# Model-Based Predictive Control of Large-Scale Systems Based on Fuzzy, Neural, and Neuro-Fuzzy Estimators

**Giorgos K. Apostolikas***

**Triatafyllos Pimenides****

**Spyros G. Tzafestas***

*Intelligent Robotics and Automation Laboratory

Division of Signals, Control and Robotics

Department of Electrical and Computer Engineering, National Technical University of Athens

Zographou 15773, Athens,

GREECE

E-mail: Giorgos.Apostolikas@gmx.net

E-mail: tzafesta@softlab.ntua.gr

**Division of Systems and Control

Department of Electrical and Computer Engineering

University of Patras, Patras 26500

GREECE

E-mail: pimenide@ee.upatras.gr

**Abstract:** In this paper use of fuzzy, neural and neuro-fuzzy estimators are employed to predict the interaction signals among subsystems in a large-scale system (LSS) controlled by the model-based predictive control (MBPC) scheme. The spreading of the non-local information within the LSS is assumed to follow the m-step delay sharing information pattern. The MBPC scheme is used for the local control of each subsystem enhanced by the interaction signal's predictive model. The interaction trajectories are assumed to be non-linear functions of the states of the various subsystems. The paper includes a thorough performance analysis of fuzzy versus neural estimators. Two nontrivial examples are studied via simulation to test and verify the capabilities of the proposed "fuzzy, neural and neuro-fuzzy estimator-based" MBPC of LSS.

**Keywords:** Large-scale-system, model-based predictive control, m-step delay sharing information pattern, interconnection signal, adaptive fuzzy / neuro-fuzzy predictor, neural estimator.

**Apostolikas Giorgos:** He received the Diploma in Electrical and Computer Engineering from the National Technical University of Athens in 1997. He is currently a Ph.D. Candidate at the Division of Signals, Control and Robotics working in Computational Intelligence Techniques and Applications in Control and Robotics. *Diploma thesis* : Fuzzy Model – Based Control of Large Scale Systems. *Teaching Experience* : Postgraduate courses in Robotics and Automation Laboratory of the National Technical University of Athens. *Research Interests* : Model – based and model – free predictive control of large scale systems including reinforcement learning in neural and fuzzy systems. He has published ten papers. Mr. Apostolikas is a member of the Technical Chamber of Greece

**Pimenides, Triataffylos.** He received Degree in Mathematics from University of Athens 1974, Diploma in Electrical Engineering from University of Patras 1981, Ph.D in 1984 from the University of Patras. He is an Associate Professor of Systems & Control of the University of Patras. His research interests include Control of Multivariable Dynamical Systems, Robotics, N-dimensional Systems, Computer–aided Design, Analysis Methods and Optimal Control. He has published over than fifty papers and three books. Mr. Pimenides T. is a member of the Technical Chamber of Greece and a member of the Greek Mathematic Society.

**Tzafestas Spyros G.:** full professor, Director of the Institute of Communication and Computer Systems (ICCS), the Signals, Control and Robotics Division and the Intelligent Robotics and Automation Laboratory (IRAL) of the National Technical University of Athens (NTUA). Holder of Ph.D. and D.Sc. in Control and Automation. Recipient of Honorary Doctorates of the International University (D.Sc. (Hon.) ) -and the Technical University of Munich (Dr.-Ing. E.h.) .Fellow of IEEE (N.Y.) and IEE (London) ; Member ofASME (N.Y.) , New York Academy of Sciences , IMACS (Rutgers, N.J.) and SIRES (Brussels). Member of IFAC SECOM and MIM TCs. Project evaluator of national european and international projects (USA, Canada, Italy, Hong Kong, Japan ). Project coordinator of national and EU projects in the fields of robotics, CIM and IT (ESPRIT, BRITE-EURAM , TIDE, INTAS , SOCRATES, EUREKA, GROWTH etc.). Publications: 30 research books, 60 book chapters, over 700 journal and conference technical papers. Editor-in-Chief of the Journal of Intelligent and Robotic Systems and the book series "Microprocessor-Based and Intelligent Systems Engineering" (Kluwer). Organizer of several international conferences (IEEE, IFAC, IMACS, IASTED, SIRES etc.). Listed in several international biographical volumes. Current interests include: control, robotics and CIM.

## 1. Introduction

Decentralized control theory and design of interconnected dynamical systems is still receiving increasing interest in the control community ([1], [2], [3]). One of the benefits of decentralized control is that large-scale systems (LSS) can be decomposed into many subsystems [4],[5], and the control system design and implementation of each one of them can be performed independently. This simplifies the overall control problem. Moreover the computational burden can be shared by all the control stations involved. The main difficulty in designing decentralized control systems is the limited information available for the control law [6],[7],[8]. The set of the control stations is constrained to have immediate access only to local information and restricted access to other subsystems' information [9]. This is exactly what characterizes a non-classical information pattern. Actually, much work has been conducted so far aiming at the analysis of stability of decentralized control problems and the development of decentralized stabilizers using output or state feedback (see e.g. [10], [11]). Moreover optimal decentralized control algorithms have been developed for dealing with large scale systems (LSS) of this type [12]. But it is well known that optimal control is not very suitable for complex industrial systems or general large-scale systems since it needs an accurate model of the controlled system, and it is sensitive to parameter variations and to the existence of stochastic disturbances.

An alternative control approach that does not have the drawbacks of standard optimal control and it is suitable for complex large scale systems is the so called **Model-Based Predictive Control (MBPC)** approach [13-16]. This approach was developed in the mid-seventies, and allows for model uncertainties, it updates the output of the model by closed-loop corrections, and optimizes the control law on a moving horizon. In this paper the decentralized control approach is combined with the model-based predictive control approach where for the estimation of the interconnection trajectories ([17]) a general technique is considered that takes advantage of the capability of fuzzy logic systems to uniformly approximate any non-linear function to any degree of accuracy. Section II of the paper presents the problem formulation, Section III introduces the concept of MBPC and deals with the computation of the local part of the control algorithm. Sections IV, V and VI present the various models implemented for the calculation of interconnection signals. Finally, Section VII presents a set of representative simulation results for particular examples including some comparisons between the models, and Section VIII gives some concluding remarks.

## 2. Problem Formulation

Consider a discrete-time, linear, possibly time-varying large scale system which consists of N-interconnected subsystems, each of which has the following state space description:

$$x_i(t+1)=A_i x_i(t)+B_i u_i(t)+E_i z_i(t) \tag{1a}$$

$$x_i(0)=x_{io} \qquad (i=1,2,\ldots,N) \tag{1b}$$

$$y_{m_i}(t)=C_i x_i(t) \tag{1c}$$

where:

$x_i(t)$: is the $n_i$-dimensional state vector of the ith subsystem at time t,

$u_i(t)$: is the $r_i$-dimensional control vector of the ith subsystem at time t,

$y_{m_i}(t)$: is the $p_i$-dimensional output vector of the model of the ith subsystem at time t, and

$z_i(t)$: is the $q_i$-dimensional interconnection vector that describes the influence of all other subsystems upon the ith one.

The vector $z_i(t)$ is considered to be a non-linear function of the states of all other subsystems, i.e.

$$z_i(t)=\Phi_i(x_1(t),x_2(t),\ldots,x_{i-1}(t),x_{i+1}(t),\ldots,x_N(t)), \quad i=1,2,\ldots,N \tag{2}$$

where $\Phi_i(\cdot)$ is assumed to be known to the ith control station. It is remarked that the output $y_{m_i}(t)$ of the model may generally have a small difference from the real output $y_i(t)$ because of modeling errors or noise which affect the whole system or parts of it.

