USC-SIPI REPORT #184

Analysis and Design of Fuzzy Logic Controller

by

Li-Xin Wang and Jerry M. Mendel

August 1991

Signal and Image Processing Institute
UNIVERSITY OF SOUTHERN CALIFORNIA
Department of Electrical Engineering-Systems
3740 McClintock Avenue, Room 404
Los Angeles, CA 90089-2564 U.S.A.

ANALYSIS AND DESIGN OF FUZZY LOGIC CONTROLLER

Li-Xin Wang and Jerry M. Mendel

July 31, 1991

Abstract

In this report, the fuzzy logic controller (FLC) with product inference, centroid defuzzification, and Gaussian membership function is proven to be capable of approximating any real continuous function on a compact set to arbitrary accuracy. The Stone-Weierstrass Theorem is used as a principal tool for the analysis. Design parameters of FLC are defined, and a parallel implementation architecture for FLC is proposed. An optimal multi-input-single-output FLC is designed which can match N given input-output data pairs to arbitrary accuracy using N fuzzy rules. In order to overcome the high complexity weakpoint of the optimal FLC, a sub-optimal design method is developed. Based on the parallel architecture of FLC, the sub-optimal FLC is designed by using a new back-propagation training algorithm. This report shows that the new back-propagation FLC (BP FLC) can utilize both numerical data and linguistic rules; specifically, the BP FLC is first trained to match the given input-output data pairs using the back-propagation algorithm, then linguistic rules from human experts are added to the trained BP FLC to form the final FLC. A method of choosing the initial parameters of the BP FLC is proposed which is based on the optimal FLC; this method is shown to be very effective and makes the training for the BP FLC very fast. The BP FLC is finally applied to approximate a controller for a non-linear system. By comparing the BP FLC with the back-propagation feedforward neural network (BP FNN), this report shows that: (1) the BP FLC can be applied to any problem which is suited for the BP FNN; (2) the BP FLC can utilize both numerical

and linguistic information, while the BP FNN can only utilize numerical information; and, (3) the training for the BP FLC is much faster than that for the BP FNN.

1 INTRODUCTION

Following the pioneering research of Mamdani and his colleagues on fuzzy logic controller (FLC) in the middle seventies [20-22], there have been many successful applications of FLC to a wide variety of practical problems. Notable applications of FLC include the control of: warm water [11], robot [2,8,33,37,42], heat exchange [28], traffic junction [29], cement kiln [16,41], activated sludge [9,38], automobile speed [23,24], automatic train operation systems [55-57], model-car parking and turning [34-36], turning [32], aircraft [15], water purification [54], automatic container crane operation systems [58,59], elevator [4], automobile transmission [10], and power systems and nuclear reactor [1,12]. Recent advances of fuzzy memory devices and fuzzy chips [39,40,50-53] make fuzzy systems especially suitable for industrial applications.

A very fundamental theoretical question about FLC remains unanswered, namely: "Why does a FLC have such excellent performance for such a wide variety of applications ?" Existing explanations are qualitative, e.g., "FLC is model-free," "FLC can utilize linguistic information from human experts," "FLC can simulate human thinking procedure," "FLC captures the approximate and inexact nature of the real world," etc.. In this report, we try to answer this fundamental question by proving that the FLC is capable of approximating any real continuous function on a compact set to arbitrary accuracy. We use the famous Stone-Weierstrass Theorem [30] to prove this fundamental result. This result can be viewed as an existence theorem of an optimal FLC for a wide variety of problems.

How can this optimal FLC be achieved (designed) for a practical problem ? In this report, we provide an optimal FLC design which is capable of approximating N given input-output data pairs

to arbitrary accuracy using N fuzzy rules. In order to overcome the high complexity weakpoint of the optimal FLC, a sub-optimal FLC is designed by using a new back-propagation training algorithm. A FLC (which is shown in this paper to have a natural parallel architecture), equipped with this new back-propagation training algorithm, is named a "BP FLC." The BP FLC can be trained to match a set of given input-output data pairs using the back-propagation algorithm. Since the trained BP FLC still has the basic structure of the FLC, it is quite easy to incorporate linguistic IF-THEN rules from human experts into the trained FLC and form the final FLC, which is therefore designed based on both numerical and linguistic information.

For many practical problems, the information concerning the system which is to be identified or controlled is often represented in two forms: one is a set of input-output data pairs, obtained by measuring the outputs of the system for some typical input signals; and, the other is a set of linguistic descriptions about the system, often in the form of IF-THEN fuzzy rules, from human experts who are very familiar with the behavior of the system. Traditional system identification methods [5,19], even including the (model-free) neural network approaches [25,31], can only utilize input-output data pairs. If the input signal is not sufficiently "rich", then some modes of the system are not excited; hence, it is impossible for these traditional methods to correctly identify the system. If, somehow, rules from experts contain some linguistic descriptions for these unexcited modes, then we can hope that a design method which utilizes both numerical input-output data pairs and linguistic IF-THEN rules can correctly identify the system. The BP FLC approach of this report is such a method.

