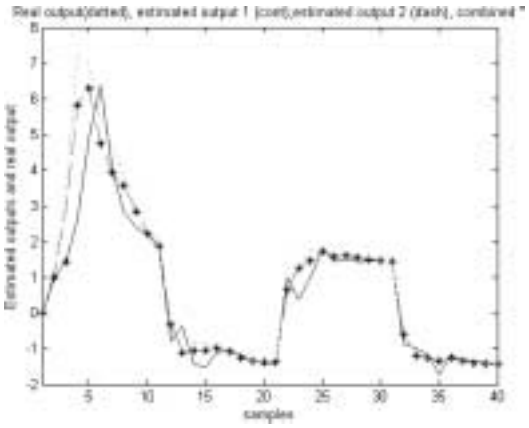

Fig. 8. Estimated outputs for the multimodel ANN.

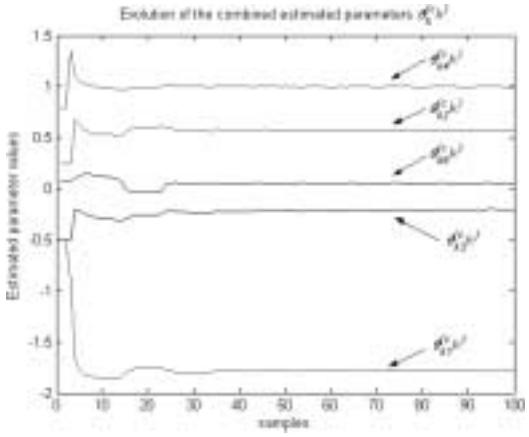

Fig. 9. Evolution of the combined estimated parameter vector.

4. Genetic Algorithms

In this Section, a genetic algorithm representation for multiestimation based control schemes is given. Genetic algorithms are usually used as optimisation tools in complex problems (Wetter and Wright, 2004). However, they have recently been applied to the intelligent control of dynamic processes (Mwembeshi *et al.*, 2004). The key idea is to use the *natural selection* and the *genetics* to obtain, at each generation, more accurate solutions to an original complex problem (Beyers, 1998). First, a codification for the solutions for the proposed problem is decided. The codification process consists of deciding how the information about our problem has to be managed by the genetic algorithm. The codification may be formed by binary (formed by 1's and 0's) or numeric (natural, integer, real, ...) vectors. These vectors are called *chromosomes* in the GA context. In the multiestimation case, the *chromosomes* will be vectors of real components containing the

plant parameter values. The *best* vector is that for which the estimated output (associated to that parameter vector) is closer to the real plant output. A general description of a genetic algorithm is given by the Fig. 10.

In the first step, there exists an initial set of (plant parameter) vectors uniformly distributed over the possible parameter space. This is a typical assumption in the adaptive control problem: the existence of a convex and compact subset of the parameter space where the real plant parameter vector is assumed to belong to. These initial vectors are denoted by $\hat{\theta}_0^{(1)}, \hat{\theta}_0^{(2)}, \ldots, \hat{\theta}_0^{(N_e)}$, i.e., they represent the estimated plant parameter vectors. Once the GA is initialised it starts running. First, one of the above vectors is chosen in order to generate the control law. The selection is made according to a performance index which evaluates the *quality* of each vector (in the first step the choice may be arbitrarily). The unique requirements about the performance index (in a GA context) are that it must be nonnegative and monotonically increasing with quality, i.e., the better vector is that which has the greater performance index. Thus, the GA selection operator is represented in the current context by the switching law which chooses the estimation algorithm used to parameterise the adaptive controller. Moreover, the adaptive counterpart of the cross over operator may be the updating rule for the estimation algorithms such as the least squares one or any of its variants for example while the mutation operator is used as an identification robustness test in the sequel. Once the process has been completed, then, the algorithm starts again and the process is repeated so on. The parallelism between multimodel based controllers and genetic algorithms is illustrated by the schematic table (Table 2).

Note that the GA representation allows to use a broad class of modification rules for the estimated vectors which may not be driven by a classical parameter updating equation. Furthermore, the number of different models (the number of chromosomes) may not be constant during the system operation. *This suggests the following interesting idea for multimodel based controllers.* If the system detects that with a reduced number of models an acceptable system behaviour is achieved, then it may suppress some of the

Table 2

Parallelism between multimodel controllers and genetic algorithms

Genetic Algorithms	***Multimodel Controllers***
Number of individuals	*Number of estimators*
Individual of the GA	*Estimated parameter vector*
Evaluation of the fitness of each individual	*Performance index for each estimation algorithm*
Selection of the best individuals	*Selection of the current estimator to parameterise the adaptive controller*
Generation of the offspring	*Generation of the new estimated vectors by the estimation algorithm updating equations*
Replace the population with the offspring	*Replace the estimated parameter vectors with the new ones*
Next generation	*Next step*