Finally the matrices $A_i$, $B_i$, $E_i$, and $C_i$ are of proper dimensions, i.e. $A_i \in M_{n_i x n_i}$, $B_i \in M_{n_i x r_i}$, $E_i \in M_{n_i x q_i}$, $C_i \in M_{p_i x n_i}$, where $M_{kxs}$ is the set of matrices of kxs dimensions. In this paper we consider time invariant systems but the algorithm can be extended to slowly time-varying systems. The problem is "*to find at every time t the best control $u_i(t)$ for the ith subsystem, which leads the output current value $y_i(t)$ to its setpoint $w_i(t)$*". The control must be "the best" in the sense of minimizing a cost function which will be defined later. Moreover, the control laws must be specified in a decentralized way, i.e. the control laws are assumed to be of the form $u_i(t)=\acute{A}_i(I_i(t))$ , $i=1,2,\dots,N$, where $\acute{A}_i(\cdot)$ is a function of the available information set $I_i(t)$ of the ith subsystem defined as follows:

$I_i(t)=\{y_i(1), y_i(2),\dots, y_i(t) ; u_i(1), u_i(2),\dots,u_i(t-1) ; x_i(0), x_i(1),\dots\dots, x_i(t)\}$

$I_i(0)=\varnothing$

This means that $I_i(t)$ involves not only the measurement of the current output $y_i(t)$ but also the past outputs $y_i(r)$ , $u_i(r)$ $r<t$. The information set $I_i(t)$ does not contain $z_i(t)$ which is necessary for the computation of $u_i(t)$ but it is not available to the ith control station because it depends on the non-available states of the other subsystems $x_j(t)$, $j\neq i$.

In the following, the predictive control technique will be used to satisfy the problem requirements. It will be shown that the control laws are of the form :

$$u_i(t)=u_i^c(t)+u_i^d(t)\ ,\ i=1,2,\dots,N \tag{3}$$

where $u_i^c(t)$ is the **local part** of the control, i.e. the part which is available to the ith control station, and $u_i^d(t)$ is the **non-local part** of the control law which depends on information not available to the ith control station, and actually depends on $z_i(t)$ and predictions of it.

## 3. MBPC and Computation of the Local Part of the Control

Model-Based Predictive Control is a control algorithm which uses a model for open-loop predictions, optimizing the control inputs on a moving horizon and updating the outputs of the model by closed loop predictions.

Especially, at each time t, the output $y_i(t+k)$ is predicted over a future period of time $k=1,2,\dots,L_y$ where $L_y$ is the prediction horizon. The predictions are symbolized by $y_{pi}(t+k/t)$ and are determined by means of a model, for example a state space model like (1a-1c). The predictions $y_{pi}(t+k/t)$ , $k=1,2,\dots,L_y$ depend on future control values $u_i(t+k/t)$ , $k=0,1,\dots,L_u-1$ , where $L_u$ is the control horizon ($L_u\leq L_y$).

We assume that the control signal for time $t' \geq t+L_u$ remains fixed in the last value of the control horizon $u_i(t+L_u-1)$, i.e.

$u_i(t+L_u+k/t)=u_i(t+L_u-1)$ , $k\geq 0$

The ith subsystem output predictions can be calculated as:

$$y_{pi}(t+k/t)=y_{mi}(t+k/t)+q_i(t+k/t) \tag{4}$$

where $y_{mi}$ is the prediction for the output derived by the available model, using (1a, b) :

$$y_{mi}(t+k/t)=C_i\left[A_i^k x_i(t)+\sum_{j=1}^{k}A_i^{t-1}B_i u_i(t+k-j/t)+\sum_{j=1}^{k}A_i^{t-1}E_i z_i(t+k-j/t)\right] \tag{5}$$
$$k=1,2,\dots,L_y$$

and $q_i(t+k/t)$ is the closed-loop correction vector based on the available information set at time t. A recommended form for $q_i(t+k/t)$ is the following:

$$q_i(t+k/t)=y_i(t)-y_{mi}(t) \tag{6}$$

where $y_i(t)$ is the measured value of the output vector at time t.

A reference trajectory $r_i(t+k/t)$, $k=1,2,\dots,L_y$ is defined over the prediction horizon, which describes how it is desirable for the output vector $y_i(t)$ to move to its set-point $w_i(t)$, i.e.

$$r_i(t+k/t)=w_i(t+k/t)+\upsilon_i(t+k/t) \tag{7}$$

where $\upsilon_i(t+k)$ is a correction vector based on the previous error information set $\{w_i(t)-y_i(t), w_i(t-1)-y_i(t-1),\dots, w_i(1)-y_i(1)\}$.

A simple form which gives good results is the following:

$$\upsilon_i(t+k/t)=a^k[w_i(t)-y_i(t)] \tag{8}$$

where $0<a<1$ is a tuning parameter that specifies the desired closed-loop dynamic ($a=0$:fast control; $a=1$:slow control).

The reference trajectory is initiated at the current measured output i.e. $r_i(t/t)=y_i(t)$.

It is noted that if the future set-point values $w_i(t+k/t)$, $k=1,2,\ldots,L_y$ are unknown at time t, one can assume that:

$$w_i(t+k/t) = w_i(t)\ ,\ k=1,2,\ldots,L_y \tag{9}$$

All the above issues are illustrated in Fig.1.

The cost function for the ith control station has the form:

$$J_i(t)=\frac{1}{2}\left[\sum_{k=L_o}^{L_y}\left\|r_i(t+k/t)-y_{pi}(t+k/t)\right\|^2_{Q_i(k)}+\sum_{k=0}^{L_u-1}\left\|u_i(t+k/t)\right\|^2_{R_i(k)}\right] \tag{10}$$

where $Q_i(k)\geq 0$ for $k=L_o,\ldots,L_y$, and $R_i(k)>0$ for $k=0,\ldots,L_u-1$.

**Since $J_i(t)$ varies with time t and has a moving optimization horizon, only the first term in the optimal solution is implemented to control the ith subsystem. The optimization parameter $L_o$ determines, together with $L_y$, the "coincidence horizon" during which the predicted output has to follow the reference trajectory over the time interval $[t+L_o,\ldots,t+L_y]$.**

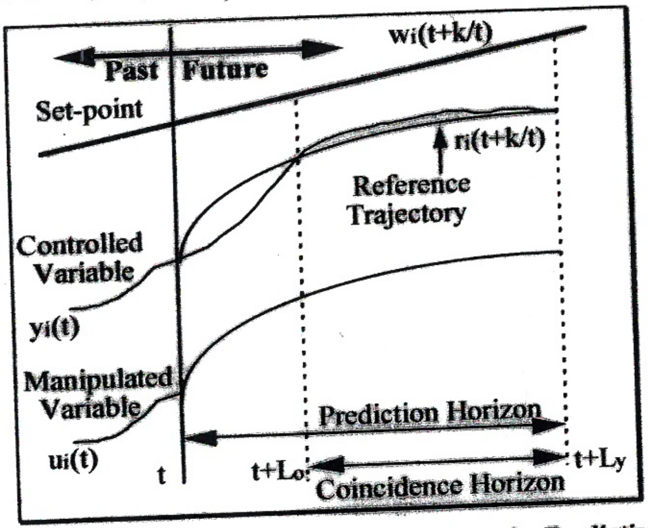


**Figure 1. The Reference Trajectory, the Setpoint Trajectory, the Prediction Horizon, and the Coincidence Horizon**

The basic concepts of the MBPC structure (applied to the ith subsystem) are illustrated in Fig.2.

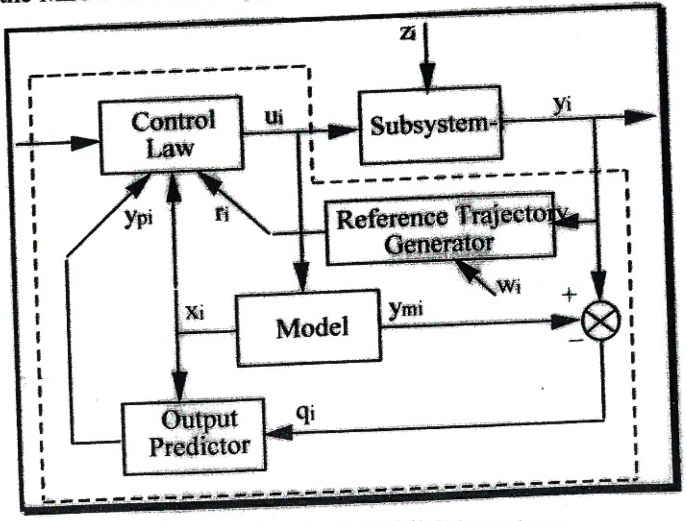


**Figure 2. The Basic MBPC Structure**

The minimization of the cost function (10) over the future control scenario (sequence) $\hat{u}_i(t) = [u_i(t/t),...,u_i(t+L_u-1/t)]^T$, is straightforward (see [3] for the complete proof):