In Section 2, we present a system description of the FLC, and define the design parameters of the FLC. In Section 3, we prove that a subset of the FLC — the FLC's with product inference, centroid defuzzification, and Gaussian membership function — is capable of approximating any real continuous function on a compact set to arbitrary accuracy; we also present a parallel architecture

to implement the FLC. In Section 4, we design an optimal FLC which can approximate N given input-output data pairs to arbitrary accuracy. In Section 5, the BP FLC is designed and applied to a simple example to show how the BP FLC can correctly approximate a system which cannot be approximated using traditional identification methods (the test input signals for this example are not sufficiently rich). In Section 6, the BP FLC is trained to approximate a controller for a nonlinear ball and beam system [44]. Section 7 concludes the report.

2 SYSTEM DESCRIPTION OF FUZZY LOGIC CONTROLLER

We first define some terminology. *A universe of discourse* U is a collection of objects which can be discrete or continuous. *A fuzzy set* F in a universe of discourse U is characterized by a membership function $\mu_F : U \rightarrow [0,1]$, and is labelled by a linguistic term, where *a linguistic term* is a word such as "small", "medium", "large", "very large", etc.. For example, let U be the values of speed of a car. Then, we can define three fuzzy sets in U, namely, "slow", "medium", and "fast", which are characterized by the membership functions shown in Fig. 1. In this report, we use (F, μ_F) to denote a fuzzy set, where F is the linguistic term labelling the fuzzy set, and μ_F is its membership function.

The basic configuration of a FLC is shown in Fig. 2. Note that there are four principal elements in a FLC: fuzzification interface, fuzzy rule base, fuzzy inference machine, and defuzzification interface. In this paper, we consider multi-input-single-output (MISO) FLC: $U \subset R^n \rightarrow R$.

The *fuzzification interface* is a mapping from the observed input universe of discourse $U \subset R^n$ to the fuzzy sets defined in U. Specifically, let (F_i, μ_{F_i}) be a fuzzy set defined in U, and let $\underline{x} \in U$ be an input to the fuzzification interface; then, the outputs of the fuzzification interface are $\mu_{F_i}(x_i)$ $(i = 1, 2, ..., n)$. For example, Fig. 1 defines a fuzzification interface which maps the speed of a car into the membership values of the speed in the three fuzzy sets labelled as "slow", "medium", and

"fast". If the speed is 45 *mph*, then the outputs of the fuzzification interface are $\mu_{slow}(45) = 0.5$, $\mu_{medium}(45) = 0.5$, and $\mu_{fast}(45) = 0$. There are two factors which determine a fuzzification interface: (1) the number of fuzzy sets defined in the input universe of discourse; and, (2) the specific membership functions for these fuzzy sets. If these two factors are specified, we obtain a fuzzification interface; hence, we can view these two factors as design parameters of a fuzzification interface, where in this report *design parameters* represent the elements one must fix or choose in order to design a FLC. Specifically, for a MISO FLC, the design parameters of a fuzzification interface are: (1) $m_i, i = 1, 2, ..., n$, the number of fuzzy sets defined in the $i'th$ subspace of U, where the $i'th$ *subspace* of U is the projection of U into the $i'th$ coordinate of R^n, i.e., it is the set $[x_i : \underline{x} = (x_1, x_2, ..., x_n) \in U] \subset R$; and, (2) $\mu_{F_i^j}$, $i = 1, 2, ..., n, j = 1, 2, ..., m_i$, the membership function of the $j'th$ fuzzy set defined in the $i'th$ subspace of U.

The *fuzzy rule base* is a set of linguistic statements in the form of "IF a set of conditions are satisfied, THEN a set of consequences are inferred", where the conditions and the consequences are associated with fuzzy concepts (i.e., linguistic terms). For example, in the case of an n-input-single-output FLC, the fuzzy rule base may consist of the following rules:

$$R_j : IF\ x_1\ is\ A_1^j\ and\ x_2\ is\ A_2^j\ and\ \cdots\ and\ x_n\ is\ A_n^j,\ THEN\ z\ is\ B^j, \quad (1)$$

where x_i $(i = 1, 2, ..., n)$ are the inputs to the FLC, z is the output of the FLC, A_i^j and B^j $(j = 1, 2, ..., K)$ are linguistic terms, and K is the number of fuzzy rules in the fuzzy rule base. These fuzzy IF-THEN rules provide a natural form in which humans represent their knowledge. By relating each linguistic term in the fuzzy rules with a membership function (e.g., as in Fig. 1), we specify the meaning of the fuzzy rules in a determined fuzzy sense. There are many different kinds of fuzzy rules; see [17] for a complete discussion. In this paper, we consider only fuzzy rules

in the form of (1). The design parameters of a fuzzy rule base are: (1) K, the number of fuzzy rules in the fuzzy rule base; and, (2) the specific statement of each fuzzy rule.

