Multimodel Control Design Using Unsupervised Classifiers

Nesrine Elfelly[1,2], Jean-Yves Dieulot[3], Mohamed Benrejeb[2], Pierre Borne[1]

[1] EC Lille, LAGIS, Cité Scientifique,
59650Villeneuve d'Ascq, France,
nesrine.elfelly@ec-lille.fr

[2] ENI Tunis,UR LARA Automatique,
BP371002 Tunis LeBelvédère, Tunisia.

[3] Polytech Lille, LAGIS, Cité Scientifique,
59650Villeneuve d'Ascq, France,
dieulot@univ-lille1.fr

Abstract: Multimodel approaches derive a smooth control law from the blending of local controllers using the concept of validities and domain overlapping. In this paper, it is demonstrated that unsupervised classification algorithms can be of a great help to design such parameters as the number of the models and their respective clusters, which will be performed using a respectively Rival Penalized Competitive Learning (RPCL) and simple or fuzzy K-means algorithms. The classical multimodel approach follows by deriving parametric model identification using the classification results for models orders and then parameters estimation. The determination of the global system control parameters results from a fusion of models control parameters. The case of a second order nonlinear system is studied to illustrate the efficiency of the proposed approach, and it is shown that this approach is much simpler that other multimodel control design methods which generally require a huge number of neighboring models.

Keywords: complex systems, multimodel, identification, control, classification

1. Introduction

The multimodel approach has arisen from the needs of process industries for which operations often include set-point changes and/or the co-existence of multiple operating modes. While nonlinear control is complicated to derive and tune, it is often more appropriate to consider a set of well-known operating points and their respective subsystems to achieve modelling or control by an accurate blending of the local systems/controllers. However, the multimodel approach owns, from its distributed structure, a high number of degrees of freedom, including the number and parameters of the different models representative of the system, the choice of the blending method and the design of a suitable control merging algorithm.

Over the last few years, many authors [e.g. 1,5] have proposed methods for identification and model structure validation, and a huge literature addresses linear models blending such as fuzzy Takagi-Sugeno models [e.g. 11]. However, the multimodel representation is more difficult to obtain when the subsystems are nonlinear and/or should be determined from raw input-output data. Some results were given in [7, 8]. In the main, classification of models with unsupervised algorithms was used to find an appropriate size for the model-base and estimate the models parameters, and the blending functions between several models were selected in the common case where the operating domains overlap.

Classification or neural techniques have been used for output to state modelling or control [1, 15, 19] and also to build multimodel representation and control from raw data; however, whereas Cho et al. [5] used Kohonen Self Organizing Map and K-means techniques [21], few other works have attempted to bridge the classification and multimodel control domains.

This paper thus proposes a practical approach for complex systems control based on classification algorithms while extending previous works in which the modelling issue was addressed [6, 8] and the preliminary results in [7]. Multimodel control is based on the multimodel representation and has been applied very successfully, for example, to chemical and biological plants (see e.g. [4, 18]). The procedure consists in designing a controller for each model of the base and to obtain a global control by some blending law. Local controls have been chosen as neural networks (e.g. [1]), PID (eg. [4]), predictive (e.g. [18]), or adaptive controllers [9], whereas the blending law can be selected either as a commutation between the partial controllers (e.g. [9]) or a fusion by using validity indexes [14]. However, again, these multimodel-based controllers suffer from

the initial choice of model number and structure, when the operating points are not chosen a priori but should emerge from insight in input-output data.

In a first place, it will be recalled how to build a set of models from input-output data using the fuzzy K-means algorithm and to determine the transition functions. An appropriate clustering method called Rival Penalized Competitive Learning (RPCL) [20] which is an extension of Kohonen competitive algorithm will enhance the determination of the number of models/clusters to be considered. Second, an adaptive – multimodel – controller will be obtained through a fusion of the parameters of the different controllers already designed for the models of the base by means of the appropriate validity indexes based on the residual approach [6, 8].

A nonlinear system which has been already presented in [5] allows confirming the relevance and the simplicity of the suggested approach, as results show that the number of local models can be reduced drastically.

