# Simplifying Fuzzy Rule-Based Models Using Orthogonal Transformation Methods

John Yen, *Senior Member, IEEE,* and Liang Wang, *Member, IEEE*

***Abstract*— An important issue in fuzzy-rule-based modeling is how to select a set of important fuzzy rules from a given rule base. Even though it is conceivable that removal of redundant or less important fuzzy rules from the rule base can result in a compact fuzzy model with better generalizing ability, the decision as to which rules are redundant or less important is not an easy exercise. In this paper, we introduce several orthogonal transformation-based methods that provide new or alternative tools for rule selection. These methods include an orthogonal least squares (OLS) method, an eigenvalue decomposition (ED) method, a singular value decomposition and QR with column pivoting (SVD-QR) method, a total least squares (TLS) method, and a direct singular value decomposition (D-SVD) method. A common attribute of these methods is that they all work on a firing strength matrix and employ some measure index to detect the rules that should be retained and eliminated. We show the performance of these methods by applying them to solving a nonlinear plant modeling problem. Our conclusions based on analysis and simulation can be used as a guideline for choosing a proper rule selection method for a specific application.**

***Index Terms*— Fuzzy logic, fuzzy modeling, model reduction, orthagonal transformation.**

## I. Introduction

FUZZY-RULE-BASED modeling has become an active research field in recent years because of its unique merits in solving complex nonlinear system identification and control problems. Primary advantages of this approach include the facility for the explicit knowledge representation in the form of If–Then rules, the mechanism of reasoning in human understandable terms, the capacity of taking linguistic information from human experts and combining it with numerical information, and the ability of approximating complicated nonlinear functions with simpler models. Unlike conventional modeling, where a single model is used to describe the global behavior of a system, fuzzy rule-based modeling is essentially a *multimodel* approach in which individual rules (where each rule acts like a "local model") are combined to describe the global behavior of the system. When using a fuzzy model to approximate an unknown system, it is desired that the model include as many rules as possible so that it can cover the input–output state space of the system with sufficient "patches;" yet it is also desired that the model include as few rules as possible because the generalizing ability of the model decreases as the number of rules increases. When we speak here of *generalization* we are referring to the system's mean performance in terms of approximation accuracy evaluated over some independent test data set. The tradeoff between goodness of fit and simplicity is a fundamental principle underlying various general theories of statistical modeling and inductive inference [1], [2].

Several research efforts have been made in the fuzzy logic community to strike a balance between reducing the fitting error and increasing the model complexity. For example, an *entropy criterion* was proposed in [35] to find a simple structure of fuzzy model by minimizing the rate of interaction between fuzzy rules. The number of fuzzy rules in this criterion was determined using an *unbiasedness criterion* (UC) [27]. In order to calculate an UC value, the available training data must be split into two subsets each of which is used to construct a fuzzy model. Since two fuzzy models have to be constructed one time, the computational cost of UC is high. Further, UC essentially tries to find the model structure by cross-validating two subsets of training data and does not take into account the complexity of the resulting model. As pointed out in [3], if enough different model structures are considered, UC can often find a model that has a low error on the two subsets of training data, but will not generalize well to new untrained data. In [4], a *pruning and merging* strategy was suggested to eliminate redundant fuzzy rules. This method is complex, also the arbitrariness associated with some predetermined parameters in the method may lead to improper models. *Genetic algorithms* based methods have also been used for extracting fuzzy rules for control and classification problems [12], [17], [19]. These methods use *ad hoc* criteria to assess the importance of rules and cannot always assure the resulting models are simplest. Moreover, they are extremely computation intensive and may not be well-suited to resource-constrained applications.

Recently, several researchers have applied *orthogonal transformation* methods for selecting important fuzzy rules from a given rule base [22], [32], [37]–[39]. However, the performance and usage of these methods are unknown for most fuzzy modeling workers. This paper will provide a more thorough analysis of these existing methods and introduce several new methods. In particular, we will conduct a comparative study

Manuscript received August 28, 1997; revised July 18, 1998. This work was supported by the National Science Foundation Young Investigator Award IRI-9257293.

J. Yen is with the Center for Fuzzy Logic, Robotics, and Intelligent Systems, Department of Computer Science, Texas A&M University, College Station, TX 77843-3112 USA (e-mail: yen@cs.tamu.edu).

L. Wang is with the Center for Adaptive Systems Applications, Inc., Los Alamos, NM 87544 USA.

Publisher Item Identifier S 1083-4419(99)00765-7.

of the performance of these proposed methods. We expect that our work will communicate these methods to the fuzzy logic community and provide a guideline for choosing a proper method for a specific application.

## II. Formulation of the Problem

### A. Fuzzy Models

A fuzzy model is a set of fuzzy If–Then rules that maps inputs to outputs. The basic structure of a fuzzy model consists of three conceptual components: a *rule base*, which contains a set of fuzzy rules, a *database* or *dictionary*, which defines the membership functions used in the fuzzy rules, and a *reasoning mechanism*, which performs the inference procedure upon the rules and a given condition to derive a reasonable output or conclusion. As Jang and Sun [14] pointed out, the spirit of fuzzy models quite resembles that of "divide and conquer"—the antecedents of fuzzy rules partition the input space into a number of local fuzzy regions, while the consequents describe the behavior within a given region via various constituents. The consequent constituent could be a membership function [20], a constant [13], or a linear equation [28]. Different consequent constituents result in different types of fuzzy models, but their antecedents are always the same.

In this paper, we have adopted the fuzzy model with constant consequent constituents to illustrate the proposed methods. This type of fuzzy model has the following form [13]:

$$R_i: \quad \text{if } x_1 \text{ is } A_{i1} \text{ and } \ldots \text{ and } x_m \text{ is } A_{im} \quad \text{then } y = b_i, \qquad i = 1, 2, \ldots, M \tag{1}$$

where $m$ and $M$ are the number of input variables and rules, respectively; $x_i$ and $y$ are the input and output variables, respectively; $A_{ij}$ are the membership functions of input variables; and $b_i$ are the constant consequent constituents. The membership functions of input variables are assumed to be the *Gaussian form*[1]

$$A_{ij}(x_j) = \exp\left(-\frac{(x_j - c_{ij})^2}{2\sigma_{ij}^2}\right) \tag{2}$$

where $c_{ij}$ and $\sigma_{ij}$ are the *centers* and *widths* of Gaussian functions, respectively.

This is a multi-input and single-output fuzzy model. Using the *center average defuzzifier*, the total output of the model can be computed as [33]

$$y = \sum_{i=1}^{M} v_i b_i \tag{3}$$

[1]The one-dimensional Gaussian membership function for each input variable can be obtained by decomposing an $m$-dimensional Gaussian membership function with a diagonal weighting matrix, where $m$ corresponds to the number of input variables. This is a unique property of the Gaussian membership function, i.e., it is *factorizable*.

where $v_i$ is the *normalized firing strength* of the $i$th rule, which is defined by

$$v_i = \frac{\prod_{j=1}^{m} \exp\left(-\frac{(x_j - c_{ij})^2}{2\sigma_{ij}^2}\right)}{\sum_{i=1}^{M}\prod_{j=1}^{m} \exp\left(-\frac{(x_j - c_{ij})^2}{2\sigma_{ij}^2}\right)}. \tag{4}$$

Equation (3) can be viewed as a special case of the linear regression model [33]

$$y(k) = \sum_{i=1}^{M} p_i(k)\theta_i + e(k) \tag{5}$$

with $p_i(k)$ and $\theta_i$ given by

$$p_i(k) \equiv v_i, \quad \theta_i \equiv b_i \tag{6}$$

where $p_i(k)$ are known as the *regressors*, $\theta_i$ are the parameters, and $e(k)$ is an error signal which is assumed to be uncorrelated with the regressors $p_i(k)$. Given $N$ input–output pairs $\{\boldsymbol{x}(k), y(k)\}$, $k = 1, 2, \ldots, N$, where $\boldsymbol{x}(k) = (x_1(k), x_2(k), \ldots, x_p(k))$, it is convenient to express (5) in the matrix form:

$$\boldsymbol{y} = P\theta + \boldsymbol{e} \tag{7}$$

where $\boldsymbol{y} = [y(1), y(2), \ldots, y(N)]^T \in \Re^N$, $P = [\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_M] \in \Re^{N\times M}$ with $\boldsymbol{p}_i = [p_i(1), p_i(2), \ldots, p_i(N)]^T \in \Re^N$, $\theta = [\theta_1, \theta_2, \ldots, \theta_M]^T \in \Re^M$, and $\boldsymbol{e} = [e(1), e(2), \ldots, e(N)]^T \in \Re^N$. Note that each column of $P$ corresponds to one of the fuzzy rules in the rule base. We will call $P$ the *firing strength matrix* and $P\theta$ the *predictor* throughout this paper for notational simplicity. In building a fuzzy model, the number of available training data points is usually larger than the number of fuzzy rules in the rule base. This implies that the row dimension of the matrix $P$ is larger than its column dimension, that is, $N > M$.

In this paper, the centers $c_{ij}$ and the widths $\sigma_{ij}$ of the Gaussian membership functions are predetermined using a *k-means clustering algorithm* and a *nearest-neighbor heuristic*, respectively [21]. This means that the antecedents of fuzzy rules are known *a priori*. In this case the only unknown parameters in (7) are $\theta_i$ and the problem is *linear-in-parameters*. Thus, many conventional linear optimization methods can be used to solve this problem.

