A New Scheme for Fuzzy Rule-Based System Identification and Its Application to Self-Tuning Fuzzy Controllers

Kuhu Pal, Rajani K. Mudi, and Nikhil R. Pal, *Senior Member, IEEE*

***Abstract*—There are many important issues that need to be resolved for identification of a fuzzy rule-based system using clustering. We address three such important issues: 1) deciding on the proper domain(s) of clustering; 2) deciding on the number of rules; and 3) getting an initial estimate of parameters of the fuzzy systems. We justify that one should start with separate clustering of X (input) and Y (output). We propose a scheme to establish correspondence between the clusters obtained in X and Y. The correspondence dictates whether further splitting/merging of clusters is needed or not. If X and Y do not exhibit strong cluster substructures, then again clustering of X^* (input data augmented by the output data) exploiting the results of separate clustering of X and Y, and of the correspondence scheme is recommended. We justify that usual cluster validity indices are not suitable for finding the number of rules, and the proposed scheme does not use any cluster validity index. Three methods are suggested to get the initial estimate of membership functions (MFs). The proposed scheme is used to identify the rule base needed to realize a self-tuning fuzzy PI-type controller and its performance is found to be quite satisfactory.**

***Index Terms*—Fuzzy clustering, fuzzy rule extraction, self-tuning controller, system identification.**

I. Introduction

GIVEN A SET of input–output data, there have been several attempts to identify a rule-based system to characterize the relation between the input and output by various clustering methods [3]–[12]. Let the set of p-dimensional input vectors be $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subset \Re^p$ and the associated set of q-dimensional output vectors be $Y = \{\mathbf{y}_1, \ldots, \mathbf{y}_n\} \subset \Re^q$. The set with input–output vectors taken together will be denoted by $X^* = \{\mathbf{x}_i^* = \binom{\mathbf{x}_i \in \Re^p}{\mathbf{y}_i \in \Re^q} \in \Re^{p+q}, i = 1, \ldots, n\}$. Sugeno and Yasukawa [3] used the fuzzy c-means (FCM) algorithm [2] and clustered Y. The membership values of the input clusters were obtained by projecting the membership values of the extracted clusters on the input axes. They approximated these clusters by trapezoidal fuzzy sets and used a heuristic method to adjust the parameters of the trapezoidal membership functions (MFs).

Manuscript received September 16, 1998; revised December 8, 1999, March 26, 2001, and January 26, 2002. This work was supported in part by DST, India, under Grant III.5(21)/96-ET and by CSIR, India. This paper was recommended by Associate Editor W. Pedrycz.

K. Pal is with Pragati Nagar, Chinsurah, Hooghly 712102, India.

R. K. Mudi is with the Department of Instrumentation Engineering, Jadavpur University, Calcutta 700 098, India.

N. R. Pal is with the Electronics and Communication Sciences Unit, Indian Statistical Institute, Calcutta 700 035, India (e-mail: nikhil@isical.ac.in).

Publisher Item Identifier S 1083-4419(02)04375-3.

Yoshinari *et al.* [4] cluster X^* using fuzzy c-linear varieties [2] where each cluster is interpreted as a multidimensional local linear relationship. After clustering, given an input vector $\mathbf{x}_i \in \Re^p$, the predicted output $\mathbf{y}_i \in \Re^q$ from the model is obtained using a two-step procedure. Sin and deFigueiredo [15] used FCM for clustering and suggested to use the Xie–Beni index [16] for selecting the number of clusters. Each cluster obtained from X^* is represented by a Takagi–Sugeno (TS) type rule [25]. Yager and Filev [5] used the mountain clustering method (MCM) on X^k. The optimal number of clusters is chosen based on a user-defined threshold on the mountain potential. They used the height method [17] of defuzzification with the Mamdani–Assilian (MA) model [24]. Nakamori and Royke [18] classified the variables involved in rule extraction into three categories: 1) objective variables or output variables; 2) explanatory variables or the variables used in the consequent functions of the TS model; and 3) conditional variables or the variables used in defining the premises. For clustering, they used the objective variables and a subset of explanatory and conditional variables chosen carefully. The authors used a hyperellipsoidal crisp clustering algorithm, which dynamically determines the number of clusters based on several user-defined parameters. Each cluster is then translated into a TS rule with linear function for the consequents, and if the TS model is not satisfactory, they use the MA model.

Babuska and Kaynak [8] cluster X^* using Gustafson–Kesel's [20] fuzzy c-means (GKFCM) algorithm with a large value of c (the number of clusters) for TS modeling. Then the compatible cluster merging criteria [21] is used to merge compatible clusters. After this, GKFCM is again run with the reduced number of clusters. The process is repeated until no more clusters can be merged. The fuzzy partition thus obtained is used to generate MFs for the antecedent variables. Finally, the consequent parameters are estimated using the least square technique. Kaynak *et al.* [9] and Babuska *et al.* [22] discuss methods for system optimization through removal and merging of fuzzy sets extracted by clustering of X^* using a measure of similarity between fuzzy sets. Runkler and Palm [11] defined a regular fuzzy system as a fuzzy system with complete rule base defined with equispaced unimodal MFs and the first-order TS consequents. Here a consequent represents a known physical model for the system and its parameters are identified clustering X^* using the regular fuzzy c-elliptotypes (FCE) algorithm [2], [11]. Delgado *et al.* [12] presented several methods for fuzzy modeling that use clustering. The first method clusters X^k with FCM. Each

cluster is transformed into a rule of the form R_k: *If* $\boldsymbol{x}$ *is* A_k *then* $\boldsymbol{y}$ *is* B_k; $k = 1, 2, \ldots, c$. The membership values of fuzzy sets A_k and B_k are defined by the FCM membership formula [2] with $\mathbf{v}_i = \mathbf{v}_k^x$ and $\mathbf{v}_i = \mathbf{v}_k^y$, respectively where $\mathbf{v}_k^* = (\mathbf{v}_k^x \mathbf{v}_k^y)^T$ is the centroid of the kth cluster in X^* and $\mathbf{v}_i$ is the centroid of the ith cluster used in the FCM membership formula. For the TS model, the kth rule takes the form R_k: *If* $\boldsymbol{x}$ *is* A_k *then* $\boldsymbol{y}$ *is* $\mathbf{v}_i^y$. Sin and deFigueiredo [15] also used the same multidimensional FCM MFs for computing the firing strength, but they used the FCM formula defined using $\mathbf{v}_k^* \in \Re^{p+q}$. It requires to get an approximate value of the output $\boldsymbol{y}$ before the firing strength of any rule can be computed. These approximate values (different for different rules) are obtained from the consequent functions. The rules are then written using information produced by both clustering of X^* and X. Delgado *et al.* [12] also proposed to cluster X and Y separately to generate fuzzy sets $A_1, \ldots, A_{c_1}$ for antecedents and $B_1, \ldots, B_{c_2}$ for the consequents. This results in $c_1 \times c_2$ rules of the form *If* $\boldsymbol{x}$ *is* A_i *then* $\boldsymbol{y}$ *is* B_j. A certainty value w_{ij} is associated with each rule. This scheme may result in more rules than required. In [12] several cluster validity indices [13], [16], [23] are used to get a good choice for the number of clusters (rules). Finally, they optimized the system with respect to rms error using genetic algorithms (GA).

The preceding discussion shows that different researchers have used different domains of clustering, different clustering algorithms, and different cluster validity indices to decide on the number of rules. Our search through the literature revealed that there are several issues which need to be addressed before a clustering-based rule extraction schemes can be effectively used. Here, we address only three important issues: *where to cluster*, *how to decide on the number of rules* (note that we do not say number of clusters but the number of rules), and *how to get an initial estimate of* MFs. We justify that usual cluster validity indices may not be useful (preferably be avoided) to decide on the number of rules. We also justify that separate clustering of X and Y is preferable when X and Y exhibit distinct cluster substructures; otherwise, separate clustering of X and Y followed by clustering in X^* with the information gathered in the preceding step (separate clustering of X and Y) would be required. The separate clustering of X and Y raises another problem, namely, how to establish a correspondence between clusters found in X and Y. We provide a solution to this problem and hence, the problem of deciding on the number of rules. Three different schemes have been suggested to get the initial estimate of the peak (the point at which the membership value is 1) of a triangular MF. Finally, the proposed methodology is illustrated to identify the rules required for updating the multiplicative gain factor (α) of a self-tuning fuzzy PI controller (STFPIC) [26]. In this regard, two different strategies have been used to generate the data. Our simulation exercise with several processes showed quite satisfactory performance of the proposed schemes. We emphasize, although our scheme is demonstrated for identification of a rule base required for tuning of the output scaling factor (SF) of STFPIC [26], the proposed methodology is quite general in nature and may be successfully applied to other system identification problems.

II. Issues Relating to System Identification

Following Pal *et al.* [1], we enumerate the issues first and then we propose solutions to a few of them, and finally use them to extract rules for the gain adjustment of STFPIC [26]. There are at least six questions, as raised in [1], that must be answered to fully exploit the features of exploratory data analysis for rule extraction. We list these questions next.

Given the data set X^*, can we just use some clustering algorithm? Or, first of all, do we need to find whether cluster substructures are present in the data or not. If there is no cluster structure, can or should we still use clustering?

The existence of cluster substructures in X or Y or X^* can be assured using a cluster tendency assessment technique [19]. We do not pursue this issue further in this investigation. If we decide to cluster, the next question is *what is the proper domain for clustering? Is it enough to cluster in only one of the three possible domains, or do we need more than one domain?*

When $\mathbf{v}_i^*$ is associated with a good cluster, it is taken as a signal that if $\|\mathbf{x}_k - \mathbf{v}_i^x\|$ is small, $\|\mathbf{y}_k - \mathbf{v}_i^y\|$ is also small. This roughly indicates that such a cluster represents a locally continuous or even smooth input–output relation. In this case the ith cluster is translated into a fuzzy rule of the form

MA models [24]: *If* $\mathbf{x}$ *is CLOSE to* $\mathbf{v}_i^x$ *then* $\mathbf{y}$ *is CLOSE to* $\mathbf{v}_i^y$;
TS models [25]: *If* $\mathbf{x}$ *is CLOSE to* $\mathbf{v}_i^x$ *then* $\mathbf{y} = \mathbf{u}_i(\mathbf{x})$.