2. Designing Multimodel Base using Classification Algorithms

The multimodel representation assumes that it is possible to replace a unique nonlinear representation by a combination of simpler models describing the system dynamics at specific operating points, the interaction between the models being captured via activation functions. The decision unit (Figure 1) selects the relevant contribution of each model of the base while the output unit computes the multimodel output from the local models outputs and weights.

When no a priori knowledge is assumed and only I/O data are available, a logical procedure and the help of unsupervised classification algorithms are necessary to extract information from data to build the model base. In a first time, the relevant number of the models is to be determined, and, then, local clusters and domains should be found. Finally, local – possibly linear or not – model structure and parameters should be estimated, which is related to model identification.

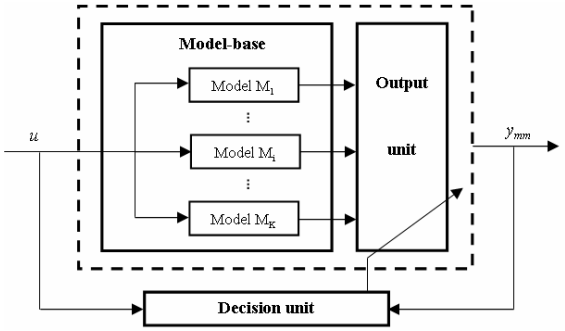

Figure 1. Multimodel approach

Clusters selection and estimation

Most existing clustering algorithms do not handle the selection of a relevant number of clusters, whereas it is a key feature of the multimodel approach, resulting in a tradeoff between algorithmic complexity and modelling accuracy. However, the RPCL algorithm [20], which is a variant of the Kohonen rule, has shown to be able to discard irrelevant extra units by driving away their clusters from the data cloud. The corresponding neural network rewards the winning neuron but also penalizes the second winning unit (called the rival), allowing not only better cluster separation but also moving away some of the units. The learning rate is much greater than the penalization rate [e.g. 16]. Given a competitive learning neural network, i.e. a layer of units with the output u_i of each unit and its weight vector w_i for $i=1…K$; K is the number of clusters.

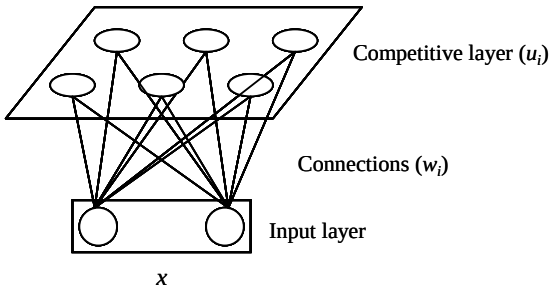

Figure 2. Competitive learning neural network

The RPCL algorithm can be described by the following steps.

1. Initialize weight vectors w_i randomly.
2. Take a sample x from a data set D, and for $i = 1…K$, let

$$u_i = \begin{cases} 1 & if\ i = c \\ -1 & if\ i = r \\ 0 & otherwise \end{cases}; \quad (1)$$

with:

$$\gamma_c \left\| x - w_c \right\|^2 = \min_j \gamma_j \left\| x - w_j \right\|^2$$
$$\gamma_r \left\| x - w_r \right\|^2 = \min_{j \neq c} \gamma_j \left\| x - w_j \right\|^2 \quad (2)$$

||*||: Euclidean distance;

c : index of the unit which wins the competition (winner);

w_c : weight vector of the winner;

r : second winner (rival) index;

w_r : weight vector of the rival;

γ_j : conscience factor (relative winning frequency) used to reduce the winning rate of the frequent winners.

$$\gamma_j = \frac{n_j}{\sum_{i=1}^{K} n_i} \quad (3)$$

where n_j refers to the cumulative number of occurrences the node j has won the competition ($u_j = 1$).