The *fuzzy inference machine* is decision making logic which employs fuzzy rules from the fuzzy rule base to determine fuzzy outputs of a FLC corresponding to the fuzzified inputs to the FLC. It is the fuzzy inference machine that simulates a human decision making procedure based on fuzzy concepts and linguistic statements. There are many different kinds of fuzzy logic which may be used in a fuzzy inference machine; see [18] for a comprehensive review. The design parameter of a fuzzy inference machine is: which specific fuzzy logic is used.

The *defuzzification interface* defuzzifies the fuzzy output of a FLC to generate a nonfuzzy output which is used as the actual control to the plant (see Fig. 2). There are three existing defuzzification methods, namely: centroid (i.e, center of area), max-criterion, and mean of maximum (see [18] for details). The design parameters of a defuzzification interface are: (1) number of fuzzy sets defined in the output universe of discourse R; (2) specific membership functions of these fuzzy sets; and, (3) which defuzzification method is used.

In summary, an FLC has the following design parameters:

(d.1) number of fuzzy sets defined in the input and output universes of discourse;

(d.2) membership functions of these fuzzy sets;

(d.3) number of fuzzy rules in the fuzzy rule base;

(d.4) linguistic statements of the fuzzy rules;

(d.5) decision making logic used in the fuzzy inference machine; and,

(d.6) defuzzification method.

We see that a MISO FLC is a complicated nonlinear system which maps a nonfuzzy $U \subset R^n$ into the nonfuzzy R. There are a wide variety of FLC design parameters which give its design great flexibility. Next we briefly discuss the roles of each design parameter.

The number of fuzzy sets defined in the input and output universes of discourse and the number of fuzzy rules in the fuzzy rule base heavily influence the complexity of a FLC, where complexity includes time complexity, i.e., the computational requirements of the FLC, and space complexity, i.e., the storage requirements of the FLC. These parameters can be viewed as structure parameters of a FLC. In general, the larger these parameters are, the more complex is the FLC, and the higher is the expected performance of the FLC (see [43] for examples). Hence, there is always a trade-off between complexity and accuracy in the choice of these parameters; and, their choice is usually quite subjective.

The membership functions of the fuzzy sets heavily influence the "smoothness" of the input-output surface determined by the FLC. In general, the "sharper" the membership functions are, the less smooth is the input-output surface. The choice of membership functions is also quite subjective.

The linguistic statements of the fuzzy rules are the heart of a FLC in the sense that it is these linguistic statements that contain most of the information concerning the FLC design; all other design parameters assist in the effective representation and use of the information. The fuzzy rules usually come from two sources: human experts, and training data. A general method to generate fuzzy rules from numerical data was proposed in [43], and was successfully applied in [44] to temperature prediction, truck backer-upper control [27], and chaotic time-series prediction [14]. Another method was proposed in [13] to generate fuzzy rules from numerical data using vector quantization. It seems that except for the work of [43,44] and [13], the derivation of fuzzy rules is quite subjective.

The decision making logic used by the fuzzy inference machine is very important, and may be the most flexible component in the FLC. If we compare a FLC with a human controller, then the fuzzification interface corresponds to our sensory organs (e.g., eye, ear, etc.), the defuzzification

interface corresponds to our action organs (e.g., arms, feet, etc.), the fuzzy rule base corresponds to our memory, and the fuzzy inference machine corresponds to our thought process. A sophisticated FLC may need a sophisticated fuzzy inference machine.

The role of the defuzzification strategy in a FLC is somewhat unclear because there are only three defuzzification methods available, among which the centroid method seems to provide the best performance for most applications [18].

In this section, we only presented a very brief description of the FLC. Our purpose was to highlight and specify its design parameters. For a comprehensive description and discussion of the FLC, see [17,18].

3 ANALYSIS OF THE FUZZY LOGIC CONTROLLER AND ITS PARALLEL IMPLEMENTATION

We have shown that MISO FLC's can be viewed as mappings from a nonfuzzy universe of discourse $U \subset R^n$ to R, and, these mappings are characterized by the six design parameters (d.1)-(d.6). All these mappings consist of a function space X. Since there are great flexibilities in choosing the design parameters, X is a very large space. In this section, we analyse the properties of a subset of X which is determined by fixing some design parameters in the following way:

(a) fuzzy rules in the fuzzy rule base are all in the form of (1);

(b) all membership functions are of the following *Gaussian form*:

$$\mu_{A_i^j}(x_i) = a_i^j exp[-\frac{1}{2}(\frac{x_i - \bar{x}_i^j}{\sigma_i^j})^2], \tag{2}$$

where $i = 0, 1, 2, ..., n, j = 1, 2, ..., K$ (K is the number of fuzzy rules in the fuzzy rule base), $i = 0$ represents the membership functions for the output space, and, $i = 1, 2, ..., n$ represent the

membership functions for the input space; A_i^j has the same meaning as in (1) ($A_0^j \equiv B^j$); $0 < a_i^j \leq 1$; $\bar{x}_i^j$ is the point in the $i'th$ input subspace (for $i = 1, 2, ..., n$) or the output space (for $i = 0$) at which the fuzzy set $(A_i^j, \mu_{A_i^j})$ achieves its maximum membership value; and, $\sigma_i^j \in (0, \infty)$ characterizes the shape of the Gaussian membership function;

(c) product inference logic [16] (which is specified below) is used in the fuzzy inference machine; and,

(d) centroid defuzzification method (which is also specified below) is used in the defuzzification interface.