Usually the antecedent part, *If* $\mathbf{x}$ *is CLOSE to* $\mathbf{v}_i^x$, is written as a conjunction of p atomic clauses: *If* x_1 *is CLOSE to* v_{i1}^x and *if* x_2 *is CLOSE to* $v_{i2}^x \ldots$ and *if* x_p *is CLOSE to* v_{ip}^x. The function $\mathbf{u}_i(\mathbf{x})$ in the TS case primarily models the behavior of the input–output relation in the neighborhood of $\mathbf{v}_i^*$. When clustering is performed in X alone, there are several methods for partitioning the output data. For example, if a crisp clustering algorithm is used, points in Y associated to a cluster in X are expected to form a cluster in $\Re^q$ (assuming a smooth relation between X and Y). This association enables us to write rules using the centroids $\{\mathbf{v}_i^y\}$ of the crisp clusters in the output domain. In a similar manner rules can be generated when Y is clustered, and the centers $\{\mathbf{v}_i^x\}$ are obtained as centroids of the associated crisp clusters in X. When a fuzzy clustering algorithm is used one can simply transfer each membership from clusters in X (or Y) to the corresponding points in the other domain Y (or X).

The third important question that needs to be answered is the choice of a clustering model: *What clustering model should be used for defining the initial structure?*

A good answer to this question depends on the input–output data and the information we want to use in subsequent steps of system identification (SI). If membership values from the clustering algorithm are required to estimate the MFs involved in the rules, we need a fuzzy clustering algorithm. If we expect the data to exhibit, say, hyperspherical structures then a model like the FCM may be a good choice. On the other hand, if the data possess ellipsoidal structures (including linear structure as a special limiting case), then FCE [11] may be appropriate. For multidimensional data, it is difficult to guess the type of clusters that the data might have in advance—this makes the problem more difficult.

Choosing the right number of clusters (i.e., number of rules), $c, 2 \leq c \leq n$, is the next major issue faced by system identifiers. Thus, *what validity indices are useful for determining the best number of rules? Should one or a vote of several be used? Can assumptions about the data, if available, be used to guide the selection of a validity index or some new validity index is required?*

The conventional cluster validity indices that are designed to assess goodness of clusters extracted from a data set may not be useful for deciding on the desired number of rules. We shall elaborate this issue later and propose some schemes to solve the problem.

Once we find the "optimal" number of clusters and an associated c-partition of the data, the next important issue is *How to estimate the parameters of the MFs and consequent functions. How should we transfer the MF associated with a p-dimensional vector to its one-dimensional components; should we use LSE estimate or some other schemes?*

Whether the clusters are found in X, Y, or X^*, their centroids and membership values can be projected onto the domain of each input/output variable. If a crisp clustering algorithm such as the MCM [5]–[7] is used, then MF (mi_{ik}, the kth fuzzy set on the ith input feature) for CLOSE to v_{ik}^p must be defined about the projected centers. If a fuzzy clustering algorithm is used, then estimates of the MFs $\{mi_{ik}\}$, and also of $\{mo_{ik}\}$ where mo_{ik} is the kth fuzzy set on the ith output feature (for the MA model), can be obtained by several methods using the clustering results. One may also assume that all MFs belong to some particular family such as Gaussian, triangular, or trapezoidal. Use of fuzzy clustering algorithms often helps to guess the desired shapes of MFs. Once parametric forms are defined, their parameters can be estimated using techniques such as gradient descent, LSE, etc. For the TS model, the functional form for the $\{u_i\}$ is assumed and its parameters are often estimated using LSE, gradient descent, or GA.

Cluster-based SI should produce good rules. However, each of the five issues raised here has no definitive answer, so any system produced by this approach can be inadequate for many reasons. Therefore, system validation is a very important step in SI and this becomes the next issue: *How should the system be tested? Is it enough to consider the square error? How do we measure the robustness of the system? How do we assess the generalization ability of the identified system?*

III. The Proposed Scheme

If a cluster tendency assessment technique signals existence of good substructure in the data, then it may be easier to find an "optimal" number of rules. However, irrespective of whether the input–output data have cluster substructure or not, it is always possible to partition it into a number of hyperspherical subsets and each such subset can be converted into a rule. If the data indeed have hyperspherical clusters, then the number of rules (subsets) would be smaller compared to the case when the data do not have any cluster substructure. For example, if the input–output relation is linear, the data will not exhibit any cluster structure, yet it can be partitioned into a number of small hyperspherical clusters to generate a set of rules to identify such linear systems. Clustering algorithms that seek hyperspherical clusters like the FCM can make such partitions easily. Hence, in this investigation, we do not check for cluster tendency and use the FCM algorithm for rule extraction.

A. Deciding on the Clustering Domains

There are five possibilities:

1) clustering of X and then imposing the cluster structure obtained in X to Y;
2) clustering of Y and imposing the cluster structure in Y to X;
3) clustering of X^*;
4) clustering separately X and Y and then establishing a correspondence between the clusters in X and Y;
5) any combination of two or three of the previous cases.

If we cluster X and transfer that structure to Y, we may not get the actual substructure present in Y. Similarly, if we cluster Y only, then we may not extract the actual substructures present in X. On the other hand, if we cluster X^*, we may fail to extract the substructures present in either of X and Y. Let us elaborate this.

Suppose each feature in Y lies in [0, 1] and for X, each feature value lies in say [200, 500]. In such a case, the distance between a pair of vectors in X^* say $d(\mathbf{x}_i^*, \mathbf{x}_j^*)$ will primarily be governed by $d(\mathbf{x}_i, \mathbf{x}_j)$ where $\mathbf{x}_i^* \in \Re^{p+q}$ and $\mathbf{x}_i \in \Re^p$. Thus the clustering algorithm will primarily extract the substructures present in X. Similarly, we can think of data sets where the clustering algorithm will essentially extract the clusters in Y although the algorithm runs on X^*. Note that global normalization of X^* may not solve this problem, as the relative contribution of the x part and the y part will remain the same. Moreover, we should not normalize each feature separately, as then we will lose the actual structure (the shape of the clusters) present in the data.

We emphasize here that existence of very clear cluster substructures in one of the three possible domains X, Y, or X^* does not necessarily imply that the system is identifiable by exploratory data analysis. Use of clustering must be done cautiously. Consider the scatter diagram of a data set in Fig. 1(a). We assume that both input and output are one-dimensional and Fig. 1(a) gives the scatter plot of X^*. X^* has four distinct clusters. If we cluster X, we will lose two of the clusters, while clustering only Y will also result in loss of two other clusters. This suggests that for this data set, clustering of X^* would possibly be the right choice. But this again, although it appears better than the previous two cases, is not quite satisfactory. We will get four clusters, but when membership values of clusters 1 and 2 are projected on, say, the y axis, we will get two MFs over almost the same domain of the y axis, and projecting clusters 3 and 4 we again get two MFs practically over the same area of the y axis. Similar is the case for projection on the x axis. Considering X and Y separately, it is clear that two membership functions for each of X and Y would be enough. For the given example, it thus appears that neither clustering of X, Y, nor X^* only can produce the desirable results, although the best cluster substructures can be extracted clustering X^*. One can argue that such data sets are not very relevant for our purpose

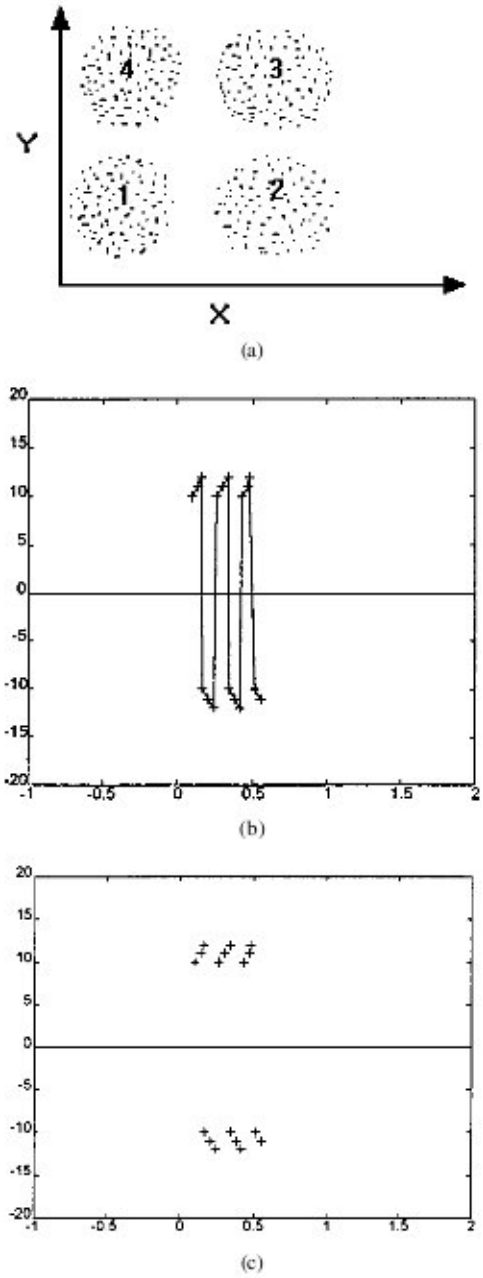


Fig. 1. (a) Scatter plot of an input–output data set. (b) The system has been sampled at the points marked with+. (c) Scatter plot of the sampled points.

because for such data sets, points from the same area are mapped to two widely separated places suggesting lack of continuity of the function and possibly no rule-based system will be able to identify such a system completely. This artificial data set has been created to emphasize the point that the presence of good clusters does not necessarily mean that the underlying system can be identified. However, such data sets are possible even for identifiable systems. For example, consider the single-input single-output system in Fig. 1(b). An almost uniform sampling of the input x can pick up the points marked by —. Fig. 1(c) shows the scatter plot of the sampled points. Clearly, all arguments given for Fig. 1(a) are equally applicable to Fig. 1(c). In this case, although there is a system, the training data are such that the extracted fuzzy rules will appear inconsistent. Thus, the existence of nice cluster structure does not necessarily mean that translation of the clusters into rules will result in a rule base to model the system. One possible way to screen out such data sets is to check for the consistency of the extracted rules. We do not consider this issue further and in this investigation we assume that the data sets under consideration are such that fuzzy rules can model the underlying system.