3. Update the weight vectors as follows:

$$w_j(t+1) = w_j(t) + \Delta w_j; \quad (4)$$

with:

$$\Delta w_j = \begin{cases} \alpha_c(t)(x - w_j(t)) & \text{if } u_j = 1 \\ -\alpha_r(t)(x - w_j(t)) & \text{if } u_j = -1 \\ 0 & \text{otherwise} \end{cases}; \quad (5)$$

$0 \le \alpha_c(t)$ and $0 \le \alpha_r(t) \ll \alpha_c(t)$ are respectively the winner learning rate and the rival de-learning rate. Several empirical functions have been proposed for the update of the learning and de-learning rates [16].

4. Repeat steps 2 and 3 until the whole learning process has converged.

The RPCL only provides the adequate number of clusters, but does not estimate the operating domains; a fuzzy-based method such as the fuzzy K-means classification algorithm allows, with an easy workout, to estimate both clusters and overlapping operating domains [3]. Every data can belong fuzzily – following a membership degrees between 0 and 1 – to several clusters, while the optimal K-partition is obtained by minimizing the fuzzy objective function:

$$J_m = \sum_{i=1}^{N} \sum_{j=1}^{K} u_{ij}^m \left\| x_i - c_j \right\|^2, \quad 1 \le m < \infty \quad (6)$$

with:

||*||: any norm expressing the similarity between any measured data and a cluster centre;

m : weighting exponent (real number greater than 1) which is a constant that influences the membership values;

u_{ij} : degree of membership of x_i to the cluster j, such as $u_{ij} \in [0,1]$, $\sum_{j=1}^{K} u_{ij} = 1 \quad \forall i$ and $0 < \sum_{i=1}^{N} u_{ij} < N \quad \forall j$;

x_i : i^{th} data point;

c_j : centre vector (node) of the cluster j;

N : number of observations;

K : number of clusters ($2 \le K < N$).

Fuzzy partitioning is carried out through an iterative optimization of the objective function shown above, with the update of membership u_{ij} and the cluster centres c_j [17].

The algorithm is composed of the following steps:

5. Initialize the matrix U(k)=[uij(k)], U(0).
6. At k-step: calculate the centres vectors C(k)=[cj]:

$$c_j = \frac{\sum_{i=1}^{N} u_{ij}^m \, x_i}{\sum_{i=1}^{N} u_{ij}^m} \quad (7)$$

7. Update U(k), U(k+1):

$$u_{ij} = \left[\sum_{r=1}^{K} \left(\frac{\left\| x_i - c_j \right\|}{\left\| x_i - c_r \right\|} \right)^{\frac{2}{m-1}} \right]^{-1} \quad (8)$$

8. If || U(k+1) - U(k)||< ε then STOP; otherwise return to step 2.

For representation purposes, a point will belong to the cluster for which its membership degree is the highest.

Local model estimation and multimodel computation

The repartition of the data set is the key part of the designing procedure, as further identification procedures can be applied on each local model, which use classical estimation methods. For further details, the

reader is referred to [2]. When the sub-models are supposed to be linear, their parameters and order can be estimated along with their data kernel, i.e. data for which the degree of membership is equal to 1. Model order *m* can be determined using the Instrumental Determinants Ratio (IDR)

$$IDR(m) = \left| \frac{\det(Q_m)}{\det(Q_{m+1})} \right|. \quad (9)$$

where Q_m is the well-known information matrix, δ is a threshold, the model order m is given by:

$$m = \min \arg \left(IDR(m) - IDR(m-1) > \delta \right). \quad (10)$$

When the model order is found, a classical estimation procedure using I/O data such as the Recursive Least-Squares method (RLS) is applied to achieve the parameters estimation.

Now that the model-base is complete, the multimodel output can be computed using the decision and output units (Figure 1).

$$y_{mm}(k) = \sum_{i=1}^{K} y_i(k)\, v_i(k); \quad \sum_{i=1}^{K} v_i(k) = 1. \quad (11)$$

The relevance of each model is estimated in real-time according to the validity index which is computed, in this study, via the residues' approach, i.e. the distance measurement between the outputs of the process and of the considered model, e.g.:

$$r_i = \left| y - y_i \right| \quad i = 1, \ldots, K; \quad (12)$$

with:

y: process output;

y_i: output of the model M_i.