### B. SVD, Minimum 2-Norm Solution, and Fuzzy Rule Selection

The *singular value decomposition* (SVD) of a matrix is a factorization of the matrix into a product of three matrices. For the firing strength matrix $P$, the decomposition can be written as

$$P = U\Sigma V^T \tag{8}$$

where $U \in \Re^{N\times N}$ and $V \in \Re^{M\times M}$ are orthogonal matrices, $\Sigma = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_M) \in \Re^{N\times M}$ is a diagonal matrix with $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_M \geq 0$. The diagonal elements of $\Sigma$ are called the *singular values* of $P$.

An important property of SVD is that it reveals the *rank* of $P$. From (8) it follows that $\text{rank}(P) = \text{rank}(\Sigma)$. Consequently, the number of nonzero singular values indicates the rank of the matrix $P$. Let $s = \text{rank}(P)$. Then (8) can be rewritten in an alternative form

$$P = \sum_{i=1}^{s} \sigma_i \boldsymbol{u}_i \boldsymbol{v}_i^T \tag{9}$$

where $\sigma_1, \sigma_2, \ldots, \sigma_s$ are the $s$ nonzero singular values of $P$, $\boldsymbol{u}_i$, and $\boldsymbol{v}_i$ are the $i$th column of $U$ and $V$, respectively. Substituting (9) into (7) and after simple algebraic manipulations, we obtain

$$\theta = \sum_{i=1}^{s} \frac{\boldsymbol{u}_i^T \boldsymbol{y}}{\sigma_i} \boldsymbol{v}_i. \tag{10}$$

It can be proved [8] that this solution minimizes $||\boldsymbol{y} - P\theta||_2$ and has the smallest 2-norm of all minimizers.

In practice, the minimum 2-norm solution is usually approximated by

$$\theta' = \sum_{i=1}^{r} \frac{\boldsymbol{u}_i^T \boldsymbol{y}}{\sigma_i} \boldsymbol{v}_i \tag{11}$$

where $r \leq s$ is some numerically determined estimate of $s$. Note that $\theta'$ minimizes $||\boldsymbol{y} - P'\theta'||_2$ where

$$P' = \sum_{i=1}^{r} \sigma_i u_i v_i^T \tag{12}$$

is the closest matrix to $P$ that has rank $r$ [8].

Replacing $P$ by $P'$ amounts to filtering the small singular values and can make a great deal of sense in those situations where $P$ is derived from noise data. In our application, however, the existence of small singular values implies the presence of *redundant* or *less important rules*[2] among the rules that comprise the underlying model [22], [37]. In this case, we are not interested in a predictor such as $P'\theta'$ that involves all $M$ rules. Instead, a predictor $P\theta$ should be sought where $\theta$ has at most $r$ nonzero components. The position of the nonzero entries determines which columns of $P$, i.e., which rules in the rule base, are to be used in constructing the model and in approximating the observation vector $\boldsymbol{y}$. How to pick these columns is the problem of *rule selection* and is the subject of this paper.

## III. Rule Selection Using Orthogonal Transformation

Orthogonal transformation is one of the most useful and powerful tools of numerical linear algebra and arises in many application areas particularly in control and signal processing [9], [18]. It is known that orthogonal transformation can lead to relative decorrelation and compaction of information into salient modes in a data set. The first proposal of applying this technique to fuzzy modeling was offered in Wang and Mendel [32] where an *orthogonal least-squares* (OLS) method was suggested to select the most important *fuzzy basis functions* each of which was associated with a fuzzy rule. This method was also used in [31] to remove less significant consequent terms in the *TSK model* [28]. In this section, we first introduce the OLS method, and then provide several new orthogonal transformation-based rule selection methods. Comparison of these methods will be conducted in the next section.

[2]The redundant rules can appear in a rule base when several rules have identical (or nearly identical) or linearly dependent (or nearly linearly dependent) firing strengths in the whole input space. The less important rules can occur in a rule base when several rules have zero (or near-zero) firing strengths in the whole input space.

### A. OLS Method

The OLS method involves the transformation of the regressors $\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_M$ (i.e., the column vectors of the firing strength matrix $P$) into a set of orthogonal basis vectors denoted by $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_M$. In particular, using a *classical Gram–Schmidt orthogonalization* procedure [8], the firing strength matrix $P$ is decomposed into

$$P = UR \tag{13}$$

where $U \in \Re^{N \times N}$ is a matrix with orthogonal columns $\boldsymbol{u}_i$, and $R \in \Re^{M \times M}$ is an upper triangular matrix with unity diagonal elements. Substituting (13) into (7) yields

$$y = UR\theta + \boldsymbol{e} = U\boldsymbol{g} + \boldsymbol{e} \tag{14}$$

where $\boldsymbol{g} = R\theta$. Since $\boldsymbol{u}_i$ and $\boldsymbol{u}_j$ are orthogonal for $i \neq j$, the sum of squares of $y(k)$ can be written as

$$\boldsymbol{y}^T \boldsymbol{y} = \sum_{i=1}^{M} \boldsymbol{g}_i \boldsymbol{u}_i^T \boldsymbol{u}_i + \boldsymbol{e}^T \boldsymbol{e}. \tag{15}$$

Dividing $N$ on both side of (15), it can be seen that $\Sigma \boldsymbol{g}_i \boldsymbol{u}_i^T \boldsymbol{u}_i / N$ is the part of the output variance $\boldsymbol{y}^T \boldsymbol{y}/N$ which can be explained by the regressors and $\boldsymbol{e}^T \boldsymbol{e}/N$ is the unexplained variance of $\boldsymbol{y}$. Thus, $\boldsymbol{g}_i \boldsymbol{u}_i^T \boldsymbol{u}_i / N$ is the increment to the output variance introduced by $\boldsymbol{u}_i$, and an *error reduction ratio* due to $\boldsymbol{u}_i$ can be defined as [5]

$$[\text{err}]_i = \frac{\boldsymbol{g}_i^2 \boldsymbol{u}_i^T \boldsymbol{u}_i}{\boldsymbol{y}^T \boldsymbol{y}}, \qquad 1 \leq i \leq M. \tag{16}$$

This ratio offers a simple means of seeking a subset of important regressors in a forward-regression manner. For fuzzy modeling problem, the subset of important regressors corresponds to a subset of important fuzzy rules. If it is decided that $r$ rules are used to construct a fuzzy model, then the first $r$ rules with the largest error reduction ratios will be selected. The complete OLS algorithm is summarized as follows.

**OLS Algorithm**

1) For $1 \leq i \leq M$, compute

$$\begin{aligned} \boldsymbol{u}_1^{(i)} &= \boldsymbol{p}_i \\ \boldsymbol{g}_1^{(i)} &= \left(\boldsymbol{u}_1^{(i)}\right)^T \boldsymbol{y} \Big/ \left(\left(\boldsymbol{u}_1^{(i)}\right)^T \boldsymbol{u}_1^{(i)}\right) \\ [\text{err}]_1^{(i)} &= \left(\boldsymbol{g}_1^{(i)}\right)^2 \left(\boldsymbol{u}_1^{(i)}\right)^T \boldsymbol{u}_1^{(i)} / (\boldsymbol{y}^T \boldsymbol{y}) \end{aligned} \tag{17}$$

Find

$$[\text{err}]_1^{(i_1)} = \max\{[\text{err}]_1^{(i)}, 1 \leq i \leq M\} \tag{18}$$

and select

$$\boldsymbol{u}_1 = \boldsymbol{u}_1^{(i_1)} = \boldsymbol{p}_{i_1}, \boldsymbol{g}_1 = \boldsymbol{g}_1^{(i_1)}. \tag{19}$$

2) For $2 \le k \le r$ and $1 \le i \le M, i \ne i_1, \ldots, i \ne i_{k-1}$, compute

$$\alpha_{jk}^{(i)} = \boldsymbol{u}_j^T \boldsymbol{p}_i / (\boldsymbol{u}_j^T \boldsymbol{u}_j),\ 1 \le j \le k$$

$$\boldsymbol{u}_k^{(i)} = \boldsymbol{p}_i - \sum_{j=1}^{k-1} \alpha_{jk}^{(i)} \boldsymbol{u}_j$$

$$\boldsymbol{g}_k^{(i)} = \left(\boldsymbol{u}_k^{(i)}\right)^T \boldsymbol{y} \Big/ \left(\left(\boldsymbol{u}_k^{(i)}\right)^T \boldsymbol{u}_k^{(i)}\right)$$

$$[\mathrm{err}]_k^{(i)} = \left(\boldsymbol{g}_k^{(i)}\right)^2 \left(\boldsymbol{u}_k^{(i)}\right)^T \boldsymbol{u}_k^{(i)} / (\boldsymbol{y}^T \boldsymbol{y}) \tag{20}$$

Find

$$[\mathrm{err}]_k^{(i_k)} = \max\{[\mathrm{err}]_k^{(i)},\ 1 \le i \le M,\ i \ne i_1, \ldots, i \ne i_{k-1}\} \tag{21}$$

and select

$$\boldsymbol{u}_k = \boldsymbol{u}_k^{(i_k)},\ \boldsymbol{g}_k = \boldsymbol{g}_k^{(i_k)}. \tag{22}$$