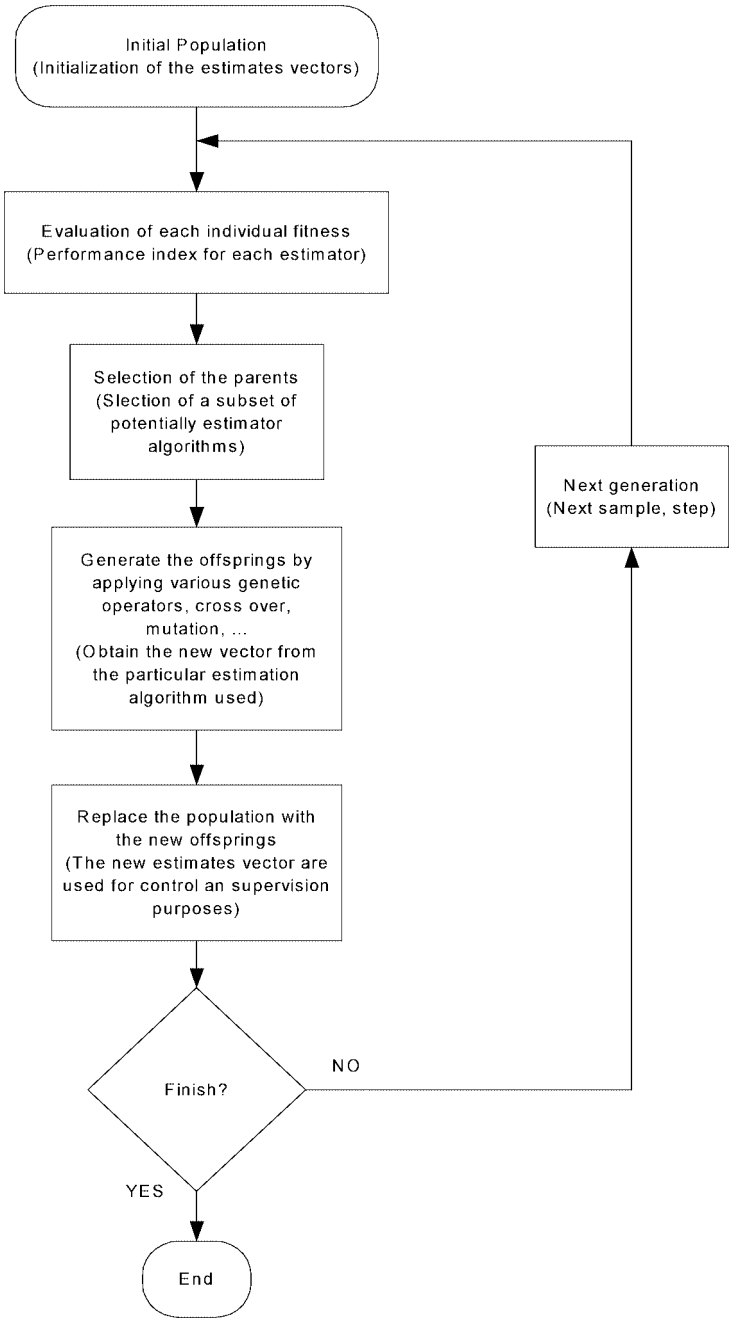

Fig. 10. General structure of a genetic algorithm.

models (chromosomes) in order to prune unnecessary computations. Thus, the multiple models are classified into *priority sets* in such a way that models with similar performance indexes belong to the same set (according to some performance criteria, for example, all models with their performance indexes inside a prescribed range belong to the same set). *In the context of the current multiestimation scheme, the above idea translates into eliminating online through time a subset of the identification algorithms that fit worse the identification performance.* The architecture of the multiestimation scheme is displayed in the Fig. 11, where σ denotes a permutation of the $\{1, 2, \ldots, N_e\}$ estimation algorithms. Thus, the sets associated to models with the worst performance may be pruned from the GA process while those sets containing the most accurate models may be recompensed by increasing the number of models inside them. Thus, from a general uniformly distributed different estimation models according to a swept of the initial conditions, the system is able to obtain an improved number of estimation models achieving an acceptable system performance. The genetic algorithm representation has led to new ideas to be incorporated to the classical multiestimation schemes.

Furthermore, the mutation operation, typical in GA processes can be applied to our problem in the following concrete way. At each time instant, mutate randomly the set of estimated parameter vectors $\{\hat{\theta}_k^{(1)}, \hat{\theta}_k^{(2)}, \ldots, \hat{\theta}_k^{(N_e)}\}$ to obtain a set of perturbed estimated parameter vectors $\{\delta\hat{\theta}_k^{(1)}, \delta\hat{\theta}_k^{(2)}, \ldots, \delta\hat{\theta}_k^{(N_e)}\}$. This new set is obtained by summing a gaussian random perturbation with zero mean, variance unity and amplitude A to

