ARTIFICIAL INTELLIGENCE REPRESENTATIONS OF MULTI-MODEL BASED CONTROLLERS

Asier Ibeas, Manuel de la Sen
Instituto de Investigación y Desarrollo de Procesos, Facultad de Ciencia y Tecnología
Universidad del País Vasco, Apdo. 644, 48080 Bilbao, Spain

Keywords: Multimodel control, artificial intelligence, neural networks, genetic algorithms, fuzzy logic, switching.

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 multimodel based controllers. A method for synthesizing multimodel based neural network controllers from already designed single model based ones is presented. Some applications of the genetic algorithms and fuzzy logic to multimodel controller design are also proposed.

1 INTRODUCTION

Multi-model based controllers have been broadly studied during the last years (Narendra et al, 1997, Gregorcic et al, 2001, Ibeas et al, 2003). This kind of control architecture allows to design intelligent control systems able to modify their behavior according to the characteristics of a changing environment or operation point. This intelligent behavior, allows the stability and improvement of the closed-loop output for complex systems. Thus, a general multimodel based control scheme is formed by a set of different plant models running in parallel. These models, which may be fixed (Narendra et al, 1994) or adaptive (Ibeas et al, 2003), are different one from each other in what it is concerned with its structure or its parameter values. Thus, each one contains different characteristics of the controlled process. Moreover, a higher level switching structure between the various models chooses at each time the model which will be used to calculate the control law at that time instant. The switching structure chooses the control model according to a performance index for the closed-loop system. Thus, the switching law acts as a supervisor of the system behavior. The structure and operation of the switching law has been studied from an artificial intelligence point of view in an expert systems context (De la Sen et al, 2002). However, multimodel structures itself, have always been modeled in a classical control theory frame (Narendra et al, 1997). This paper proposes a possible interpretation of multimodel schemes in an artificial intelligence frame. The artificial intelligence structures chosen for such a goal have been, artificial neural networks (ANN), genetic algorithms (GA) and fuzzy logic. This interpretation will allows 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. Also, some applications of genetic algorithms and fuzzy logic to multimodel control design are presented. An adaptive, being more general than that related to the use of fixed models, formalism is used for making the interpretation.

2 BASIC MULTIESTIMATION SCHEME

In this Section, a brief description on the multiestimation scheme used for discussion is presented. It has been considered the adaptive case since the fixed case is included in this as a particular case. The aim is to design a multimodel control for the discrete (the continuos case can be treated in the
same way) time invariant linear SISO plant described by:

\[ A(q^{-1})y_k = B(q^{-1})u_k \tag{1} \]

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

\[ A(q^{-1}) = 1 + a_1q^{-1} + a_2q^{-2} + \cdots + a_nq^{-n} \tag{2.1} \]
\[ B(q^{-1}) = b_0 + b_1q^{-1} + \cdots + b_nq^{-n} \tag{2.2} \]

with \( n \geq m \). The above Equations (1-2) represent a linear difference equation which is usually written in adaptive control as the inner product of two vectors

\[ y_k = \phi^T_k \theta = \phi^T_k \theta \tag{3} \]

\[ \phi^T_k = [-y_{k-1}, -y_{k-2}, \ldots, -y_{k-m}, u_k, u_{k-1}, \ldots, u_{k-n}] \]

being the so called regressor and

\[ \theta = [a_1, a_2, \ldots, a_n, b_0, b_1, \ldots, b_n] \]

symbolising the true plant parameter vector (Ibeas et al, 2003). 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 \( k \). This estimated vector is used for control calculations at the inner product. If this estimated vector is far away 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 to consider a set of estimation algorithms running in parallel, each one with its own estimated parameter vector \( \hat{\theta}_k^{(1)}, \hat{\theta}_k^{(2)}, \ldots, \hat{\theta}_k^{(N_e)} \), where \( N_e \) is the number of total estimators. Each estimated vector is updated at each sample according to input and output measurements of the plant. The multiestimation scheme block diagram is displayed in Figure 1. A switching logic between the various estimation algorithms chooses the estimated vector that achieves the best system behavior improvement according to a prescribed performance index \( J_i^*(\theta_k^*, \hat{\theta}_k^*, \ldots, \hat{\theta}_k^{(N_e)}) \). The switching law must respect a minimum dwell or residence time between consecutive switchings in order to guarantee closed-loop stability.

A complete discussion of the stability issues is available in (Ibeas et al, 2003). In the next sections an artificial intelligence representation of the above multiestimation scheme is given for various typical artificial intelligence structures (Da Ruan, 1997).