Between the methods proposed for the calculation of validities (see [6, 13]), only the simple and the reinforced validities approaches are here considered. In general, the expression of the normalized validities is given by:

$$v_i^{simp} = \frac{1 - \frac{r_i}{\sum_{j=1}^{K} r_j}}{K - 1}. \quad (13)$$

A reinforcement step can be considered, for example by the normalized validities v_i^{renf} :

$$v_i'^{renf} = v_i \prod_{\substack{j=1 \\ j \neq i}}^{K} (1 - v_j), \; v_i^{renf} = \frac{v_i'^{renf}}{\sum_{j=1}^{K} v_j'^{renf}}. \quad (14)$$

Previous studies have shown that the designer should select discarding methods (K-means and reinforced validities) to allow a better separation of data when the operating points are quite independent, whereas operating domain overlap require more smooth methods (e.g. fuzzy classifiers). Once this step is complete, the validation of the global modelling scheme is carried out through a comparison between the real and the multimodel outputs, obtained through a fusion of the *K* models' outputs y_i weighted by their respective validity indexes v_i, for different input sequences.

3. Design of Multimodel Control

Multimodel control is a replication of the distributed modelling method presented above, where the global adjustable controller of the system (Figure 3) depends on a decision mechanism which computes the real-time estimation of the contribution of each local model and corresponding adequate local controller. After local controller synthesis, the fusion can be designed using the controllers' output blending, the global system control *u* is obtained here through a simple fusion of model controls u_i weighted by their relative validity coefficients v_i,

$$\forall\, i = 1, 2, \ldots, K, \; u(k) = \sum_{i=1}^{K} v_i(k)\, u_i(k) \quad (15)$$

Whenever controllers exhibit the same structure, parameter fusion can be proposed according to the weights - validities - defined in (13-14). In the latter case, the multimodel controller can be seen as an adaptive-like controller, where parameters are updated according to the relevance of the local models v_i, for which the corresponding parameter is p_i (Figure 4).