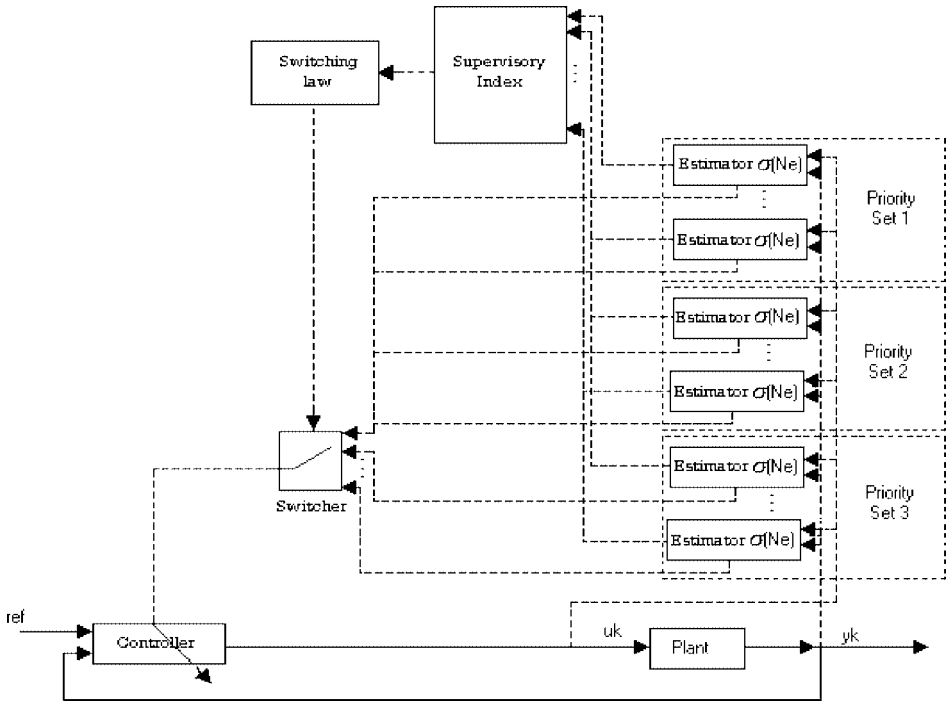

Fig. 11. Priority set diagram for the multi model scheme.

each component of the estimated plant parameter vector. The amplitude of the mutation is expressed as a % of the values of the original estimates vector

$$\left|\frac{\delta\hat{\theta}_{kj}^{(i)} - \hat{\theta}_{kj}^{(i)}}{\hat{\theta}_{kj}^{(i)}}\right| \times 100 \leqslant A,$$

where $\hat{\theta}_{kj}^{(i)}$ is the *j*-th component of the $\hat{\theta}_{k}^{(i)}$ vector. These perturbed estimated plant parameter vectors are then used to calculate a perturbed estimated plant output $\{\delta\hat{y}_k^{(1)}, \delta\hat{y}_k^{(2)}, \ldots, \delta\hat{y}_k^{(N_e)}\}$ through the equations:

$$\delta\hat{y}_k^{(i)} = \varphi_k^T \delta\hat{\theta}_k^{(i)}, \quad i = 1, 2, \ldots, N_e. \tag{10}$$

Thus, the perturbed estimated plant output can be compared with the unperturbed estimated plant one over a time interval in order to evaluate the robustness quality of each estimator. Then, the proposed performance index is given by

$$\begin{aligned} J_{k,GA}^{(i)-1} = & \sum_{\ell=k-M}^{k} \lambda_\ell^{k-\ell}\Big[\beta_1\big(y_\ell - \hat{y}_\ell^{(i)}\big)^2 + \beta_2\big(\hat{u}_\ell^{(c_{k-1})} - \hat{u}_\ell^{(i)}\big)^2\Big] \\ & + \beta_3 \sum_{\ell=k-N}^{k} \lambda_\ell^{k-\ell}\big(\hat{y}_\ell^{(i)} - \delta\hat{y}_\ell^{(i)}\big)^2, \end{aligned} \tag{11}$$

with $\beta_1 + \beta_2 + \beta_3 = 1,\ \ \beta_1, \beta_2, \beta_3 \geqslant 0$. Note that the inverse of the index (5) has been considered in this case since a larger value for $J_k^{(i)}$ reveals a worse behaviour of the corresponding estimator while the GA philosophy states that the chromosomes quality function must be increasing with quality. Thus, the estimator selected to parameterise the adaptive controller, once the residence time constraint has been fulfilled, is the one associated to the maximum performance index $J_{k,GA}^{(i)}$ (which is equivalent to select the estimator with the minimum value of the $J_{k,GA}^{(i)-1}$ indexes). The proposed evaluation function (11) has three parts. To the previous defined two ones (5) a measure about the identification robustness quality of each estimator has been added. Note that the robustness test is performed over a window of N samples size. The inclusion of this term is specially adequate when the plant is operating in a noisy environment. Thus, the deviation of the perturbed plant estimated outputs form the estimated unperturbed ones can be interpreted as a measure of the robustness property of the corresponding estimator and included in the criteria for switching between estimators.

4.1. *Refinement of the Robust System Working*

Moreover, the number of estimation algorithms can be decreased through time in order to reduce the computational cost of the algorithm. Thus, the number of estimation algorithms running in parallel can be pruned according to the following algorithm:

$\{N_{e0}, \Delta N_e > 0, N_{eThreshold}, N_{test} \neq N_D$ *are selected by the designer*$\}$
$N_e(0) \leftarrow N_{e0}, k_{lastTest} \leftarrow 0$
for $k > 0$
  if $k = k_{lastTest} + N_{test}$ *then*
   if $N_e(k) > N_{eThreshold}$
    $\{\hat{\theta}_k^{(\sigma(1))}, \hat{\theta}_k^{(\sigma(2))}, \ldots, \hat{\theta}_k^{(\sigma(N_e))}\}$
      $\leftarrow$*DecreasingOrder*$\left(\{\hat{\theta}_k^{(1)}, \hat{\theta}_k^{(2)}, \ldots, \hat{\theta}_k^{(N_e)}\}, J_{k,GA}^{(i)}\right)$
    $\{\hat{\theta}_k^{(\sigma(1))}, \hat{\theta}_k^{(\sigma(2))}, \ldots, \hat{\theta}_k^{(\sigma(N_e))-\Delta N_e}\} \leftarrow \{\hat{\theta}_k^{(\sigma(1))}, \hat{\theta}_k^{(\sigma(2))}, \ldots, \hat{\theta}_k^{(\sigma(N_e))}\}$
    $N_e(k) \leftarrow N_e(k-1) - \Delta N_e$
    $k_{lastTest} \leftarrow k$
   else
    $N_e(k) \leftarrow N_e(k-1)$
   end_if
  else
   $N_e(k) \leftarrow N_e(k-1)$
  end_if
end_for

where σ represents a permutation of the $N_e(k)$ estimation algorithms. When the residence number of samples for the prune algorithm N_{test} is fulfilled, then the estimators are ordered in a decreasing sequence according to the value of their performance indexes $J_{k,GA}^{(i)}$. Then, those (a number of ΔN_e) estimators with the minimum value are pruned for the estimation process, i.e., ΔN_e of the estimators are eliminated at regular time instants $t = kT_sN_{test}$ (T_s being the sampling period) where k is a positive integer number, until the number of estimators reaches a minimum threshold of estimates $N_{eThreshold}$, when the prune process is stopped. After that, the system is only able to switch between the rest of the existing estimators. The prune algorithm stops when the number of estimators reaches a minimum threshold. Then, the same number of estimators is maintained. The main usefulness of the scheme relies on reducing the computational cost of the multiestimation scheme (in memory requirements basically) removing those estimators with the worst performance index, as the priority set idea exposed above states. As it is typical in intelligent control (de la Sen *et al.*, 2004; de la Sen and Almansa, 2002), both supervisors have been designed to act with a different rate over the system. This choice avoids conflictive decisions between both supervisors.

The following simulation example illustrate the working of the proposed schemes. The simulation compares the working of a single model based adaptive control scheme, the classical multimodel based adaptive control scheme for which $\beta_3 = 0$, and the genetic algorithm approach in both schemes: the first one in which the number of estimators remain impassive through time and the second one in which the number of estimators is reducing as time grows.

4.2. *Simulation Example*

Now, a simulation example comparing the adaptive schemes is presented. We will suppose that the plant input and output are affected by a gaussian random perturbation with zero mean, unity variance and a maximum amplitude of 4% of the original plant input and output signals. Note that in order to generate the perturbed plant vectors $\delta\hat{\theta}_k^{(i)}$, a gaussian random perturbation is summed to the estimated vectors as well. This perturbation is taken into account in the performance index through the perturbed plant outputs (10). This test over the robustness quality of each estimator is used in the verification step, when the plant is working under the effect of a noise which will be simulated as a gaussian random perturbation independent from the one used to define the perturbed plant outputs. In order to make an statistical meaningful interpretation of the proposed schemes, each scheme has been simulated 75 times affected by the random noise, and then the results have been averaged. Furthermore, the following performance index is proposed in order to compare the schemes:

$$J_m(k) = \sum_{\ell=1}^{k} \left(y_\ell - y_{m\ell}\right)^2, \tag{12}$$

where y_k is the real plant output and y_{mk} is the desired closed-loop output. The above index indicates the mean separation of the real plant output from the desired one for each scheme. Then, the value of this index (12) is averaged 75 times. There are 30 estimators running in parallel, while the single model based adaptive control is initialised by the first one. The estimators are initialised by

$$\begin{aligned}\hat{\theta}_0^{(i)T} &= \left[\, -0.85 \;\; 0.2 \;\; -0.1 \;\; 0.7 \;\; -0.35 \;\; 0.075 \,\right] \\ &\quad +(i-1) * \left[\, -0.0717 \;\; 0.0433 \;\; -0.0083 \;\; 0.0433 \;\; -0.017 \;\; 0.0007 \right],\end{aligned}$$

for $i = 1, 2, \ldots, 30$. The amplitude of the test perturbation is $A = 5\%$. $N_{e0} = 30, \Delta N_e = 2, N_{eThreshold} = 4, N_{test} = 6$ samples. For the standard multiestimation scheme $\beta_1 = 0.85$; $\beta_2 = 0.15$; $\beta_3 = 0$ while for the robust schemes $\beta_1 = 0.85$; $\beta_2 = 0.05$; $\beta_3 = 0.1$. $\lambda = 0.95$ in all cases. The results are obtained in Fig. 12.

It can be concluded that multiestimation based techniques may improve the transient response of the adaptive system by the convenient parameterisation of the adaptive controller through time. Thus, the performance index (12) is smaller for the multiestimation schemes than for the single model based one. *Furthermore, the robustness term of the performance index* (11) *allows to improve the behaviour of the standard multiestimation scheme in the presence of noise as Fig.* 12 *reveals for the performance index* (12) *for the robust multiestimation scheme with a constant number of estimation algorithms.* Also, if a prune over the multiestimation scheme is done, then the scheme improves its behaviour in comparison to the standard multiestimation one but it gets worse in comparison to the original genetic algorithm one which incorporates robustness issues. However, in this case we have pruned some computations and reduce some memory storage requirements

with the drawback that a worse behaviour than the original genetic algorithm scheme which maintains the number of estimators constant through time has been achieved.

