Designing Fuzzy Logic Systems

Jerry M. Mendel, *Fellow, IEEE*, and George C. Mouzouris

(Invited Paper)

***Abstract*— We present a formulation of a fuzzy logic system (FLS) that can be used to construct nonparametric models of nonlinear processes, given only input–output data. In order to effectively construct such models, we discuss several design methods with different properties and features. We compare and illustrate systems designed with each one of the methods, using an example on the predictive modeling of a nonlinear dynamic (chaotic) system.**

***Index Terms*— Backpropagation method, chaotic time series, design, forecasting, fuzzy logic systems, one-pass method, RLS method.**

I. Introduction

FUZZY LOGIC SYSTEMS (FLS's) are nonlinear systems capable of inferring complex nonlinear relationships between input and output variables. The nonlinearity property is particularly important when the underlying physical mechanism to be modeled is inherently nonlinear. The system can "learn" the nonlinear mapping by being presented a sequence of input signal and desired response pairs, which are used in conjunction with an optimization algorithm to determine the values of the system parameters. This is one of the most commonly used learning paradigms, called *supervised learning*. Even if the process to be modeled is nonstationary, the system can be updated to reflect the changing statistics of the process. Unlike conventional stochastic models used to model such processes, FLS's do not make any assumptions regarding the structure of the process, nor do they invoke any kind of probabilistic distribution model, i.e., they belong to the general family of model-free, data driven, nonparametric methods.

Designing a FLS can be viewed as approximating a function, or fitting a complex surface in a (probably) high dimensional space. Given a set of input–output pairs, the task of learning is essentially equivalent to determining a system that provides an optimal fit to the input–output pairs, with respect to a cost function. In addition, the system produced by the learning algorithm should be able to generalize to certain regions of the multidimensional space where no training data was given, i.e., it should be able to interpolate the given input–output data. Within the framework of approximation, and interpolation theory, it is common among many approximation/interpolation methods to generate the desired surface using a linear combination of basis functions (typically, nonlinear transformations of the input). As we will show in Section II, a FLS can be expressed as a linear combination of nonlinear functions.

There exists a variety of design methods, such as [1]–[3], [5], [7]–[9], [14], [15], [18]–[28], [30], [31], [37], [39], [41], [45], [46], to name a few, that can be used to construct FLS's with different properties and characteristics. Some of these design methods are data intensive, some are aimed at computational simplicity, some are recursive (thus giving the FLS an adaptive nature), some are offline, and some are application specific. One of the most widely used methods for constructing FLS's, mainly because of its simplicity, is table-lookup. The one-pass (OP) method [43], is a table-lookup method that allows us to generate fuzzy rules from given input–output pairs, by performing a simple OP operation on the given numerical and/or linguistic data. The generated fuzzy rules are collected into a common rulebase and combined using fuzzy set theory techniques to construct a final FLS.

The backpropagation (BP) algorithm is one of the main reasons that artificial neural networks have gained so much in popularity. It is a powerful training technique that can be applied to networks with feedforward structure, to turn them into adaptive systems. Since FLS's can be represented as layered feedforward networks [42], the same concept of BP can be applied, to train all design parameters of the FLS. This can be very advantageous, especially in cases where there exists complex interaction between the explanatory variables of the system we are trying to model. Such systems are sometimes referred to as *neuro–fuzzy* systems.

In general, adaptive FLS's designed with recursive algorithms such as BP, can have several advantages over similarly designed nonlinear systems: 1) they can utilize information other systems cannot (such as linguistic, or expert information), so they can conceivably achieve a better solution; 2) they can be initialized using relevant expert information, so that the search space during optimization is constrained, which leads to faster convergence; and 3) in some applications, the parameters of a FLS have physical meaning, so that intuition and expertise allow us to appropriately select the values of many of these parameters, in which case only a few need to be updated during training.

As we will show in Section II, FLS's that use the *height defuzzifier* to map the final output fuzzy set to a crisp value, can be expressed as a linear combination of nonlinear basis functions. Since they are linear in the output space (i.e., the space of the consequent fuzzy sets), linear methods such as

Manuscript received August 29, 1996; revised February 18, 1997. This work was supported by the National Science Foundation under Grant MIP-9 419 386. This paper was recommended by Editor J. Choma, Jr.

The authors are with the Signal and Image Processing Institute, Department of Electrical Engineering-Systems, University of Southern California, Los Angeles, CA 90089 USA.

Publisher Item Identifier S 1057-7130(97)07672-6.

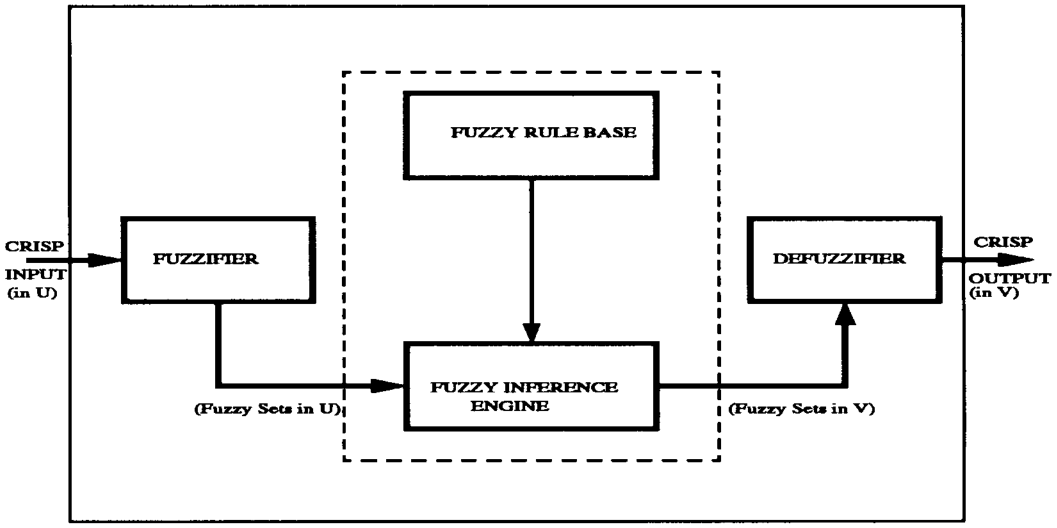


Fig. 1. Structure of fuzzy logic system.

recursive least-squares [44], or *orthogonal least-squares* [18], can be used to optimally select the centers of the consequent fuzzy sets in a least-squares sense.

A very important issue in the design of FLS's is determining the number of basis functions, or rules necessary to adequately represent a given system. Given an initial set of basis functions, one wants to select the best possible subset of basis functions for an effective representation. The SVD-QR design procedure [35] enables us to obtain an estimate of the number of necessary basis functions, and then discard basis functions that don't have a significant contribution in the rulebase. In addition, it produces estimates of the centers of the consequent fuzzy sets that are optimal in a least-squares sense.

In Section II, we present an overview of general FLS's, and show how certain classes of FLS's can be expressed as a linear expansion of nonlinear basis functions. In Section III, we describe the problem used to compare different FLS design methods; these methods are presented in Section IV. In Section V, we present our conclusions.

II. Description of the FLS

FLS's are both intuitive and numerical systems, that map crisp inputs, $\mathbf{x}$, into a crisp output, y. Every FLS is associated with a set of rules with meaningful linguistic interpretations, such as

$$\begin{aligned} R^l:\ & \text{IF } u_1 \text{ is } F_1^l \text{ and } u_2 \text{ is } F_2^l \text{ and } \dots \text{ and } u_n \text{ is } F_n^l \\ & \text{THEN } v \text{ is } G^l \end{aligned}$$

(note that without loss of generality, the rule is assumed to have only a single output), which can be obtained either from numerical data, or experts familiar with the problem at hand. Based on this kind of statement, actions are combined with rules in an antecedent/consequent format, and then aggregated according to approximate reasoning theory, to produce a nonlinear mapping from the input space $U = U_1 \times U_2 \times \cdots \times U_n$ to the output space V, where $F_k^l \subset U_k$, $k = 1, 2, \cdots, n$, are the antecedent membership functions, and $G^l \subset V$ is the consequent membership function. The input linguistic variables are denoted by u_k, $k = 1, 2, \cdots, n$, and the output linguistic variable is denoted by v.

A FLS consists of four basic elements (Fig. 1): the *fuzzifier*, the *fuzzy rulebase*, the *inference engine*, and the *defuzzifier*. The fuzzy rulebase is a collection of rules of the form of R^l, which are combined in the inference engine, to produce a fuzzy output (in essence, the inference engine produces mappings from fuzzy sets to fuzzy sets). The fuzzifier maps the crisp inputs into fuzzy sets, which are subsequently used as inputs to the inference engine, whereas the defuzzifier maps the fuzzy sets produced by the inference engine into crisp numbers.