$$\hat{u}_i(t) = \hat{u}_i^c(t) + \hat{u}_i^d(t) \tag{10a}$$

$$\hat{u}_i^c(t) = \hat{D}_i(t)\hat{\gamma}_i(t) \tag{10b}$$

$$\hat{u}_i^d(t) = \hat{E}_i(t)\hat{z}_i(t) \tag{10c}$$

where the gain matrices $\hat{D}_i(t)$ and $\hat{E}_i(t)$ depend on the ith subsystem parameters and the weight matrices $Q_i(t)$, and $R_i(t)$ of the cost function. The extended vector $\hat{\gamma}_i(t)$ depends on information which is fully available to the ith control station, namely: current and future values of the closed-loop correction vector $q_i(t)$ in (6), the distance between the reference trajectory and the setpoint $\upsilon_i(t)$ in (8), the setpoint $w_i(t)$ itself in (9), and the associated subsystem's parameters. On the other hand the extended vector $\hat{z}_i(t) = [z_i(t/t),...,z_i(t+Ly-1/t)]^T$ depends on present and future values of the interaction vectors. None of the previous vectors is available to the ith control station because all these vectors depend on the states (present and future) of the other subsystems. So, it is required to approximate or to predict them using interaction models. In the sequel, an adaptive fuzzy, a neuro-fuzzy and a neural predictor are employed, focused on the approximation of the present and future values of the interconnections necessary for the computation of the non-local part of the control.

The method is implemented here, for the case of LSS that have an m-step delay sharing information pattern. A decentralized control problem is said to have an *m-step delay sharing information pattern* when it permits the spreading of its information through the subsystems after a delay of m time units. Clearly, each control station obtains *instantaneously* all the information about its associated subsystem, and after a delay of m time units all the information available to *all* control stations. For the problem considered here, this means that at time t, in the ith control station, the vectors $x_j(t-m)$, $j \neq i$ and all the past values $x_j(t-m-k)$, $k>0$ are known. Then one can calculate $z_i(t-m)$ using (2). Without loss of generality, it can be assumed that the jth component of the vector $z_i(t-m)$, denoted as $z_i^j(t-m)$, is the output of a MISO non-linear system:

$$z_i^j(t-m) = F_i^j(z_i^j(t-m-1), z_i^j(t-m-2),...,z_i^j(t-m-p)) \tag{11}$$

where $p \geq 1$ is the order of the model (design parameter). The task of the fuzzy system implemented here is to estimate the function $F_i^j(\cdot)$ at every time t for the known set of numerical data $\{z_i^j(t-m), z_i^j(t-m-1), z_i^j(t-m-2),...,z_i^j(t-m-p)\}$. After this estimation, $\overline{F}_i^j(\cdot)$ becomes available and can be iteratively used as:

$$\begin{aligned} &\overline{z}_i^j(k/t) = \overline{F}_i^j(\overline{z}_i^j(k-1/t), \overline{z}_i^j(k-2/t),...,\overline{z}_i^j(k-p/t)) \\ &\quad k = t-m+1, t-m+2,..,t+L_y-1 \end{aligned} \tag{12}$$

for the calculation of the extended vector $\hat{z}(t)$ and the non-local part of the control in (10c). It is noted that the estimation of $\overline{F}_i^j(\cdot)$ is updated on-line at every time t, since new input - output data are available.

## 4. Adaptive Fuzzy Predictors Based on Lookup Tables as Approximators for the Interconnections

In this section, a simple and effective method is presented for adaptive fuzzy system design based on the work of Wang [18], which performs a one-pass operation on the numerical input-output pairs. In the following, a brief description of the proposed fuzzy system used is given.

Consider a set of desired input-output data pairs is given:

$$(x_1^{(1)}, x_2^{(1)}; y^{(1)}), (x_1^{(2)}, x_2^{(2)}; y^{(2)}),... \tag{13}$$

where $x_1$ and $x_2$ are inputs, and y is the output. This simple two-input one-output case is chosen in order to emphasize and clarify the basic ideas of the proposed approach; extensions to general multi-input-multi-output cases are straightforward and will be discussed later in this section. The task here is to generate a set of fuzzy rules from the desired input output pairs of (13) and use these fuzzy IF-THEN rules to determine a fuzzy logic system. This approach consists of the following five steps:

**Step 1:** Divide the Input and Output Spaces into Fuzzy Regions

Assume that the domain intervals of $x_1$, $x_2$ and y are $[x_1^-, x_1^+], [x_2^-, x_2^+]$ and $[y^-, y^+]$, respectively, where the domain interval of a variable means that most probably this variable will lie in this interval (the values of a variable are allowed to lie outside its domain interval). Divide each domain interval into 2N + 1 regions (N can be different for different variables, and the lengths of these regions can be equal or unequal) and assign to each region a fuzzy membership function. It is recommended that a normalization procedure of the data in the interval [0,1] takes place before they feed the fuzzy system. Figure 3 shows an example where the domain interval of $x_1$, is divided into seven regions (N = 3), and the domain region of $x_2$ is divided into five regions (N = 2). The shape of each membership function is triangular; one vertex lies at the center of the region and has membership value unity; the other two vertices lie at the centers of the two neighboring regions, respectively, and have membership values equal to zero. Of course, other divisions of the domain regions and other shapes of membership functions are possible.

**Step 2:** Generate Fuzzy Rules from Given Data Pairs

First, determine the degrees of the given $x_1^{(i)}, x_2^{(i)}$ and $y^{(i)}$ in different regions. Second, assign a given $x_1^{(i)}, x_2^{(i)}$ or $y^{(i)}$ to the region with the maximum degree. Finally, obtain one rule from each pair of desired input-output data according to their associated degrees. The rules generated in this way, are "AND" rules, i.e. rules in which the conditions of the IF part must be met simultaneously in order for the result of the THEN part to occur.

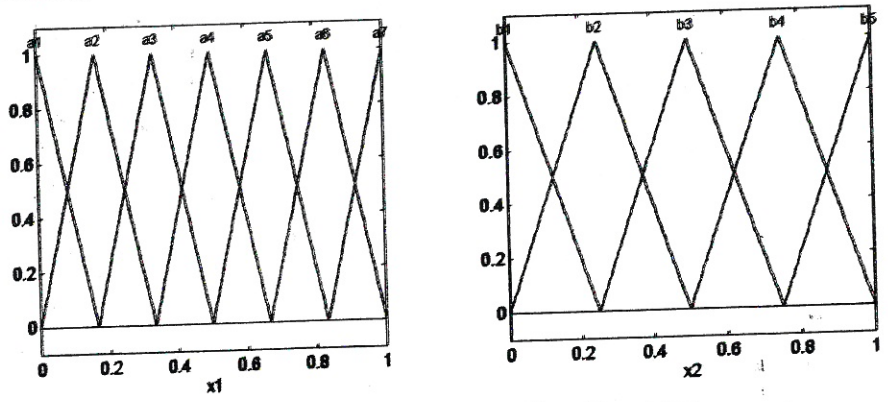


**Figure 3. Membership Functions Examples**

For the problem considered here, i.e. the problem of generating fuzzy rules from numerical data, only "AND" rules are required since the antecedents are different components of a single input vector.

**Step 3:** Assign a Degree to Each Rule

Since there are usually many data pairs, and each data pair generates one rule, it is highly probable that there will be some conflicting rules, that is rules which have the same IF part but a different THEN part. One way to resolve this conflict is to assign a degree to each rule generated from data pairs and accept only the rule from a conflict group that has maximum degree. In this way not only is the conflict problem resolved, but also the number of rules is greatly reduced. The following product strategy is used, for the assignment of each rule with a degree: for the rule "IF $x_1$, is A and $x_2$ is B, THEN y is C," the degree of this rule, denoted by D(Rule), is defined as

$$D(\text{Rule}) = \mu_A(x_1)\mu_B(x_2)\mu_C(y) \tag{14}$$

where $\mu_L(x)$ is the membership function value of fuzzy set L for the crisp value x.