### 3 ARTIFICIAL NEURAL NETWORKS

In this section, an artificial neural network (ANN) representation is developed for the above multiestimation based control scheme (Fausett, 1998). In (Etchebarria, 1994), a two layered ANN is presented for a discrete time single adaptive control. The difference Equation (1)-(2) is implemented for estimation purposes by the ANN displayed in Figure 2, where the activation functions are linear for all neurons. The ANN output can be written as:

\[ y_k = \sum_{i=1}^{m} w_{i,k} y_{k-i} + \sum_{j=1}^{n} w_{n+j,k} u_{k-j} = \varphi^T_k \hat{\theta}_k \tag{4} \]

where:

\[ \varphi^T_k = [w_{1,k}, w_{2,k}, \ldots, w_{m+k}] \]

and \( \varphi \) is the so called regressor. Comparing the above Equation (4) with Equation (3), it can be observed that network weights \( w_{i,k} \) represents the estimated plant parameters. Networks weights (or plant parameters) are updated by using the well known Widrow-Hoff rule for single-output multiple-input ANN:

\[ w_{k} = w_{k-1} + \alpha (y_k - \hat{y}_k) \varphi_{k-1} \tag{5} \]

where \( \hat{y} \) denotes ANN output while \( y \) denotes real measured plant output and \( \alpha > 0 \), \( \alpha \in (0,2) \).

(Etchebarria, 1994). Thus, network weights are updating by comparing the network output with the real plant output (which it is the target value).
plant parameters vector) is used for controller design purposes.

\[ y_{k+1} = y_{k} + w_{1,k} u_k \]

Figure 2: Single neural network estimator

\[ y_{k+1} = y_{k} + w_{1,k} u_k \]

Figure 3: Multiestimator neural network

Now, the multiestimation scheme presented in Section 2 can be represented by increasing the number of neurons in the output layer to a number of neurons equal to the number of different estimators used in the multiestimation scheme. Since the output layer has one neuron in this case, a multiestimation scheme with \( N_e \) estimators running in parallel will have \( N_e \) neurons in its output layer as the Figure 3 displays for the case of two estimators. Hence, the number of connections and weights between neurons is increased. Thus, the proposed ANN is a structure containing itself the \( N_e \) estimated parameter vectors (which are represented by the corresponding weights). 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. If the original target value was the real measured plant output, \( y_1 \), in the case with two estimators, the new target vector is defined by:

\[ y^T = [y_1, y_1] \]

while in the general case with \( N_e \) estimators, it is:

\[ y^T = \left[ y_1, y_2, \ldots, y_{N_e} \right] \]

The switching logic compares each output of the ANN with the real plant output and chooses the set of weights associate with the best estimated output in order to calculate the control law. The training rule is the generalization of the above Widrow-Hoff single output training rule (5) to the multiple output case:

\[ w_{d,k} = w_{d,k-1} + \alpha \left( y_k - \hat{y}_k^{(i)} \right) \psi_{d,k-1} \]

where \( \psi_{d,k-1} \) stands for the \( j \)-th component of the vector \( \varphi_{d,k-1} \). Note that the updating law for the estimated parameters vectors (network weights) is formulated for the multiple output ANN as a unique identity. In the following, a simulation example is presented containing two estimation algorithms and the above training rule. The switching logic is assumed to respect a minimum residence time between successive switchings in order to guarantee closed-loop stability (Ibeas et al, 2003). The discrete plant has the real plant parameter vector \( \theta^T = [-1.9, 0.73, -0.195, 1, 0.6, 0.087] \) and the reference model is:

\[ m^T = [0.6, 0.11, 0.006, 1, 0.32, 0.025] \]

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

\[ \hat{\theta}_1^{TT} = [-0.5, 0.25, -0.5, 0.79, -0.5, 0.08] \]

\[ \hat{\theta}_2^{TT} = [-1.5, 0.7, -0.2, 0.9, -0.5, 0.08] \]

It is taken \( \varepsilon = 0.001 \) and \( \alpha = 1 \). The input signal is a unity square wave with a 20 samples period. The residence time is 2 samples and the performance index to decide switches is:

\[ J_e^{(i)}(k) = \sum_{t=k}^{k+N_e-1} \lambda^{t-k} \left( y_i - \hat{y}_i^{(i)} \right)^2 \]

with the forgetting factor \( \lambda = 0.95 \). The single adaptive control scheme is initialised with the first estimator. Simulations are showed in Figures (4-6).
It is showed that the system improves its behaviour by using the best weight set at each time (respecting the residence time constraint) Figures (4-5). The switching map $c_i$ illustrating the switching process between both set of weights (parameters) is showed in Figure 6. The above idea can be extended to the most 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. 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. Now, define a new ANN 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 $N_o \times e_N$, where $e_N$ is the number of estimators considered. The target vector in this case is 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. 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 Figure 7.