Fuzzy sets can be interpreted as membership functions μ_X that associate with each element x of the universe of discourse, U, a number $\mu_X(x)$ in the interval $[0, 1]$:

$$\mu_X: U \to [0, 1]. \tag{1}$$

The fuzzifier maps a crisp point $x \in U$ into a fuzzy set $X \in U$. In the case of a *singleton* fuzzifier, the crisp point $x \in U$ is mapped into a fuzzy set X with support x_i, where $\mu_X(x_i) = 1$ for $x_i = x$ and $\mu_X(x_i) = 0$ for $x_i \neq x$, i.e., the *single* point in the support of X with nonzero membership function value is $x_i = x$. In the case of a *nonsingleton* fuzzifier, the point $x \in U$ is mapped into a fuzzy set X with support x_i, where μ_X achieves maximum value at $x_i = x$ and decreases while moving away from $x_i = x$. We assume that fuzzy set X is normalized so that $\mu_X(x) = 1$.

Nonsingleton fuzzification is especially useful in cases where the available training data, or the input data to the fuzzy logic system, contain any kind of uncertainty (such as noise, or linguistic imprecision). Conceptually, the nonsingleton fuzzifier implies that the given input value x is the most likely value to be the correct one from all the values in its immediate neighborhood; however, because of the presence of uncertainty, neighboring points are also likely to be the correct values, but to a lesser degree.

The shape of the membership function μ_X can be determined by the system designer, based on an estimate of the kind and quantity of uncertainty present. It would be the logical choice, though, for the membership function to be symmetric about x, since the effect of noise is most likely to be equivalent on all points. Examples of such membership functions are 1) the Gaussian

$$\mu_X(x_i) = \exp[-\frac{(x-x_i)^2}{2\sigma^2}]$$

where the variance σ^2 reflects the width (spread) of $\mu_X(x_i)$; 2) triangular

$$\mu_X(x_i) = \max(0, 1 - |\frac{x-x_i}{c}|)$$

and, 3)

$$\mu_X(x_i) = \frac{1}{(1+|\frac{x-x_i}{c}|^p)}$$

where x and c are, respectively, the mean and spread of the fuzzy sets. Note that larger values of the spread of the above membership functions imply that a higher degree of uncertainty is anticipated to exist in the given data.

Each input to the FLS, after having been processed through the fuzzifier, will activate each rule in the rulebase to a (possibly) different degree. The fuzzy rules R^l, $l = 1, 2, \cdots, M$ will produce output fuzzy sets Y^l which will then be aggregated, typically using t-conorm operations, to produce the total output fuzzy set

$$\Upsilon = \biguplus_{l=1}^{M} Y^l \tag{2}$$

where $\biguplus_{l=1}^{M}$ denotes a sequence of t-conorm operations.

Most engineering applications of FLS's require a crisp output, therefore the fuzzy set Υ is mapped to a single point, f, using a defuzzification operation $\mathcal{D}$:

$$f = \mathcal{D}(\Upsilon). \tag{3}$$

Comparisons of several defuzzification methods based on their computational complexity, weight counting, plausibility, disambiguity, and continuity properties have been performed in [10] and [17]. In those studies, it was determined that the center-of-sums and height methods have more desirable overall properties than the remaining methods.

A general way to express an n-input single-output FLS with M rules in its rulebase is as follows:

$$f = \mathcal{D}\left(\biguplus_{l=1}^{M} \int_V \mu_{G^l}(y) \star \mathcal{T}_{k=1}^{n} \mu_{Q_k^l}(x_{k,\sup}^l)/(y)\right) \tag{4}$$

where $\star$ denotes t-norm operation; $\mathcal{T}_{k=1}^{n}$ denotes a sequence of t-norm operations; $\int$ denotes union of points in the continuum; $\mathcal{D}$ is a general defuzzifier that maps fuzzy sets in the output space V to crisp points in V; $\mu_{Q_k^l}(x_{k,\sup}^l) \equiv \mu_{F_k^l}(x_{k,\sup}^l) \star \mu_{X_k}(x_{k,\sup}^l)$; $\mu_{F_k^l}(x_{k,\sup}^l)$ is the membership function for the kth antecedent of the lth rule; $\mu_{X_k}(x_{k,\sup}^l)$ is the membership function for the kth input fuzzy set; and $x_{k,\sup}^l$ is the point that maximizes $\mu_{Q_k^l}$.

In the special case of the *modified Height defuzzifier*, ([32], [33]) (4) can be expressed as a *fuzzy basis function* (FBF) expansion

$$f_{\text{ns}}(\mathbf{x}) = \sum_{l=1}^{M} \bar{y}^l p_{\text{ns}}^l(\mathbf{x}) = \frac{\sum_{l=1}^{M} \bar{y}^l [\mathcal{T}_{k=1}^{n} \mu_{Q_k^l}(x_{k,\sup}^l)]/(\delta^l)^2}{\sum_{l=1}^{M} [\mathcal{T}_{k=1}^{n} \mu_{Q_k^l}(x_{k,\sup}^l)]/(\delta^l)^2} \tag{5}$$

where $\bar{y}^l$ denotes the center of the lth consequent fuzzy set G^l and δ^l is a constant proportional to the uncertainty in G^l (e.g., δ^l could be chosen equal to the spread of G^l). Equation (5) is recognized as a *nonsingleton fuzzy logic system* (NSFLS), where

$$p_{\text{ns}}^l(\mathbf{x}) = \frac{[\mathcal{T}_{k=1}^{n} \mu_{Q_k^l}(x_{k,\sup}^l)]/(\delta^l)^2}{\sum_{l=1}^{M} [\mathcal{T}_{k=1}^{n} \mu_{Q_k^l}(x_{k,\sup}^l)]/(\delta^l)^2} \tag{6}$$

are the nonsingleton FBF's. In the absence of uncertainty (i.e., when the input fuzzy set becomes a single point), the NSFLS in (5) reduces to the singleton FLS

$$f(\mathbf{x}) = \sum_{l=1}^{M} \bar{y}^l p^l(\mathbf{x}) = \frac{\sum_{l=1}^{M} \bar{y}^l [\mathcal{T}_{k=1}^{n} \mu_{F_k^l}(x_k)]/(\delta^l)^2}{\sum_{l=1}^{M} [\mathcal{T}_{k=1}^{n} \mu_{F_k^l}(x_k)]/(\delta^l)^2} \tag{7}$$

where the FBF's are now given by

$$p^l(\mathbf{x}) = \frac{[\mathcal{T}_{k=1}^{n} \mu_{F_k^l}(x_k)]/(\delta^l)^2}{\sum_{l=1}^{M} [\mathcal{T}_{k=1}^{n} \mu_{F_k^l}(x_k)]/(\delta^l)^2}. \tag{8}$$

In our treatment here, we will focus on the singleton FLS (7). For further details on NSFLS's and comparisons with singleton FLS's, see [29] and [32]–[35].

III. Predictive Modeling of Chaotic Time Series

Many dynamical systems displaying the phenomenon of chaos fall under the classification of nonlinear oscillators [38]. The Van der Pol oscillator and the Duffing oscillator are some very well-known examples, whose dynamics have been widely studied. In the following sections, we will construct predictive models of a chaotic time series obtained as the solution of Duffing's equation [36]

$$\begin{aligned} \dot{x} &= y \\ \dot{y} &= x - x^3 - \epsilon y + \gamma \cos(\omega t). \end{aligned} \tag{9}$$

The approach we adopt here is based on determining a FLS as a model of the chaotic system (9), given past time-series data. The problem of constructing predictive models of chaotic systems is a challenging one, because of the low degree of correlation among consecutive time-series points. The solution of this problem is illustrative of how someone can utilize a nonparametric method such as a FLS to model an unknown system (e.g., a complicated "*RC*" circuit) treated as a black box. Fig. 2 depicts phase plane plots and amplitude spectra, as well as trajectories of the solution of (9), before and after transition to chaos. It is evident from these plots that after transition to chaos (i.e., when $\epsilon = 0.15$, $\gamma = 0.30$, and $\omega = 1$) there is no apparent structure in the time-series data, in terms of periodic behavior.