Based on the preceding discussion, it appears that the best choice left is to cluster X and Y separately and then establish a correspondence between the clusters. Separate clustering of X and Y is logical also because in this case the substructures present in both X and Y will be extracted. The correspondence between the clusters obtained from the two domains may not (and usually will not) be one to one. A particular input cluster may correspond to more than one output cluster and vice versa. However, we shall elaborate later that for some cases after separate clustering of X and Y, clustering of X^e using the information obtained from separate clustering of X and Y would be necessary.

B. Extraction of Rules

We decided to cluster X and Y separately. But how many clusters to look for? Unlike most attempts, we do not use conventional cluster validity index as it is *not* appropriate here. We illustrate this with an example. Consider a data set that has three reasonably separated clusters and the volume of each cluster is quite large. In this case, any cluster validity index will suggest $c = 3$, but with only three rules, it is almost impossible to describe the behavior of the data with reasonable accuracy. First, since three clusters are quite separated, only one rule will be used to describe the data points in a cluster, but one rule may not be adequate to model the variation of the data within a cluster. For the sake of argument, one may stretch the MFs so that all three rules are fired for every input point. But in that case, the identified system would be too coarse to be of any use, because the clusters are well separated and each cluster has significant variation within itself. We now present the proposed scheme.

Let A be any clustering algorithm (here, A = FCM) that we use to cluster separately both X and Y. Let $c1$ and $c2$ be the number of clusters in X and Y, respectively. $c1$ and $c2$ can be heuristically chosen based on the expected number of rules. Note that this expectation does not have to be accurate. In fact, one can safely assume $c1 = c2 = 10$ for almost all applications. The reason is that with $c1 = c2 = 10$ we can have, if dictated by the data, $10 \times 10 = 100$ rules, which should be more than enough for most of the system identification task. Thus the guideline may be, if there is no reason to expect more than 100 rules, one can safely start with $c1 = c2 = 10$. The proposed scheme is such that it will not matter much if $c1$ (or $c2$) is a little more or less than the ideal number of clusters required for X and Y. Let $u_{ik;X}$ be the membership value of $\mathbf{x}_k \in X \subset \Re^p$ to cluster $i, i = 1, \ldots, c1$ and $u_{ik;Y}$ be the membership value of $\mathbf{y}_k \in Y \subset \Re^q$ to cluster $i, i = 1, \ldots, c2$. If a hard clustering algorithm is used, it is easy to identify the physical clusters. Let the hard clusters obtained from X be denoted by $X_1, X_2, \ldots, X_{c1}$ and the clusters from Y be denoted

TABLE I
MATRIX C

	Y_1	Y_2	Y_3	Y_4	Y_5
X_1	50	3			
X_2		30	45	3	
X_3	5			55	
X_4		4			33
X_5	7			2	51

by $Y_1, Y_2, \ldots, Y_{c2}$. If a fuzzy clustering algorithm, such as the FCM, is used then we can generate the hard clusters as follows:

$$X_i = \{\mathbf{x}_k / \mathbf{x}_k \in X \subset \Re^p, u_{ik;X} = \max_j \{u_{jk;X}\}\},$$
$$i = 1, \ldots, c1$$

and

$$Y_i = \{\mathbf{y}_k / \mathbf{y}_k \in Y \subset \Re^q, u_{ik,Y} = \max_j \{u_{jk,Y}\}\},$$
$$i = 1, \ldots, c2$$

We now generate a correspondence table $C = [C_{lm}]_{c1 \times c2}$ of size $c1 \times c2$ such that C_{lm} gives the number of data points $\mathbf{x}_k^* = (\mathbf{x}_k\ \mathbf{y}_k)^T$ for which $\mathbf{x}_k \subset X_l$ and $\mathbf{y}_k \subset Y_m$. *Algorithm Generate_ Correspondence_Matrix* gives the computational steps.

Algorithm ***Generate_Correspondence_Matrix***
Begin
$C_{ij} = 0\ \forall i = 1, 2, \ldots, c1;\ j = 1, 2, \ldots, c2$
for $k = 1$ to n
$l = \arg\max_i \{u_{ik;X}\};\ m = \arg\max_j \{u_{jk;Y}\}$
$C_{lm} = C_{lm} + 1$
endfor
EndAlgorithm

Thus, the (l, m)th entry of the table C gives the number of occurrences for which a data point in cluster X_l of X has its associated output data point in cluster Y_m of Y. Therefore, if $c1 = c2 =$ the right number of clusters and if the clustering algorithm has extracted the correct clusters, then in C we may have exactly one nonzero entry in each row and column indicating the correspondence. This is a very ideal situation and is not likely to happen in practice. In reality, we can have more complex situations that is illustrated with an example. Let $c1 = 5, c2 = 5$. Table I shows the matrix C.

From Table I, it is clear that cluster X_1 is strongly related to only cluster Y_1 because neither the row X_1 nor the column Y_1 has any other significant frequency. The three points of X_1 that go to Y_2, and the five points from X_3 and seven points from X_5 that go to Y_1 can be ignored compared to 50. For X_2, the situation is a bit more complex. The input cluster X_2 exhibits a strong relation with two output clusters Y_2 and Y_3. Again, the three points from X_2 that correspond to Y_4 can be ignored—possibly they correspond to some overlapped regions. X_2 indicates that X has been under clustered, i.e., $c1$ should have been more than 5. So we have to split X_2 to two subclusters, say, X_{21} and X_{22}, so that X_{21} corresponds to Y_2 and X_{22} corresponds to Y_3. X_{21} and X_{22} usually will

TABLE II
MATRIX C IN TABLE I AFTER SPLITTING OF ROWS AND COLUMNS

	Y_1	Y_2	Y_3	Y_4	Y_{51}	Y_{52}
X_1	50	3				
X_{21}		30		3		
X_{22}			45			
X_3	5			55		
X_4		4			33	
X_5	7			2		51

correspond to two adjacent regions in X that are mapped to two different areas in Y. This can happen in many situations including relations like a step function. X_3 is clearly related to Y_4. X_4 is strongly related to Y_5, hence one would be tempted to assume that (X_4, Y_5) corresponds to a good rule. This is not true. Note that, the column for Y_5 has another very high frequency corresponding to input cluster X_5. Thus two input clusters X_4 and X_5 both correspond to the same output cluster Y_5. There could be two possible reasons. First, Y might have been under clustered; the value of $c2$ should have been $c2 = 6$. Hence, we should split cluster Y_5 into Y_{51} and Y_{52} with 33 and 55 points, respectively. Second, it is also possible that two different regions of input space are mapped to the same output area. In this case, splitting of Y_5 is not recommended because Y_{51} and Y_{52} are too close to form two different clusters. To distinguish between the two cases we proceed as follows. Let the centroids of Y_{51} and Y_{52} be $\mathbf{v}_{51}^y$ and $\mathbf{v}_{52}^y$, respectively. If $\|\mathbf{v}_{51}^y - \mathbf{v}_{52}^y\| < \varepsilon$, then splitting of the output cluster Y_5 is not needed. Here ε is a small positive quantity which can be taken as 10% of the average width of a MF. For example, if we have, say q MFs defined on the domain of length L then $\varepsilon = (L/q) \times 0.1$. If we split X_2 and Y_5 we get a new correspondence table, as shown Table II. In Table II, if we ignore the smaller frequencies, the matrix C attains the ideal situation where there is a one-to-one correspondence between clusters in X and Y. Table II results in six rules based on the following cluster pairs: $(X_1, Y_1), (X_{21}, Y_2), (X_{22}, Y_3), (X_3, Y_4), (X_4, Y_{51})$, and (X_5, Y_{52}).

In order to implement this scheme, we need to choose a threshold θ on the frequencies. If $C_{ij} > \theta$, then only we assume the association between X_i and Y_j as strong enough to form a rule. A schematic description of the algorithm for extracting the clusters forming the rules is given next.

Algorithm ***Cluster_Forming_Rules***
Begin
If $(C_{ij} < \theta)$ then $C_{ij} = 0\ \forall i = 1, 2, \ldots, c1;\ j = 1, 2, \ldots, c2$
for $i = 1$ to $c1$
for $j = 1$ to $c2$
If $\exists$ only one nonzero entry in the ith row and jth column
then the cluster pair (X_i, Y_j) forms a rule with $\mathbf{v}_i^x$ and
$\mathbf{v}_j^y$ as the centroids for defining MFs;
Else If the ith row has n_i nonzero entries then split X_i into
n_i sub-clusters $X_{i1}, X_{i2}, \ldots, X_{in_i}$; increase $c1$ by $(n_i - 1)$
and break loop on j;
Else If the jth column has n_j nonzero entries then
temporarily split Y_j to n_j sub-clusters $Y_{j1}, Y_{j2}, \ldots, Y_{jn_j}$;

increase $c2$ by $(n_j - 1)$
compute $\mathbf{v}_{jl}^y, l = 1, 2, \ldots, n_y$
compute $d_{lm} = \|\mathbf{v}_{jl}^y - \mathbf{v}_{jm}^y\| \; \forall l, m = 1, 2, \ldots, n_j, l \neq m$
find $d_{l'm'} = \min_{l,m,l \neq m}\{d_{lm}\}$
If $d_{l'm'} < \varepsilon$ then replace $Y_{jl'}$ and $Y_{jm'}$ by $Y_{lm} = Y_{jm'} \cup Y_{jl'}$;
accordingly adjust $c2$
The process is repeated till no pair of subclusters is left
with $d_{l'm'} < \varepsilon$ and loop on i is broken;
endfor
endfor
Every nonzero entry of C is then translated into a rule.