Definition 1: The set of *FLC's with product inference, centroid defuzzification, and Gaussian membership functions*, denoted by Y in the sequel, consists of all functions of the form

$$f(\underline{x}) = \frac{\sum_{j=1}^{K}(\bar{z}^j \Pi_{i=1}^{n} \mu_{A_i^j}(x_i))}{\sum_{j=1}^{K}(\Pi_{i=1}^{n} \mu_{A_i^j}(x_i))}, \tag{3}$$

where: f: $U \subset R^n \rightarrow R$, $\underline{x} = (x_1, x_2, ..., x_n) \in U$, K is the number of fuzzy rules in the fuzzy rule base, $\mu_{A_i^j}(x_i)$ is the Gaussian membership function in (2), and $\bar{z}^j$ is the point in the output space R at which μ_{B^j} achieves its maximum value (in the notation of (2) we have $\bar{z}^j = \bar{x}_0^j$). We assume $K \geq 1$, and that U is compact [26,30].

From (3), observe that if we view the fuzzy inference machine and defuzzification interface as an integrated part, then product inference logic can be explained as that the "weight" of Rule j to the contribution of determining the output of the FLC for input $\underline{x}$ equals $\Pi_{i=1}^{n} \mu_{A_i^j}(x_i)$. Additionally, centroid defuzzification means that the nonfuzzy output of the FLC is a weighted sum of the K points in R at which the membership functions characterizing the linguistic terms in the conclusion parts of the K rules achieve their maximum values, where the "weights" are determined by the product inference machine. We also see from (3) that the "shape" (characterized by the a_i^j and σ_i^j)

of the membership functions defined in the output space has no influence on the set Y, because only the "centers" $\bar{z}^j$ enter the right-hand side of (3).

The design parameters of the FLC in Y are:

(d.i) m_i, $i = 1, 2, ..., n$, the number of fuzzy sets defined in the $i'th$ subspace of the input universe of discourse U, and, m_0, the number of fuzzy sets defined in the output space R;

(d.ii) $\bar{x}_i^j, \sigma_i^j, a_i^j$ and $\bar{z}^j$ $(i = 1, 2, ..., n, j = 1, 2, ..., K)$, the parameters of the Gaussian membership functions of the fuzzy sets defined in the input and output spaces;

(d.iii) K, the number of fuzzy rules in the fuzzy rule base, with $K \geq 1$; and,

(d.iv) the specific statements of the fuzzy rules which are in the form of (1).

Let $d_\infty(f_1, f_2)$ be the *sup-metric* [26,30] defined by

$$d_\infty(f_1, f_2) = sup_{\underline{x} \in U}(|f_1(\underline{x}) - f_2(\underline{x})|), \tag{4}$$

then (Y, d_∞) is a metric space. To analyse this metric space, one may first ask: "Is Y empty ?," and, "Is (Y, d_∞) *well-defined* ?" (i.e., for any $\underline{x} \in U$ and any $f \in Y$, does there exist $z \in R$ such that $f(\underline{x}) = z$?)

LEMMA 1: Y is non-empty.

Proof: This is an obvious conclusion from the assumption $K \geq 1$. Q.E.D..

LEMMA 2: (Y, d_∞) is well-defined.

Proof: Since Y is non-empty, we only need to prove that the denominator of (3) is nonzero for any $\underline{x} \in U$. Based on (2), the Gaussian membership functions are nonzero (we assumed $0 < a_i^j \leq 1$); hence, the denominator of (3) is nonzero. Q.E.D..

From the proof of Lemma 2 we see that if we change the membership functions into the triangular form, e.g., the membership function for the "medium" speed in Fig. 1, then the resulting (Y, d_∞)

may not be well-defined, because for an arbitrary f in such Y we cannot guarantee that the denominator of f is nonzero for every $\underline{x} \in U$. We need other stronger conditions in order for such Y to be well-defined; a set of such conditions was given in [43,44].

Next, we use the Stone-Weierstrass Theorem to prove that (Y, d_∞) is dense in $(C[U], d_\infty)$, where $C[U]$ is the set of all real continuous functions defined on the compact set U.