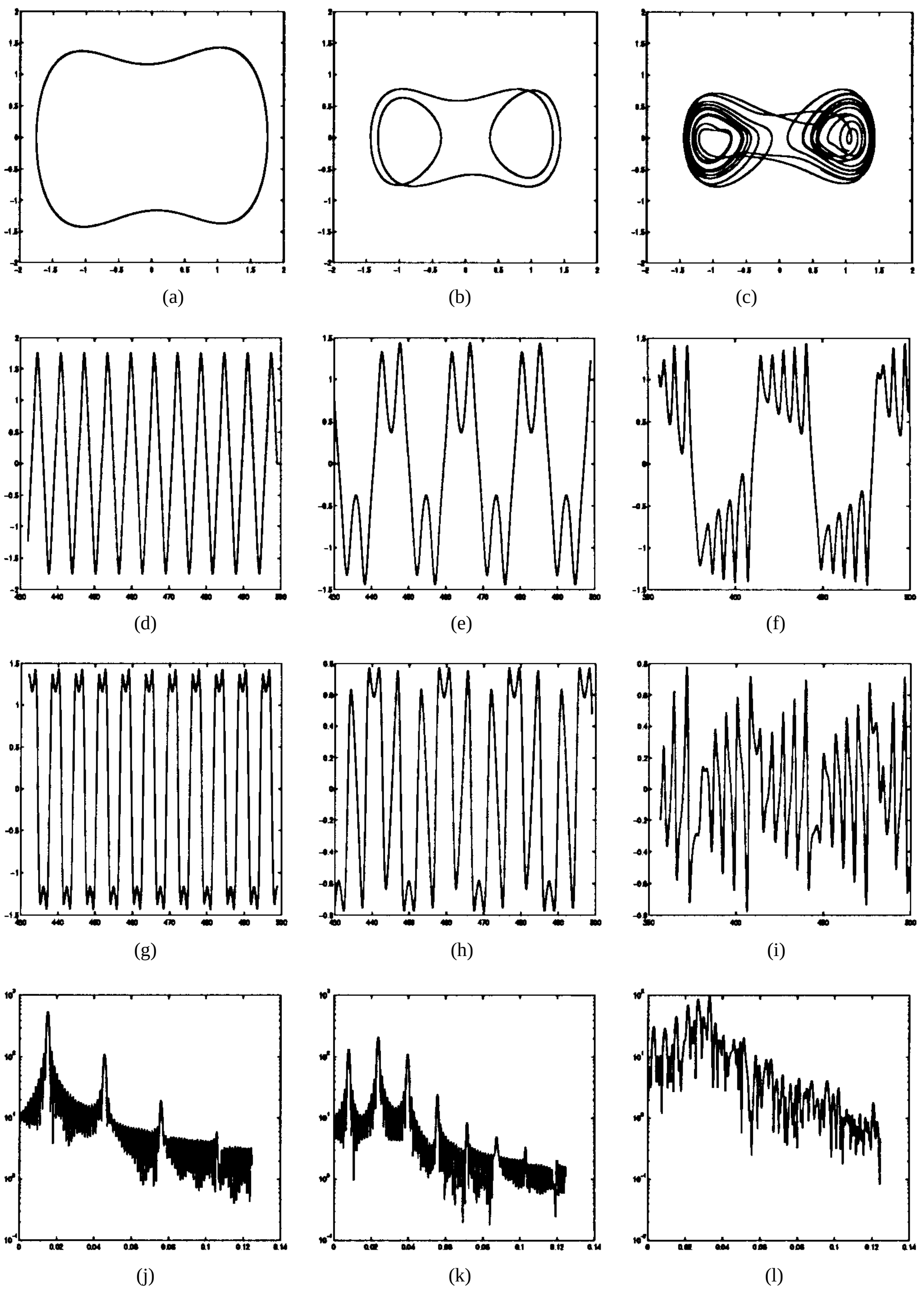


Fig. 2. (a)–(c) Phase plane plots of Duffing oscillator. (d)–(f) x-component time-history of Duffing oscillator. (g)–(i) y-component time-history of Duffing oscillator. (j)–(l) Amplitude spectrum of y-component of Duffing oscillator. The leftmost and center columns demonstrate the behavior of system (9) while it is still periodic. The rightmost column depicts the system's behavior after transition to chaos.

IV. Training Methods

In this section, we give descriptions of a representative subset of methods for designing FLS's, and discuss the characteristics of each one. We apply all methods to the same problem, and compare their performance. We also examine how some design methods can be used in conjunction with others.

A. OP Method

The OP method (also known as the "Wang-Mendel Method" [6]) is a simple design method that generates a set of IF–THEN rules by performing a OP operation on the given input–output data, and then combines the rules in a common rulebase, to construct a final FLS. Given a set of input–output pairs

$$\left(x_1^{(1)}, x_2^{(1)}, \cdots, x_n^{(1)}; y^{(1)}\right), \left(x_1^{(2)}, x_2^{(2)}, \cdots, x_n^{(2)}; y^{(2)}\right), \cdots \tag{10}$$

where $x_1, x_2, \cdots, x_n$ are inputs and y is the output, we proceed as follows to construct a FLS:

1) Let

$$[x_1^-, x_1^+], [x_2^-, x_2^+], \cdots, [x_n^-, x_n^+]; [y^-, y^+]$$

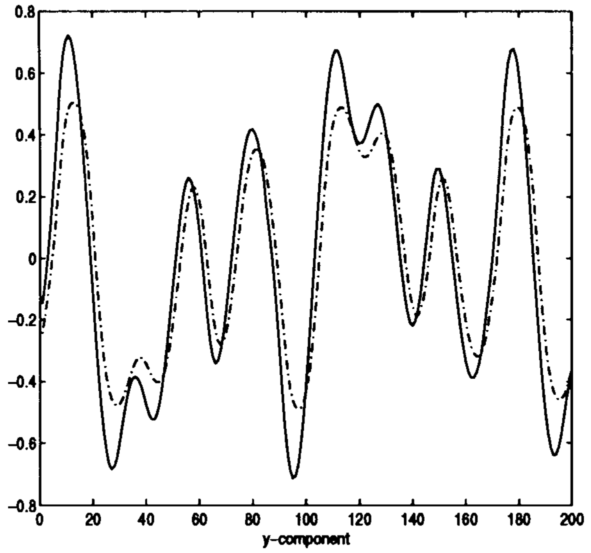
be the domain intervals of the input and output variables, respectively, where domain interval implies the interval a variable is most likely to lie in. We divide each domain interval into $2N+1$ regions, where N can be different for each variable. Then, we assign membership functions to the regions, labeled as SN (Small N), $\cdots, S1$ (Small 1), CE (Center), $B1$ (Big 1), $\cdots, BN$ (Big N).

2) We evaluate the membership of each input–output point in regions where it may occur, and assign the given $x_1^{(i)}, x_2^{(i)}, \cdots, x_n^{(i)}$, or $y^{(i)}$ to the region with maximum membership.
3) In order to resolve conflicting rules, i.e., rules with the same antecedent membership functions and different consequent membership functions, we assign a degree to each rule as follows: let $\mu_{X_k}(x_k)$ denote the membership of the kth input variable in the region X_k with maximum membership, and $\mu_Y(y)$ the membership of the output variable in the region Y with maximum membership, where X_k and Y are labels from their corresponding sets $SN, \cdots, S1, CE, B1, \cdots, BN$. Then, the degree for the lth rule, R^l, is defined as

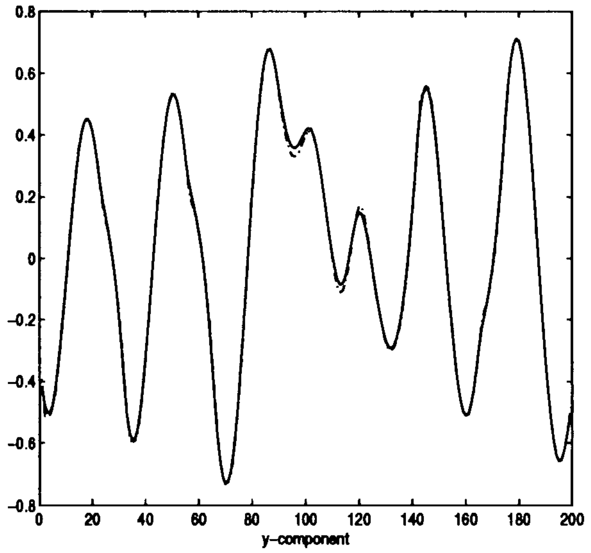
$$D(R^l) = \prod_{k=1}^{n} \mu_{X_k}(x_k)\mu_Y(y). \tag{11}$$

In the event of conflicting rules, the rule with the highest degree (11) is kept in the rulebase, and all other conflicting rules are discarded.
4) We generate a combined rulebase comprised both of numerically generated fuzzy rules (as described above) and linguistic information provided by experts.
5) After the combined rulebase is generated, we employ a defuzzification method (such as Center-of-Area, Center-of-Sums, Height defuzzifier), to obtain the crisp output of the FLS.