$$p(k) = \sum_{i=1}^{K} v_i(k)\, p_i(k) \quad (16)$$

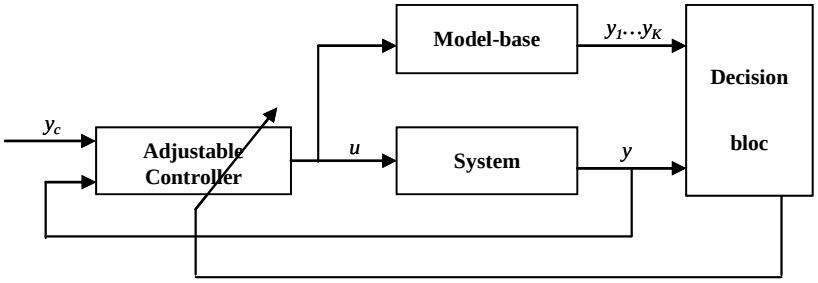

Figure 3. Multimodel control principle

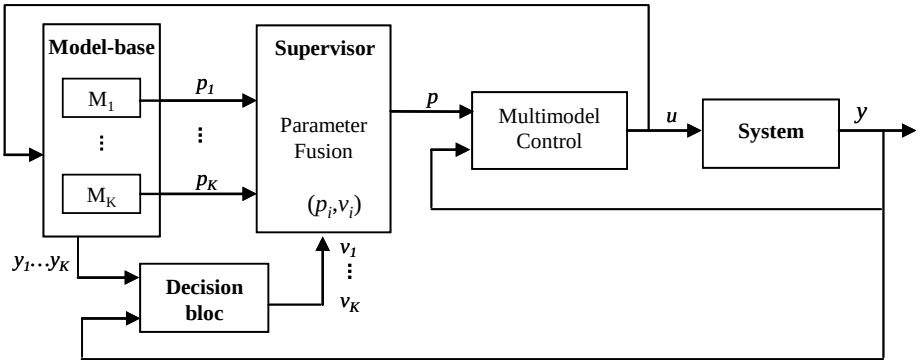

Figure 4. Principle of the fusion of control parameters

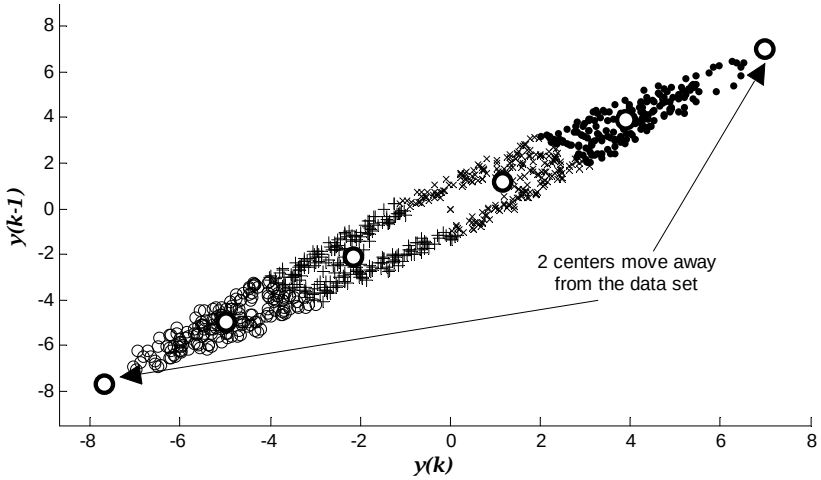

Figure 5. RPCL algorithm applied to nonlinear example

4. Nonlinear Process Control

The control of a discrete nonlinear system found in the work of [5] is tackled:

$$\begin{cases} x_1(k+1) = x_2(k) \\ x_2(k+1) = -\dfrac{3}{16}\left[\dfrac{x_1(k)}{1+\left(x_2(k)\right)^2}\right] + x_2(k) + u(k) \\ y(k) = x_1(k) \end{cases}$$

where u and y are the system's input and output.

Multimodel representation

Before control design, a multimodel representation was given, and Figure 5 shows that some of the centres move away, leaving only 4 classes, to be compared with the 64 models found in [5]. Since domain overlapping is important as can be seen from Figure 5, fuzzy K-means was selected to generate the 4 clusters, and 4 linear models were estimated, each of order 4. Introducing the NRMSE (Normalized Root Mean Square Error)

$$NRMSE = \frac{\sqrt{\frac{\sum_{i=1}^{N}\left(y_{mm}-y\right)^2\right)}{N}}}{y_{\max}-y_{\min}}, \qquad (17)$$

where y_{mm} and y are respectively the multimodel and system outputs, N is the number of samples, one obtains an accuracy of NRMSE = 0.002 compared to NRMSE = 0.0006 – best over Monte Carlo simulations – in [5] and NRMSE = 0.006 for a Time Delay Neural Network. Accuracy is thus fairly comparable whereas the design and workout complexity is reduced with respect to the work of [5].

The input sequence in Figure 6 was applied, and Figure 7 shows the difference between multimodel and real outputs.

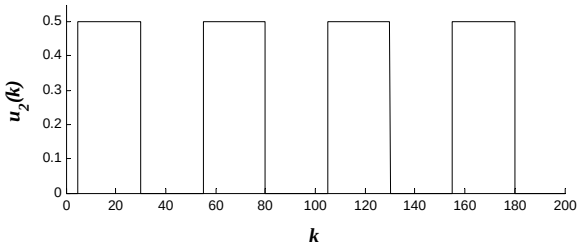

Figure 6. Input u_2

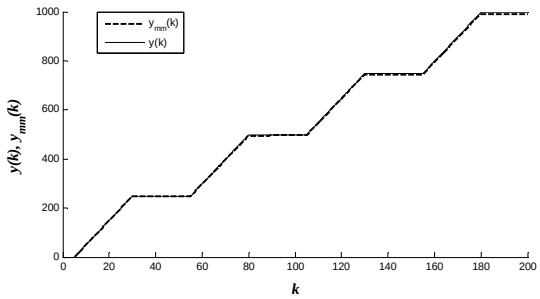

Figure 7. Real and multimodel outputs (input u_2)