By these two steps we can establish the matrix $U$ and the parameters vector $\boldsymbol{g}$. To obtain the original parameter vector $\theta$, Wang and Mendel [32] suggested to solve the triangular equation

$$R^{(r)} \theta^{(r)} = \boldsymbol{g}^{(r)} \tag{23}$$

where

$$R^{(r)} = \begin{bmatrix} 1 & \alpha_{12}^{(i_2)} & \alpha_{13}^{(i_3)} & \ldots & \alpha_{1r}^{(i_r)} \\ 0 & 1 & \alpha_{23}^{(i_3)} & \ldots & \alpha_{2r}^{(i_r)} \\ \ldots & \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 1 & \ldots & \alpha_{r-1,r}^{(i_r)} \\ 0 & 0 & 0 & \ldots & 1 \end{bmatrix} \tag{24}$$

$$\boldsymbol{g}^{(r)} = [\boldsymbol{g}_1, \boldsymbol{g}_2, \ldots, \boldsymbol{g}_r]^T, \quad \theta^{(r)} = [\theta_1, \theta_2, \ldots, \theta_r]^T \tag{25}$$

by *back substitution* [since $R^{(r)}$ is upper triangular]. However, the parameter vector $\theta^{(r)}$ obtained this way will still contain information from all $M$ initial rules, even though only the most important $r$ rules have been established by OLS. This is due to the normalization step used in constructing the firing strength matrix $P$ that has taken into account all $M$ rules. A better solution is to construct a new firing strength matrix $P^{(r)} \in \Re^{N \times r}$ by solely using the $r$ most important rules and then solve $\theta^{(r)}$ based on $P^{(r)} \theta^{(r)} = \boldsymbol{y}$. This may be performed by running the OLS a second time [11] or using the SVD method introduced in Section II-B. The latter method is used in this paper.

The OLS method tries to select the most important fuzzy rules based on their contributions of variance to the variance of the output. This is quite similar to the strategy for selecting components in *principal component regression* [16] where those components with large variances are remained in the regression model[3].

[3] Selecting principal components on the basis of variance reduction is a standard practice in principal component regression. However, some researchers argue that low variance for a principal component does not necessarily imply that the corresponding component is unimportant in the regression and deletion based solely on variance can be dangerous if low-variance components have predictive value. For an interesting discussion of this controversy, we refer the reader to Jolliffe [16].

### B. Eigenvalue Decomposition (ED) Method

This method was proposed in Nisbet, Mulgrew, and McLaughlin [24] to construct reduced *radial basis function* (RBF) *networks* and *Volterra series polynomials*. In particular, instead of considering the original linear equation, which has the same form as (7), Nisbet *et al*. focused on the following *normal equation*:

$$\Phi_{pp} \theta = \Phi_{py} \tag{26}$$

where $\Phi_{pp} = P^T P \in \Re^{M \times M}$ and $\Phi_{py} = P^T \boldsymbol{y} \in \Re^M$ are called the *correlation matrix* and the *cross-correlation vector*, respectively [9]. Computing the *eigenvalue decomposition* of $\Phi_{pp}$ yields

$$\Phi_{pp} = V \Lambda V^T \tag{27}$$

where $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_M) \in \Re^{M \times M}$ is a diagonal matrix whose diagonal entries are the *eigenvalues* of $\Phi_{pp}$, and $V = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_M] \in \Re^{M \times M}$ is an orthogonal matrix whose columns are the corresponding *eigenvectors*. Since $\Phi_{pp}$ is symmetric and nonnegative definite, the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_M$ are all real and nonnegative [9]. Thus, the decomposition can be arranged such that the eigenvalues appear in a descending order, i.e., $\lambda_1 \ge \lambda_2 \ge \ldots \ge \lambda_M \ge 0$.

If the matrix $\Phi_{pp}$ contains zero or near-zero eigenvalues, the matrix is *rank deficient* [26]. Nisbet *et al*. just used this information to remove the redundant radial basis functions and Volterra terms which amounts to eliminating some entries in $\theta$. In order to give an indication as to which entries of $\theta$ should be removed, Nisbet *et al*. defined the following *measure index vector* $\boldsymbol{I}_{\mathrm{ED}} \in \Re^M$

$$\begin{aligned} \boldsymbol{I}_{\mathrm{ED}} &= [I_{\mathrm{ED}_1}, I_{\mathrm{ED}_2}, \ldots, I_{\mathrm{ED}_M}]^T = \mathrm{diag}(V_r V_r^T) \\ &= \mathrm{diag}(\boldsymbol{v}_1 \boldsymbol{v}_1^T + \boldsymbol{v}_2 \boldsymbol{v}_2^T + \ldots + \boldsymbol{v}_r \boldsymbol{v}_r^T) \end{aligned} \tag{28}$$

where $V_r = [\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_r] \in \Re^{M \times r}$ contains the first $r$ columns of $V$. The position of the $r$ largest indexes in $\boldsymbol{I}_{\mathrm{ED}}$ indicates the position of the entries in $\theta$ that should be retained, while the position of the rest $M - r$ indexes in $\boldsymbol{I}_{\mathrm{ED}}$ indicates the position of the entries in $\theta$ that should be removed.

The resulting $r$-dimensional parameter vector, denoted by $\theta^{(r)} = [\theta_1, \theta_2, \ldots, \theta_r]^T$, was solved from the reduced normal equation

$$\Phi_{pp}^{(r)} \theta^{(r)} = \Phi_{py}^{(r)} \tag{29}$$

where $\Phi_{pp}^{(r)} \in \Re^{r \times r}$ was obtained by removing the rows and columns of $\Phi_{pp}$ that were associated with the $M - r$ smallest indexes in $\boldsymbol{I}_{\mathrm{ED}}$, and similarly, $\Phi_{py}^{(r)} \in \Re^r$ was obtained by removing the rows of $\Phi_{py}$. In this way, Nisbet *et al*. obtained a reduced RBF network and a reduced Volterra series polynomial model that had better numerical property and prediction performance.

Due to the functional similarity between RBF networks and fuzzy inference systems [15], this method can also be used to construct a reduced fuzzy model. The removal of the entries of $\theta$ and the associated rows and columns of $\Phi_{pp}$ as well as the associated rows of $\Phi_{py}$ amounts to removing the redundant or less important rules from a rule base. However,

unlike RBF and Volterra series polynomial modeling where no normalization step is done before forming the matrices $\Phi_{pp}$ and $\Phi_{py}$, fuzzy modeling does require such a step. This implies that, after the removal of the associated rows and columns of $\Phi_{pp}$ as well as the associated rows of $\Phi_{py}$, the resulting $\Phi_{pp}^{(r)}$ and $\Phi_{py}^{(r)}$ still contain information from the deleted rules. In order to eliminate the effect, a new normalization step should be implemented for the entries of $\Phi_{pp}^{(r)}$ and $\Phi_{py}^{(r)}$. Note that removing the rows and columns of $\Phi_{pp}$ as well as the rows of $\Phi_{py}$ is tantamount to eliminating the associated columns of $P$. Thus $\Phi_{pp}^{(r)}$ and $\Phi_{py}^{(r)}$ can be directly formed by computing $(P^{(r)})^T P^{(r)}$ and $(P^{(r)})^T y$, respectively, where $P^{(r)} \in R^{N\times r}$ corresponds to the $r$ columns of $P$ which are associated with the $r$ most important rules in the rule base. The complete algorithm is given below.

**ED Algorithm**

1) Compute the eigenvalue decomposition of $\Phi_{pp}$ using (27) and get $\Lambda$ and $V$. Determine the number of fuzzy rules that should be retained (or equivalently that should be removed) by checking the eigenvalues $\Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_M)$. The number of the first $r$ largest eigenvalues indicates the number of fuzzy rules that should be retained.
2) Partition $V$ into $V = [V_r, V_{M-r}]$, where $V_r$ and $V_{M-r}$ correspond to the first $r$ columns and the last $M-r$ columns of $V$, respectively.
3) Compute the measure index vector $\boldsymbol{I}_{\text{ED}}$ using (28). The position of the $r$ largest indexes in $\boldsymbol{I}_{\text{ED}}$ indicates the position of the rows and columns of $\Phi_{pp}$ as well as the rows of $\Phi_{py}$ (or equivalently the position of the columns of the firing strength matrix $P$) that should be remained, i.e., the position of the $r$ most important rules in the rule base.
4) Construct a reduced $N$-by-$r$ firing strength matrix $P^{(r)}$ based on the selected $r$ most important fuzzy rules, and form $\Phi_{pp}^{(r)}$ and $\Phi_{py}^{(r)}$.
5) Solve $\theta^{(r)}$ from (29) using the SVD method introduced in Section II-B.

Our computer simulations showed that the algorithm could usually result in a compact fuzzy model with low fitting and prediction error. However, a possible drawback of this algorithm is that it needs to form the correlation matrix $\Phi_{pp}$ and then computes its eigenvalue decomposition. It is known that such a procedure often causes numerical instability [18]. A better practice is to work directly on the firing strength matrix $P$ and avoid explicitly forming the correlation matrix. The SVD based methods introduced in the following sections can better meet the concern.