We used the OP method to construct a FLS as a single step predictive model of the y-component of the Duffing equation [see Fig. 3(a)]. We solved (9) using a Runge–Kutta integration routine, and allowed 2000 points of the y-component time-series to elapse, for the transients to die out. Then, we used the next $G = 400$ points to design our system, and tested it on 200 out-of-sample points following the training segment. The OP designed system had $M = 100$ rules (obtained from the training segment), each with three input variables, (i.e., antecedents), $x_k(t) = y(t-k)$, $k = 1, 2, 3$, and one output (i.e., consequent) $y(t)$. The output of the system $f(t)$ was the predicted value of $y(t)$. The rules related $y(t-k)$ ($k = 1, 2, 3$) to $y(t)$ at time points $(G/M)\cdot l - k, l = 1, 2, \cdots, M$, from the 400 hundred training points. The out-of-sample mean-squared error, $\mathrm{mse_{OP}}$, was 0.0218, and the standard deviation of the out-of-sample residuals $\mathrm{std_{OP}}$ was 0.1474. The OP FLS gives reasonably good results; but, as is evident from Fig. 3(a), it is not that accurate.



(a)



(b)

Fig. 3. (a) Single step prediction of the y-component of the Duffing equation (dash–dotted line) with a OP Method designed FLS. (b) Single step prediction of the y-component of the Duffing equation (dash–dotted line) with an RLS-designed FLS.

B. Recursive Least Squares

In order to achieve fast convergence rate while maintaining good approximation capability, we can design a *recursive least-squares* (RLS) FLS based on the general formulation (7) of a FLS as a linear expansion of nonlinear FBF's. The RLS-designed FLS only updates the centers, $\bar{y}^l$, of the consequent fuzzy sets; therefore, overall performance depends on the successful selection of $\sigma_{F_k^l}$ and $m_{F_k^l}$, which remain constant during the adaptation procedure.

Let $\{x_k(t)\} \in U_k, k = 1, 2, \cdots, n$, and $\{d(t)\} \in R, t = 0, 1, \cdots$, denote real-valued sequences. Let $\{\mathbf{x}(t)\} = \{[x_1(t), x_2(t), \cdots, x_n(t)]^T\} \in U$. Given the input–output pairs $[\mathbf{x}(t); d(t)]$, the problem is to design a FLS $f\colon U \to R$ such that

$$J_{\mathrm{RLS}}(t) = \sum_{i=1}^{t} \lambda^{t-i}[d(i) - f(\mathbf{x}(i))]^2 \tag{12}$$

is minimized, where $\lambda \in (0, 1]$ is a forgetting factor.

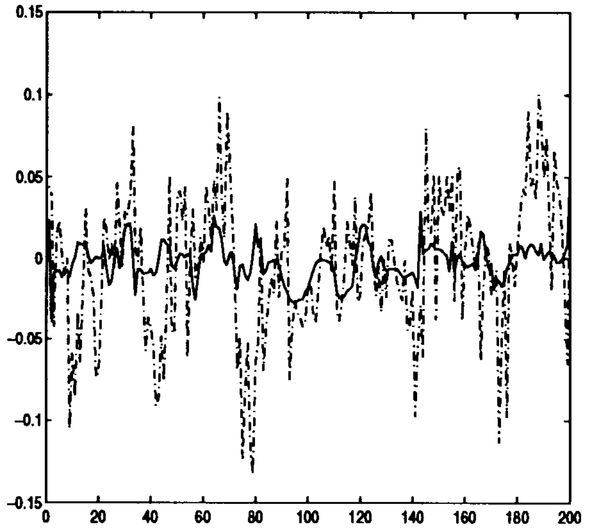


Fig. 4. Residual errors produced by the OP Method designed FLS (dash–dotted line), and residual errors produced by the RLS-designed FLS (solid line).

Using a method such as the OP (see, also, [42]), we construct a set of fuzzy rules of the same form as in Section II, and subsequently a FLS of the form of (7). Collecting all $\bar{y}^l$ in an M-dimensional vector $\bar{\mathbf{y}}$, and all p^l in an M-dimensional vector $\mathbf{p}(\mathbf{x}(t))$, we can write (7) in a vector form as

$$f(\mathbf{x}(t)) = \mathbf{p}(\mathbf{x}(t))^T\bar{\mathbf{y}}. \tag{13}$$

Since f is linear in $\bar{\mathbf{y}}$, the RLS algorithm [16] can be used to update the centers of the consequent fuzzy sets. The recursions for $\bar{\mathbf{y}}$ can be obtained by minimizing $J_{\text{RLS}}(t)$ in (12), given that f is of the form in (7). If we let $P(t) \equiv \mathbf{p}(\mathbf{x}(t))$ denote the FBF vector at time instant t, the recursions are given by [34], [42]

$$\Phi(t) = \frac{1}{\lambda}[\Phi(t-1) - \Phi(t-1)P(t)(\lambda + P^T(t) \times \Phi(t-1)P(t))^{-1}P^T(t)\Phi(t-1)] \tag{14}$$

$$K(t) = \Phi(t-1)P(t)[\lambda + P^T(t)\Phi(t-1)P(t)]^{-1} \tag{15}$$

$$\bar{\mathbf{y}}(t) = \bar{\mathbf{y}}(t-1) + K(t)[d(t) - P^T(t)\bar{\mathbf{y}}(t-1)] \tag{16}$$

for $t = 1, 2, \cdots$. Matrix Φ is initialized to $\Phi(0) = \beta I$, where I is an $M \times M$ identity matrix, and β is a small positive constant. The initial value of $\bar{\mathbf{y}}, \bar{\mathbf{y}}(0)$, and the values of the fixed system parameters, can be selected from linguistic information, or from the OP method, so that the supports of the resulting membership functions cover the corresponding universes of discourse.

We employed an RLS-designed system with 100 rules (initially obtained from the OP method) and three inputs, $x_k(t) = y(t-k)$, $k = 1, 2, 3$, to produce single step forecasts of the y-component of the Duffing equation (see Fig. 3(b)). The out-of-sample mean-squared error for the RLS-designed system was $\text{mse}_{\text{RLS}} = 0.000\,31$, and the standard deviation of the residuals was $\text{std}_{\text{RLS}} = 0.0175$. Fig. 4 depicts the pointwise residual errors produced by the OP designed FLS (dash–dotted line), and the pointwise residual errors produced by the RLS designed FLS. Clearly, performance of the RLS FL forecaster is better than that of the OP forecaster.

C. Back-Propagation (BP) Method

For the development of the BP training algorithm for updating the design parameters of a FLS, we focus on a system with Gaussian antecedent and consequent membership functions, product inference, and height defuzzification, for which:

$$f(\mathbf{x}) = \frac{\sum_{l=1}^M \bar{y}^l \prod_{k=1}^n \mu_{F_k^l}(x_k)}{\sum_{l=1}^M \prod_{k=1}^n \mu_{F_k^l}(x_k)} = \frac{\sum_{l=1}^M \bar{y}^l \prod_{k=1}^n \exp\left(-\frac{\left(x_k - m_{F_k^l}\right)^2}{2\sigma^2_{F_k^l}}\right)}{\sum_{l=1}^M \prod_{k=1}^n \exp\left(-\frac{\left(x_k - m_{F_k^l}\right)^2}{2\sigma^2_{F_k^l}}\right)} \tag{17}$$

$$= \sum_{l=1}^M \bar{y}^l p^l(\mathbf{x}). \tag{18}$$

Given an input–output training pair $(\mathbf{x}^r, d^r)$, $\mathbf{x}^r \in R^n$ and $d^r \in R$, we wish to design a FLS in the form of (17) such that the following error function is minimized:

$$e^r = \frac{1}{2}[f(\mathbf{x}^r) - d^r]^2. \tag{19}$$

It is evident from (17), that f is completely characterized by $\bar{y}^l, m_{F_k^l}$, and $\sigma^2_{F_k^l}$. Using a steepest descent algorithm to minimize e^r, it is straight forward to obtain the following recursions to update all design parameters of a FLS:

$$m_{F_k^l}(q+1) = m_{F_k^l}(q) - \alpha(f-d)\frac{x_k - m_{F_k^l}}{\sigma^2_{F_k^l}}(\bar{y}^l - f)p^l \tag{20}$$

$$\bar{y}^l(q+1) = \bar{y}^l(q) - \alpha(f-d)p^l \tag{21}$$

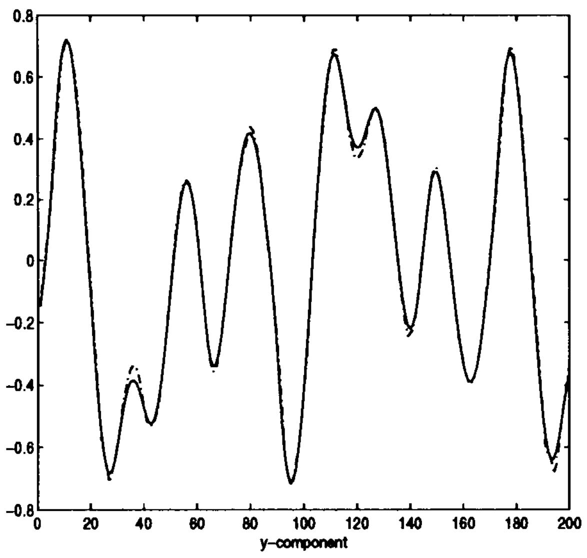
$$\sigma_{F_k^l}(q+1) = \sigma_{F_k^l}(q) - \alpha(f-d)\frac{(x_k - m_{F_k^l})^2}{\sigma^3_{F_k^l}}(\bar{y}^l - f)p^l \tag{22}$$

where p^l denotes the lth basis function. Equations (20)–(22) are referred to as a backpropagation algorithm because of their dependence on error $(f-d)$.