Stone-Weierstrass Theorem [30]: Let Z be a set of real continuous functions on a compact set U. If:

1) Z is an *algebra*, i.e., the set Z is closed under addition, multiplication, and scalar multiplication;

2) Z *separates points on* U, i.e., for every $\underline{x}, \underline{y} \in U, \underline{x} \neq \underline{y}$, there exists $f \in Z$ such that $f(\underline{x}) \neq f(\underline{y})$; and,

3) Z *vanishes at no point of* U, i.e., for each $\underline{x} \in U$ there exists $f \in Z$ such that $f(\underline{x}) \neq 0$;

then, the uniform closure of Z consists of all real continuous functions on U, i.e., (Z, d_∞) is dense in $(C[U], d_\infty)$.

In order to use the Stone-Weierstrass Theorem to prove that (Y, d_∞) is dense in $(C[U], d_\infty)$, we need to show that Y is an algebra, Y separates points on U, and, Y vanishes at no point of U. The following Lemmas 3-5 prove that Y has these properties.

LEMMA 3: (Y, d_∞) is an algebra.

Proof: Let $f_1, f_2 \in Y$, so that we can write them as

$$f_1(\underline{x}) = \frac{\sum_{j=1}^{K1} (\bar{z1}^j \Pi_{i=1}^n \mu_{A1_i^j}(x_i))}{\sum_{j=1}^{K1} (\Pi_{i=1}^n \mu_{A1_i^j}(x_i))}, \tag{5}$$

$$f_2(\underline{x}) = \frac{\sum_{j=1}^{K2} (\bar{z2}^j \Pi_{i=1}^n \mu_{A2_i^j}(x_i))}{\sum_{j=1}^{K2} (\Pi_{i=1}^n \mu_{A2_i^j}(x_i))}; \tag{6}$$

hence,

$$f_1(\underline{x}) + f_2(\underline{x}) = \frac{\sum_{j1=1}^{K1}\sum_{j2=1}^{K2}(\bar{z1}^{j1} + \bar{z2}^{j2})(\Pi_{i=1}^{n}\mu_{A1_i^{j1}}(x_i)\mu_{A2_i^{j2}}(x_i))}{\sum_{j1=1}^{K1}\sum_{j2=1}^{K2}(\Pi_{i=1}^{n}\mu_{A1_i^{j1}}(x_i)\mu_{A2_i^{j2}}(x_i))}. \tag{7}$$

Since $\mu_{A1_i^{j1}}$ and $\mu_{A2_i^{j2}}$ are Gaussian in form , their product $\mu_{A1_i^{j1}}\mu_{A2_i^{j2}}$ is also Gaussian in form (this can be verified by straightforward algebraic operations); hence, (7) is the same form as (3), so that $f_1 + f_2 \in Y$. Similarly,

$$f_1(\underline{x})f_2(\underline{x}) = \frac{\sum_{j1=1}^{K1}\sum_{j2=1}^{K2}(\bar{z1}^{j1}\bar{z2}^{j2})(\Pi_{i=1}^{n}\mu_{A1_i^{j1}}(x_i)\mu_{A2_i^{j2}}(x_i))}{\sum_{j1=1}^{K1}\sum_{j2=1}^{K2}(\Pi_{i=1}^{n}\mu_{A1_i^{j1}}(x_i)\mu_{A2_i^{j2}}(x_i))}, \tag{8}$$

which is also in the same form of (3); hence, $f_1 f_2 \in Y$. Finally, for arbitrary $c \in R$,

$$cf_1(\underline{x}) = \frac{\sum_{j=1}^{K1}(c\bar{z1}^{j})(\Pi_{i=1}^{n}\mu_{A1_i^{j}}(x_i))}{\sum_{j=1}^{K1}(\Pi_{i=1}^{n}\mu_{A1_i^{j}}(x_i))}, \tag{9}$$

which is again in the form of (3); hence, $cf_1 \in Y$. Q.E.D..

LEMMA 4: (Y, d_∞) separates points on U.

Proof: We prove this by constructing a required f, i.e, we specify the number of fuzzy sets defined in U and R, the parameters of the Gaussian membership functions, the number of fuzzy rules, and the statements of fuzzy rules, such that the resulting f (in the form of (3)) has the property that $f(\underline{x}^0) \neq f(\underline{y}^0)$ for arbitrarily given $\underline{x}^0, \underline{y}^0 \in U$ with $\underline{x}^0 \neq \underline{y}^0$. Let $\underline{x}^0 = (x_1^0, x_2^0, ..., x_n^0)$ and $\underline{y}^0 = (y_1^0, y_2^0, ..., y_n^0)$. If $x_i^0 \neq y_i^0$, we define two fuzzy sets, $(A_i^1, \mu_{A_i^1})$ and $(A_i^2, \mu_{A_i^2})$, in the $i'th$ subspace of U, with

$$\mu_{A_i^1}(x_i) = exp[-\frac{(x_i - x_i^0)^2}{2}], \tag{10}$$

$$\mu_{A_i^2}(x_i) = exp[-\frac{(x_i - y_i^0)^2}{2}]. \tag{11}$$

If $x_i^0 = y_i^0$, then $A_i^1 = A_i^2$ and $\mu_{A_i^1} = \mu_{A_i^2}$, i.e., only one fuzzy set is defined in the $i'th$ subspace of