In practice, a priori information about the data pairs is often available. For example, if an expert checks given data pairs, the expert may suggest that some are very useful and crucial, but others are very unlikely and may be caused just by measurement errors. Although it was not necessary to implement it in our examples, generally the designer can assign a degree to each data pair which represents the belief of its usefulness. Suppose the data pair $(x_1^{(i)}, x_2^{(i)}, y^{(i)})$ has degree $\mu^{(i)}$, then the degree of Rule is redefined as

$$D(\text{Rule}) = \mu_A(x_1)\mu_B(x_2)\mu_C(y)\mu^{(i)} \tag{15}$$

That is, the degree of the rule is defined as the product of the degrees of its components and the degree of the data pair which generates this rule. This is important in practical applications because real numerical data have different reliabilities. For good data, higher degrees are assigned, whereas for bad data lower degrees are assigned. In this way human experience about the data is used in a common base as other information. If one emphasizes objectivity and does not want a human to judge the numerical data this strategy still works by setting all the degrees of the data pairs equal to unity.

**Step 4:** Determine a Mapping Based on the Fuzzy Rule Base

The following defuzzification strategy is used to determine the output control y for given inputs $(x_1,x_2)$. First for given inputs $(x_1,x_2)$, the antecedents of the ith fuzzy rule are combined using product operations to determine the degree of $\mu^i_{O^i}$ of the output control corresponding to $(x_1,x_2)$ that is,

$$\mu^i_{O^i} = \mu_{I^i_1}(x_1)\mu_{I^i_2}(x_2) \tag{16}$$

where $O^i$ denotes the output region of Rule i, and $I^i_j$ denotes the input region of Rule i for the jth component. Then the *center of gravity (COG)* defuzzification formula is used to determine the output

$$y = \frac{\sum_{i=1}^{M} \mu^i_{O^i} \bar{y}^i}{\sum_{i=1}^{M} \mu^i_{O^i}} \tag{17}$$

where $\bar{y}^i$ denotes the center value of region $O^i$ (the center of a fuzzy region is defined as the point which has the smallest absolute value among all the points at which the membership function for this region has membership value equal to 1), and M is the number of fuzzy rules in the fuzzy rule base.

From Steps 1 to 4 one can see that the proposed method is simple and straightforward in the sense that it is a one-pass build-up procedure that does not require time-consuming training; hence, it has the same advantage that the fuzzy approach has over the neural approach, namely, it is simple and quick to construct.

This four-step procedure can easily be extended, to general multi-input-multi-output cases. Steps 1 to 3 are independent of how many inputs and how many outputs there are. In Step 4, $\mu^i_{O^i}$ in (16) needs to be replaced by $\mu^i_{O^i_j}$, where j denotes the jth component of the output vector ($O^i_j$ is the region of Rule i for the jth output component; $\mu^i_{O^i_j}$, is the same for all j), and the COG formula (17) must be changed to

$$y_j = \frac{\sum_{i=1}^{M} \mu^i_{O^i_j} \bar{y}^i_j}{\sum_{i=1}^{M} \mu^i_{O^i_j}} \tag{18}$$

where $\bar{y}^i_j$ denotes the center of the region $O^i_j$.

## 5. Adaptive Neuro – Fuzzy Predictors as Approximators for the Decentralized Part of the Control

In this section the proposed neuro-fuzzy predictor is presented.

The configuration of the fuzzy logic system is shown in Figure 4 where the fuzzy rule base consists of a collection of fuzzy IF-THEN rules, and the fuzzy inference engine uses these rules to determine a mapping from fuzzy sets in the input universe of discourse to fuzzy sets in the output universe of discourse based on fuzzy logic principles. As real world variables are numbers, fuzzifier and defuzzifier modules are incorporated to the system.

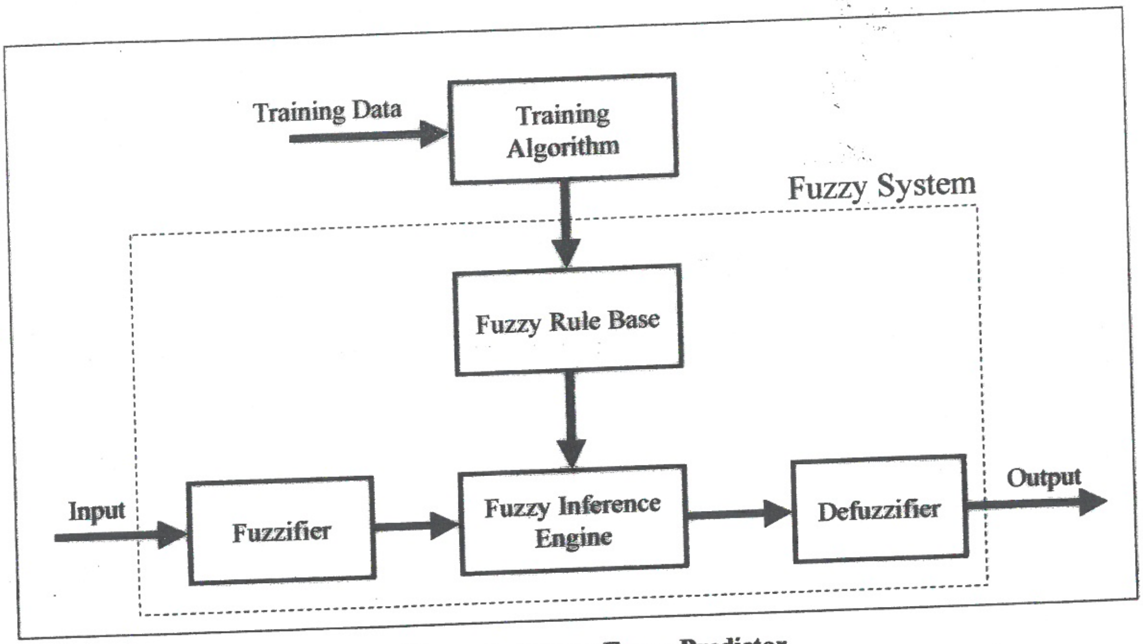


**Figure 4. Neuro-Fuzzy Predictor**

The fuzzy predictor is a MISO system. The input vector length is denoted by N. All elements of the fuzzy system are described next :

**Fuzzifier :**

The fuzzifier maps crisp points in $\Re$ to fuzzy sets defined in $\Re$. The fuzzifier outputs are singletons fuzzy sets with their unique not zero value set to the value of the corresponding real number input. A fuzzy set A defined on $\Re$ is called singleton if for a unique $x \in \Re$ $A(x)=1$ and for all other $y \in \Re$ $A(y)=0$. Non-Singleton fuzzifier are only used to reflect the measuring uncertainty of the corrupted by noise variables fed in the fuzzy system. In the fuzzy predictor the use of non-singleton fuzzifier would introduce a significant computational cost overhead and only a minor improvement in prediction results.

**Fuzzy Rule Base :**

The fuzzy rule base consists of a collection of M (l=1,2,…,M) fuzzy IF-THEN rules of the general form:

$$R^l : \text{if} \quad X_1 \text{ is } F_1^l \quad \text{and} \quad X_2 \text{ is } F_2^l \quad \text{and} \ldots \text{and} \quad X_N \text{ is } F_N^l \quad \text{then} \quad Y \text{ is } G^l$$

Where $F_i^l$ (i=1,2,…N) and G are fuzzy sets, $X_i$ (i=1,2,…,N) and Y are the input and output linguistic variables, respectively. These fuzzy IF-THEN rules are mostly used as they provide a convenient framework for a straightforward mapping from the fuzzy input space to the output one. As the fuzzifier is singleton the linguistic variables $X_i$ are singleton fuzzy sets. Output linguistic variables are also chosen to be singleton fuzzy sets for computational cost reasons. As we will see next, both “Fuzzy Inference Engine” and “Defuzzifier” modules have dramatically decreased computational burden due to the use of singleton output linguistic variables. Moreover, this choice does not seem to influence at all the quality of the fuzzy systems used in control applications.