4 GENETIC ALGORITHMS

In this Section, a genetic algorithm representation is given for multiestimation based control schemes. Genetic algorithms are usually used as optimisation tools in complex problems (Beyers, 1998). 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. 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, 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 Figure 8. In the first step, there exists an initial set of vectors uniformly distributed over the possible parameter space.
performance index which evaluates the control law. The selection is made according to the above vectors is chosen in order to generate the parameter vector is assumed to belong to. Once the subset of the parameter space where the real plant problem, the existence of a convex and compact modification rules are called, selection, crossover and mutation. The adaptive counterpart is the updating rule for the estimation algorithms. Inference relations over fuzzy classical set theory operations are extended to its grade of membership ranging from one to zero. Such a set is characterised by a membership characteristic function which assigns to each object with a continuum grade of membership. classical set theory (Tilli, 1992). It allows a class of known, fuzzy set theory is a generalization of the multiestimation scheme presented in Section 2, an estimated parameter vector is chosen from a set of parameter estimated vectors 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: where $0 \leq \alpha_i \leq 1$, $1 \leq i \leq N$, and $\forall k \geq 0$. This linear combination (8), is convex in the sense that $\sum_{i=1}^{N} \alpha_i = 1$, $\forall k \geq 0$. In the standard cases (considered above and in (Ibeas et al, 2003)), only one coefficient $\alpha_i$ is different to zero and equal to unity. However, it is also possible to let each coefficient $\alpha_i$ take a value between one to zero. Then, we can interpret each one as a membership coefficient is different to zero.

5 FUZZY LOGIC APPROACH

In this Section, a fuzzy logic approach is given for multieistimation 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. Such a set is characterised by a membership (characteristic) function which assigns to each object its grade of membership 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. In the multieistimation scheme presented in Section 2, an estimated parameter vector is chosen from a set of parameter estimated vectors 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: 

$$\hat{\theta} = \sum_{i=1}^{N} \alpha_i \hat{\theta}_i \geq 0$$

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 models (chromosomes) in order to prune unnecessary computations. Thus, the multiple models are classified into priority sets in such a way that models with a similar performance belong to the same set (according to some performance criteria, for example, all models with performance index inside a prescribed range belong to the same set). Thus, the sets associated to models with the worst performance may be pawn 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 spaced different models, the system is able to obtain an improved number of models achieving an acceptable system performance.

Figure 8: General structure of a genetic algorithm

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. 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. Then, the parameter estimated vector are modified by applying some modification rules. The modification rules may depend on the value of the performance index associate with each vector. In GA terms, these modification rules are called, selection, crossover and mutation. Furthermore, the number of different models (the number of chromosomes) may not be constant during the system operation.
function of the combined estimated vector $\hat{\theta}_i$ to the corresponding estimation algorithm with vector $\hat{\theta}^{(i)}$. The following membership function is proposed in order to clarify the interpretation:

$$\alpha_{ik} = \frac{\sum_{i=1}^{n} J_i^{(i)}(\theta)}{n}$$

where the $J_i^{(i)}$ symbolizes the performance indexes for evaluating the quality of each estimation scheme. A bigger performance index for an estimation algorithm leads to a less membership function for the combined estimated vector to the corresponding estimation algorithm associate estimated vector. The fuzzy logic approach allows that the membership functions may be determined by linguistic rules as

If $f(\text{condition}_1, \text{condition}_2, \ldots, \text{condition}_N)$ is true
Then modify membership functions as (rules)

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 control algorithms. 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. The updating of the membership functions must respect a minimum residence time in order to guarantee closed-loop stability. The following simulations show the usefulness of the proposed scheme. The plant, the input signal and the performance index used in (9) are the same as in the ANN example (7). The estimation algorithm is of least squares type. The residence time is 5 samples. There are five estimators initialized by:

$\hat{\theta}_0^{(i)} = [-0.5, 0.2, -0.5, 0.79, -0.35, 0.082]$

$\hat{\theta}_1^{(i)} = [-1, 0.4, -0.4, 0.9, -0.45, 0.084]$

$\hat{\theta}_2^{(i)} = [-1.5, 0.6, -0.3, 1, -0.55, 0.086]$

$\hat{\theta}_3^{(i)} = [-2, 0.8, -0.2, 1.2, -0.65, 0.088]$

$\hat{\theta}_4^{(i)} = [-2.5, 1, -0.15, 1.5, -0.75, 0.088]$

The initial values for the membership functions are:

$\alpha_0 = [1/5, 1/5, 1/5, 1/5, 1/5]$ and they are updated by Equation (9) respecting the residence time constraint. The single adaptive control scheme is initialized by the first estimator. Figure (9) show a simulation example of the proposed scheme.

### 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 based approach has been given to multiestimation based schemes. These artificial intelligence techniques suggest new ideas and directions to be incorporated to the classical multimodel controllers.

### ACKNOWLEDGEMENTS:

The authors are grateful to MEC and to the UPV-EHU by its partial support of this work through projects DPI2000-0244, DPI 2003-00164, UPV 1/UPV/EHU 00106.106.310 EB 8235/2000 and 9/UPV 00106.106-15263/2003.
REFERENCES