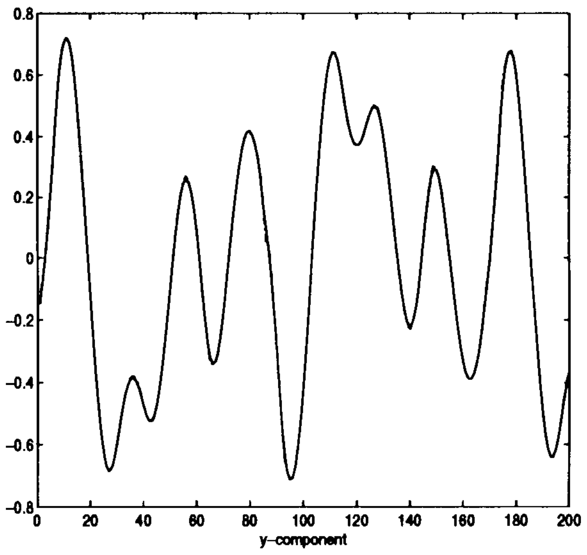
Singleton FLS's, and the BP update recursions can be generalized to the nonsingleton FLS case [34]; nonsingleton FLS's can also be trained to account for uncertainty at the input, by training for a parameter proportional to the spread (or width) of the input fuzzy sets.

A 100-rule, three input BP-trained FLS, designed as a single step predictor of the y-component of the Duffing equation [see Fig. 5(a)], had out-of-sample mean-square error $\text{mse}_{\text{BP}} =$ 0.000 48, and residual standard deviation $\text{std}_{\text{BP}} = 0.0219$. The input and output variables were the same as the OP- and RLS-trained FLS's, and the initial parameter values used in the BP algorithm were determined using the OP method. A total of 700 parameters [(2 per input $\times$ 3 inputs $+$ 1 consequent center) $\times$ 100 rules] were updated using the BP algorithm.

For this data, the RLS FL forecaster outperformed the BP FL forecaster. This is not a surprising result, because even though all the design parameters of the BP trained FLS are updated, the BP algorithm converges (typically to a local minimum) much slower than the RLS algorithm.



(a)



(b)

Fig. 5. (a) Single step prediction of the y-component of the Duffing equation (dash–dotted line) with a back-propagation designed FLS. (b) Single step prediction of the y-component of the Duffing equation (dash–dotted line) with an SVD-QR designed FLS.

D. Singular-Value-QR Decomposition Method

In this section, we show that, by appropriate analysis of a FBF matrix $\mathbf{P}$, we can extract useful information about the FLS, so as to design systems with optimally selected $\bar{\mathbf{y}}$'s that have minimal redundancy in their rulebase. The FBF matrix can be formed by evaluating each FBF $p^l(\mathbf{x}(t))$, $l = 1, 2, \cdots, M$, at N points and then ordering them in an $N \times M$ matrix

$$\mathbf{P} = \begin{bmatrix} \mathbf{p}(\mathbf{x}(1)) \\ \mathbf{p}(\mathbf{x}(2)) \\ \vdots \\ \mathbf{p}(\mathbf{x}(N)) \end{bmatrix} \tag{23}$$

where $\mathbf{p}(\mathbf{x}(t)) \equiv [p^1(\mathbf{x}(t)), p^2(\mathbf{x}(t)), \cdots, p^M(\mathbf{x}(t))]$.

It is well known that if the rank of $\mathbf{P}$ is less than M, then the least-squares problem

$$\min_{\bar{\mathbf{y}}} \|\mathbf{P}\bar{\mathbf{y}} - \mathbf{f}\|_2 \tag{24}$$

has no unique solution. Several orthogonalization algorithms can be adapted to handle this [4], but the only reliable way to treat rank deficiency is by the computation of the *singular value decomposition* (SVD) of $\mathbf{P}$ [13]. Furthermore, the SVD provides a natural way to separate a space into dominant and subdominant subspaces. If we view the FBF matrix $\mathbf{P}$ as a span of the input subspace, then the SVD decomposes the span into an equivalent orthogonal span, from which we can identify the dominant and subdominant spans [40], i.e., we can identify which FBF's contribute the most to the system, and how many of them are needed to effectively represent the system. The FBF's that contribute the least can be discarded, and a reduced, parsimonious system can be designed. Note that the SVD approach only reorders the original FBF's to form a set of independent FBF's; thus, it preserves the meaning of linguistic information initially incorporated into the system.

Let $N > M$, and $\text{rank}(\mathbf{P}) = r \leq M$ denote the rank of $\mathbf{P}$. The SVD of $\mathbf{P}$ is given by [16]

$$\mathbf{P} = \mathbf{U} \begin{bmatrix} \mathbf{\Sigma} & 0 \\ 0 & 0 \end{bmatrix} \mathbf{V}^T = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^T \tag{25}$$

where $\mathbf{U}$ is an $N \times N$ matrix of orthonormalized eigenvectors of $\mathbf{P}\mathbf{P}^T$, $\mathbf{V}$ is an $M \times M$ matrix of orthonormalized eigenvectors of $\mathbf{P}^T\mathbf{P}$, $\mathbf{v}_i$ and $\mathbf{u}_i$ denote the column vectors of $\mathbf{V}$ and $\mathbf{U}$, respectively, and $\mathbf{\Sigma}$ is the diagonal matrix $\mathbf{\Sigma} = \text{diag}(\sigma_1, \sigma_2, \cdots, \sigma_r)$, where σ_i denotes the ith singular value of $\mathbf{P}$, and $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$.

The pseudoinverse of $\mathbf{P}$ is given by [16]

$$\mathbf{P}^+ = \mathbf{V} \begin{bmatrix} \mathbf{\Sigma}^{-1} & 0 \\ 0 & 0 \end{bmatrix} \mathbf{U}^T = \sum_{i=1}^{r} \frac{1}{\sigma_i} \mathbf{v}_i \mathbf{u}_i^T \tag{26}$$

where $\mathbf{\Sigma}^{-1} = \text{diag}\ (\sigma_1^{-1}, \cdots, \sigma_r^{-1})$. The optimal centers of the consequent fuzzy sets which represent the solution to (24), can then be calculated, as

$$\bar{\mathbf{y}}' = \mathbf{P}^+\mathbf{f} = \mathbf{V} \begin{bmatrix} \mathbf{\Sigma}^{-1} & 0 \\ 0 & 0 \end{bmatrix} \mathbf{U}^T\mathbf{f} = \sum_{i=1}^{r} \frac{1}{\sigma_i} \mathbf{v}_i \mathbf{u}_i^T \mathbf{f}. \tag{27}$$

Out of the many vectors that solve the least-squares problem in the rank deficient case, the one defined by (27) is unique in that it simultaneously satisfies two requirements [12]: 1) it is optimal in a least-squares sense and 2) it has the smallest Euclidean norm.

Assuming that the *numerical rank* of $\mathbf{P}$ is given by r', the minimum norm solution for $\bar{\mathbf{y}}'$ can be approximated by

$$\bar{\mathbf{y}}_{r'} = \sum_{i=1}^{r'} \frac{1}{\sigma_i} \mathbf{v}_i \mathbf{u}_i^T \mathbf{f} \tag{28}$$

where $\bar{\mathbf{y}}_{r'}$ minimizes

$$\|\mathbf{P}_{r'}\bar{\mathbf{y}} - \mathbf{f}\|_2 \tag{29}$$

and $\mathbf{P}_{r'} = \sum_{i=1}^{r'} \sigma_i \mathbf{u}_i \mathbf{v}_i^T$. Replacing $\mathbf{P}$ with $\mathbf{P}_{r'}$ can be viewed as filtering out very small singular values, which is especially useful in cases where the FBF matrix is formed using noisy data. Small singular values are also associated with redundant factors with singular values larger than the ones due to noise, but considerably smaller than the largest singular value, σ_1. In such a case, it would not be desirable to construct a system such as $\mathbf{P}_{r'}\bar{\mathbf{y}}_{r'}$ which includes all redundant factors; instead, a system $\mathbf{P}\bar{\mathbf{y}}$ can be designed, where $\bar{\mathbf{y}}$ has $\hat{r} \leq r'$ nonzero components (i.e., $\bar{\mathbf{y}}$ may have less than r' components). The system $\mathbf{P}\bar{\mathbf{y}}$ can be viewed as reducing (or eliminating) the redundancy among the factors (FBF's) comprising the underlying model, as reflected in the initial FBF matrix determination. The location of the nonzero components of $\bar{\mathbf{y}}$ can be used to determine a subset of the M initial FBF's, which are used to approximate the desired response $\mathbf{f}$.