### C. SVD-QR with Column Pivoting Method

The *SVD-QR with column pivoting algorithm* was originally proposed by Golub, Klema, and Stewart [7] for solving the problem of *subset selection* in regression analysis and was suggested in Mouzouris and Mendel [22] to extract the most important fuzzy rules from a given rule base. This basic idea of this method is to replace $P\theta$ in (7) by $P^{(r)}\theta^{(r)}$ where $P^{(r)} \in \Re^{N\times r}$ consists of $r$ columns of $P$. The position of these columns in $P$ determines which rules in the rule base are to be used in approximating the observation vector $\boldsymbol{y}$. The principle of this method is introduced as follows.

Let the SVD of $P \in \Re^{N\times M}$ be given by (8), and define the matrix $P^{(r)} \in \Re^{N\times r}$ by

$$\begin{matrix} P\Pi = [P^{(r)} & P^{(M-r)}] \\ \quad r & \quad M-r \end{matrix} \tag{30}$$

where $\Pi \in \Re^{M\times M}$ is a *permutation matrix*. Golub *et al*. showed that if

$$\begin{matrix} \Pi^T V = \begin{bmatrix} V'_{11} & V'_{12} \\ V'_{21} & V'_{22} \end{bmatrix} & \begin{matrix} r \\ M-r \end{matrix} \\ \quad r* \quad\quad M-r \end{matrix} \tag{31}$$

and $V'_{11} \in \Re^{M\times M}$ is nonsingular, then

$$\frac{\sigma_r(P)}{\|(V'_{11})^{-1}\|_2} \le \sigma_r\left(P^{(r)}\right) \le \sigma_r(P) \tag{32}$$

where $\sigma_r(P^{(r)})$ and $\sigma_r(P)$ are the $r$th singular value of $P^{(r)}$ and $P$, respectively.

This result suggests that in order to obtain a sufficiently independent subset of columns (correspondingly the most important subset of fuzzy rules), the permutation matrix $\Pi$ should be chosen so that the resulting $V'_{11}$ submatrix is as well-conditioned as possible and hence $\|(V'_{11})^{-1}\|_2$ is as small as possible. This implies that the computed subset $P^{(r)}$ tends to maximize its minimal singular value $\sigma_r(P^{(r)})$. A heuristic solution to this problem, suggested by Golub *et al*. is obtained by computing the *QR with column pivoting decomposition* [8] of the matrix $[V_{11}^T \quad V_{21}^T]$, where $V_{11}^T \in \Re^{M\times M}$ and $V_{21}^T \in \Re^{(M-r)\times M}$ are defined by

$$\begin{matrix} V = \begin{bmatrix} V_{11} & V_{12} \\ V_{21} & V_{22} \end{bmatrix} & \begin{matrix} r \\ M-r \end{matrix} . \\ \quad r \quad\quad M-r \end{matrix} \tag{33}$$

In particular, if we apply QR with column pivoting to compute

$$\begin{matrix} Q^T[V_{11}^T \quad V_{21}^T]\Pi = \begin{bmatrix} R_{11} & R_{12} \end{bmatrix} \\ \quad r \quad\quad M-r \end{matrix} \tag{34}$$

where $Q \in \Re^{r\times r}$ is orthogonal, $\Pi \in \Re^{M\times M}$ is the permutation matrix, and $R_{11} \in \Re^{r\times r}$ is upper triangular, then (31) implies

$$\begin{bmatrix} V'_{11} \\ V'_{21} \end{bmatrix} = \Pi^T \begin{bmatrix} V_{11} \\ V_{21} \end{bmatrix} = \begin{bmatrix} R_{11}^T Q^T \\ R_{12}^T Q^T \end{bmatrix}. \tag{35}$$

Note that $R_{11}$ is nonsingular and that $\|(V'_{11})^{-1}\|_2 = \|R_{11}^{-1}\|_2$. Heuristically, column pivoting tends to produce a well-conditioned $R_{11}$, and so the overall process tends to produce a well-conditioned $V'_{11}$.

Each column of $\Pi$ contains exactly one "1" (other entries are all 0's). The position of the entries 1's in $\Pi$ determines the position of columns of $P$ in $P\Pi$ as well as the position

of the associated rules in the rule base [22]. To illustrate this, suppose we have

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \\ p_{41} & p_{42} & p_{43} \end{bmatrix} \quad \text{and} \quad \Pi = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}. \tag{36}$$

Then

$$P\Pi = \begin{bmatrix} p_{12} & p_{13} & p_{11} \\ p_{22} & p_{23} & p_{21} \\ p_{32} & p_{33} & p_{31} \\ p_{42} & p_{43} & p_{41} \end{bmatrix}. \tag{37}$$

If we want to build a model using two fuzzy rules, then the second and the third rules should be selected.

The consequent constituents of the $r$ most important fuzzy rules can be solved from the reduced equation

$$P^{(r)}\theta^{(r)} = \boldsymbol{y} \tag{38}$$

where $\theta^{(r)} = [\theta_1, \theta_2, \ldots, \theta_r]^T$, and $P^{(r)}$ needs to be renormalized as before by solely using the $r$ most important fuzzy rules.

In summary, the SVD-QR with column pivoting consists of the following main steps.

**SVD-QR Algorithm**

1) Compute the SVD of $P$, i.e., $P = U\Sigma V^T$. Save $\Sigma$ and $V$.
2) Check the singular values in $\Sigma = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_M)$, and determine the number of fuzzy rules (denoted by $r$) used to construct the model, where $r \leq \text{rank}(P)$.
3) Partition $V$ according to (33) and form $[V_{11}^T \quad V_{21}^T]$. Apply QR with column pivoting to $[V_{11}^T \quad V_{21}^T]$ and get the permutation matrix $\Pi$. The position of 1's in the first $r$ columns of $\Pi$ indicates the position of the $r$ most important fuzzy rules in the rule base.
4) Construct the normalized matrix $P^{(r)} \in \Re^{N\times r}$ based on the $r$ most important fuzzy rules.
5) Solve $P^{(r)}\theta^{(r)} = \boldsymbol{y}$ for $\theta^{(r)}$ using the SVD method introduced in Section II-B.

This method is closely related to the eigenvalue decomposition method introduced in the preceding section. This can be seen by substituting (8) into $\Phi_{pp}$

$$\Phi_{pp} = P^T P = (U\Sigma V^T)^T U\Sigma V^T = V\Sigma^2 V^T. \tag{39}$$

Comparing (39) and (27) and noticing that $\Sigma^2$ is identical to $\Lambda$ (since $\Phi_{pp}$ is symmetric and nonnegative definite), we know that the two algorithms are equivalent by this stage. The only difference between the two algorithms lies in the determination of the entries of $\theta$ (or equivalently the columns of $P$) that should be retained: ED uses the measure index vector $\boldsymbol{I}_{\text{ED}}$, while SVD-QR uses the permutation matrix $\Pi$. However, recalling that the implication of the QR with column pivoting is to make column interchange during each orthogonalization step of the QR decomposition of the matrix $[V_{11}^T \quad V_{21}^T](= V_r^T)$ so that the columns with the largest 2-norms are first orthogonalized and the largest pivots are moved to the upper left-hand corner of $[R_{11} \quad R_{12}]$, and noticing that the effect of the column interchange is the same as that of making the corresponding interchange in the columns of $[V_{11}^T \quad V_{21}^T]$ before the QR decomposition is begun [34], the permutation matrix $\Pi$ can thus actually be determined by computing the measure index vector

$$\begin{aligned} \boldsymbol{I}_{\text{SVD-QR}} &= [I_{\text{SVD-QR}_1}, I_{\text{SVD-QR}_2}, \ldots, I_{\text{SVD-QR}_M}]^T \\ &= [\|\boldsymbol{v}_1\|_2, \|\boldsymbol{v}_2\|_2, \ldots, \|\boldsymbol{v}_M\|_2]^T \\ &= [(\boldsymbol{v}_1\boldsymbol{v}_1^T)^{1/2}, (\boldsymbol{v}_2\boldsymbol{v}_2^T)^{1/2}, \ldots, (\boldsymbol{v}_M\boldsymbol{v}_M^T)^{1/2}]^T \end{aligned} \tag{40}$$

where $\boldsymbol{v}_i \in \Re^r$ is the $i$th column of $[V_{11}^T \quad V_{21}^T]$. The entry "1" in each column of $\Pi$ corresponds to one of the indexes in $\boldsymbol{I}_{\text{SVD-QR}}$. For example, the entry "1" in the first column of $\Pi$ corresponds to the largest index of $\boldsymbol{I}_{\text{SVD-QR}}$, and its row coordinate corresponds to the position of the largest index in $\boldsymbol{I}_{\text{SVD-QR}}$. In other words, if the $i$th entry of $\boldsymbol{I}_{\text{SVD-QR}}$ is the largest, the $(i, 1)$th entry of $\Pi$ will be 1; if the $j$th entry of $\boldsymbol{I}_{\text{SVD-QR}}$ is the second largest, the $(j, 2)$th entry of $\Pi$ will be 1, and so on. In particular, if we substitute (40) with (28) or substitute (28) with (40), the two algorithms are essentially the same. Nevertheless, the SVD-QR algorithm does not need to explicitly form the correlation matrix and thus avoid the introduction of unnecessary numerical and rounding errors. Moreover, it is known that the singular values can be computed more efficiently with much greater numerical and computational stability than the eigenvalues [18], and hence the SVD-QR algorithm is superior to the ED algorithm from an implementation point of view.