U. We define two fuzzy sets, (B^1, μ_{B^1}) and (B^2, μ_{B^2}), in the output universe of discourse R, with

$$\mu_{B^j}(z) = exp[-\frac{(z - \bar{z}^j)^2}{2}], \tag{12}$$

where $j = 1, 2$, and $\bar{z}^j$ will be specified later. We choose two fuzzy rules in the form of (1) for the fuzzy rule base (i.e., K=2). Now we have specified all the design parameters except $\bar{z}^j$ $(j = 1, 2)$, i.e., we have already obtained a function f which is in the form of (3) with $K = 2$ and $\mu_{A_i^j}$ given by (10) and (11). With this f, we have

$$f(\underline{x}^0) = \frac{\bar{z}^1 + \bar{z}^2 \Pi_{i=1}^n exp[-(x_i^0 - y_i^0)^2/2]}{1 + \Pi_{i=1}^n exp[-(x_i^0 - y_i^0)^2/2]} = \alpha \bar{z}^1 + (1 - \alpha)\bar{z}^2, \tag{13}$$

$$f(\underline{y}^0) = \frac{\bar{z}^2 + \bar{z}^1 \Pi_{i=1}^n exp[-(x_i^0 - y_i^0)^2/2]}{1 + \Pi_{i=1}^n exp[-(x_i^0 - y_i^0)^2/2]} = \alpha \bar{z}^2 + (1 - \alpha)\bar{z}^1, \tag{14}$$

where

$$\alpha = \frac{1}{1 + \Pi_{i=1}^n exp[-(x_i^0 - y_i^0)^2/2]}. \tag{15}$$

Since $\underline{x}^0 \neq \underline{y}^0$, there must be some i such that $x_i^0 \neq y_i^0$; hence, we have $\Pi_{i=1}^n exp[-(x_i^0 - y_i^0)^2/2] \neq 1$, or, $\alpha \neq 1 - \alpha$. If we choose $\bar{z}^1 = 0$ and $\bar{z}^2 = 1$, then $f(\underline{x}^0) = 1 - \alpha \neq \alpha = f(\underline{y}^0)$. Q.E.D..

LEMMA 5: (Y, d_∞) vanishes at no point of U.

Proof: By observing (3) and (2), we simply choose all $\bar{z}^j > 0$ $(j = 1, 2, ..., K)$, i.e., any $f \in Y$ with $\bar{z}^j > 0$ serves as the required f. Q.E.D..

Now we state the main result of this section.

THEOREM 1: For any given real continuous function g on the compact set $U \subset R^n$ and arbitrary $\epsilon > 0$, there exists $f \in Y$ (Y is defined in Definition 1) such that

$$|g(\underline{x}) - f(\underline{x})| < \epsilon \tag{16}$$

for all $\underline{x} \in U$.

Proof: From (3) and (2), it is obvious that Y is a set of real continuous functions on U. Theorem 1 is therefore a direct consequence of the Stone-Weierstrass Theorem and Lemmas 3-5. Q.E.D..

Theorem 1 is an existence theorem. It shows that for any control problem, if an optimal controller (in whatever sense) performs a real continuous mapping from a compact set $U \subset R^n$ into R, then it is possible to design a FLC, in fact a very special FLC — the FLC in Y (Definition 1), such that the FLC approximates the optimal controller arbitrarily well. Theorem 1 gives a theoretical explanation for some of the successes of FLC in practical applications.

Some comments are now in order:

1) From the proof of Lemma 3 we see that, in order for (Y, d_∞) to be an algebra, it is essential that the product of two membership functions preserves the functional form of each individual membership function. The triangular type of membership functions (e.g., Fig. 1), which are most commonly used in the FLC literature, do not have this reproducing property.

2) From the proof of Lemma 3 we see that if f_1 and f_2 are constructed using $K1$ and $K2$ fuzzy rules, respectively, then we need $K1 \times K2$ fuzzy rules to construct $f_1 + f_2$ or $f_1 f_2$. Consequently, if the number of fuzzy rules is bounded, then the resulting Y will not be an algebra. To permit an unbounded number of fuzzy rules is a strong requirement which, as we will see next, forces us to permit the number of fuzzy sets defined in the input universe of discourse to also be unbounded.

3) If only *nonconflicting fuzzy rules*, i.e., fuzzy rules which must have the same THEN parts if the IF parts are the same, are allowed in the fuzzy rule base, then the number of fuzzy rules is bounded from above by $m_1 m_2 \cdots m_n$, i.e., $K \leq m_1 m_2 \cdots m_n$, where m_i is the number of fuzzy sets defined in the $i'th$ input subspace. This result is due to the facts that the maximum number of combinations of the fuzzy sets defined in the input universe of discourse is $m_1 m_2 \cdots m_n$, and only

nonconflicting fuzzy rules are allowed. Since K must be unbounded in order for the FLC set to be an algebra (which is an essential requirement in order to obtain our main result in Theorem 1), it seems difficult to use the Stone-Weierstrass Theorem to analyse the FLC with fixed number of fuzzy sets defined in the input universe of discourse.