Although several subset selection methods exist [4], a singular value decomposition method is preferable in rank deficient problems [13]. Here, we use the following algorithm that is similar to the one in [11], [13] to select a set of independent FBF's that minimize the residual error in a least-squares sense.

(1) Determine a numerical estimate r' of the rank of the FBF matrix $\mathbf{P}$ by calculating the singular value decomposition $\mathbf{P} = \mathbf{U}[\begin{smallmatrix}\mathbf{\Sigma} & 0\\ 0 & 0\end{smallmatrix}]\mathbf{V}^T$, and select $\hat{r} \leq r'$.

(2) Calculate a permutation matrix $\mathbf{\Pi}$ such that the columns of the matrix $\mathbf{\Gamma}_1 \in R^{N\times\hat{r}}$ in

$$\mathbf{P\Pi} = [\mathbf{\Gamma}_1, \mathbf{\Gamma}_2] \tag{30}$$

are independent. The permutation matrix $\mathbf{\Pi}$ is obtained from the QR decomposition of the submatrix comprised of the right singular vectors, which correspond to the $\hat{r}$ ordered *most-significant* singular values.

(3) Approximate $\mathbf{f}$ with $\mathbf{P}\bar{\mathbf{y}}$ where

$$\bar{\mathbf{y}} = \mathbf{\Pi}\begin{bmatrix}\mathbf{z}\\ 0\end{bmatrix} \tag{31}$$

so that $\mathbf{P}\bar{\mathbf{y}} = \mathbf{\Gamma}_1\mathbf{z}$, and $\mathbf{z} \in R^{\hat{r}}$ minimizes

$$||\mathbf{\Gamma}_1\mathbf{z} - \mathbf{f}||_2. \tag{32}$$

Given $\mathbf{P} \in R^{N\times M}$, $\mathbf{f} \in R^N$, and $\hat{r}$, the following algorithm computes a permutation matrix $\mathbf{\Pi}$ and a vector $\mathbf{z} \in R^{\hat{r}}$ such that the first $\hat{r}$ columns of $\mathbf{P\Pi}$ are independent and

$$\left\|\mathbf{P\Pi}\begin{bmatrix}\mathbf{z}\\ 0\end{bmatrix} - \mathbf{f}\right\|_2 \tag{33}$$

is minimized.

- Calculate the SVD of $\mathbf{P}$ and save $\mathbf{V}$.
- Estimate $\hat{r} \leq$ numerical $\mathrm{rank}(\mathbf{P})$ and partition

$$\mathbf{V} = \begin{bmatrix}\mathbf{V}_{11} & \mathbf{V}_{12}\\ \mathbf{V}_{21} & \mathbf{V}_{22}\end{bmatrix}, \tag{34}$$

where $\mathbf{V}_{11} \in R^{\hat{r}\times\hat{r}}$, $\mathbf{V}_{12} \in R^{\hat{r}\times(M-\hat{r})}$, $\mathbf{V}_{21} \in R^{(M-\hat{r})\times\hat{r}}$, and $\mathbf{V}_{22} \in R^{(M-\hat{r})\times(M-\hat{r})}$. In many practical cases, σ_1 is much larger than $\sigma_{r'}$; thus, $\hat{r}$ can be chosen much smaller than the estimate r' of $\mathrm{rank}(\mathbf{P})$.

- Using QR decomposition with column pivoting, determine $\mathbf{\Pi}$ such that

$$\mathbf{Q}^T[\mathbf{V}_{11}^T, \mathbf{V}_{21}^T]\mathbf{\Pi} = [\mathbf{R}_{11}, \mathbf{R}_{12}]$$

where $\mathbf{Q}$ is a unitary matrix, and $\mathbf{R}_{11}$ and $\mathbf{R}_{12}$ form an upper triangular matrix; then form $\mathbf{P\Pi} = [\mathbf{\Gamma}_1, \mathbf{\Gamma}_2]$, where $\mathbf{\Gamma}_1 \in R^{N\times\hat{r}}$ and $\mathbf{\Gamma}_2 \in R^{N\times(M-\hat{r})}$.
- Determine $\mathbf{z} \in R^{\hat{r}}$ such that $||\mathbf{\Gamma}_1\mathbf{z} - \mathbf{f}||_2$ is minimized.

TABLE I
MEAN-SQUARED ERROR AND STANDARD DEVIATION OF OUT-OF-SAMPLE RESIDUALS FOR THE PREDICTIVE MODELING OF THE DUFFING CHAOTIC ATTRACTOR, USING THE SVD-QR DESIGN METHOD. THE INITIAL FLS HAD $M = 100$ RULES IN ITS RULEBASE, AND ANTECEDENT STANDARD DEVIATION EQUAL TO $\sigma_F = 0.5$

Number of rules	mse_{out}	std_{out}
100	3.625×10^{-5}	0.006017
80	3.687×10^{-5}	0.006051
60	3.852×10^{-5}	0.006186
40	5.162×10^{-5}	0.007181
20	1.485×10^{-4}	0.012150
10	3.074×10^{-4}	0.017470
5	1.953×10^{-2}	0.139750

TABLE II
MEAN-SQUARED ERROR AND STANDARD DEVIATION OF OUT-OF-SAMPLE RESIDUALS FOR THE PREDICTIVE MODELING OF THE DUFFING CHAOTIC ATTRACTOR USING THE SVD-QR DESIGN METHOD. THE INITIAL FLS HAD $M = 100$ RULES IN ITS RULEBASE AND ANTECEDENT STANDARD DEVIATION EQUAL TO $\sigma_F = 1$

Number of rules	mse_{out}	std_{out}
100	3.676×10^{-5}	0.006050
80	3.678×10^{-5}	0.006051
60	3.936×10^{-5}	0.006263
40	4.182×10^{-5}	0.006454
20	4.814×10^{-5}	0.006937
10	3.172×10^{-4}	0.017760
5	4.524×10^{-4}	0.021250

Although the SVD-QR design method provides optimal estimates of $\bar{\mathbf{y}}$ in a least-squares sense, it does not provide any estimates for the remaining parameters of a FLS. The backpropagation algorithm described in Section IV-C allows us to do that. Once backpropagation training is completed, the SVD-QR method can be used again with the updated parameters, to recalculate $\bar{\mathbf{y}}$.

We employed the SVD-QR method to produce single step predictions of the y-component of the Duffing equation [see Fig. 5(b)]. We initially designed a FLS with 100 rules obtained from the OP method as in Section IV-A, and three input variables $x_k(t) = y(t-k)$, $k = 1, 2, 3$; then, we used the SVD-QR design method to reduce the number of rules. The results are given in Tables I and II, for two different values of antecedent standard deviation. Observe, from Table I, that a 40-rule FLS achieves about the same performance as 100-rule FLS (when $\sigma_F = 0.5$), and, from Table II, that a 20-rule FLS achieves about the same performance as a 100-rule FLS (when $\sigma_F = 1$).