EndAlgorithm

C. Formation of Membership Functions

The next task is to find an initial estimate of the centroids of each cluster. If there is no splitting of clusters, then for every $C_{ij} > \theta$, we use $\mathbf{v}_i^x$ and $\mathbf{v}_j^y$ to define the peaks of the initial triangular MFs. But when clusters are split, we recompute the centroids as follows. Let P be the set of points corresponding to the (i, j)th entry of C. We compute

$$\bar{\mathbf{v}}_i^x = \left(\sum_{\mathbf{x}_j \in P} \mathbf{x}_j\right) \Big/ |P|$$

and

$$\bar{\mathbf{v}}_j^y = \left(\sum_{\mathbf{y}_j \in P} \mathbf{y}_j\right) \Big/ |P|. \tag{1}$$

When a fuzzy clustering algorithm is used, we can compute $\mathbf{v}_i^x$ and $\bar{\mathbf{v}}_j^y$ as

$$\bar{\mathbf{v}}_i^x = \left(\sum_{\mathbf{x}_j \in P} (u_{ij})^m \mathbf{x}_j\right) \Big/ \sum_{\mathbf{x}_j \in P} (u_{ij})^m$$

and

$$\bar{\mathbf{v}}_j^y = \left(\sum_{\mathbf{y}_j \in P} (u_{ij})^m \mathbf{y}_j\right) \Big/ \sum_{\mathbf{y}_j \in P} (u_{ij})^m, \tag{2}$$

Here, u_{ij} is the membership of $\mathbf{x}_j$ (or of $\mathbf{y}_j$, as the case may be).

We call this scheme *Strategy* 1. When some clusters are split, one might think that a better strategy would be to recluster X and Y separately with $\mathbf{v}_i^x$ and $\mathbf{v}_i^y$ for further refinement of the cluster centroids. This is neither required, as the initial MFs will be tuned, nor it is desirable, because if we recluster X and Y, the same correspondence may not be preserved, and we should again generate the correspondence table C. The process may have to be repeated many times, and even then we may not get the same C in two successive iterations. Consequently, we do not cluster X and Y again. We recommend to use *Strategy* 1 when there are not many entries in C which are less than θ. In such cases, X and Y have distinct clusters and the algorithm has been able to extract that. Even if the underlying input–output relation is not smooth or has abruptness, the identified rule-based system should be adequate.

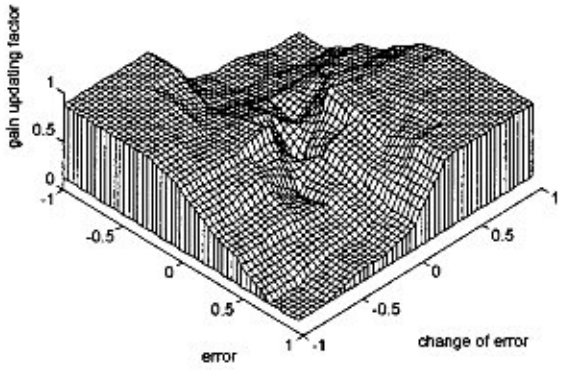


Fig. 2. Variation of α as a function of e and Δe.

There could be another variant in which we recluster X^* with $\mathbf{v}_i^* = (\mathbf{v}_j^x \mathbf{v}_k^y)^T; i = 1, \ldots, c$ as the initial centroid, where c is the number of input–output associations identified and $\bar{\mathbf{v}}_j^x$ and $\bar{\mathbf{v}}_k^y$ are the centroids of the clusters that are associated and computed either using (1) or (2). There are a few reasons for this. The first one is the use of a threshold θ to extract the clusters (to determine the rules). Sometimes, the data points that are ignored by the threshold θ may be important. We ignored these points to compute $\bar{\mathbf{v}}_i^x$ and $\bar{\mathbf{v}}_j^y$. As a result, the initial structure of the system may not be adequate to produce a smooth response; in fact, the extracted rules may not be complete even after tuning. There are two factors to consider: the smoothness (absence of abruptness) of the relation that we are trying to identify and the existence of clusters in the data. To make it clear, consider the variation of α as a function of error (e) and change of error (Δe) shown in Fig. 2. (Fig. 2 will be elaborated later.) α is a multiplicative factor used for online updating of the output SF of a fuzzy controller [26]. It has abrupt changes, i.e., the relation is not quite smooth. Now if we generate the training data based on uniform sampling of e and Δe, then X will not exhibit any cluster substructure, but if the data are generated running the process for different initial conditions then X will exhibit cluster structures. For example, a lot of points will be generated around the set point, $e \approx 0$ and $\Delta e \approx 0$ resulting in some cluster substructure in X. So for the same system, in the former case X will not have distinct cluster substructures (this does not necessarily mean that *Strategy* 1 will not work; in fact, we shall show that in the current case, it does a good job) while in the latter case X exhibits cluster substructure. The SI scheme should be good enough to take care of both types of data. Therefore, when X and Y do not have distinct cluster structures (which may be due to improper sampling) or the underlying relation has abruptness, we recommend clustering again (reclustering) of X^* initializing the clustering algorithm with $\mathbf{v}_i^* = (\mathbf{v}_j^x \mathbf{v}_k^y)^T; i = 1, \ldots, c$ where c is the number of cells with $C_{ij} > \theta$. Here, the clusters are expected to grow around $\mathbf{v}_i^*$ taking into account the interaction between X and Y. This, reclustering of X^* will help to maintain the smoothness of the input–output relation in the extracted rules. We call this scheme *Strategy* 2.

In the present investigation, we propose three *types* of symmetric triangular initial MFs with equal base for both the antecedent and consequent part of the rule, i.e., e, Δe and α. The extracted fuzzy model consists of rules like R_k: *if* e is $\mu_k(e)$ and Δe is $\mu_k(\Delta e)$ *then* α is $\mu_k(\alpha)$. Since we are using the *height*

method of defuzzification [17], it is enough to consider the peak r of the consequent MF $\rho_k(\alpha)$. Peaks of the triangular MFs for antecedent and consequent of the ith rule are computed as follows.

Type I: Weighted average of x_{kj}'s with $u_{ik}(\mathbf{x}_k^*)$ as the weight for x_{kj}, $\forall k = 1, 2, \ldots n$. We denote the generated MFs as $\mathrm{MF}_{\mathrm{wav}\,1}$.

Type II: Weighted average of x_{kj}'s with $u_{ik}(\mathbf{x}_k^*)$ as the weight for x_{kj}, $\forall k = 1, 2, \ldots n$, such that $u_{ik}(\mathbf{x}_k^* \geq 0.25$. Here, the points which do not have adequate support for the cluster i are ignored for computing the peak value. We call such MFs $\mathrm{MF}_{\mathrm{wav}\,2}$.

Type III: Cluster centroids as extracted by the FCM algorithm. This type is referred as $\mathrm{MF}_{\mathrm{cen}}$.

After we obtain the peak of different linguistic values, the next problem is to get an initial estimate of the base width of each MF, so that the rule base becomes complete (i.e., for every $e, \Delta e$ pair at least one rule is fired). The first necessary step toward achieving this is to ensure that the set of MFs corresponding to every linguistic value covers its respective domain. The choice of the base width thus depends on the number of MFs for a particular linguistic variable. Since our domain is -1 to $+1$, and all MFs are symmetric triangles with equal base width, the two MFs at the extreme ends should have a reasonably high membership grade at -1 and $+1$. This means the effective domain for the MFs should be a little beyond -1 and $+1$. We also assume that neighboring MFs will have a significant overlap. Taking into account all these, we compute base width of each MF. We emphasize here that this is just an initial guess for the base width of each MF. This choice is not at all critical because finally we tune them using *gradient descent*. Of course, a good initial guess will take less computation time while tuning. This will ensure coverage of the entire domain of each input variable i.e., e and Δe, but it does not guarantee completeness of the rule base. To enhance our confidence about the completeness of the rule base, we proceed as follows.

Each domain of e and Δe is uniformly quantized within the range $[-1, 1]$. Then for every quantized pair $\{e, \Delta e\}$, the firing strength of each rule is calculated. Existence of at least one rule with nonzero firing strength for every pair of $e, \Delta e$ ensures completeness of the rule base with respect to the generated data. If the resolution of quantization is sufficiently high, then such a condition will ensure the completeness of the rule base with respect to the operating domain of the input variables. If the condition of completeness is not satisfied, then the base widths of all MFs are increased by a small percentage (10%) of its previous value. Again, the rule completeness is checked with the new base width and this iterative process is continued until completeness is achieved. In this way, we transform the rule base to a complete one, which is then refined using *gradient descent*.

IV. Implementation and Results

The proposed methodology is used for identification of a self-tuning mechanism of a fuzzy PI controller proposed in [26]. For the sake of completeness, we provide a brief description of the self-tuning mechanism.