Multimodel control

Once the multimodel structure is elaborated, a polynomial controller can be designed for each linear model of the base.

$$u_i(k) = \frac{1}{S_i(q^{-1})}\left[T_i(q^{-1})\,y_c(k) - R_i(q^{-1})\,y_i(k)\right] \quad i = 1,\ldots,K \qquad (18)$$

where q^{-1} is the delay operator, y_c is the reference, u_i , y_i are the i[th] model input and output, and

$$R_i(q^{-1}) = r_{i0} + r_{i1}\,q^{-1} + \ldots + r_{in_R}\,q^{-n_R}; \qquad (19)$$

The same kind of equation applies for S, T, parameters are obtained by pole placement [2].

Since the controller's structure is the same, it is chosen to blend controllers parameters in the multimodel way (equation 20). For example, one has [2]

$$R_g(k) = \sum_{i=1}^{4} v_i^{simp}(k)\,R_i \qquad (20)$$

and so on for S and T.

One expects a pole at $z = 0.5$ which yields e.g.

$$R_1 = \begin{bmatrix}1.1942 & -0.9502 & 0.013\end{bmatrix}$$
$$S_1 = \begin{bmatrix}1 & -0.4105 & -0.6102 & 0.0168 & 0.004\end{bmatrix}$$
$$T_1 = \begin{bmatrix}1.0267 & -1.0267 & 0.2567\end{bmatrix}$$

and so on for the remaining 3 models.

Taking the set-point

$$y_{c2} = 2 + 0.5\sin(0.2k) + 0.7\sin(0.04k),$$

Figure 8 enlightens the relevance of the proposed scheme, as the output and set-point are nearly the same. The value of the NRMSE (where NRMSE in (17) is now defined as a measure between system output and set-point) is 0.046, which yields a small relative error.

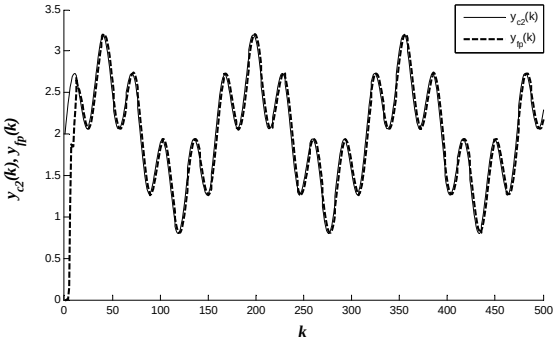

Figure 8. Evolution of output and set-point

5. Conclusion

The multimodel technique can be of a great help for blending local models or controllers of a more complicated system. However, the designer faces a high number of degrees of freedom, which is directly proportional to the number of local model and to their parameters set size. This paper shows how classification algorithm can automatically allocate an adequate number of models in the sense of a trade-off between modelling complexity and accuracy. It is also shown that the results of classification allow to direct the designer to a an algorithm which will accordingly separate or blend the different models. The same method can be replicated for multimodel control design.

This method has proved to be able to represent and control a nonlinear system, with the same accuracy and far less complexity than other multimodel or neural networks schemes.