Given the means of the antecedent membership functions, the antecedent standard deviations determine the overlap between consecutive antecedents. They therefore determine the

TABLE III
MEAN-SQUARED ERROR AND STANDARD DEVIATION OF OUT-OF-SAMPLE RESIDUALS FOR THE PREDICTIVE MODELING OF THE DUFFING CHAOTIC ATTRACTOR USING THE SVD-QR DESIGN METHOD AFTER THE SYSTEM WAS FIRST INITIALIZED USING THE BP METHOD. THE INITIAL FLS HAD $M = 100$ RULES IN ITS RULEBASE AND ANTECEDENT STANDARD DEVIATION PRIOR TO RUNNING THE BP METHOD, EQUAL TO $\sigma_F = 1$

Number of rules	mse_{out}	std_{out}
100	2.560×10^{-5}	0.005030
80	3.133×10^{-5}	0.005587
60	3.756×10^{-5}	0.006112
40	4.421×10^{-5}	0.006649
20	4.987×10^{-5}	0.007056
10	3.103×10^{-4}	0.017570
5	3.803×10^{-4}	0.019470

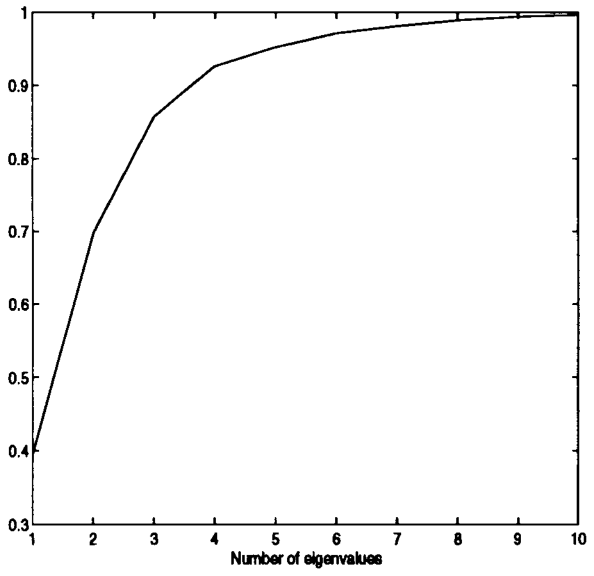
smoothness of the interpolation, or approximation surface generated by the fuzzy system, and how well the antecedents cover their corresponding input space. From Tables I and II, we observe that for 100 rules, the system with smaller antecedent standard deviation, $f^{(1)}$, performs better than the system with larger one, $f^{(2)}$. But when the number of rules is reduced to five, the performance of $f^{(1)}$ degrades abruptly, whereas $f^{(2)}$ performs reasonably well, in comparison to the initial systems with all the rules included. This is due to the fact that for a small number of rules, the input space of $f^{(1)}$ is not sufficiently covered, whereas $f^{(2)}$ is still able to generate a smooth approximation surface.

Fig. 6(a) and (b) depicts the normalized sum of singular values corresponding to $f^{(1)}$ and $f^{(2)}$, respectively. Because of the greater antecedent standard deviation, there exists a higher degree of overlap, and thus more redundancy in the rulebase of $f^{(2)}$. Observe that after the fifth singular value, the contribution of the remaining singular values to the normalized sum is very small, whereas in the case of $f^{(1)}$ there still exist significant singular values after the fifth one. This implies that we can find a smaller subset of rules that could represent $f^{(2)}$ rather than $f^{(1)}$.

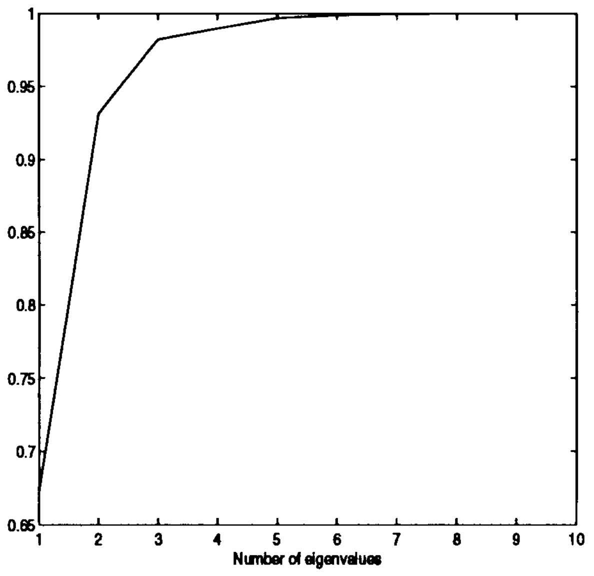
In another experiment, we designed an initial FLS with 100 rules and three inputs, again using the OP method, which was then updated using the BP method, and then further refined using the SVD-QR method. Table III summarizes the results for different number of rules, after employing the SVD-QR method. Observe, by comparing Table III to Tables I and II, that performance is usually better for a combined design than for just the SVD-QR design.

V. CONCLUSION

The ability of FLS's to produce accurate models of complicated systems is not surprising, since they can uniformly approximate any real continuous function on a compact set [33]. The uniform approximation theorem guarantees the existence of a FLS capable of approximating a given function to any degree of accuracy, but does not tell us how to obtain that system. However, since FLS's are trainable, we can begin with an initial system, which can be updated to come very close to the desired FLS. We have given a sample of the many training methods available for designing FLS's, and have shown that some of them can be used in conjunction with each other, to produce improved results.



(a)



(b)

Fig. 6. (a) Normalized sum of singular values for SVD-QR designed system, with $\sigma_F = 0.5$. (b) Normalized sum of singular values for SVD-QR designed system, with $\sigma_F = 1$.

Selection of which design method to use should be based on the particular problem to be solved, and the advantages and disadvantages of each method. The OP method is a quick and simple way to design a FLS, when a very high degree of accuracy is not critical. It does however provide good insight about how a FLS can be set up, and it can be used to initialize more sophisticated design methods. When the system designer can provide fairly good choices for the antecedent parameters, the RLS design method can be used to estimate the centers of the consequent fuzzy sets. The RLS method converges very fast, and can be used to update a FLS in slowly varying nonstationary environments. In general, it can be viewed as an effective tool for designing very accurate fuzzy systems, especially when compared to the OP method.

Backpropagation trained FLS's also belong to the family of adaptive systems. They can be particularly useful in cases where complex interaction among independent variables necessitates training for all system parameters. Solutions arrived at by BP training can be conceivably better than solutions produced by other methods, but more often than not, they are only locally optimal (i.e., the algorithm becomes trapped at a local minimum on the error curve, and can never reach the global minimum). BP trained FLS's are typically slow to converge, and their overall performance depends on the initial values of the system parameters. However, as is evidenced by the examples presented in this paper, they can still be very useful in producing good models of complicated processes, or even in providing improved initial parameter values for another design method used in conjunction with BP.

The SVD-QR method is a very powerful method that allows us to design highly accurate and parsimonious FLS's, given little training data. Unlike the other training methods, the SVD-QR method can be used to: 1) obtain an estimate of the number of rules that have a significant contribution in the fuzzy model, given an initial set of rules; 2) select the best possible subset of rules from the initial set of rules; and 3) obtain an estimate of the vector containing the centers of the consequent fuzzy sets that is both optimal in a least-squares sense, and has the smallest Euclidean norm. The disadvantage of the SVD-QR method is that it cannot be used to adaptively update parameter estimates as new information becomes available.

It should also be noted that these training methods can be extended to design and train nonsingleton FLS's, which are a generalization of singleton FLS's.

Acknowledgment

This work was conducted at the University of Southern California, Los Angeles, CA.

References

[1] A. Bárdossy and L. Duckstein, *Fuzzy Rule-Based Modeling with Applications to Geophysical, Biological and Engineering Systems*. Boca Raton, FL: CRC, 1995.

[2] C.-L. Chen and F.-Y. Chang, "Design and analysis of neural/fuzzy variable structural PID control systems," *Proc. Inst. Elect. Eng. Contr. Theory Applicat.*, 1996, vol. 143, pp. 200–208.

[3] C.-L. Chen and Y.-M. Chen, "Self-organizing fuzzy logic controller design," *Comput. Indust.*, vol. 22, pp. 249–261, Oct. 1993.

[4] S. Chen, S. A. Billings, and W. Luo, "Orthogonal least squares methods and their application to nonlinear system identification," *Int. J. Contr.*, vol. 50, no. 5, pp. 1873–1896, 1989.

[5] M. G. Cooper, "Evolving a rule-based fuzzy controller," *Simulation*, vol. 65, pp. 67–72, 1995.

[6] E. D. Cox, *Fuzzy Logic for Business and Industry*. Rockland, MA: Charles River Media, 1995.

[7] S. Daley and K. F. Gill, "A design study of a self-organizing fuzzy logic controller," in *Proc. Inst. Mechan. Eng. Part C, Mechan. Eng. Sci.*, 1986, vol. 200, pp. 59–69.

[8] J. A. Dickerson and B. Kosko, "Fuzzy function learning with covariance ellipsoids," in *Proc. IEEE Int. Conf. Neural Networks (IEEE ICNN93)*, 1993, pp. 1162–1167.

[9] ——, "Ellipsoidal Learning and fuzzy throttle control for platoons of smart cars," in *Fuzzy Sets, Neural Networks, and Soft Computing*, R. R. Yager and L. A. Zadeh, Eds.. New York: Van Nostrand Reinhold, 1994, pp. 63–84.

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

[11] G. H. Golub, "Numerical methods for solving least squares problems," *Numerische Mathematik*, no. 7, pp. 206–216, 1965.

[12] G. H. Golub and C. Reinsch, "Singular value decomposition and least squares solutions," *Numerische Mathematik*, no. 14, pp. 403–420, 1970.

[13] G. H. Golub and C. F. Van Loan, *Matrix Computations*. Baltimore, MD: Johns Hopkins Univ. Press, 1983.