A. Self-Tuning Fuzzy PI Controller (STFPIC)

The simplified block diagram of the STFPIC [26] is shown in Fig. 3(a). The output SF of the controller is modified by a self-tuning mechanism, which is shown by the dotted boundary. All MFs for controller inputs (i.e., e and Δe) and incremental change in controller output (i.e., Δu) are defined on the common normalized domain $[-1, 1]$, whereas the MF for α is defined on the normalized domain [0, 1]. In [26], except for the trapezoidal MFs at the two extreme ends, we used isosceles triangles with equal base width, as shown in Fig. 3(b) and (c). However, like the triangular partition of Sudkamp and Hammell [28], triangular MFs can also be used at the two extreme ends. Both of these choices impose a fuzzy partition on the respective domain. The relationships between the SFs and the input and output variables of the STFPIC are

$$e_{\mathcal{N}} = G_e \cdot e, \quad \Delta_{e_{\mathcal{N}}} = G_{\Delta e} \cdot \Delta e, \quad \Delta u = (\alpha \cdot G_u) \cdot \Delta u_{\mathcal{N}}. \tag{3}$$

Here, $G_e, G_{\Delta e}$, and G_u are the SFs for $e, \Delta e$, and Δu, respectively, and $e_{\mathcal{N}}, \Delta e_{\mathcal{N}}$, and $\Delta u_{\mathcal{N}}$ are normalized quantities. Unlike fuzzy PI controllers (FPIC), which use only G_u, the actual output (Δu) for STFPIC is obtained by using the effective SF $(\alpha \cdot G_u)$, as shown in Fig. 3(a). In [26], we proposed to compute α on-line using a model independent fuzzy rule base defined on e and Δe. The operation of a PI-type FLC can be described by

$$u(k) = u(k-1) + \Delta u(k). \tag{4}$$

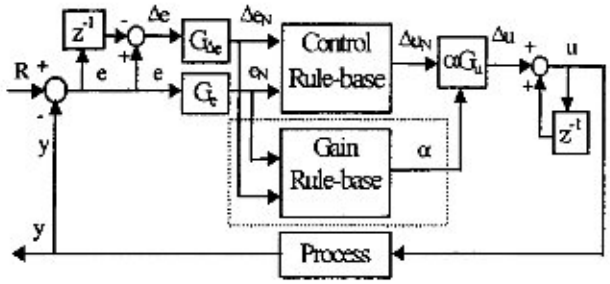
In (4), k is the sampling instance and Δu is the incremental change in controller output, which is determined by the rules of the form R_{PI} : If e is E and Δe is ΔE, then Δu is ΔU. The rule base for computing Δu is shown in Fig. 3(d). For STFPIC, the required nonlinear controller output $(\Delta u_{\mathrm{STFPIC}})$ is generated by modifying the output of a simple FLC $(\Delta u_{\mathrm{FPIC}})$ with the updating factor α, i.e.

$$\Delta u_{\mathrm{STFPIC}} \propto \alpha \cdot (\Delta u_{\mathrm{FPIC}}) \text{ or } \Delta u_{\mathrm{STFPIC}} = K \cdot \alpha \cdot (\Delta u_{\mathrm{FPIC}}) \tag{5}$$

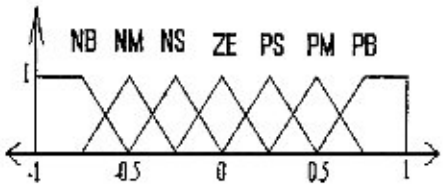
where K is the proportionality constant. The gain updating factor α is calculated using fuzzy rules of the form R_α: If e is E and Δe is ΔE, then α is $\boldsymbol{\alpha}$. The rule base in Fig. 3(e) is used for the computation of α. This is designed in conjunction with the rule base in Fig. 3(d). The rule base in Fig. 3(e) is derived from the knowledge of control engineering with a view to mimicking an operator's strategy while running a plant. We emphasize here that the determination of the rule base for α is dependent on the controller rule base. Detailed justification for using the rule base in Fig. 3(e) can be found in [26].

B. Identification of the Self-Tuning Controller

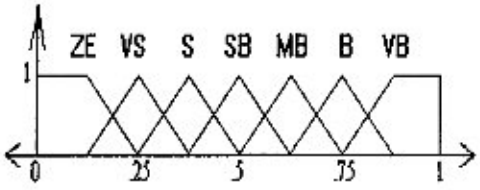
The STFPIC in Fig. 3(a) uses 49 rules to compute the gain updating factor α, and exhibits good performances [26]. In this section, we investigate the following: Given some data describing the gain variation α as a function of e and Δe, can we extract a set of rules to realize such a highly nonlinear system (Fig. 2)? Do we really need all 49 rules, or we can extract a smaller set of rules using our proposed SI approach



(a)



(b)



(c)

Δe/e	NB	NM	NS	ZE	PS	PM	PB
NB	NB	NB	NB	NM	NS	NS	ZE
NM	NB	NM	NM	NM	NS	ZE	PS
NS	NB	NM	NS	NS	ZE	PS	PM
ZE	NB	NM	NS	ZE	PS	PM	PB
PS	NM	NS	ZE	PS	PS	PM	PB
PM	NS	ZE	PS	PM	PM	PM	PB
PB	ZE	PS	PS	PM	PB	PB	PB

(d)

Δe/e	NB	NM	NS	ZE	PS	PM	PB
NB	VB	VB	VB	B	SB	S	ZE
NM	VB	VB	B	B	MB	S	VS
NS	VB	MB	B	VB	VS	S	VS
ZE	S	SB	MB	ZE	MB	SB	S
PS	VS	S	VS	VB	B	MB	VB
PM	VS	S	MB	B	B	VB	VB
PB	ZE	S	SB	B	VB	VB	VB

(e)

Fig. 3. (a) Block diagram of the STFPIC. (b) MFs of e and Δe. **N**: Negative, **P**: Positive, **ZE**: Zero, **B**: Big, **M**: Medium, **S**: Small. (c) MFs of gain updating factor (α). **ZE**: Zero, **V**: Very, **B**: Big, **M**: Medium, **S**: Small. (d) Fuzzy rules for computation of Δu. (e) Fuzzy rules for computation of α.

TABLE III
CORRESPONDENCE MATRIX FOR UNIFORMLY SAMPLED INPUT DATA

	Y_1	Y_2	Y_3	Y_4	Y_5
X_1	0	18	49	55	11
X_2	86	10	0	0	30
X_3	86	10	0	5	27
X_4	6	44	5	26	32
X_5	0	18	49	50	8

to do the same? How does the performance of the identified system compare with the original one?

To identify the STFPIC, we need some data, i.e., we need a set of two-dimensional input vectors as $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subset \Re^2$ and the associated set of one-dimensional output vectors as $Y = \{y_1, \ldots, y_n\} \subset \Re$ where $X = \{e, \Delta e\}$ and $Y = \{\alpha\}$. Here, we have generated data using two schemes: 1) by sampling input variables $(e, \Delta e)$ uniformly and computing the value of α for each sampled point, and 2) by running the process in close loop. In our implementation, we considered $c1 = c2 = 5$ and used FCM algorithm [2] with $m = 2$. As an illustration, the correspondence table C obtained for a set with 625 points generated by scheme 1) is shown in Table III. As discussed earlier, the number of rules to be extracted for α is dependent on the threshold θ. For example, $\theta = 11$ leads to 14 rules, while $\theta = 8$ gives rise to 17 rules. Note that the data generation using scheme 1) does not require the process to be controlled, while for scheme 2), we need the process to run in a close loop.

We have tested our schemes for several second-order linear and nonlinear processes with different values of dead time (L) and observed satisfactory results in each case. However, here we report results only for two of them, shown in (6) and (7) with a single value of L for each. These processes are also used in [26].

$$\ddot{y} + \dot{y} = u(t - L) \quad \text{(marginally stable)} \tag{6}$$

and

$$\ddot{y} + \dot{y} + 0.25y^2 = u(t - L) \quad \text{(nonlinear)}. \tag{7}$$

Table IV shows the correspondence table C for the marginally stable system in (6) when the data (1000 points) are generated by scheme 2), i.e., by running the process in close loop. Choice of $\theta = 11$ results in twelve rules. The input centroid corresponding to the first row of Table IV is (0.000 06, 0.001 27). This clearly indicates, as explained earlier, that there is a strong cluster in X near the steady state, i.e., $(e, \Delta e) \approx (0, 0)$. Similarly, Table V depicts the correspondence for the process in (7) when data (800 points) are generated by scheme 2). Here again, we find a strong cluster around $(e, \Delta e) \approx (0, 0)$. The second row of Table V corresponds to the centroid $(e, \Delta e) = (0.002\,85, -0.000\,40)$. Here, $\theta = 11$ leads to 13 rules.

TABLE IV
CORRESPONDENCE MATRIX FOR THE DATA GENERATED BY RUNNING THE PROCESS IN (6) IN CLOSE LOOP

	Y_1	Y_2	Y_3	Y_4	Y_5
X_1	140	226	314	0	37
X_2	28	1	0	39	55
X_3	20	1	0	39	48
X_4	8	13	5	0	0
X_5	8	13	5	0	0

TABLE V
CORRESPONDENCE MATRIX FOR THE DATA GENERATED BY RUNNING THE PROCESS IN (7) IN CLOSE LOOP

	Y_1	Y_2	Y_3	Y_4	Y_5
X_1	28	78	0	16	41
X_2	147	61	220	10	36
X_3	0	6	0	16	56
X_4	10	36	0	0	0
X_5	0	21	0	6	12

To demonstrate the effectiveness of our scheme, we have used the same controller parameters and implementation strategy as used in [26]. We consider $L = 0.1$ s and 0.5 s for the processes in (6) and (7), respectively. Load disturbances for the process in (6) and (7) are respectively applied at 55 and 40 s (Figs. 4–7). For notational convenience, the untuned and tuned versions of the FLCs will be denoted by $(\mathrm{FPIC}_{\mathrm{wav}\,1}, \mathrm{TFPIC}_{\mathrm{wav}\,1})$, $(\mathrm{FPIC}_{\mathrm{cent}}, \mathrm{TFPIC}_{\mathrm{cent}})$ and $(\mathrm{FPIC}_{\mathrm{wav}\,2}, \mathrm{TFPIC}_{\mathrm{wav}\,2})$ for *three* different types of MFs. We have computed a number of performance indices such as peak overshoot (%os), settling time (t_s), rise time (t_r), integral-absolute error (IAE), and integral of time-multiplied absolute error (ITAE) [27] for a detailed performance comparison of the identified system and the original STFPIC. These performance indices for both processes with dead time L are provided in tabular forms. In each table row corresponding to STFPIC presents the performance of the original system. We also provide response characteristics for each process due to step set-point change as well as load disturbance. We report the response characteristics corresponding to $\mathrm{FPIC}_{\mathrm{cent}}$ and $\mathrm{TFPIC}_{\mathrm{cent}}$ only, as we found that their performances are almost the same as those of $\mathrm{FPIC}_{\mathrm{wav}\,2}$ and $\mathrm{TFPIC}_{\mathrm{wav}\,2}$, and better than those of $\mathrm{FPIC}_{\mathrm{wav}\,1}$ and $\mathrm{TFPIC}_{\mathrm{wav}\,1}$. Of course, we provide the detailed performance comparison in terms of various indices in tabular form for all the three types of FLCs with the original STFPIC [26].