### D. Total Least-Squares Method

The *total least-squares* (TLS) *method* is a technique designed to compensate for data errors for linear parameter estimation problem which has the form of (7). In solving such a problem, the ordinary *least-squares* (LS) *method* assumes the matrix $P$ (which is usually called *measurement matrix* in parameter estimation problems) to be free of error and all errors to be confined to the observation vector $\boldsymbol{y}$. However, this assumption is frequently unrealistic: sampling error, human errors, modeling errors, and instrument errors may imply inaccuracies of the matrix $P$ as well. This is especially true for the parameter estimation problem of fuzzy models where the matrix $P$ is usually constructed based on the membership functions provided by human experts which inevitably contains errors. TLS offers a promising parameter estimation method that is appropriate when there are errors in both the observation vector $\boldsymbol{y}$ and the measure matrix $P$. A detailed introduction of the TLS, its theory, computing algorithms, and many interesting applications, can be found in [30].

A subset selection method based on TLS was developed in Van Huffel and Vandewalle [29] and used here to extract the most important fuzzy rules from a given rule base. This method extends the SVD-QR with column pivoting method by incorporating information included in the observation vector $\boldsymbol{y}$. In particular, the SVD is applied to an *extended matrix*, defined by $[P \quad \boldsymbol{y}]$

$$[P \quad \boldsymbol{y}] = \tilde{U}\tilde{\Sigma}\tilde{V}^T \tag{41}$$

where $\tilde{U} \in \Re^{N\times N}$ and $\tilde{V} \in \Re^{(M+1)\times(M+1)}$, $\tilde{\Sigma} = \text{diag}(\tilde{\sigma}_1, \tilde{\sigma}_2, \ldots, \tilde{\sigma}_{M+1}) \in \Re^{N\times(M+1)}$. Partition $\tilde{V}$ as

$$\tilde{V} = \begin{matrix} \begin{bmatrix} \tilde{V}_{11} & \tilde{V}_{12} \\ \tilde{V}_{21} & \tilde{V}_{22} \\ \tilde{V}_{31} & \tilde{V}_{32} \end{bmatrix} & \begin{matrix} r \\ M-r \\ 1 \end{matrix} \\ \begin{matrix} r & M-r \end{matrix} & \end{matrix} \tag{42}$$

where $r \le \text{rank}[P \quad \boldsymbol{y}]$ corresponds to the number of fuzzy rules which are selected to construct the model. Apply QR with column pivoting to compute

$$\tilde{Q}^T[\tilde{V}_{11}^T \quad \tilde{V}_{21}^T]\tilde{\Pi} = \tilde{Q}^T[\tilde{V}_{11}^T \quad \tilde{V}_{21}^T] \begin{matrix} \begin{bmatrix} \tilde{R}_{11} & \tilde{R}_{12} \end{bmatrix} \\ \begin{matrix} r & M-r \end{matrix} \end{matrix} \tag{43}$$

where $\tilde{Q} \in \Re^{r\times r}$ is orthogonal, $\tilde{\Pi} \in \Re^{M\times M}$ is a permutation, and $\tilde{R}_{11} \in \Re^{r\times r}$ is upper triangular. Set

$$P\tilde{\Pi} = \begin{matrix} [\tilde{P}^{(r)} & \tilde{P}^{(M-r)}] \\ r & M-r. \end{matrix} \tag{44}$$

Similarly to (32), it can be proved [29]

$$\frac{\sigma_r([P \quad \boldsymbol{y}])}{\|(\tilde{V}_{11}')^{-1}\|_2} \le \sigma_r([\tilde{P}^{(r)} \quad \boldsymbol{y}]) \le \sigma_r([P \quad \boldsymbol{y}]). \tag{45}$$

Like the permutation matrix $\Pi$ in the SVD-QR method, the permutation matrix $\tilde{\Pi}$ orders the columns of $P$ (and correspondingly the rules in the rule base) such that the $r$ "most important columns" enter $\tilde{P}^{(r)}$ in (44). The most important columns can also be equivalently determined using a measure index similar to that defined in (40), which is given by

$$\begin{aligned} \boldsymbol{I}_{\text{TLS}} &= [I_{\text{TLS}_1}, I_{\text{TLS}_2}, \ldots, I_{\text{TLS}_M}]^T \\ &= [\|\tilde{\boldsymbol{v}}_1\|_2, \|\tilde{\boldsymbol{v}}_2\|_2, \ldots, \|\tilde{\boldsymbol{v}}_m\|_2]^T \\ &= [(\tilde{\boldsymbol{v}}_1\tilde{\boldsymbol{v}}_1^T)^{1/2}, (\tilde{\boldsymbol{v}}_2\tilde{\boldsymbol{v}}_2^T)^{1/2}, \ldots, (\tilde{\boldsymbol{v}}_M\tilde{\boldsymbol{v}}_M^T)^{1/2}]^T \end{aligned} \tag{46}$$

where $\tilde{\boldsymbol{v}}_i \in \Re^r$ is the $i$th column of $[\tilde{V}_{11}^T \quad \tilde{V}_{21}^T]$.

As before, the consequent constituents of the $r$ most important fuzzy rules can be solved from the reduced equation

$$\tilde{P}^{(r)}\tilde{\theta}^{(r)} = \boldsymbol{y} \tag{47}$$

where $\tilde{P}^{(r)}$ is reconstructed by solely using the $r$ most important fuzzy rules.

Instead of seeking the LS solution, the TLS solution of $\tilde{\theta}^{(r)}$ has been advocated in Van Huffel and Vandewalle [29]. In particular, the solution can be computed by

$$\tilde{\theta}^{(r)} = \frac{1}{\hat{v}_{r+1,\,r+1}}[\hat{v}_{1,\,r+1}, \hat{v}_{2,\,r+1}, \ldots, \hat{v}_{r,\,r+1}]^T \tag{48}$$

where $\hat{v}_{1,\,r+1}, \hat{v}_{2,\,r+1}, \ldots, \hat{v}_{r,\,r+1}$ are the entries of the last column of $\hat{V} \in \Re^{(r+1)\times(r+1)}$, which is obtained from the SVD of $[\tilde{P}^{(r)} \quad \boldsymbol{y}]$, i.e.,

$$[\tilde{P}^{(r)} \quad \boldsymbol{y}] = \hat{U}\hat{\Lambda}\hat{V}^T. \tag{49}$$

In summary, the TLS method consists of the following main steps.

**TLS Algorithm**

1) Compute the SVD of $[P \quad \boldsymbol{y}]$ using (41). Save $\tilde{\Sigma}$ and $\tilde{V}$.
2) Check the singular values in $\tilde{\Sigma} = \text{diag}(\tilde{\sigma}_1, \tilde{\sigma}_2, \ldots, \tilde{\sigma}_{M+1})$, and determine the number of fuzzy rules (denoted by $r$) used to construct the model, where $r \le \text{rank}([P \quad \boldsymbol{y}])$.
3) Partition $\tilde{V}$ according to (42) and save $[\tilde{V}_{11}^T \quad \tilde{V}_{21}^T]$. Apply QR with column pivoting to $[\tilde{V}_{11}^T \quad \tilde{V}_{21}^T]$ and get the permutation matrix $\tilde{\Pi}$. The position of 1's in the first $r$ columns of indicates the position of the first $r$ most important fuzzy rules in the rule base. Alternatively, the measure index vector $\boldsymbol{I}_{\text{TLS}}$ defined in (46) can also be used to determine the position of the first $r$ most important fuzzy rules in the rule base with the same results.
4) Construct the normalized matrix $\tilde{P}^{(r)} \in \Re^{N\times r}$ based on the $r$ most important fuzzy rules.
5) Compute the SVD of using (49) and save the $(r+1)$th column of $\tilde{V}$. Solve $\tilde{\theta}^{(r)}$ from (48).

The TLS method has taken into account the errors on all variables. Also, by using the information provided by the observation vector $\boldsymbol{y}$, the accuracy of the selected subset model with respect to parameter estimation and prediction can usually be improved.

### *E. Direct SVD Method*

The orthogonal transformation methods introduced in the preceding sections share a common two-step procedure: 1) determine the number of rules that should be retained or removed based on the singular values (eigenvalues), and 2) identify the position of these rules in the rule base (i.e., which rules should be retained or deleted) based on some measure index. An alternative method was proposed in [37] where, besides indicating the number of the most important fuzzy rules, the singular values themselves served as a measure index to picking these rules. Because this method uses a single SVD to determine both the number and position of the most important fuzzy rules without resort to any additional measure index, we call it *direct SVD method*.

The key of this method lies in the identification of the original position of the singular values in the firing strength matrix $P$, since each singular value is related to one of fuzzy rules and its position in $P$ indicates the position of the associated rule in the rule base. The SVD algorithm used in the previous sections is based on that of Golub and Reinsch [8]. This algorithm first uses *householder transformation* to bidiagonalize the given matrix and then an iterative algorithm, known as the *QR algorithm* (not QR decomposition) to compute the singular values of the resultant bidiagonal form. This algorithm has been implemented in some well-established numerical computation packages such as MATLAB. However, the singular values produced using these packages usually appear in a descending order. These ordered singular values do not indicate the position where they appear in the matrix $P$. This poses a difficulty if we want to relate these singular values with their corresponding rules.