$F_i^l$ fuzzy sets are Gaussian membership functions (figure 5) :

$$\mu_{F_i}^l(x_i) = \exp\left[-\left(\frac{x_i - \bar{x}_i^l}{\sigma_i^l}\right)^2\right]$$

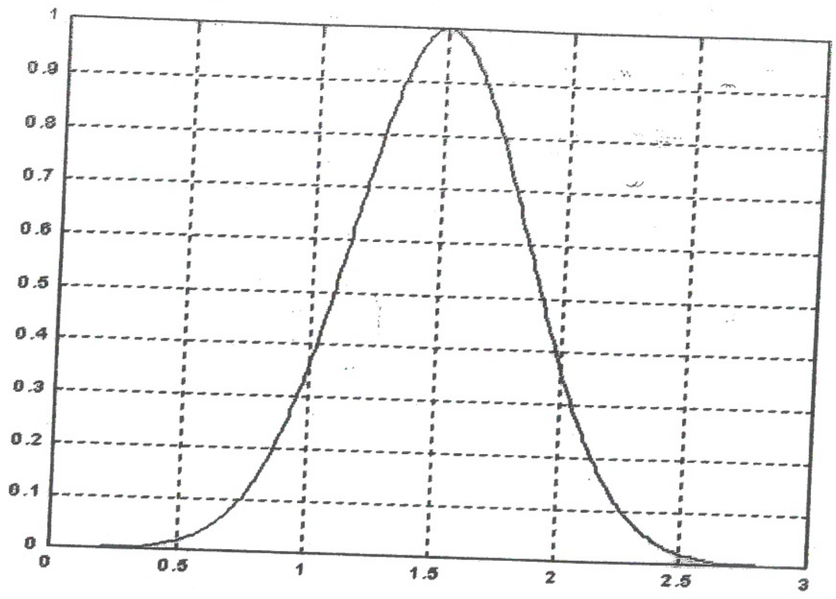


**Figure 5. Gaussian Membership Function**

Gaussian membership functions provide good generalization for the fuzzy system which is very important as the input space is multi-dimensional (usually 4-dimensional). A narrow membership function, like triangular, would need at least four times more training data to cover the whole input space and to give predictions with the same accuracy.

**Fuzzy Inference Engine :**

The inference engine combines the information stored in the fuzzy rule base with a number of past values of the interconnection signals to predict future values of those signals. The inference engine is implemented with the product fuzzy implication :

$$\mu_{A\rightarrow B}(\underline{x}, y) = \mu_A(\underline{x}) \cdot \mu_B(y) \tag{19}$$

where $\mu_A$ is defined according to the product operation rule :

$$\mu_A(\underline{x}) = \mu_{F_1\times\cdots\times F_N}(\underline{x}) = \mu_{F_1}(x_1) \cdot \mu_{F2}(x_2) \cdots \mu_{F_N}(x_N) \tag{20}$$

From the common fuzzy implications (Min, Arithmetic, Maxmin, Boolean) this is the only one that can be used in a fuzzy system trained by the back propagation (BP) algorithm, since BP requires the calculation of some derivatives associated with the fuzzy system function.

**Defuzzifier :**

The defuzzifier performs a mapping from fuzzy sets space to real numbers. Having used singleton fuzzy sets as output membership functions, the defuzzifier performs a non-linear interpolation using the real numbers each of which is the only non-zero element of the corresponding singleton output. So it is logical to utilize the *center average* defuzzifier which gives as output the most probable value. The center average defuzzifier is defined as :

$$y = \frac{\sum_{l=1}^{M} \overline{y}^l \cdot \left(\mu_{B^l}(\overline{y}^l)\right)}{\sum_{l=1}^{M} \left(\mu_{B^l}(\overline{y}^l)\right)} \tag{21}$$

where $\overline{y}^l$ is the center of the output fuzzy set $G^l$ of rule l.

Having defined all the modules of the fuzzy system we can now extract its corresponding function which has the form:

$$\sigma_i^l(t+1) = \sigma_i^l(t) - a_\sigma \cdot \frac{f(\underline{x}^t) - d^t}{b} \cdot \left(\overline{y}^l - f(\underline{x}^t)\right) \cdot h_l \cdot \frac{2\left(x_i^t - \overline{x}_i^l(t)\right)^2}{\left(\sigma_i^l(t)\right)^3} \tag{26}$$

where $\alpha_y$, $\alpha_x$, and $\alpha_\sigma$ are the learning steps used for the three sets of parameters.

## 6. Neural Estimation for the non Local Part of the Control

Multilayer Perceprtons (MLPs) are networks where the processing units (neurons) are arranged in layers. The nodes of each layer take as input the outputs of the nodes of the previous layer and perform a non-linear (usually sigmoidal) transformation on them in order to produce their outputs. The units belonging to one layer are connected only to the nodes of the previous and the next layer and not to each other (Fig.7). The network learns by adjusting its weights according to the back-propagation algorithm, which computes the error between the actual and the desired output for a number of training patterns (supervised learning) and "backpropagates" it to the units which adjust their assossiated weight values so that the actual response of the network moves closer to the desired response. To express the above in a formal mathematical way, the output of the i-th node is assumed to be:

$$o_i = \sigma\left(\sum_j w_{ij} + \theta_l\right), \quad \sigma(x) = 1/(1-e^{-x}) \tag{27}$$

where $w_{ij}$ is the weight corresponding to the connections from node j to node i, and $\theta_i$ a fixed value called "threshold". The algorithm minimizes the value of the "energy" function

$$E(\mathbf{w}) = \sum_p (t_p - o_p)^2 \tag{28}$$

where $t_p$, $o_p$ are the desired and the actual output when the network receives as input the pattern p. To do so, it uses the delta rule, thus

$$w_{ij}(n+1) = w_{ij}(n) - n \cdot \nabla E(\mathbf{w}) \tag{29}$$

MLPs can be trained either off-line when all of the training patterns are known beforehand or on-line in the opposite case [25]. It can be proved that an MLP trained with the back-propagation algorithm can approximate arbitrarily well, every function belonging to the $L_2$ class. Another very important characteristic of the MLP networks is their ability to generalize, i.e. to give valid (meaningful) output for input patterns for which they have not been trained (since after their training they implement a continuous mapping function).

In this paper a multi-layer perceptron is used at each control station to predict the interaction trajectrories over the prediction horizon.

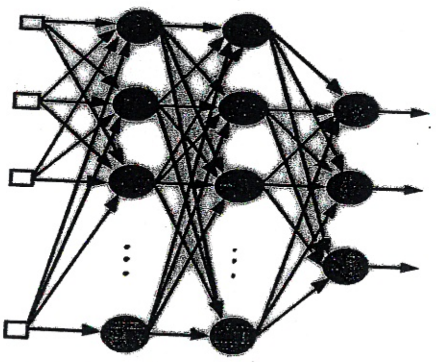


**Figure 7. A Multi-layer Perceptron with two Hidden Layers**

An estimation of $z_i(t+k/t)$, k=0,1,...,$L_y$-1 can be obtained by an appropriate model of $z_i(t-m)$ as a function of $z_i(t-m-j)$ j=1,2,...,p, where p is a design parameter. For this purpose the vector $z_i(t-m)$ is computed as the output of an MLP with inputs the $z_i(t-m-j)$ values, j=1,2,...,p. In this case of course the training is done on-line since each estimated value is used for the estimation of the next one. The training algorithm is a variant of the standard back-propagation algorithm. That is, the energy function to be minimized is the following:

$$E(\mathbf{w}) = \sum_{p}(t_p - o_p)^2 + \frac{n_w \sum_i \sum_j w_{ij}^2}{\sum_i \sum_j w_{ij}^2 + n_w K_2} \qquad (30)$$

where $w_{ij}$ is the weight of the connection from node j to node i, and $K_2$, $n_W$ are appropriate constants. The extra term added increases the stability of the algorithm and the generalization capability of the network by reducing the number of active weights (i.e. weights with values not practically equal to zero). The weights are again adapted according to the delta rule where the gradient of the new energy function is computed. The initial weight values are random but small, so that the derivatives of the network output attain their maximum values. During the first m time-steps before the real value of the predicted variable is communicated to the subsystem, the network output is random. From that time on, the network is trained on-line such that to minimize the energy function (30), where p increases as more real values are communicated to the subsystem. The training stops when the value of E(w) becomes smaller than a preset limit depending on the specifications of each problem.

## 7. Simulation Results

Here, two simple but nontrivial examples of the application of the proposed predictors to the decentralized control scheme are provided. These examples illustrate the predictors' capabilities and effectiveness, and provide a comparison of their performance in the specified task.