We established in [26] that STFPIC provides an improved performance over the conventional FLC. The objective here is to justify whether the identified systems ($\mathrm{FPIC}_{\mathrm{cent}}$ or $\mathrm{TFPIC}_{\mathrm{cent}}$) can provide the same level of performance as that of the original one (STFPIC). In our subsequent discussion, we say that the performance of $\mathrm{FPIC}_{\mathrm{cent}}/\mathrm{TFPIC}_{\mathrm{cent}}$ is good or satisfactory only when its performance is close to that of STFPIC. We emphasize that an identified system is called satisfactory only with respect to its closeness to the target system, here STFPIC. Therefore, if for some reason or other

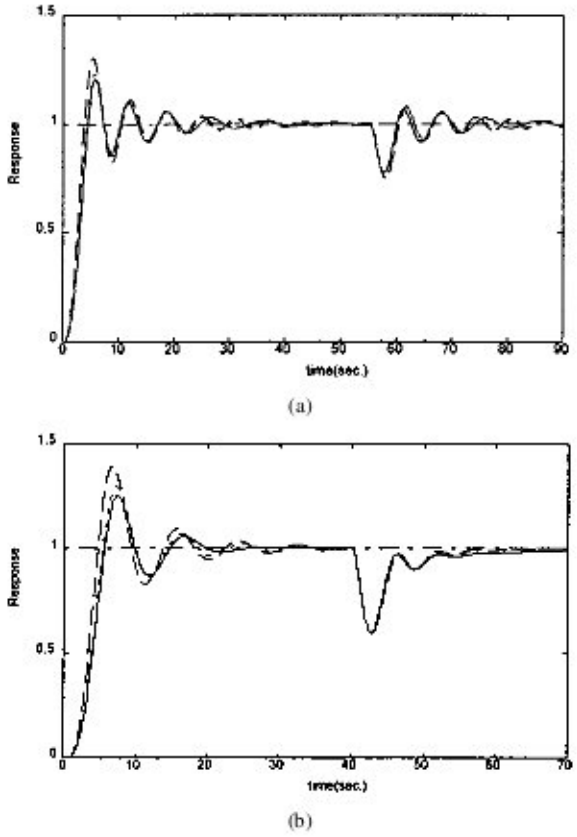


Fig. 4. (a) Responses of (6) for *Strategy* 1 with uniformly sampled data. [— STFPIC; -- $\mathrm{FPIC}_{\mathrm{cent}}$; ..... $\mathrm{TFPIC}_{\mathrm{cent}}$]. (b) Responses of (7) for *Strategy* 1 with uniformly sampled data. [— STFPIC; -- $\mathrm{FPIC}_{\mathrm{cent}}$; ..... $\mathrm{TFPIC}_{\mathrm{cent}}$].

TABLE VI
PERFORMANCE ANALYSIS FOR *STRATEGY* 1 WITH UNIFORMLY SAMPLED DATA

Process	FLC	%os	t_s(s)	t_r(s)	ITAE	IAE
(6)	STFPIC	20.3	19.0	4.4	101.0	5.19
	$\mathrm{FPIC}_{\mathrm{cent}}$	30.3	19.2	3.8	116.4	5.48
	$\mathrm{TFPIC}_{\mathrm{cent}}$	22.8	16.5	4.2	110.0	5.36
(7)	STFPIC	25.2	17.1	5.5	109.3	6.63
	$\mathrm{FPIC}_{\mathrm{cent}}$	39.0	17.2	4.7	103.0	6.70
	$\mathrm{TFPIC}_{\mathrm{cent}}$	29.5	17.3	5.3	104.3	6.65

$\mathrm{FPIC}_{\mathrm{cent}}$ or $\mathrm{TFPIC}_{\mathrm{cent}}$ shows much better performance compared to STFPIC then that will indicate a bad/unsatisfactory performance of our SI scheme as it fails to track the original one.

1) Results for Strategy 1:

Data Generated by Uniform Sampling of e and Δe: Here, we use $\theta = 8$ in Table III. After splitting the input and output clusters as suggested by the proposed scheme described in Section III-B we extracted 17 rules. Response characteristics of (6) and (7) are respectively shown in Fig. 4(a) and (b), and Table VI provides various performance indices. Response characteristics for each process show higher %os even in the case of tuned FLC ($\mathrm{TFPIC}_{\mathrm{cent}}$), and are comparatively more oscillatory than STFPIC, specially due to step set-point change. For example, corresponding to the process in (6), the %os for $\mathrm{FPIC}_{\mathrm{cent}}$ is 30.3 and for $\mathrm{TFPIC}_{\mathrm{cent}}$ is 22.8 (Table VI). These are higher

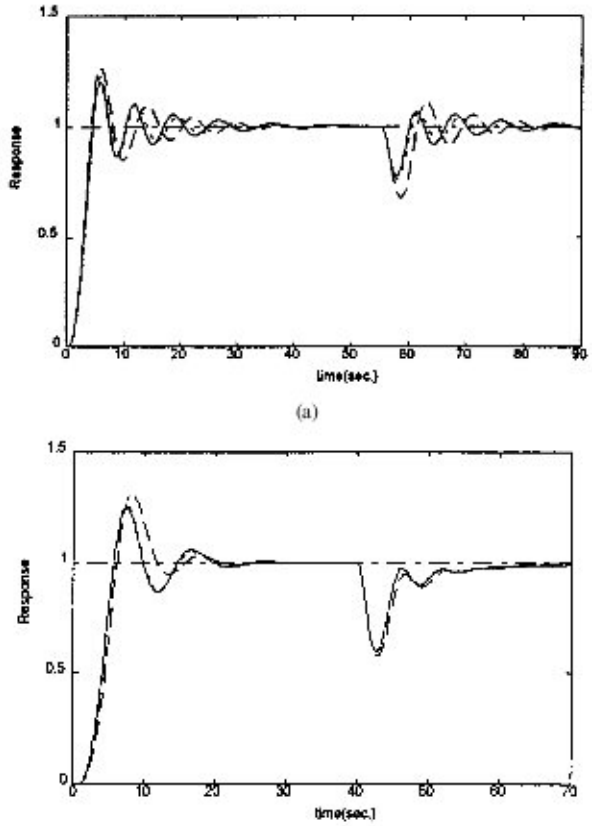


Fig. 5. (a) Responses of (6) for *Strategy* 1 with data generated in close loop. [— STFPIC; -- $FPIC_{cent}$; ..... $TFPIC_{cent}$]. (b) Responses of (7) for *Strategy* 1 with data generated in close loop. [— STFPIC; -- $FPIC_{cent}$; ..... $TFPIC_{cent}$].

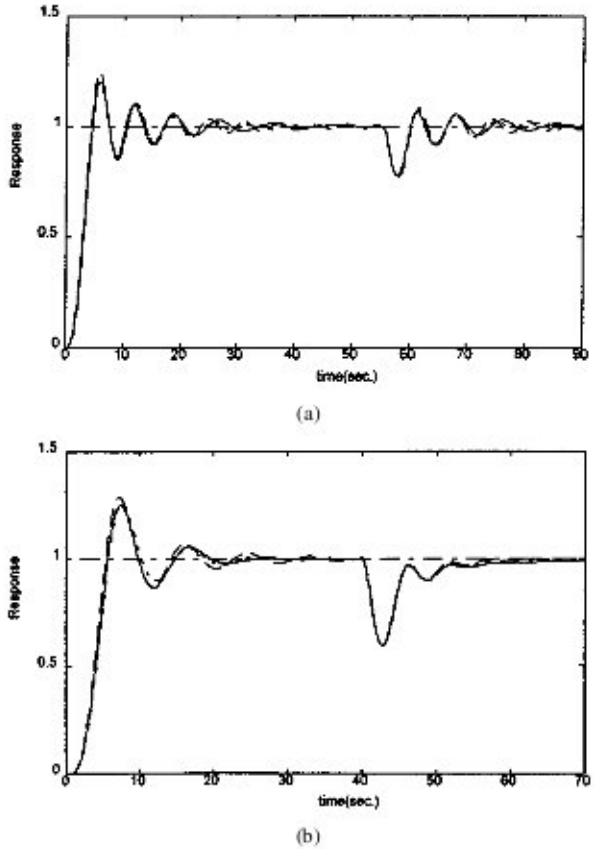


Fig. 6. (a) Responses of (6) for *Strategy* 2 with uniformly sampled data when $\theta = 8$. [— STFPIC; -- $FPIC_{cent}$; ..... $TFPIC_{cent}$]. (b) Responses of (7) for *Strategy* 2 with uniformly sampled data when $\theta = 8$. [— STFPIC; -- $FPIC_{cent}$; ..... $TFPIC_{cent}$].