Theorem 1 shows that the FLC in Y can approximate continuous functions. The following corollary generalizes the result of Theorem 1 to discrete functions.

COROLLARY 1: For any $g \in L_2(U)$ and arbitrary $\epsilon > 0$, there exists $f \in Y$ such that

$$(\int_U |f(\underline{x}) - g(\underline{x})|^2 d\underline{x})^{1/2} < \epsilon, \tag{17}$$

where $U \subset R^n$ is compact, $L_2(U) = [g : U \rightarrow R | \int_U |g(\underline{x})|^2 d\underline{x} < \infty]$, and the integrals are in the Lebesgue sense.

Proof: Since U is compact, $\int_U d\underline{x} = V < \infty$. Since continuous functions on U form a dense subset of $L_2(U)$ [30], for any $g \in L_2(U)$ there exists a continuous function $\bar{g}$ on U such that $(\int_U |g(\underline{x}) - \bar{g}(\underline{x})|^2 d\underline{x})^{1/2} < \epsilon/2$. By Theorem 1, there exists $f \in Y$ such that $sup_{\underline{x} \in U} |f(\underline{x}) - \bar{g}(\underline{x})| < \epsilon/(2V^{1/2})$; hence, we have

$$\begin{aligned}
(\int_U |f(\underline{x}) - g(\underline{x})|^2 d\underline{x})^{1/2} &\leq (\int_U |f(\underline{x}) - \bar{g}(\underline{x})|^2 d\underline{x})^{1/2} + (\int_U |\bar{g}(\underline{x}) - g(\underline{x})|^2 d\underline{x})^{1/2} \\
&< (\int_U (sup_{\underline{x} \in U} |f(\underline{x}) - \bar{g}(\underline{x})|)^2 d\underline{x})^{1/2} + \epsilon/2 \\
&< (\frac{\epsilon^2}{2^2 V} V)^{1/2} + \epsilon/2 = \epsilon.
\end{aligned} \tag{18}$$

Q.E.D..

Finally, by analysing the computations involved in the FLC in Y, i.e., by analysing (3) and (2), we observe a very important fact: most of these computations can be performed in parallel. For given $\underline{x} \in U$, the computations involved in the FLC in Y consist of two steps: first, compute the

$K \times n$ membership values, $\mu_{A_i^j}(x_i)$, $i = 1, 2, ..., n, j = 1, 2, ..., K$, according to (2); then, substitute these $K \times n$ values into (3) to obtain the output $f(\underline{x})$. We see from (2) that the computations of the $K \times n$ $\mu_{A_i^j}(x_i)$ are independent, i.e., all these computations can be performed in parallel. Consequently, we have a parallel implementation scheme for the FLC in Y, shown in Fig. 3. From Fig. 3 we see that the FLC can be viewed as a three-layer feedforward network (it may also be viewed as a structured network [45-49]). In Section 5, we will develop a new back-propagation algorithm to train this network.

4 DESIGN OF AN OPTIMAL FUZZY LOGIC CONTROLLER

Consider the following design problem: Given N input-output pairs $(\underline{x}^j, z^j)$, where $j = 1, 2, ..., N$, $\underline{x}^j \in U \subset R^n$, $z^j \in R$, and $\underline{x}^j \neq \underline{x}^l$ for $j \neq l$, design an FLC f: $U \rightarrow R$ with product inference, centroid defuzzification, and Gaussian membership function (Definition 1), such that

$$J = \sum_{j=1}^{N} [f(\underline{x}^j) - z^j]^2 \tag{19}$$

is minimized.

We propose the following design procedure:

Step 1: Define N fuzzy sets, $(A_i^j, \mu_{A_i^j})$, $j = 1, 2, ..., N$, in the $i'th$ subspace of U, with

$$\mu_{A_i^j}(x_i) = exp[-\frac{1}{2}(\frac{x_i - x_i^j}{\sigma})^2], \tag{20}$$

where $i = 1, 2, ..., n$, x_i^j is the $i'th$ component of the given $\underline{x}^j$, and $\sigma > 0$ is a *free* design parameter, e.g., we can choose σ such that (22) is satisfied;

Step 2: Define N fuzzy sets, (B^j, μ_{B^j}), $j = 1, 2, ..., N$, in the output universe of discourse R,

with

$$\mu_{B^j}(z) = exp[-\frac{1}{2}(\frac{z - z^j}{\sigma})^2], \tag{21}$$

where z^j are the outputs in the given input-output pairs, and σ is the same free design parameter as in (20); and,

Step 3: Use N fuzzy rules in the form of (1) ($K = N$) in the fuzzy rule base.