REFERENCES

1. BARUCH, I. S., R. B. LOPEZ, J-L. OLIVARES, J-M. FLORES, **A Fuzzy-neural Multi-model for Nonlinear Systems Identification and Control,** Fuzzy sets and systems, vol. 159, 2008, pp. 2650-2667.
2. BEN ABDENNOUR, R., P. BORNE, M. KSOURI, F. M'SAHLI, **Identification et commande numérique des procédés industriels,** Editions Technip, Paris, France, 2001.
3. BEZDEK, J. C., **Pattern Recognition with Fuzzy Objective Function Algorithms**, Plenum Press, New York, 1981.
4. BÖLING, J. M., D. E. SEBORG, J. P. HESPANHA, **Multi-model Adaptive Control of a Simulated pH Neutralization Process**, Control Engineering Practice, vol. 15, 2007, pp. 663-672.
5. CHO, J., J. C. PRINCIPE, D. ERDOGMUS, M. A. MOTTER, **Quasi-sliding Mode Control Strategy based on Multiple-linear Models**, Neurocomputing, vol. 70, 2007, pp. 960-974.
6. ELFELLY, N., J-Y. DIEULOT, P. BORNE, **A Neural Approach of Multimodel Representation of Complex Processes,** International Journal of Computers, Communications & Control, vol. 3, 2008, pp. 149-160.
7. ELFELLY, N., J.-Y. DIEULOT, P. BORNE, M. BENREJEB, **A Multimodel Approach of Complex Systems Identification and Control using Neural and Fuzzy Clustering Algorithms**, Proceeding of the 9th International Conference on Machine Learning and Applications, Washington (USA), 2010, pp. 93-98.
8. ELFELLY, N., J.-Y. DIEULOT, M. BENREJEB, P. BORNE, **A New Approach for Multimodel Identification of Complex Systems based on Both Neural and Fuzzy Clustering Algorithms,** Engineering Applications of Artificial Intelligence, vol. 3, 2010, pp. 1064-1071.
9. FU, Y., CHAI, T., **Nonlinear Multivariable Adaptive Control using Multiple Models and Neural Networks**, Automatica, vol. 43, 2007, pp. 1101-1110.
10. GHARSALLAOUI, H., M. AYADI, M. BENREJEB, P. BORNE, **Robust Flatness-based Multi-controllers Approach,** Studies in Informatics and Control, Vol. 19, No.4/2010, pp.357-358.
11. JAMEL, W., A. KHEDHER, N. BOUGUILA, K. BEN OTHMAN, **State Estimation via Observers with Unknown Inputs: Application to a Particular Class of Uncertain T-S Systems,** Studies in Informatics and Control, vol.19, No. 3/2010, pp. 219-228.
12. M. KSOURI-LAHMARI, P. BORNE, M. BENREJEB, **Multimodel: the Construction of Model Bases**, Studies in Informatics and Control, Vol. 13, No.3/2004, pp. 199-210.
13. LORIMIER, L., P. BORNE, **Local Observer-based Multi-control and Online Validity Estimation for Multiple-model Control of Complex Systems**, IEEE International Conference Systems, Man, and Cybernetics, vol. 1, 2003, pp. 822-827.
14. MANIOUDAKIS, G. D., E. N. DEMIRIS, S. D. LIKOTHANASSIS, **A Self-organized Neural Network based on the Multi-model Partitioning Theory**, Neurocomputing, vol. 37, 2001, pp. 1-29.
15. MURLIDHARAN NAIR, T., C. L. ZHENG, J. LYNN FINK, R. O. STUART, M. GRIBSKOV, **Rival Penalized Competitive Learning (RPCL): A Topology-determining Algorithm for Analyzing Gene Expression Data**, Computational Biology and Chemistry, 27, 2003, pp. 565-574.
16. NASCIMENTO, S., B. MIRKIN, F. MOURA-PIRES, **A Fuzzy Clustering Model of Data and Fuzzy C-means**, The Ninth IEEE International Conference on Fuzzy Systems, vol. 1, 2000, pp. 302-307.
17. ÖZKAN, L., M. V. KOTHARE, C. GEORGAKIS, **Control of a Solution**

Copolymerization Reactor using Multi-model Predictive Control, Chemical Engineering Science, vol. 58, 2003, pp. 1207-1221.

18. PATIC, P. C., R. M. ZEMOURI, L. DUTA, **Recurrent Neural Networks in Linear Systems Controlling**, Studies in Informatics and Control, vol. 19, No. 2/2010, pp. 153-158.
19. TAMBE, S. S., B. D. KULKARNI, P. B. DESHPANDE, **Elements of Artificial Neural Networks with Selected Applications on Chemical Engineering, and Chemical & Biological Sciences**, Simulations & Advanced Controls, Louisville-KY, USA, 1996.
20. XUE, Z. K., S. Y. LI, **Multi-model Modelling and Predictive Control based on Local Model Networks,** Control and Intelligent Systems, 34, 2006, pp. 105-112.