5. Fuzzy Logic Approach

In this Section, a fuzzy logic approach is given for multiestimation based control schemes. As it is known, fuzzy set theory is a generalization of the classical set theory (Tilli, 1992). It allows a class of objects with a continuum grade of membership to a set. Such a set is characterised by a membership (characteristic) function which assigns to each object its grade of membership to the set ranging from one to zero. The classical set theory operations are extended to the fuzzy case as well. Inference relations over fuzzy set objects define the so called *fuzzy logic*. Fuzzy logic, together with artificial neural networks and genetic algorithms, completes the set of techniques used in *intelligent control* (Liutkevicius, 2003; Wang and Qing, 2004; Mwembeshi *et al.*, 2004; Wetter and Wright, 2004; Feng, 2004; Etxebarria and de la Sen, 1996; Melin and Castillo, 2003). In the multiestimation scheme presented in Section 2, an estimated parameter vector $\hat{\theta}_k^{(c_k)}$ is chosen from a set of parameter estimated vectors $\{\hat{\theta}_k^{(1)}, \hat{\theta}_k^{(2)}, \ldots, \hat{\theta}_k^{(N_e)}\}$ to parameterise the adaptive controller at each sampling time. However, instead of choosing a single estimated vector, it is also possible to define a combined estimated vector as

$$\hat{\theta}_k = \alpha_{1,k}\hat{\theta}_k^{(1)} + \alpha_{2,k}\hat{\theta}_k^{(2)} + \cdots + \alpha_{N_e,k}\hat{\theta}_k^{(N_e)}, \tag{13}$$

where $0 \leqslant \alpha_{i,k} \leqslant 1$, $1 \leqslant i \leqslant N_e$ and $\forall k \geqslant 0$. This linear combination (13), is convex in the sense that$\sum_{i=1}^{N_e} \alpha_{i,k} = 1, \forall k \geqslant 0$. In the standard cases (considered above and in (Ibeas *et al.*, 2003)), only one coefficient $\alpha_{i,k}$ is different to zero and equal to unity while the rest of the parameters take a value of zero. However, it is also possible to let each coefficient $\alpha_{i,k}$ take a value running from zero to unity. Then, we can interpret each

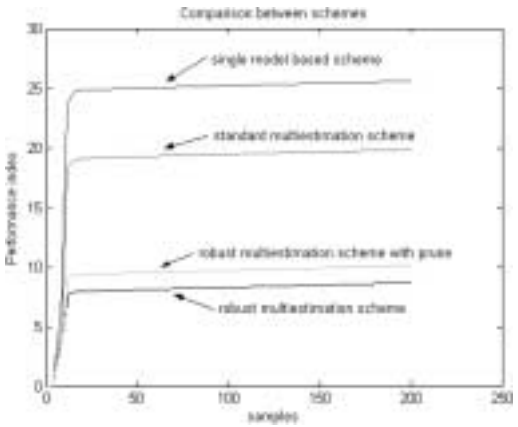

Fig. 12. Comparison between the single model based adaptive control and the multiestimation schemes.

one as a membership function of the combined estimated vector $\hat{\theta}_k$ to the corresponding estimation algorithm with vector $\hat{\theta}_k^{(i)}$. According to a recent work (Alonso-Quesada *et al.*, 2004), the following membership function is proposed in order to clarify the interpretation:

$$\alpha_{i,k} = \frac{J_k^{(i)^{-1}}}{\sum_{\ell=1}^{N_e} J_k^{(\ell)^{-1}}}, \tag{14}$$

where the $J_k^{(\ell)}$ symbolizes the performance indexes defined above (5) and used for evaluating the quality of each estimation scheme. A larger performance index for an estimation algorithm leads to a less membership function of the combined estimated vector to the corresponding estimation algorithm. However, the updating of the membership functions must respect a minimum residence time in order to guarantee the closed-loop stability. Thus, the following updating rule is proposed:

$$\alpha_{i,k} = \begin{cases} \frac{J_k^{(i)^{-1}}}{\sum_{i=1}^{N_e} J_k^{(i)^{-1}}} & \text{if } \; k = \mu N_D, \;\; \mu \in \mathbb{N}, \\ \alpha_{i,k-1} & \text{otherwise,} \end{cases} \tag{15}$$

where $N_D > 0$ is the residence number of samples with $\alpha_{i,0}$ being an arbitrary initialisation. This kind of combined multiestimation control schemes have been recently applied to the adaptive control of robotic manipulators as well (Ibeas and de la Sen, 2004). The Fig. 13 illustrates the diagram of the proposed multiestimation scheme.