[14] A. Gonzalez and R. Perez, "Structural learning of fuzzy rules from noised examples," in *Proc. Fourth IEEE Conf. Fuzzy Syst.*, Yokohama, Japan, 1995, vol. 3, pp. 1323–1330.

[15] A. R. Hasan, T. S. Martis, and A. H. M. S. Ula, "Design and implementation of a fuzzy controller based automatic voltage regulator for a synchronous generator," *IEEE Trans. Energy Conv.*, vol. 9, pp. 550–556, 1994.

[16] S. Haykin, *Adaptive Filter Theory*. Englewood Cliffs, NJ: Prentice-Hall, 1991.

[17] H. Hellendoorn and C. Thomas, "Defuzzification in fuzzy controllers," *J. Intell. Syst.*, vol. 1, pp. 109–123, 1993.

[18] J. Hohensohn and J. M. Mendel, "Two-pass orthogonal least-squares algorithm to train and reduce fuzzy logic systems," in *Proc. Third IEEE Conf. Fuzzy Syst.*, Orlando, FL, 1994, vol. 1, pp. 696–700.

[19] Y.-Y. Hsu and C.-H. Chen, "Design of fuzzy power system stabilisers for multimachine power systems," *Proc. Inst. Elect. Eng. Part C, Generation, Transmission and Distribution*, May 1990, vol. 137, pp. 233–238.

[20] J.-S. R. Jang, "Self-learning fuzzy controllers based on temporal backpropagation," *IEEE Trans. Neural Networks*, vol. 3, pp. 714–723, Sept. 1992.

[21] R. J. Jang and C. Sun, "Neuro–fuzzy modeling and control," *Proc. IEEE*, vol. 83, pp. 378–406, Mar. 1995.

[22] R. Langari and L. Wang, "Fuzzy models, modular networks, and hybrid learning," in *Proc. Fourth IEEE Conf. Fuzzy Syst.*, Yokohama, Japan, 1995, vol. 3, pp. 1291–1298.

[23] L. I. Larkin, "A fuzzy logic controller for aircraft flight control," in *Industrial Applications of Fuzzy Control*, M. Sugeno, Ed. Amsterdam, The Netherlands: Elsevier Science (North-Holland), 1985.

[24] C. J. Lin and C. T. Lin, "Reinforcement learning for art-based fuzzy adaptive learning control networks," in *Proc. Fourth IEEE Conf. Fuzzy Syst.*, Yokohama, Japan, 1995, vol. 3, pp. 1299–1306.

[25] D. A. Linkens and H. O. Nyongesa, "Genetic algorithms for fuzzy control. Part 1: Offline system development and application," *Inst. Elect. Eng. Proc. Contr. Theory Applicat.*, 1995, vol. 142, pp. 161–176.

[26] ——, "Genetic algorithms for fuzzy control. Part 2: Online system development and application," *Inst. Elect. Eng. Proc. Contr. Theory Applicat.*, 1995, vol. 142, pp. 177–185.

[27] B. Liu and C. Huang, "Systematic design approach for multivariable fuzzy expert System," in *Proc. Third IEEE Conf. Fuzzy Syst.*, Orlando, FL, 1994, vol. 3, pp. 2094–2099.

[28] M. Mancuso, P. Moretti, and R. Tamagnini, "Fuzzy algorithms for machine vision," *Electron. Eng.*, vol. 67, pp. 51–52, Feb. 1995.

[29] J. M. Mendel, "Fuzzy logic systems for engineering: A tutorial," *Proc. IEEE*, vol. 83, pp. 345–377, Mar. 1995, .

[30] J. A. Momoh, X. W. Ma, and K. Tomsovic, "Overview and literature survey of fuzzy set theory in power systems," *IEEE Trans. Power Syst.*, vol. 10, pp. 1676–1690, 1995.

[31] H. Mori and H. Kobayashi, "Optimal fuzzy inference for short-term load forecasting," *IEEE Trans. Power Syst.*, vol. 11, pp. 390–396, 1996.

[32] G. C. Mouzouris and J. M. Mendel, "Non-Singleton fuzzy logic systems," in *Proc. Third IEEE Int. Conf. Fuzzy Syst.*, June 1994, vol. 1, pp. 456–461.

[33] ——, "Non-Singleton fuzzy logic systems: Theory and application," *USC-SIPI Rep. 262*, 1994; also, *IEEE Trans. Fuzzy Syst.*, vol. 5, pp. 56–71, Feb. 1997.

[34] ——, "Nonlinear time series analysis with nonsingleton fuzzy logic systems," in *IEEE/IAFE Conf. Computat. Intell. Financial Eng.*, New York, Apr. 1995, pp. 47–56.

[35] G. C. Mouzouris and J. M. Mendel, "A singular-value-QR decomposition based method for training fuzzy logic systems in uncertain environments,"*J. Intell. Fuzzy Syst.*, 1996.

[36] T. S. Parker and L. O. Chua, *Practical Numerical Algorithms for Chaotic Systems*. New York: Springer-Verlag, 1989.

[37] W. Pedrycz, "Design of fuzzy control algorithms with the aid of fuzzy models," in *Industrial Applications of Fuzzy Control*, M. Sugeno, Ed. Amsterdam, The Netherlands: North-Holland, 1992, pp. 139–151.

[38] S. N. Rasband, *Chaotic Dynamics of Nonlinear Systems*. New York: Wiley, 1990.

[39] R. M. Tong, "An annotated bibliography of fuzzy control," in *Industrial Applications of Fuzzy Control*, M. Sugeno, Ed. Amsterdam, The Netherlands: Elsevier Science (North-Holland), 1985.

[40] R. Vaccaro, Ed., *SVD and Signal Processing Algorithms, II, Algorithms, Analysis and Applications*. New York: Elsevier, 1991.

[41] F. Vidal-Verdu and A. Rodriguez-Vazquez, "Using building blocks to design analog neuro–fuzzy controllers," *IEEE Micro*, vol. 15, pp. 49–57, 1995.
[42] L. X. Wang, *Adaptive Fuzzy Systems and Control*. Englewood-Cliffs, NJ: Prentice-Hall, 1994.
[43] L. X. Wang and J. M. Mendel, "Generating fuzzy rules from numerical data, with applications," *USC-SIPI Rep. 169*, 1991. See also *IEEE Trans. Syst., Man, Cybern.*, vol. 22, pp. 1414–1427, 1992.
[44] L. X. Wang and J. M. Mendel, "Fuzzy adaptive filters, with application to nonlinear channel equalization," *IEEE Trans. Fuzzy Syst.*, vol. 1, pp. 161–170, Aug. 1993.
[45] S. Yasunobu and S. Miyamoto, "Automatic train operation system by predictive fuzzy control," in *Industrial Applications of Fuzzy Control*, M. Sugeno, Ed. Amsterdam, The Netherlands: Elsevier Science (North-Holland), 1985.
[46] J. Zhang and A. J. Morris, "Fuzzy neural networks for nonlinear systems modeling," *Inst. Elect. Eng. Proc., Contr. Theory Applicat.*, 1995, vol. 142, pp. 551–561.

Jerry M. Mendel (S'59–M'61–SM'72–F'78) received the Ph.D. degree in electrical engineering from the Polytechnic Institute of Brooklyn, Brooklyn, NY, in 1963.

Currently he is a Professor of electrical engineering and an Associate Director for Education of the Integrated Media Systems Center at the University of Southern California, Los Angeles, where he has been since 1974. He has published more than 350 technical papers and is the author and/or editor of seven books. His present research interests include higher-order statistics and neural networks applied to array processing and prediction of nonlinear time-series; and fuzzy logic applied to prediction of nonlinear time-series, classification problems, and social science problems.

Dr. Mendel is a Distinguished Member of the IEEE Control Systems Society. He was President of the IEEE Control Systems Society in 1986. Among his awards are the 1983 Best Transactions Paper Award of the IEEE Geoscience and Remote Sensing Society, the 1992 Signal Processing Society Paper Award, and a 1984 IEEE Centennial Medal.

George C. Mouzouris was born in Cyprus, in 1966. He received the B.S. (with honors) and the M.S. degress from Brown University, Providence, RI, both in 1990, and the Ph.D. degree from the University of Southern California, Los Angeles, in 1996.

From 1990 to 1992, he was with the Digital Signal Processing Group of Texas Instruments, Houston, where he dealt with the design and implementation of signal and image processing algorithms on specialized processors. He is currently with the Signal and Image Processing Institute at the University of Southern California. His research interests lie in the areas of nonlinear dynamic modeling, fuzzy systems, radial basis functions, neural networks, and financial modeling.

Dr. Mouzouris is a member of Tau Beta Pi and Sigma Xi.