A different algorithm, known as the *Jacobi algorithm* was used in [37] to compute the SVD of $P$. This algorithm first applies a sequence of *one-sided Jacobi transformation* [10] to

$P$ and get

$$B = U^T P \tag{50}$$

where $B$ is row orthogonal and $U$ is both row and column orthogonal, then computes the QR decomposition of $B^T$ and its transpose, that is

$$B = (B^T)^T = (Q_B R_B)^T = R_B^T Q_B^T \tag{51}$$

where $R_B$ is actually a diagonal matrix whose entries consist of the singular values of $P$ (Assume that a nonnegative handing step has been done). Let

$$\Sigma = R_B^T \quad \text{and} \quad V^T = Q_B^T \tag{52}$$

then from (50)–(52) a complete SVD

$$P = U \Sigma V^T \tag{53}$$

is formed. Using this algorithm we can obtain the singular values of the firing strength matrix $P$ that appear in their original order.

The main steps of the direct SVD method are summarized as follows.

**D-SVD Algorithm**

1) Compute the SVD of $P$ using the Jacobi method and get the singular values that appear in their original order in the matrix.
2) Select $r$ fuzzy rules to construct the model, where $r$ corresponds to the $r$ largest singular values whose position indicate the position of the $r$ most important fuzzy rules in the rule base.
3) Construct the reduced firing strength matrix $P^{(r)} \in \Re^{N \times r}$ based on the $r$ most important fuzzy rules.
4) Solve $P^{(r)}\theta^{(r)} = \boldsymbol{y}$ for $\theta^{(r)}$ using the SVD method introduced in Section II-B.

The advantage of this method is its simplicity and intuitiveness; it uses a single SVD procedure to determine both the number and position of important fuzzy rules in the rule base and does not need to introduce any additional measure index. Moreover, the Jacobi algorithm for SVD implementation in this method is computationally more reliable than the Golub–Reinsch algorithm [6].

## IV. Comparative Study

We have conducted numerous computer simulations to evaluate the performance of the above five rule selection methods [38], [39]. Here only one typical example is presented. We consider the second-order nonlinear plant

$$y(k) = f(y(k-1), y(k-2)) + u(k), \tag{54}$$

where

$$f(y(k-1), y(k-2)) = \frac{y(k-1)y(k-2)[y(k-1) - 0.5]}{1 + y^2(k-1) + y^2(k-2)}. \tag{55}$$

The nonlinear component $f$ in this plant, which is usually called the "unforced system" in control literature (see, e.g., [23]), has an equilibrium state $(0, 0)$ in the state space. This implies that while in equilibrium without an input, the output of the plant is the sequence $\{0\}$. Fig. 1 shows the trajectory of the unforced system in the state space.

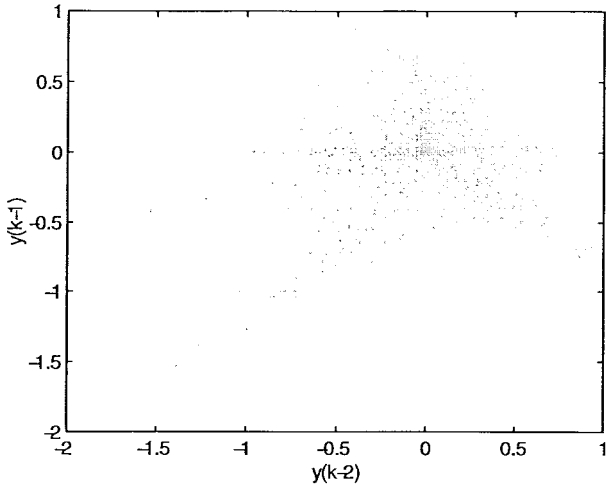



Fig. 1. Trajectory of the unforced system.

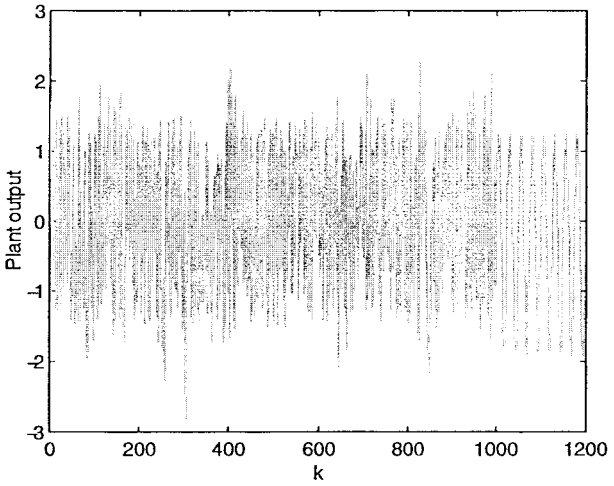
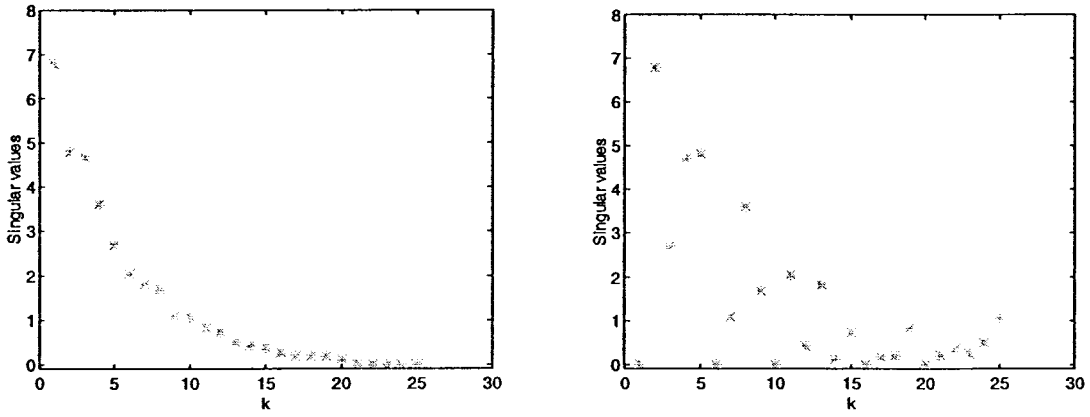



Fig. 2. Output of the plant model.

We want to approximate the nonlinear component $f$ using the fuzzy model (1). For this purpose, 1200 simulated data points were generated from the plant model (54). The first 1000 data points were obtained by assuming a random input signal $u(k)$ uniformly distributed in the interval $[-1.5, 1.5]$, while the last 200 data points were obtained by using a sinusoid input signal $u(k) = \sin(2\pi k/25)$. The 1200 simulated data points are shown in Fig. 2.

We used the first 1000 data points to build a fuzzy model. The performance of the resulting model was tested using the remaining 200 data points. We chose $y(k-1)$ and $y(k-2)$ as the input variables and arbitrarily set the number of fuzzy rules to 25. The Gaussian functions defined in (2) were used to express the membership functions of $y(k-1)$ and $y(k-2)$ with the parameters shown in Table I.[4]

[4] Initially, only 20 two-dimensional Gaussian membership functions were considered, each of which can be seen as the product of two one-dimensional membership functions for the input variables $y(k-1)$ and $y(k-2)$. The centers $c_{ij}$ and the widths $\sigma_{ij}$ of the 20 Gaussian membership functions were determined using the $k$-means clustering algorithm and the nearest-neighbor heuristic, respectively [21]. We then introduced five additional Gaussian membership functions with artificially selected parameters in order to emulate the effect caused by the redundant and less important fuzzy rules.




(a) (b)

Fig. 3. Singular values of the 1000-by-25 firing strength matrix: (a) descending order and (b) original order.

TABLE I
PARAMETERS OF GAUSSIAN MEMBERSHIP FUNCTIONS

| i | $c_{i1}$ | $c_{i2}$ | $\sigma_{i1}$ | $\sigma_{i2}$ |
|---|---|---|---|---|
| **1** | **0.0930** | **-0.3630** | **0.7095** | **0.7095** |
| **2** | **0.0933** | **-0.3632** | **0.7095** | **0.7095** |
| 3 | 1.3828 | -0.6617 | 0.6271 | 0.6271 |
| 4 | -1.0414 | 1.5397 | 0.7969 | 0.7969 |
| **5** | **-1.8130** | **-1.6470** | **1.3205** | **1.3205** |
| **6** | **-1.8125** | **-1.6469** | **1.3205** | **1.3205** |
| 7 | 0.7776 | -1.1555 | 0.7800 | 0.7800 |
| 8 | 0.1898 | 1.0142 | 0.6141 | 0.6141 |
| 9 | -0.4052 | 0.2798 | 0.8099 | 0.8099 |
| **10** | **-0.6613** | **-0.4846** | **0.0100** | **0.0100** |
| 11 | -0.6613 | -0.4846 | 0.7051 | 0.7051 |
| 12 | 0.9529 | -0.3965 | 0.6313 | 0.6313 |
| 13 | 0.7860 | 0.7723 | 0.6177 | 0.6177 |
| 14 | 0.4329 | 0.1910 | 0.6652 | 0.6652 |
| **15** | **1.2940** | **1.0740** | **0.6474** | **0.6474** |
| **16** | **1.2942** | **1.0738** | **0.6474** | **0.6474** |
| 17 | 0.6801 | 1.4083 | 0.6370 | 0.6370 |
| 18 | 1.2656 | 0.2698 | 0.7156 | 0.7156 |
| 19 | -0.3846 | 1.1827 | 0.6772 | 0.6772 |
| **20** | **-1.2642** | **-0.1808** | **0.0100** | **0.0100** |
| 21 | -1.2642 | -0.1808 | 0.7907 | 0.7907 |
| 22 | -0.9099 | -1.1750 | 0.7728 | 0.7728 |
| 23 | -0.1008 | -1.1384 | 0.8046 | 0.8046 |
| 24 | -1.1533 | 0.7037 | 0.8517 | 0.8517 |
| 25 | 1.7691 | -1.2798 | 0.8746 | 0.8746 |

These 25 rules were labeled by the number $1, 2, \ldots, 25$ which indicated the position of the rules in the rule base as well as the associated combinations of membership functions. For example, the number "1" indicates the first rule in the rule base which is associated with the membership functions combination $\{A_{11}(x_1), A_{12}(x_2)\}$ [where $x_1 \equiv y(k-1)$, $x_2 \equiv y(k-2)$], while the number "25" indicates the 25th rule in the rule base which is associated with the membership functions combination $\{A_{25,1}(x_1), A_{25,2}(x_2)\}$.