The fuzzy logic approach allows that the membership functions may be determined by linguistic rules as

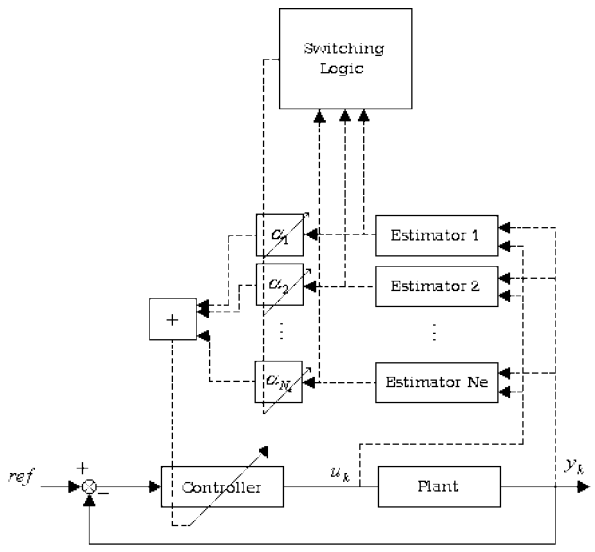

Fig. 13. Fuzzy Logic inspired multiestimation scheme.

If $f(condition1, condition2, \ldots, conditionN)$ *is true*
Then modify membership functions as $(\ldots rules \ldots)$,

where $f(\cdot)$ is a logical function of its arguments. As an example, it may be possible to avoid control singularities associated with pole-zero cancellations in pole placement based control algorithms. Thus, given a set of estimated parameter vectors, add another vector (or vectors) to the set. This vector (or vectors, which may be fixed or updated at each sample) represents coprime pole-zero polynomials. If the system is near a control singularity (condition that can be detected with a prescribed threshold by using the determinant of the *Sylvester* matrix for example), then modify membership functions in such a way that singularities in the control law are avoided. Membership functions are modified in order to make more representative the coprime vectors in such a way that the combined estimated vector remains coprime. Thus, linguistic rules for specifying the system behavior can be included in the system operation increasing the way in which multimodel based controllers can be designed. Each estimated parameter vector is updated according to its corresponding estimation scheme.

5.1. *Simulation Example*

The following simulations comparing the single model based and the fuzzy logic based multiestimation scheme show the usefulness of the proposed scheme. The plant, the input signal and the performance index used in (15) are the same as in the ANN example. The estimation algorithm is of least squares type for all the estimation algorithms. The least squares algorithm is described by the following updating equations:

$$\hat{\theta}_k^{(i)} = \hat{\theta}_{k-1}^{(i)} + \frac{P_{k-1}^{(i)}\varphi_k\left(y_k - \varphi_k^T\hat{\theta}_{k-1}^{(i)}\right)}{1+\varphi_k^T P_{k-1}^{(i)}\varphi_k},$$
$$P_k^{(i)} = P_{k-1}^{(i)} - \frac{P_{k-1}^{(i)}\varphi_k\varphi_k^T P_{k-1}^{(i)}}{1+\varphi_k^T P_{k-1}^{(i)}\varphi_k},$$

where φ_k is the above defined regressor, $\hat{\theta}_0^{(i)}$ arbitrary and bounded and $P_0^{(i)} = P_0^{(i)T} > 0$ for $i = 1, 2, \ldots, N_e$. The residence time is 5 samples. There are five estimators running in parallel and initialized by

$$\hat{\theta}_0^{(1)T} = \left[\begin{array}{cccccc} -0.5 & 0.2 & -0.5 & 0.79 & -0.35 & 0.082 \end{array}\right],$$
$$\hat{\theta}_0^{(2)T} = \left[\begin{array}{cccccc} -1 & 0.4 & -0.4 & 0.9 & -0.45 & 0.084 \end{array}\right],$$
$$\hat{\theta}_0^{(3)T} = \left[\begin{array}{cccccc} -1.5 & 0.6 & -0.3 & 1 & -0.55 & 0.086 \end{array}\right],$$
$$\hat{\theta}_0^{(4)T} = \left[\begin{array}{cccccc} -2 & 0.8 & -0.2 & 1.2 & -0.65 & 0.088 \end{array}\right],$$
$$\hat{\theta}_0^{(5)T} = \left[\begin{array}{cccccc} -2.5 & 1 & -0.15 & 1.5 & -0.75 & 0.088 \end{array}\right]$$

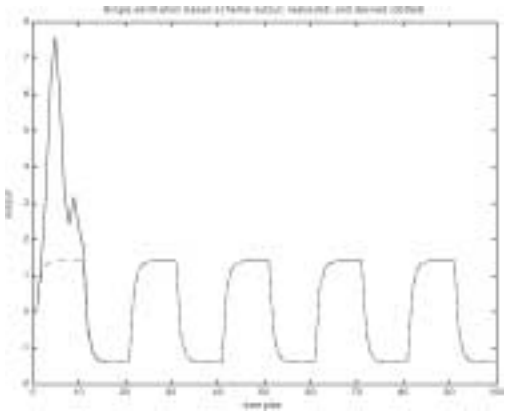

Fig. 14. Classical adaptive control scheme.