TABLE VII
PERFORMANCE ANALYSIS FOR *STRATEGY* 1 WITH DATA GENERATED IN CLOSE LOOP

Process	FLC	%os	t_s(s)	t_r(s)	ITAE	IAE
(6)	STFPIC	20.3	19.0	4.4	101.0	5.19
	$FPIC_{wav1}$	23.0	18.2	4.2	135.3	5.80
	$TFPIC_{wav1}$	19.4	16.4	4.5	90.8	5.04
	$FPIC_{cent}$	26.6	18.8	4.2	138.3	5.99
	$TFPIC_{cent}$	22.6	16.6	4.4	91.2	5.13
	$FPIC_{wav2}$	26.5	18.8	4.2	138.0	5.97
	$TFPIC_{wav2}$	22.5	16.6	4.4	91.3	5.13
(7)	STFPIC	25.2	17.1	5.5	109.3	6.63
	$FPIC_{wav1}$	28.9	11.2	6.0	119.2	7.17
	$TFPIC_{wav1}$	26.4	14.2	5.6	109.7	6.71
	$FPIC_{cent}$	31.0	11.3	6.0	118.5	7.20
	$TFPIC_{cent}$	26.3	14.1	5.6	107.1	6.66
	$FPIC_{wav2}$	30.7	11.3	6.0	117.2	7.17
	$TFPIC_{wav2}$	26.4	14.1	5.6	106.9	6.66

TABLE VIII
PERFORMANCE ANALYSIS FOR *STRATEGY* 2 WITH UNIFORMLY SAMPLED DATA WHEN $\theta = 8$

Process	FLC	%os	t_s(s)	t_r(s)	ITAE	IAE
(6)	STFPIC	20.3	19.0	4.4	101.0	5.19
	$FPIC_{wav1}$	29.4	19.2	3.9	116.1	5.54
	$TFPIC_{wav1}$	20.2	16.6	4.4	109.7	5.38
	$FPIC_{cent}$	24.2	19.0	4.2	114.2	5.46
	$TFPIC_{cent}$	20.1	16.5	4.4	102.0	5.25
	$FPIC_{wav2}$	23.7	19.0	4.2	115.2	5.47
	$TFPIC_{wav2}$	20.1	16.5	4.4	102.5	5.25
(7)	STFPIC	25.2	17.1	5.5	109.3	6.63
	$FPIC_{wav1}$	34.6	17.3	4.9	101.0	6.63
	$TFPIC_{wav1}$	26.5	17.1	5.5	102.2	6.57
	$FPIC_{cent}$	28.9	17.3	5.3	99.2	6.51
	$TFPIC_{cent}$	25.3	13.9	5.7	103.2	6.57
	$FPIC_{wav2}$	28.8	17.3	5.3	98.19	6.48
	$TFPIC_{wav2}$	25.4	13.9	5.6	102.8	6.56

than the desired value of 20.3 corresponding to STFPIC. Similarly, t_r, t_s, ITAE and IAE of $FPIC_{cent}$ or $TFPIC_{cent}$ are close (but not very close) to those of STFPIC. In case of the identified system [Fig. 4(a)] the number of oscillations before the occurrence of load disturbance (at $t = 55$ s) is found to be six, which is five for STFPIC. Similar performance is exhibited by the identified system for (7) [Fig. 4(b), Table VI]. Thus for *Strategy* 1, the performance of the identified system is not quite satisfactory, though not very bad. This is possibly due to the uniformly sampled data and presence of abruptness in the system as explained in the earlier sections.

Data Generated by Running the Process in Close Loop: Table VII and Fig. 5(a) and (b) present the performances of (6) and (7). For the process in (6), we used the correspondence in Table IV with $\theta = 11$ resulting in twelve

TABLE IX
PERFORMANCE ANALYSIS FOR *STRATEGY* 2 WITH UNIFORMLY SAMPLED DATA WHEN $\theta = 14$

Process	FLC	%os	t_s(s)	t_r(s)	ITAE	IAE
(6)	STFPIC	20.3	19.0	4.4	101.0	5.19
	$FPIC_{way1}$	29.9	19.4	4.0	115.3	5.59
	$TFPIC_{way1}$	20.2	16.4	4.4	105.6	5.28
	$FPIC_{cent}$	23.2	19.6	4.3	118.6	5.60
	$TFPIC_{cent}$	22.6	16.7	4.3	106.8	5.32
	$FPIC_{way2}$	22.5	17.1	4.4	119.2	5.61
	$TFPIC_{way2}$	22.7	16.7	4.2	107.6	5.31
(7)	STFPIC	25.2	17.1	5.5	109.3	6.63
	$FPIC_{way1}$	34.7	17.6	5.1	102.0	6.77
	$TFPIC_{way1}$	28.6	17.4	5.5	107.4	6.76
	$FPIC_{cent}$	29.0	17.8	5.6	104.7	6.84
	$TFPIC_{cent}$	27.7	17.5	5.4	107.3	6.65
	$FPIC_{way2}$	28.5	17.8	5.7	104.9	6.84
	$TFPIC_{way2}$	27.9	17.6	5.3	106.9	6.61

TABLE X
PERFORMANCE ANALYSIS FOR *STRATEGY* 2 WITH DATA GENERATED IN CLOSE LOOP

Process	FLC	%os	t_s(s)	t_r(s)	ITAE	IAE
(6)	STFPIC	20.3	19.0	4.4	101.0	5.19
	$FPIC_{way1}$	21.4	18.1	4.3	131.9	5.76
	$TFPIC_{way1}$	16.6	17.7	4.4	107.7	5.34
	$FPIC_{cent}$	19.0	18.2	4.4	130.4	5.76
	$TFPIC_{cent}$	14.8	17.6	4.5	105.8	5.30
	$FPIC_{way2}$	17.2	18.2	4.4	130.4	5.72
	$TFPIC_{way2}$	14.9	17.6	4.5	105.7	5.30
(7)	STFPIC	25.2	17.1	5.5	109.3	6.63
	$FPIC_{way1}$	32.3	13.9	5.5	140.6	7.38
	$TFPIC_{way1}$	26.7	14.5	5.6	116.7	6.85
	$FPIC_{cent}$	30.4	13.8	5.6	145.3	7.46
	$TFPIC_{cent}$	26.3	14.5	5.6	112.7	6.77
	$FPIC_{way2}$	30.1	13.9	5.6	142.4	7.41
	$TFPIC_{way2}$	26.2	14.5	5.6	112.5	6.77

rules. Since the data sets are generated by running the processes, they are supposed to have good cluster structures, which are not expected in the case of uniformly sampled data. This fact is reflected in the results (Table VII and Fig. 5). For the process in (6), Table VII shows that %os, t_r, and IAE of $TFPIC_{cent}$ is much closer to those of STFPIC compared to the earlier case (Table VI). The improvement in performance achieved by the data generated in close loop is more clearly visible comparing Fig. 4(a) and Fig. 5(a). Similar observation can also be made from Fig. 4(b) and Fig. 5(b) for the process in (7). Sometimes even the performances of the identified system become comparable to those of the original STFPIC [Fig. 5(b)].

2) Results for Strategy 2:

Data Generated by Uniform Sampling of e and Δe: Here, we report results for two different values of θ: 1) $\theta = 8$ and 2) $\theta = 14$. For $\theta = 8$, we have 17 rules, whereas we get only 13 rules for $\theta = 14$, against 49 original rules used in STFPIC. In both cases, performances are very good; even the performances with 13 rules are better than those of *Strategy* 1 with 17 rules (Fig. 4 and Table VI).

The performance with 17 rules ($\theta = 8$) for (6) and (7) are shown in Table VIII and response characteristics are shown in Fig. 6(a) and (b). In each case, performance of the identified system is seen to be extremely good (i.e., the identified system very closely matches the STFPIC) even without any tuning, and for the tuned versions, it is almost identical to that of original STFPIC. This is further established by the closeness of rows corresponding to $FPIC_{cent}$ and $TFPIC_{cent}$ with that of STFPIC.

For $\theta = 14$, we get 13 rules and in this case, too, we observe good performances as exhibited by Table IX. As expected, results are not as good as those with 17 rules under *Strategy* 2. For example, we find that the %os for each FLC is little higher than that of the original one (STFPIC). But the overall performance for *Strategy* 2 with 13 rules (Table IX) are much better than those for *Strategy* 1 with even 17 rules (Table VI).

Data Generated by Running the Process in Close Loop: Fig. 7(a) and (b) and Table X depict the performance of (6) and (7). Comparing Fig. 7(b) and Table X with Fig. 5(b) and Table VII, we find that for the process in (7) the overall performance of *Strategy* 1 and *Strategy* 2 are almost the same when data are generated in close loop. An almost similar observation [Fig. 7(a), Table X and Fig. 5(a), Table VII] also holds for the process in (6).

To summarize, when the data set is generated running the process, it exhibits cluster structures and both *Strategy* 1 and

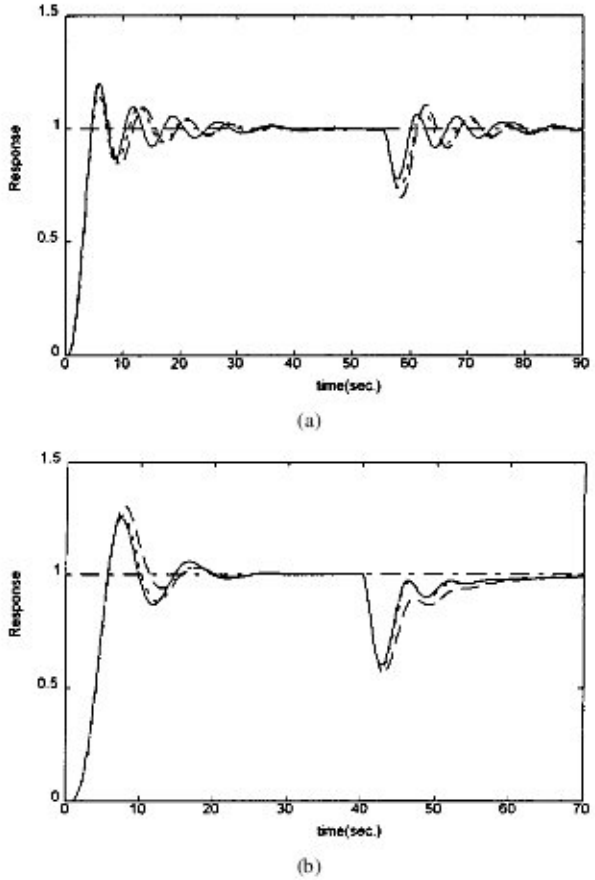


Fig. 7. (a) Responses of (6) for *Strategy* 2 with data generated in close loop. [— STFPIC; -- $FPIC_{cent}$; ..... $TFPIC_{cent}$]. (b) Responses of (7) for *Strategy* 2 with data generated in close loop. [— STFPIC; -- $FPIC_{cent}$; ..... $TFPIC_{cent}$].