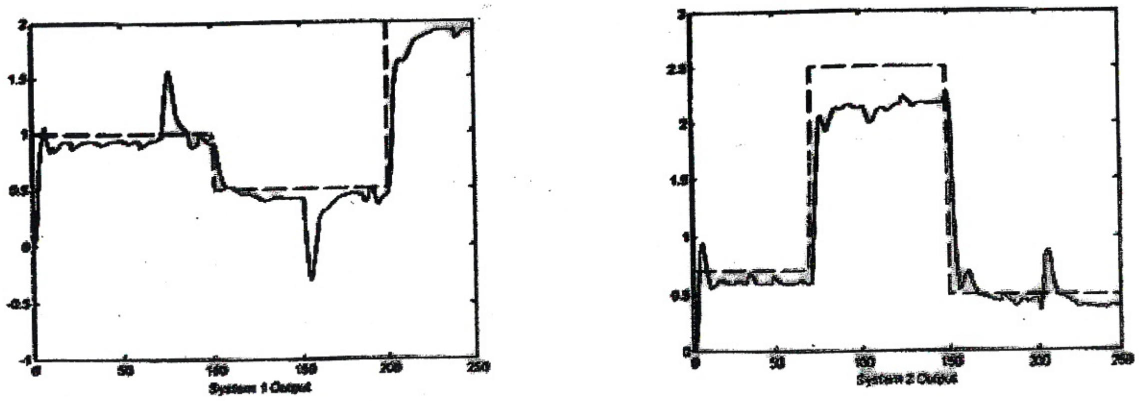
**System (i)**

This system consists of two subsystems and a 3-step delay sharing information pattern. The poles of the first subsystem are -25±10i (in the s-domain) and the ones of the second subsystem are -5±35i. For both subsystems a discrete-time state-space model was derived using the Control Toolbox of Matlab. The Model Based Predictive Controller has the following parameters: $L_o$=1, $L_y$=10, $L_u$=8, $\alpha$=0.4, Q=I, R=$10^{-3}$I (I = the identity matrix of proper dimensions) for both subsystems.

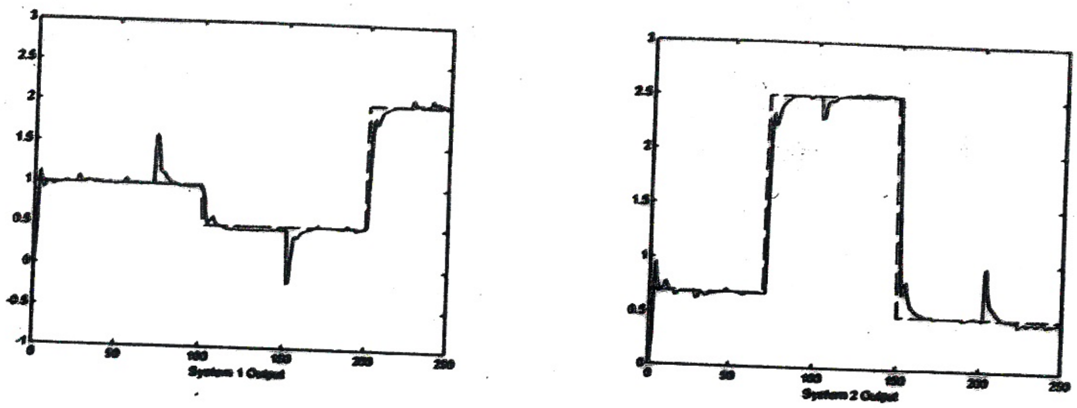
The interaction model is:

$$z_1 = x_1^{(2)} + 2x_2^{(2)}$$
$$z_2 = x_2^{(1)} - x_1^{(1)}$$

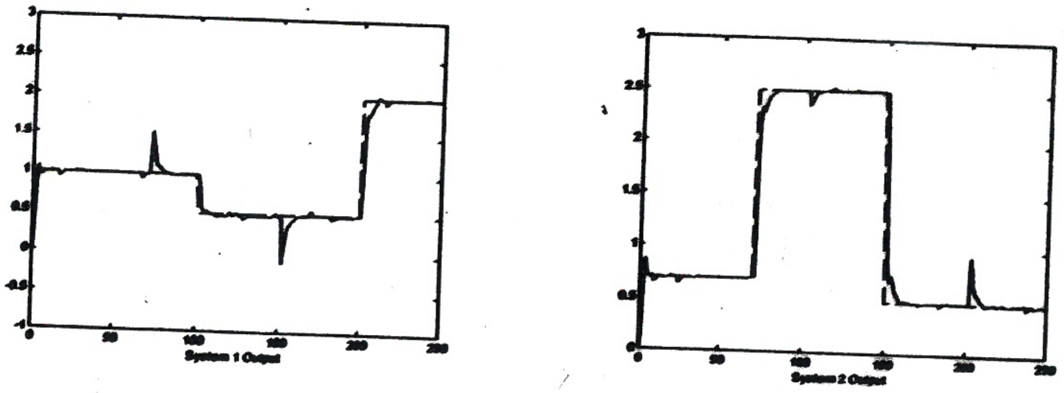
The interaction gain matrices in the state space model are $E_1$=[0.3 0.1], $E_2$=[-0.4 0.3]. Four past values of the interconnection signal (that is p=4 in (11) and (12)) were used as input variables in all models to predict the next value of the interconnection signal. The plots of Fig. 8a show the outputs of the two subsystems in the case where the MBPC controller acts alone and no care is taken concerning the interconnections. Figures 8(b-d) show the outputs of the subsystems when each one of the three proposed models is used to predict the interconnection signals.



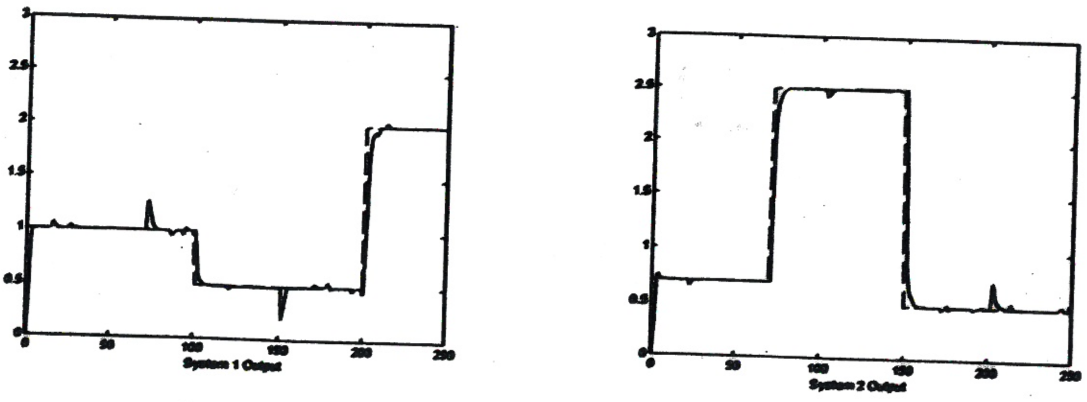
(a) No Decentralized Control



**(b) Fuzzy Predictor**



**(c) Neurofuzzy Predictor**



**(d) Neural Predictor**

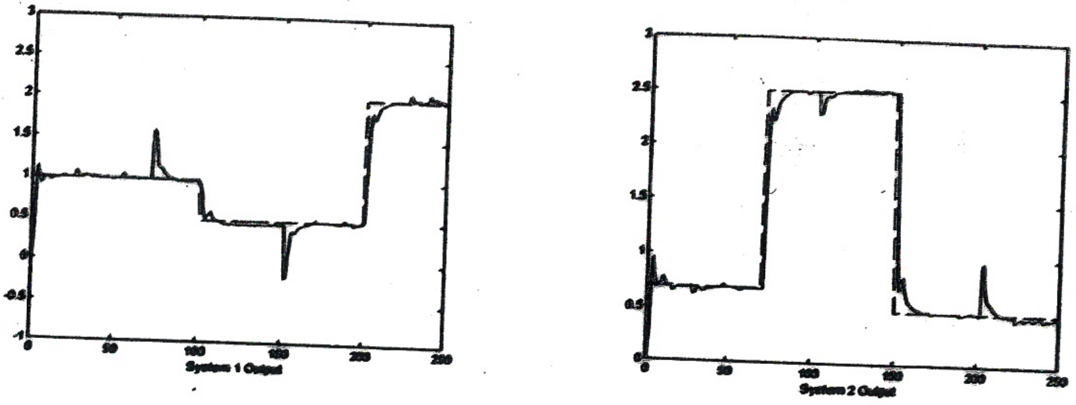
**Figure 8. Simulation results of system (i). (a) no decentralized control, (b) results with fuzzy predictor, (c) results with neuro-fuzzy predictor, (d) results with neural predictor**