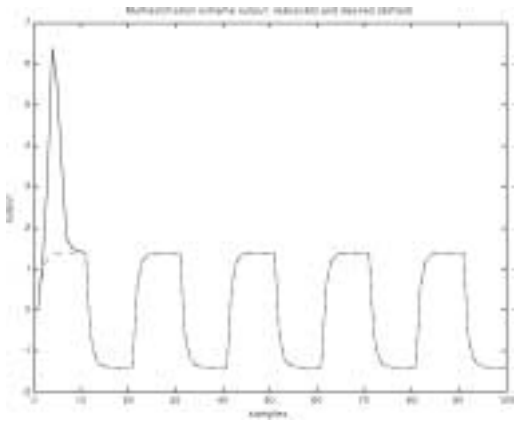

Fig. 15. Combined adaptive control scheme.

with $P_0^{(i)} = 10^{10} I_6$ for $i = 1, 2, \ldots, 5$. The initial values for the membership functions are:

$$\alpha_0 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \end{bmatrix},$$

and they are updated by Eq. 15 respecting the residence time constraint (with $\beta_2 = \beta_3 = 0$). The single adaptive control scheme is initialized by the first estimator. The simulations are obtained in Fig. 14.

It is shown in Figs. 14–16 that the fuzzy logic based combined multiestimation scheme improves the transient response of the adaptive system. The transient overpeak is reduced due to switching to a more convenient parameterisation of the adaptive controller through the convex linear combination of the estimators of the multiestimation scheme according to the updating rule (15). The transient response improvement is achieved once

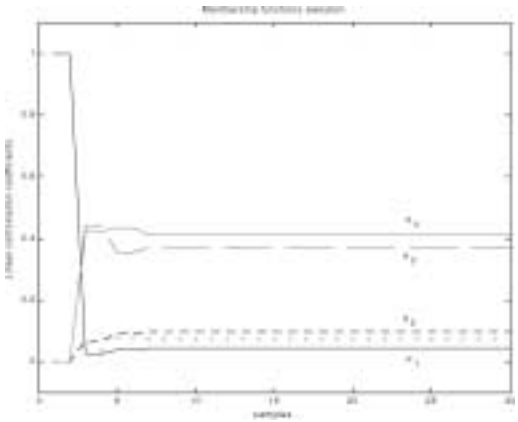

Fig. 16. Membership functions $\alpha_{i,k}$.

the residence number of samples constraint is fulfilled since the updating rule is not allowed to modify the weight values until then.

6. Conclusions

In this paper, an artificial intelligence representation of multiestimation based controllers has been developed. A neural network interpretation of multimodel based controllers has been given while a method for generating multimodel based artificial neural networks controllers from pre designed single model ones has been proposed. A genetic algorithm and fuzzy logic based approach have been given to represent multiestimation based schemes. These artificial intelligence techniques have suggested new ideas and directions to be incorporated into the classical multimodel controllers. The genetic algorithm approach has inspired a robustness test over the set of estimation algorithms which allows to improve the working of the multiestimation scheme in noisy environments. Also, the fuzzy logic approach suggests a way in which the estimation algorithms can be combined to build an estimated parameter vector to parameterise the adaptive controller. Some simulation examples show the usefulness of the proposed multiestimation structures for improving the transient response of adaptive systems.

Acknowledgements. The authors are grateful to MEC and to the UPV-EHU by its partial support of this work through projects DPI 2003-00164, UPV 9/UPV 00I06.I06-15263/2003. Also, the authors are grateful to reviewers who have helped to improve the final version of this manuscript and to Professor J.M. Tarela whose comments about genetic algorithms have helped to improve the paper. A. I. is very grateful to MECD by its support of this work through the FPU grant P2002-2727.

References

Alonso-Quesada, S., M. de la Sen, A. Bilbao-Guillerna and A. Ibeas (2004). A semiempirical identification method by using a multiestimation technique via reduced-order nominal models. In *Proceedings of the ACC'04* (in press).

Beyers, H.G. (1998). *The Theory of Evolution Strategies*. Springer-Verlag, 1998.

Chang, M.H., and E.J. Davison (1999). Adaptive switching control of LTI MIMO systems using a family of controllers approach. *Automatica*, **35**, 453–465.

Da Ruan (Ed.) (1997). *Intelligent Hybrid Systems, Fuzzy Logic, Neural Networks and Genetic Algorithms*. Kluwer.

Etxebarria, V. (1994). Adaptive control of discrete systems using neural networks. *IEE Proc. Contr. Theory and Appl*, Part D, **141**(4), 209–215.

Etxebarria, V., M. de la Sen (1996). An approach to adaptive neural control of robot manipulators. *International Journal of System Science*, **27**(11), 1143–1152.

Fausett, L.V. (1998). *Fundamentals of Neural Networks*. Prentice Hall.

Feng, G. (2004). Stability analysis of discrete-time fuzzy dynamic systems based on piecewise lyapunov functions. *IEEE Transactions on Fuzzy Systems*, **12**(1), 22–28.

Gregorcic, G., A. Mullane and G. Lightbody (2001). Simulink implementation of adaptive control and multiple model network control. In *Proc. of the Nordic Matlab Conf.* pp. 167–173.

Hocherman-Frommer, J., S.R. Kulkarni and P.J. Ramadge (1998). Controller switching based on output prediction errors. *IEEE Transactions on Automatic Control*, **43**(5), 596–607.

Ibeas, A., M. de la Sen and S. Alonso-Quesada (2003). Adaptive stabilization of discrete linear systems via a multiestimation scheme. *Inter. Math. J.*, **3**(4), 381–407.