Each row of Table I is associated with one of the fuzzy rules in the rule base. The first two rows give the nearly same parameters of membership functions, and correspondingly the first two rules in the rule base will have the nearly same firing strengths. This implies that there exists redundancy between the two rules and removing either of them will not affect the performance of the model significantly. The same observation holds for rules 5 and 6 as well as rules 15 and 16, which correspond to the rows 5 and 6 as well as the rows 15 and 16, respectively. The widths of the Gaussian functions in the rows 10 and 20 have a small value of 0.01, and thus the resultant membership functions will have a low grade (geometrically, the membership functions will become very "narrow" [25]). This implies the rules 10 and 20 have a low firing strength and removal of the two rules will not sacrifice the accuracy of the model greatly.

For each of the 1000 input data points $\{y(k-1), y(k-2)\}$, $k = 1, 2, \ldots, 1000$ to the model, we computed the normalized firing strengths of the 25 fuzzy rules using (4) to form a 1000-by-25 firing strength matrix $P$. Fig. 3 shows the singular values of the matrix in both descending and original order. It can be seen that there exist five zero or near-zero singular values among the 25 singular values. This indicates that five rules in the rule base are redundant or less important. Based on the singular values, we determine to retain 20 rules and eliminate five rules in the rule base. Table II shows the position of the retained 20 rules and that of the eliminated five rules indicated by the five orthogonal transformation-based methods. The final four rows of this table also give the *mean squared error* (MSE's) of the simplified models constructed using the retained 20 rules and that of the original model constructed using the complete 25 rules.

From Table II, we observe the following results.

1) Compared to the original model, the simplified models give large MSE's in the training stage, but smaller MSE's in the testing stage. This implies that removal of those redundant or less important rules from the rule base can result in a fuzzy model with better generalizing ability. This finding is consistent with the results reported in several previous studies [12], [36].
2) OLS successfully detects the two less important rules (rules 10 and 20) and a redundant rule (rule 1), but it fails to find the other two redundant rules (5 or 6, and 15 or 16).
3) ED, SVD-QR, TLS, and D-SVD successfully detect both the two less important fuzzy rules and the three redundant fuzzy rules, although the position of the

TABLE II
LOCATION OF THE RETAINED AND REMOVED FUZZY RULES IN THE RULE BASE AND THE PERFORMANCE OF THE RESULTANT MODELS

| Rule & model category | i | OLS | ED | SVD-QR | TLS | D-SVD |
|---|---|---|---|---|---|---|
| Retained rules | 1 | 5 | 25 | 25 | 25 | 2 |
| | 2 | 24 | 4 | 4 | 17 | 5 |
| | 3 | 25 | 7 | 7 | 8 | 4 |
| | 4 | 16 | 19 | 19 | 12 | 8 |
| | 5 | 8 | 3 | 3 | 3 | 3 |
| | 6 | 21 | 24 | 24 | 24 | 11 |
| | 7 | 23 | 8 | 8 | 18 | 13 |
| | 8 | 11 | 23 | 23 | 7 | 9 |
| | 9 | 3 | 14 | 14 | 22 | 7 |
| | 10 | 22 | 13 | 13 | 23 | 25 |
| | 11 | 6 | 21 | 21 | 4 | 19 |
| | 12 | 7 | 18 | 18 | 21 | 15 |
| | 13 | 19 | 17 | 17 | 19 | 24 |
| | 14 | 4 | 22 | 22 | 13 | 12 |
| | 15 | 15 | 12 | 12 | 9 | 22 |
| | 16 | 14 | 9 | 9 | 14 | 23 |
| | 17 | 9 | 11 | 11 | 16 | 21 |
| | 18 | 12 | 2 | 2 | 6 | 18 |
| | 19 | 2 | 16 | 16 | 2 | 17 |
| | 20 | 18 | 5 | 5 | 11 | 14 |
| Eliminated rules | 21 | 1 | 6 | 10 | 20 | 20 |
| | 22 | 13 | 15 | 15 | 15 | 6 |
| | 23 | 17 | 1 | 20 | 10 | 16 |
| | 24 | 10 | 20 | 6 | 5 | 1 |
| | 25 | 20 | 10 | 1 | 1 | 10 |
| Simplified models (20 rules) | MSE (Training) | 2.5134e-4 | 6.8341e-4 | 6.8341e-4 | 6.8341e-4 | 6.8341e-4 |
| | MSE (Testing) | 3.9054e-4 | 2.3836e-4 | 2.3836e-4 | 2.3836e-4 | 2.3836e-4 |
| Original model (25 rules) | MSE (Training) | 2.3092e-4 | | | | |
| | MSE (Testing) | 4.0717e-4 | | | | |

redundant fuzzy rules identified are somewhat different. In particular, ED and SVD-QR identify the rules 1, 6 and 15 as the redundant fuzzy rules, while TLS and D-SVD identify the rules 1, 5, and 15 and the rules 1, 6, and 16 as the redundant rules, respectively. Thus, the four methods show the same performance in both the training stage and the testing stage.

4) ED and SVD-QR select the same 20 rules that should be retained with the exactly same position, while the eliminated five rules are only different in their position.

## V. CONCLUSIONS

We have presented the following five orthogonal transformation-based methods for selecting the most important fuzzy rules from a given rule base in order to construct a compact fuzzy model with good generalization ability: orthogonal least squares (OLS) method, eigenvalue decomposition (ED) method, singular value decomposition and QR with column pivoting (SVD-QR) method, total least squares (TLS) method, and direct singular value decomposition (D-SVD) method. A common attribute of these methods is that they all work on a firing strength matrix $P$ and employ some measure index (explicitly or implicitly) to select the rules that should be retained or removed.

The performance of the five methods has been evaluated using a simulated data set. Based on our analysis and simulation, we draw the following conclusions.

1) The OLS method may produce an inappropriate subset of fuzzy rules. A possible reason behind this is that the error reduction ratio defined in (16) is essentially a measure that tries to minimize the fitting error rather than simplify the model structure. If a redundant fuzzy rule has a larger (normalized) firing strength, it is quite possible that the rule will be included in the retained subset of fuzzy rules by the OLS because of its larger contribution to the total output.
2) The ED and the SVD-QR methods are essentially the same technique, although the measure indexes of two methods are slightly different. However, considering that explicitly forming the normal matrix $\Phi_{pp}$ may introduce additional error and computing the eigenvalue decomposition is numerically less reliable and computationally less efficient than computing the singular value decomposition, the SVD-QR method should be preferred.
3) The TLS method shows the same performance as the SVD-QR method in the present simulation. The subsets of fuzzy rules selected by the two methods are essentially the same, although the position of rules in

the selected subsets is different. However, since the TLS method takes into account the possible errors on both the firing strength matrix $P$ and the observation vector $\boldsymbol{y}$, it is expected that method should show superior performance when the membership functions of input variables are not well designed or input–output observations contain large noise. Moreover, the inclusion of the observation vector $\boldsymbol{y}$ in deriving the permutation matrix $\Pi$ (or equivalently the measure index vector $\boldsymbol{I}_{\mathrm{TLS}}$) should also aid in improving the selection accuracy.

4) The D-SVD method is the simplest among the five methods, but its performance is as good as that of the other methods in the present simulation.
5) There is no essential difference in terms of performance among the SVD-QR, TLS, and D-SVD methods. Because these methods often provide slightly different subset of fuzzy rules but with nearly the same performance, it is difficult to gauge which method is the best. Also, it is difficult to say which subset of fuzzy rules is the best. This observation indicates that the "importance of fuzzy rules" should be viewed as a *relative* measure. Sometimes it may be necessary to use more than one method to reveal various properties of fuzzy rules. From a scientific standpoint, examining fuzzy rules from different viewpoints is healthy and aids in intuitive upstanding and creative thinking.
6) The number of fuzzy rules that should be retained or removed is determined by the number of "big" singular values or the number of "small" singular values in the firing strength matrix. The decision as to how big is big and how small is small is really a subjective matter and there is no strict mathematical tool that can count on [26]. This decision has to be made based on the application at hand, and sometimes trial and error is inevitable. It can be helpful to introduce some "objective" criteria to help the choice [36].