**System (ii)**

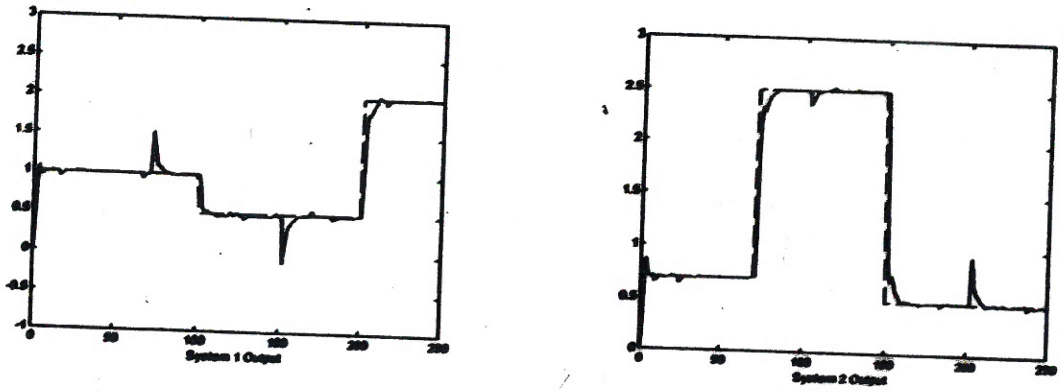
This system consists of two "harder" subsystems and a 3-step delay sharing information pattern. The poles of the first subsystem are -8±60i (in the s-domain) and the ones of the second subsystem are -1±50i. The Model Based Predictive Controller has the following parameters $L_o=1$, $L_y=7$, $L_u=5$, $\alpha=0.5$, $Q=I$, $R=10^{-4}I$, for both subsystems. The interaction model is:

$z_1 = x_1^{(2)2} - 3x_2^{(2)2}$

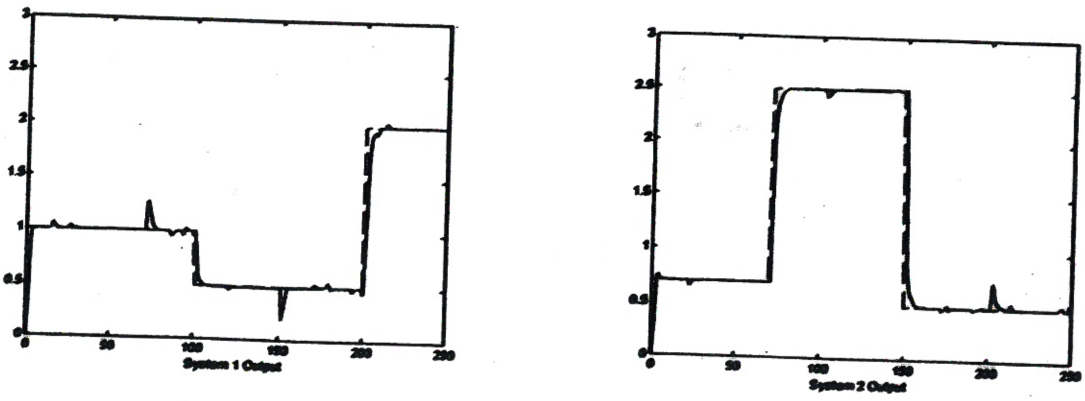
$z_2 = x_1^{(1)} / ( |x_2^{(1)}| + 1 )$



**(b) Fuzzy Predictor**



**(c) Neurofuzzy Predictor**



**(d) Neural Predictor**

**Figure 8. Simulation results of system (i). (a) no decentralized control, (b) results with fuzzy predictor, (c) results with neuro-fuzzy predictor, (d) results with neural predictor**

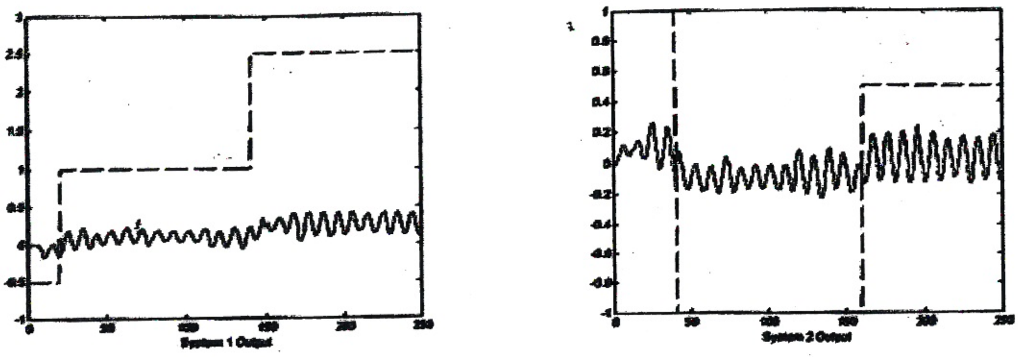
**System (ii)**

This system consists of two "harder" subsystems and a 3-step delay sharing information pattern. The poles of the first subsystem are -8±60i (in the s-domain) and the ones of the second subsystem are -1±50i. The Model Based Predictive Controller has the following parameters $L_o=1$, $L_y=7$, $L_u=5$, $\alpha=0.5$, $Q=I$, $R=10^{-4}I$, for both subsystems. The interaction model is:

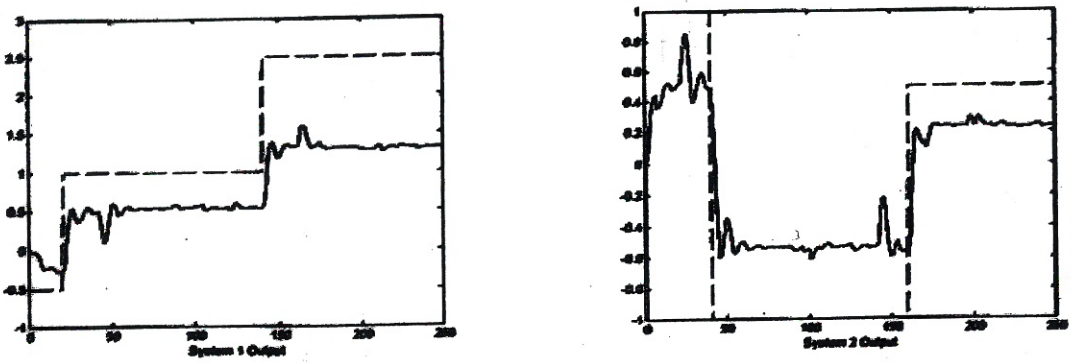
$z_1 = x_1^{(2)2} - 3x_2^{(2)2}$

$z_2 = x_1^{(1)} / ( |x_2^{(1)}| + 1 )$

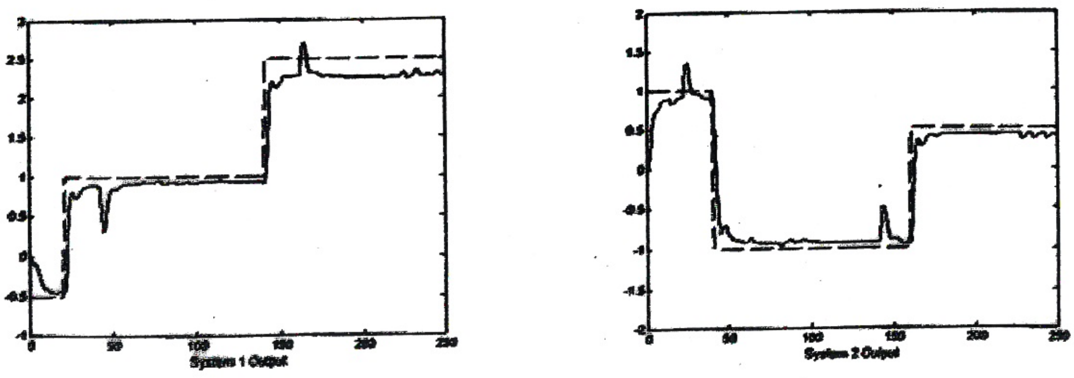
The interaction gain matrices in the state space model are $E_1=[0.6\ 0.2]$, $E_2=[0.7\ -0.1]$. Four past values of the interconnection signal were used as input variables in all models to predict the next value of the interconnection signal. The plots of Fig. 9a show the output of the two subsystems in the case where the MBPC controller acts alone and no care is taken concerning the interconnections. Figures 9(b-d) show the outputs of the subsystems when each one of the three proposed models is used to predict the interconnection signals.