Strategy 2 are equally good to identify the system. But when the data are generated uniformly sampling the input space, *Strategy* 1, as expected, does not perform quite well, but *Strategy* 2, which takes into account the interaction between input and output variables while clustering, can identify the system quite well.

V. Conclusion

After analyzing the various problems that one faces to extract rules from numerical data, we provided answers to three important issues: deciding on the proper domain(s) of clustering, deciding on the number of rules, and getting an initial estimate of parameters, such as MFs, of the fuzzy systems. We justified that if X and Y have distinct cluster substructures then separate clustering of X and Y would be enough, otherwise this should be followed by reclustering of X^*. We also suggested a scheme to establish the correspondence between the clusters obtained from X and Y. The proposed schemes have been used to extract a rule base for computation of the gain updating factor α of a self-tuning fuzzy PI controller. The proposed schemes are able to reduce the number of rules by more than 70% maintaining almost the same level of performance.

In our schemes, the number of rules depends on the choice of θ. Ideally, θ should be determined taking into account the total number of data points N. The thresholding should be done not on C_{ij}, but on its normalized value $T_{ij} = C_{ij}/N$. Although we suggested some simple schemes to get an initial estimate of the MFs, more work is needed to decide on more appropriate type (not necessarily triangular) MFs and estimation of their parameters. Further investigation is also required to establish robustness of the identified system.

References

[1] N. R. Pal, K. Pal, J. C. Bezdek, and T. Runkler, "Some issues in system identification using clustering," in *Proc. ICNN*, 1997, pp. 2524–2529.

[2] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms* New York, 1981.

[3] M. Sugeno and T. Yasukawa, "A fuzzy-logic-based approach to qualitative modeling," *IEEE Trans. Fuzzy Syst.*, vol. 1, pp. 7–31, Feb. 1993.

[4] Y. Yoshinari, W. Pedrycz, and K. Hirota, "Construction of fuzzy models through clustering techniques," *Fuzzy Sets Syst.*, vol. 54, no. 2, pp. 157–166, 1993.

[5] R. R. Yager and D. P. Filev, "Generation of Fuzzy Rules by mountain clustering," *J. Intell. Fuzzy Syst.*, vol. 2, pp. 209–219, 1994.

[6] S. L. Chiu, "Extracting fuzzy rules for pattern classification by cluster estimation," in *Proc. IFSA*, 1995, pp. 1–4.

[7] ——, "Fuzzy model identification based on cluster estimation," *J. Intell. Fuzzy Syst.*, vol. 2, pp. 267–278, 1994.

[8] R. Babuska and U. Kaynak, "Application of compatible cluster merging to fuzzy modeling of multivariable systems," in *Proc. EUFIT*, 1995, pp. 565–569.

[9] U. Kaynak, R. Babuska, and M. Setnes, "Methods for simplifications of fuzzy models," in *Proc. NATO ASI*, 1996, pp. 1–10.

[10] T. W. Cheng, D. B. Goldgof, and L. O. Hall, "Fast clustering with application to fuzzy rule generation," in *Proc. IEEE FUZZ*, 1995, pp. 2289–2295.

[11] T. A. Runkler and R. H. Palm, "Identification of nonlinear systems using regular fuzzy c-elliptotype clustering," in *Proc. IEEE FUZZ*, 1996, pp. 1026–1030.

[12] M. Delgado, A. F. Gomez-Skarmeta, and F. Martin, "A fuzzy clustering based rapid-prototyping for fuzzy modeling," *IEEE Trans. Fuzzy Syst.*, vol. 5, pp. 223–233, Apr. 1997.

[13] N. R. Pal and J. C. Bezdek, "On cluster validity for the fuzzy c-means model," *IEEE Trans. Fuzzy Syst.*, vol. 3, pp. 370–379, June 1995.

[14] Y. Fukuyama and M. Sugeno, "A new method of choosing the number of clusters for the fuzzy c-means method," in *Proc. 5th Fuzzy Systems Symp.*, 1989, pp. 247–250. in Japanese.

[15] S. K. Sin and R. P. J. deFigueiredo, "Fuzzy system design through fuzzy clustering and optimal predefuzzification," in *Proc. IEEE FUZZ*, 1993, pp. 190–195.

[16] X. L. Xie and G. A. Beni, "Validity measure for Fuzzy clustering," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 3, pp. 841–846, Aug. 1991.

[17] D. Driankov, H. Hellendoorn, and M. Reinfrank, *An Introduction to Fuzzy Control*. New York: Springer-Verlag, 1993.

[18] Y. Nakamori and M. Royke, "Identification of fuzzy prediction models through hyperellipsoidal clustering," *IEEE Trans. Syst., Man, Cybern.*, vol. 24, pp. 1153–1173, Aug. 1994.

[19] A. Jain and R. Dubes, *Algorithms for Clustering Data*. Englewood Cliffs, NJ: Prentice Hall, 1988.

[20] D. E. Gustafson and W. C. Kesel, "Fuzzy clustering with a fuzzy covariance matrix," *Proc. IEEE FUZZ*, pp. 761–766, 1997.

[21] R. Krishnapuram and C. P. Freg, "Fitting an unknown number of lines and planes to image data through compatible cluster merging," *Pattern Recognit.*, vol. 25, no. 4, pp. 385–400, 1993.

[22] R. Babuska, M. Setnes, U. Kaynak, and R. H. Van Nauta Lemke, "Simplification of fuzzy rule bases," in *Proc. EUFIT*, 1996, pp. 1115–1119.

[23] I. Gath and A. B. Geva, "Unsupervised optimal fuzzy clustering," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 11, pp. 773–781, July 1989.

[24] E. H. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *Int. J. Man-Mach. Stud.*, vol. 7, no. 1, pp. 1–13, 1975.

[25] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its application to modeling and control," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-15, pp. 116–132, Jan./Feb. 1985.

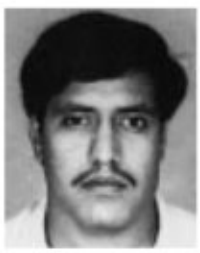
[26] R. K. Mudi and N. R. Pal, "A robust self-tuning scheme for PI and PD type fuzzy controllers," *IEEE Trans. Fuzzy Syst.*, vol. 7, pp. 2–16, Feb. 1999.

[27] K. Ogata, *Modern Control Engineering*. Englewood Cliffs, NJ: Prentice-Hall, 1970.

[28] T. Sudkamp and R. J. Hammell, II, "Interpolation, completion and learning fuzzy rules," *IEEE Trans. Syst. Man, Cybern.*, vol. 24, pp. 332–342, Feb. 1994.

Kuhu Pal received the B.Sc. degree with honors in physics in 1984 from the University of Burdwan and the M.Sc. and Ph.D. degrees in physics from Banaras Hindu University (BHU) in 1987 and 1993, respectively.

She was a Research Associate first in the Physics Department of BHU, and then from September 1995 in the Machine Intelligence Unit of the Indian Statistical Institute, Calcutta. In September 1999, she joined the MCKV Institute of Engineering as a Lecturer, and subsequently visited the Computer Science Department of the University of West Florida, Pensacola, from January to June 2000. Her research interests include pattern recognition, fuzzy sets theory, fuzzy logic controllers, neural networks, and computational material science.



Rajani K. Mudi received the B.Tech. and M.Tech. degrees in applied physics from the University of Calcutta, India, in 1990 and 1992, respectively, and the Ph.D. degree from Jadavpur University, India, in 1999.

Since 1992, he has been with the Department of Instrumentation Engineering, Jadavpur University, where he is currently a Reader. He was a Guest Lecturer in the Department of Applied Physics from 1993 to 1996, and the Department of Plastics and Rubber Technology from 1993 to 1995 at the University of Calcutta. He is a joint coordinator of the *2002 AFSS International Conference on Fuzzy Systems*. His research interests are in fuzzy and neuro-fuzzy control, qualitative modeling, and intelligent instrumentation.

Nikhil R. Pal (M'91–SM'00) received the B.Sc. degree with honors in physics and the Master of Business Management from the University of Calcutta, India, in 1978 and 1982, respectively. He received the M.Tech. and Ph.D. degrees in computer science from the Indian Statistical Institute, Calcutta, in 1984 and 1991, respectively.

He is currently a Professor in the Electronics and Communication Sciences Unit of the Indian Statistical Institute and the current Professor-in-Charge of the Computer and Communication Sciences Division. From September 1991 to February 1993, July 1994 to December 1994, October 1996 to December 1996, and January 2000 to July 2000, he visited the Computer Science Department of the University of West Florida, Pensacola. He has also been a guest faculty of the University of Calcutta. His research interests include image processing, pattern recognition, fuzzy sets theory, measures of uncertainty, neural networks, genetic algorithms, and fuzzy logic controllers. He has co-authored a book titled *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing* (Reading, MA: Kluwer, 1999), co-edited the volume *Advances in Pattern Recognition and Digital Techniques, ICAPRDT'99*, and edited a book titled *Pattern Recognition in Soft Computing Paradigm* (Singapore: World Scientific, 2001). He is an associate editor of the *International Journal of Fuzzy Systems, International Journal of Approximate Reasoning*, and an area editor of *Fuzzy Sets and Systems*.

Dr. Pal is an associate editor of IEEE TRANSACTIONS ON FUZZY SYSTEMS and IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS—Part B (electronic version).