The application of orthogonal transformation for the construction of fuzzy rule based models is just at its beginning and much work remains to be done. However, the presented results in this paper as well as those reported in several recent publications [22] and [37]–[39] show that this technique is very promising and should become a valuable addition to the fuzzy modeling workers' tool kit.

## REFERENCES


[1] H. Akaike, "A new look at the statistical model identification," *IEEE Trans. Automat. Contr.*, vol. 19, pp. 716–723, 1974.
[2] D. Angluin and C. Smith, "Inductive inference: Theory and methods," *ACM Comput. Surv.*, vol. 15, pp. 237–269, 1984.
[3] A. R. Barron, "Predicted squared error: A criterion for automatic model selection," *Self-Organizing Methods in Modeling*, S. J. Farlow, Ed. New York: Marcel-Dekker, 1984, pp. 87–103.
[4] H. R. Berenji and P. S. Khedkar, "Clustering in product space for fuzzy inference," *IEEE Trans. Fuzzy Syst.*, accepted for publication.
[5] S. Chen, C. F. N. Cowan, and P. M. Grant, "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Trans. Neural Networks*, vol. 2, pp. 302–309, 1991.
[6] J. Demmel and K. Veselic, "Jacobi's method is more accurate than QR," Dept. Comput. Sci., Courant Inst. Math. Sci., New York Univ., Tech. Rep. 468, 1989.
[7] G. H. Golub, V. Klema, and G. W. Stewart, "Rank degeneracy and least squares problems," Dept. Comput. Sci., Univ. Maryland, College Park, Tech. Rep. TR-456, 1976.
[8] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 2nd ed. Baltimore, MD: Johns Hopkins Univ. Press, 1989.
[9] S. Haykin, *Adaptive Filter Theory*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1991.
[10] M. R. Hestenes, "Inversion of matrices by biorthogonalization and related results," *J. Soc. Ind. Appl. Math.*, vol. 6, pp. 51–90, 1958.
[11] J. Hohensohn and J. M. Mendel, "Two-pass orthogonal least squares algorithm to train and reduce fuzzy logic systems," in *Proc. 4th IEEE Int. Conf. Fuzzy Systems*, 1995, pp. 696–700.
[12] H. Ishibuchi, K. Nozaki, N. Yamamoto, and H. Tanaka, "Selecting fuzzy if–then rules for classification problems using genetic algorithms," *IEEE Trans. Fuzzy Syst.*, vol. 3, pp. 260–270, 1995.
[13] J.-S. R. Jang, "ANFIS: Adaptive-network-based fuzzy inference system," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, pp. 665–685, 1993.
[14] J.-S. R. Jang and C.-T. Sun, "Neuro-fuzzy modeling and control," *Proc. IEEE*, vol. 83, pp. 378–406, 1995.
[15] ——, "Functional equivalence between radial basis function networks and fuzzy inference systems," *IEEE Trans. Neural Networks*, vol. 4, pp. 156–159, 1993.
[16] I. T. Jolliffe, *Principal Component Analysis*. New York: Springer-Verlag, 1986.
[17] C. L. Karr, "Applying genetics to fuzzy logic," *AI Expert*, vol. 6, pp. 38–43, 1991.
[18] A. J. Laub, "Numerical linear algebra aspects of control design computations," *IEEE Trans. Automat. Contr.*, vol. 30, pp. 97–108, 1985.
[19] M. A. Lee and H. Takagi, "Integrating design stages of fuzzy systems using genetic algorithms," in *Proc. 2nd IEEE Int. Conf. Fuzzy Systems*, 1993, pp. 612–617.
[20] E. H. Mamdani, "Application of fuzzy algorithms for simple dynamic plant," *Proc. Inst. Elec. Eng.*, vol. 121, pp. 1585–1588, 1974.
[21] J. Moody and C. J. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Computat.*, vol. 1, pp. 281–294, 1989.
[22] G. C. Mouzouris and J. M. Mendel, "Designing fuzzy logic systems for uncertain environments using a singular-value-QR decomposition method," in *Proc. 5th IEEE Int. Conf. Fuzzy Systems*, New Orleans, LA, Sept. 1996, pp. 295–301.
[23] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Networks*, vol. 1, pp. 4–26, 1991.
[24] K. C. Nisbet, B. Mulgrew, and S. McLaughlin, "Reduced state methods in nonlinear prediction," *Signal Process.*, vol. 28, pp. 37–49, 1996.
[25] B. G. Song, R. J. Marks, II, S. Oh, P. Arabshahi, T. P. Caudell, and J. J. Choi, "Adaptive membership function fusion and annihilation in fuzzy if–then rules," in *Proc. 2nd IEEE Int. Conf. Fuzzy Syst.*, 1993, pp. 961–967.
[26] G. W. Stewart, "Rank degeneracy," *SIAM J. Sci. Statist. Comput.*, vol. 5, pp. 403–413, 1984.
[27] M. Sugeno and G. T. Kang, "Structure identification of fuzzy model," *Fuzzy Sets Syst.*, vol. 28, pp. 15–33, 1988.
[28] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-15, pp. 116–132, 1985.
[29] S. Van Huffel and J. Vandewalle, "Subset selection using the total least squares approach in collinearity problems with errors in the variables," *Linear Algeb. Applicat.*, vol. 88/89, pp. 695–714, 1987.
[30] ——, *The Total Least Squares Problem: Computational Aspects and Analysis*. Philadelphia, PA: SIAM, 1991.
[31] L. Wang and R. Langari, "Building Sugeno-type models using fuzzy discretization and orthogonal parameter estimation techniques," *IEEE Trans. Fuzzy Syst.*, vol. 3, pp. 454–458, 1995.
[32] L. X. Wang and J. M. Mendel, "Fuzzy basis functions, universal approximation, and orthogonal least squares learning," *IEEE Trans. Neural Networks*, vol. 3, pp. 807–814, 1992.
[33] L. X. Wang, *Adaptive Fuzzy Systems and Control: Design and Stability Analysis*. Englewood Cliffs, NJ: Prentice-Hall, 1994.
[34] D. S. Watkins, *Fundamentals of Matrix Computations*. New York: Wiley, 1991.
[35] R. R. Yager and D. P. Filev, "Unified structure and parameter identification of fuzzy models," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, pp. 1198–1205, 1993.
[36] J. Yen and L. Wang, "Applying statistical information criteria for optimal fuzzy model construction," *IEEE Trans. Fuzzy Syst.*, vol. 6, pp. 362–372, 1998.
[37] ——, "An SVD-based fuzzy model reduction strategy," in *Proc. 5th IEEE Int. Conf. Fuzzy Systems*. New Orleans, LA, Sept. 1996, pp. 835–841.

[38] _____, "Application of orthogonal transformation in the construction of reduced fuzzy models," in *Proc. 7th IFSA World Congr.*, Prague, Czech Republic, June 1997, vol. 2, pp. 342–347.
[39] _____, "Simplification of fuzzy rule based systems using orthogonal transformation," in *Proc. 6th IEEE Int. Conf. Fuzzy Systems*, Barcelona, Spain, July 1997.

**John Yen** (M'91–SM'97) received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, R.O.C., in 1980, the M.S. degree in computer science from University of Santa Clara, Santa Clara, CA, in 1982, and the Ph.D. degree in computer science from the University of California, Berkeley, in 1986.

He is currently a Professor of Computer Science and the Director of the Center for Fuzzy Logic, Robotics, and Intelligent Systems at Texas A&M University, College Station. Before joining Texas A&M in 1989, he had been conducting artificial intelligence research as a Research Scientist at Information Sciences Institute, University of Southern California, Los Angele. His research interests include intelligent agents, fuzzy logic, evolutionary computing, and software engineering. He has authored or coauthored more than 90 technical journals and conference papers, and coauthored a textbook entitled *Fuzzy Logic: Intelligence, Control, and Information* (Englewood Cliffs, NJ: Prentice-Hall).

Dr. Yen is currently a Vice President of Publications for the IEEE Neural Networks Council (NNC) and the Secretary of International Fuzzy Systems Association (IFSA). He is an Associate Editor of the IEEE TRANSACTIONS OF FUZZY SYSTEMS and several other international journals on artificial intelligence and fuzzy logic. He received an NSF Young Investigator Award in 1992, the K. S. Fu Award from NAFIPS in 1994, the Dresser Industries Award and an Engineering Fellow from Texas A&M University in 1995 and 1997, respectively.

**Liang Wang** (M'95) received the B.S. degree in electrical engineering and the M.S. degree in systems engineering from Tianjin University, Tianjin, China, in 1985 and 1988, respectively. He was selected by the Liaison Committee of the European Community to continue his research in Europe in 1991 and received the Ph.D. degree in computer science with highest honors from Faculté Polytechnique de Mons, Mons, Belgium, in 1993.

He is now a Staff Scientist in the Center for Adaptive Systems Applications, Inc., Los Alamos, NM. His current research interests lie in fuzzy logic, neural networks, computer simulations, intelligent systems modeling, and control. He has coauthored more than 30 technical journal and conference papers in the above fields.