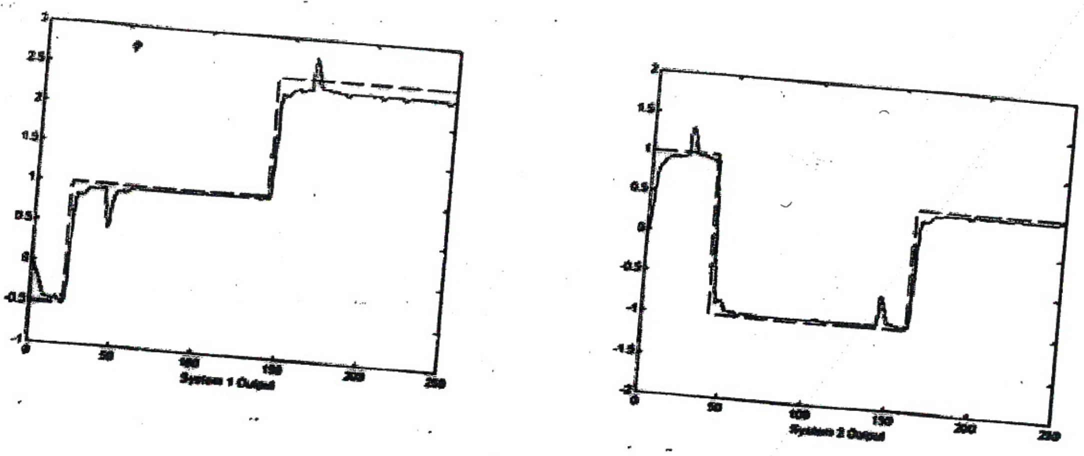
**(a) No Decentralized Control**



**(b) Fuzzy Predictor**



**(c) Neurofuzzy Predictor**



(d) Neural Predictor

**Figure 9. Simulation results of system (ii). (a) no decentralized control, (b) results with fuzzy predictor, (c) results with neuro-fuzzy predictor, (d) results with neural predictor**

## 8. Concluding Remorks

This paper has considered the class of interconnected large-scale systems and treated the decentralized control design problem under the m-step delay sharing information pattern assumption. The approach adopted is based on a combination of the model-based predictive control scheme and the prediction of the interaction variables via fuzzy, neuro-fuzzy and neural estimators. This combined scheme has improved adaptivity and robustness due to the properties of MBPC. The MBPC is capable of utilizing coarse predictions of the interconnection signals to enhance its performance. The comparison of the three models (fuzzy, neuro-fuzzy, neural) shows that neural and neuro-fuzzy models are more efficient in predicting the unknown interconnections signals. This is due to the better generalization that these models can achieve. On the other hand, the fuzzy model provides zero output (prediction) for interconnection signal sequences (past values) that lie in an input area not revisited.

On the practical side MBPC has found wide application in process industries such as refineries, pulp and paper, food, air and gas, polymers etc. Over 2200 MBPC applications are reported in [22] the majority of which (67%) concern the refinery section. Application of decentralized control can be found in [23, 24].

## REFERENCES

1. BAHNASAWI, A.A., AL-FUHAID, A.S. MAHMOUD, M.S., **Decentralized and Hierarchical Control of Interconnected Systems**, Proc. IEE, 1990, D-137, pp. 311-321.
2. JAMSHIDI, M., **Large Scale Systems: Modeling, Control and Fuzzy Logic**, Prentice Hall PTR, 1996.
3. TZAFESTAS, S.G., KYRIANNAKIS, E. KAPSIOTIS, G., **Decentralized Model Based Predictive Control of Large Scale Systems**, Proc. 3rd IEEE Mediterranean Symposium on New Directions in Control & Automation, Limassol, Cyprus, 1995.
4. LINNEMANN, A., **Decentralized Control of Dynamically Interconnected systems**, IEEE Trans. Autom. Control, 1984, AC-29, pp. 1052-1054.
5. SINGH, M.G. TITLI, A., **Systems Decomposition, Optimization and Control**, Pergamon: Oxford, 1978.
6. HO, Y.C., CHU, K.C., **Information Structure in Many-Person Optimization Problems**, Automatica *10*, 1974, pp. 149-160.
7. SANDELL, N.R., ATHANS, M., **Solution of Some Non-classical LQG Stochastic Decision Problems**, IEEE Trans. Autom. Control, 1974, AC-19, pp. 108-116.
8. YOSHIKAWA, T., KOBAYASHI, H., **Separation of Estimation and Control for Decentralized Stochastic Control Systems**, Automatica 14, 1978, pp. 623-628.

9. KURTARAM, B, SIVAN, R., **LQG Control with One-Step-Delay Sharing Pattern**, IEEE Trans., 1974, AC-19, pp. 571-574.

10. SINGH, M.G., **Decentralised Control**, North Holland: Amsterdam, 1981.

11. JAMSHIDI, M., **Large Scale Systems: Modeling and Control**, Elsevier/North-Holland, New York, 1983.

12. TZAFESTAS, S.G., WATANABE. K., **Stochastic Large-Scale Engineering Systems**, Marcel Dekker, New York, 1992.

13. RICHALET, J., **Model Based Predictive Control in the Context of Integrated Design**, Computer Integrated Design of Controlled Industrial Systems ( Richalet, J., Tzafestas, S. eds), Proc. CIM-Europe Workshop, Brussels, April 26-27, 1990.

14. DE KEYSER, R.M.C., **Model Based Predictive Control Toolbox**, Computer Integrated Design of Controlled Industrial Systems ( Richalet, J., Tzafestas, S. eds), Proc. CIM-Europe Workshop, Brussels, April 26-27, 1990.

15. CAMACHO, E.F, BORDONS, C., **Model Predictive Control in the Process Industry**, Springer, London/Berlin, 1994.

16. CLARKE D.W., **Generalized Predictive Control and its Application**, Computer Integrated Design of Controlled Industrial Systems ( Richalet, J., Tzafestas, S. eds), Proc. CIM-Europe Workshop, Brussels, April 26-27, 1990.

17. XU, X.M., XI, Y.G., ZHANG, Z.J., **Decentralized Predictive Control (DPC) of Large Scale Systems**, North-Holland, Information and Decision Technologies 14, 1988, pp. 307-322.

18. WANG, L.-X., **Adaptive Fuzzy Systems and Control**, Prentice Hall, Englewood Cliffs, NJ, 1994.

19. TZAFESTAS, S.G., PAPANIKOLOPOULOS, N.P., **Incremental Fuzzy Expert PID Control**, IEEE Trans. On Industrial Electronics, 1990, IE-37, No. 5, pp. 365-371.

20. TZAFESTAS, S.G., RIGATOS, G.G., **A Simple Robust Fuzzy Logic Controller of the Diagonal Type**, J. Intelligent and Robotic Systems, 1999, 26, Nos. 3-4, pp. 353-388.

21. TZAFESTAS, S.G. KYRIANNAKIS, E., RIGATOS, G.G. **Adaptive Model-Based Predictive and Neural Control of the Welding Process**, in: S.G. Tzafestas (ed.), **Methods and Applications of Intelligent Control**, Kluwer, Dordrecht / Boston, 1997, pp. 509-548.

22. ALLGÖWER F. BADGWELL, T.A., QIN, I.S., RAWLINGS, J.B., WRIGHT, S., **Nonlinear Predictive Control and Moving Horizon Estimation**, In : Advances in Control, P.M. Frank (ed.), Springer, Berlin/London, 1999.

23. SEZER, M.E., ŠILJAK, D.D., **Decentralized Control**, Ch. 49, The Control Handbook, William S. Levine (ed.), CRC Press / IEEE Press, 1996.

24. SINGH, M.G., TITLI, A. (eds), **Handbook of Large Scale Systems Engineering Applications**, North Holland, Amsterdam, 1977.

25. RUMELHART D. E., G.E. HINTON, R. J. WILLIAMS, **Learning Internal Representations by Error Propagation**, Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol.1, Cambridge, MA: MIT Press, 1989.