Ibeas, A., and M. de la Sen (2004). Intelligent control of robotic manipulators based on a multiestimation scheme. In *Proc.of the IASTED Int. Conf. Artificial Intelligence and Applications*. pp. 694–699.

Ibeas, A., M. de la Sen and S. Alonso-Quesada (2004). Intelligent control of discrete linear systems based on a supervised adaptive multiestimation scheme. *Journal of Intelligent and Robotic Systems* (in press).

Liutkevicius, R. (2003). Coupled adaptive fuzzy control of nonlinear, time-varying plant. *Informatica*, **14**(3), 323–336.

Melin, P., and O. Castillo (2003). Adaptive intelligent control of aircraft systems with a hybrid approach combining neural networks, fuzzy logic and fractal theory. *Applied Soft Computing*, **3**(4), 353–362.

Mosca, E., and T. Agnoloni (2001). Inference of candidate loop perfomance and data filtering for switching supervisory control. *Automatica*, **37**, 527–534.

Mwembeshi, M.M., C.A. Kent and S. Salhi (2004). A genetic algorithm based approach to intelligent modelling and control of pH in reactors. *Computers & Chemical Engineering*, **28**(9), 1743–1757.

Narendra, K.S., and J. Balakrishnan (1994). Improving transient response of adaptive control systems using multiple models and switching. *IEEE Transactions on Automatic Control*, **39**(9), 1861–1866.

Narendra, K.S., and J. Balakrishnan (1997). Adaptive control using multiple models. *IEEE Transactions on Automatic Control*, **42**(2), 171–187.

de la Sen, M., and A. Almansa (2002). Adaptive stable control of manipulators with improved adaptation transients by using on-line supervision of the free parameters of the adaptation algorithm and sampling rate. *Informatica*, **13**(3), 345–368.

de la Sen, M., S. Alonso-Quesada, A. Bilbao-Guillerna and A.J. Garrido (2003). A supervised multiestimation scheme for discrete adaptive control which guarantees robust closed-loop stability. *Inter. Math. J.*, **3**(9), 907–937.

de la Sen, M., J.J. Miñambres, A.J. Garrido, A. Almansa and J.C. Soto (2004). Basic theoretical results for expert systems. Application to the supervision of adaptation transients in planar robots. *Artificial Intelligence*, **152**(2), 173–211.

Tilli, T. (1992). *Fuzzy Logic*. Franzis-Verlag.

VanDoren, V. (Ed.) (2003). *Techniques for Adaptive Control*. Elsevier Science.

Wang, J., and M. Qing (2004). An output tracking control strategy for unknown nonlinear systems. *Applied Artificial Intelligence*, **18**(1), 1–16.

Wetter, M., and J. Wright (2004). A comparison of deterministic and probabilistic optimization algorithms for nonsmooth simulation-based optimization. *Building and Environment*, **39**(8), 989–999.

A. Ibeas received his MSc degree in applied physics (specialty electronics and automatic control) in 2000 from the Basque Country University where he is currently a PhD candidate in automatic control. His present main research interests areas are adaptive control, robotics and applications of artificial intelligence to control systems.

M. de la Sen was born in Arrigorriaga, Bizkaia in the Spanish Basque Country. He obtained the MSc degree with honors from the Basque Country University in 1975, the PhD degree in applied physics with high honors from the same University in 1979 and the degree of docteur-d'Etat-ès-Sciences Physiques (specialitè Automatique et Traitement du Signal) from the Universitè de Grenoble, France with "mention très honorable" in 1987. He has had several teaching and research positions in the University of the Basque Country in Bilbao (Spain) where he is currently Professor of systems and control engineering in the Department of Systems and Control Engineering and a head of the Institute of Research and Development of Processes IIDP. He has also had positions of visiting professor in the University of Grenoble (France), the University of Newcastle (New South Wales, Australia) and the Australian National University ANU (Canberra, Australia). He has been a member of the editorial board of the *Electrosoft Journal* (CML mechanical and computational engineering publications). He has been author or coauthor of a number of papers in the fields of adaptive systems, mathematical systems theory and ordinary differential equations, which are his research interest subjects. He acts and has acted as reviewer for several international journals and conferences of control theory and engineering and applied mathematics.

Reguliatorių, grindžiamų daugiamodelių taikymu, pateiktis dirbtinio intelekto priemonėmis

Asier IBEAS, Manuel de la SEN

Straipsnyje vystoma daugiamodelių taikymu grindžiamų reguliatorių pateiktis dirbtinio intelekto tipinėmis struktūromis, kaip antai, neuroniniais tinklais, genetiniais algoritmais bei neraiškia logika. Atlikta šių struktūrų pagrindu sudarytų reguliatorių palyginamoji analizė su atitinkamais reguliatoriais, gautais taikant klasikinius metodus, naudojančius vienintelį tiriamos sistemos modelį. Pasiūlytas robastiškumo testas, leidžiantis geriau parinkti sistemos modelį, lyginant su klasikinėmis parametrų įvertinimo schemomis, grindžiamomis daugiamodeliais. Pateikti modeliavimo ir parametrų įvertinimo rezultatai, taikant klasikinius ir dirbtinio intelekto struktūrų algoritmus.