These steps specify a FLC in Y. This FLC is *optimal* in the sense that by properly choosing the free design parameter σ, we can make the objective function J arbitrarily small.

THEOREM 2: For arbitrary $\epsilon > 0$, there exists $\sigma^* > 0$ such that the FLC f^*: $U \rightarrow R$, obtained from the preceding three-step design procedure and with the free design parameter σ^*, has the following property:

$$J^* = \sum_{j=1}^{N} [f^*(\underline{x}^j) - z^j]^2 < \epsilon. \tag{22}$$

Proof: From the design procedure and (3), we have

$$\begin{aligned} f^*(\underline{x}^j) &= \frac{\sum_{k=1}^{N} z^k (\Pi_{i=1}^{n} exp[-(x_i^j - x_i^k)^2/2\sigma^{*2}])}{\sum_{k=1}^{N} (\Pi_{i=1}^{n} exp[-(x_i^j - x_i^k)^2/2\sigma^{*2}])} \\ &= \frac{z^j + \sum_{j \neq k=1}^{N} z^k (\Pi_{i=1}^{n} exp[-(x_i^j - x_i^k)^2/2\sigma^{*2}])}{1 + \sum_{j \neq k=1}^{N} (\Pi_{i=1}^{n} exp[-(x_i^j - x_i^k)^2/2\sigma^{*2}])}, \end{aligned} \tag{23}$$

where $j = 1, 2, ..., N$. Because we assume $\underline{x}^j \neq \underline{x}^k$ for $j \neq k$, there exist some i such that $x_i^j \neq x_i^k$ for all $j, k = 1, 2, ..., N$ with $j \neq k$. Hence, for arbitrary $\epsilon_1 > 0$ and any $j, k = 1, 2, ..., N, j \neq k$, we can make $\Pi_{i=1}^{n} exp[-(x_i^j - x_i^k)^2/2\sigma^{*2}] < \epsilon_1$ by properly choosing σ^*, because $exp[-(x_i^j - x_i^k)^2/2\sigma^{*2}] \rightarrow 0$ as $\sigma^* \rightarrow 0$ if $x_i^j \neq x_i^k$. From this result and (23) we conclude that there exists $\sigma^* > 0$ such that $|f^*(\underline{x}^j) - z^j| < (\epsilon/N)^{1/2}$; hence, we have $J^* = \sum_{j=1}^{N} [f^*(\underline{x}^j) - z^j]^2 < \epsilon$. Q.E.D..

5 DESIGN OF A SUB-OPTIMAL FUZZY LOGIC CONTROLLER THROUGH BACK-PROPAGATION TRAINING

The optimal FLC designed in the last section has N fuzzy rules. If N is very large, which is often the case in practice, then the optimal FLC has a huge fuzzy rule base which makes the implementation of the optimal FLC difficult even when parallel hardware is available. In this section, we will design a FLC for which the number of fuzzy rules is fixed. Specifically, we first fix the number of fuzzy rules $K = K_0$, and then choose other design parameters (which will be defined next) such that the resulting FLC best matches the given N input-output pairs. In general, we choose $K_0 << N$. The membership functions are of the form of (2) with $a_i^j \equiv 1$; we choose $a_i^j \equiv 1$ because for practical problems it is reasonable to assume that every fuzzy membership function achieves unity membership value at some point. The design parameters for the FLC of this section are: (1) $\bar{z}^j, j = 1, 2, ..., K_0$; (2) $\bar{x}_i^j, i = 1, 2, ..., n, j = 1, 2, ..., K_0$; and, (3) $\sigma_i^j, i = 1, 2, ..., n, j = 1, 2, ..., K_0$. We will develop *a new error back-propagation training algorithm* to determine these design parameters.

For a given input-output pair $(\underline{x}^p, \hat{f}^p)$ $(p = 1, 2, ..., N)$, we define an error:

$$e(\underline{x}^p) = \frac{1}{2}(f(\underline{x}^p) - \hat{f}^p)^2. \tag{24}$$

The purpose of the error back-propagation training algorithm is to determine the design parameters such that $e(\underline{x}^p)$ is minimized. We will use e, f and $\hat{f}$ to denote $e(\underline{x}^p), f(\underline{x}^p)$ and $\hat{f}^p$, respectively.

To train $\bar{z}^j$, we use

$$\bar{z}^j(k+1) = \bar{z}^j(k) - \alpha \frac{\partial e}{\partial \bar{z}^j}|_{\bar{z}^j = \bar{z}^j(k)}, \tag{25}$$

where $j = 1, 2, ..., K_0$, and $k = 0, 1, 2, ...$. From Fig. 3 we see that f (and hence e) depends on $\bar{z}^j$

only through a, where $f = a/b$,

$$a = \sum_{j=1}^{K_0} (\bar{z}^j y^j), \tag{26}$$

$$b = \sum_{j=1}^{K_0} y^j, \tag{27}$$

and

$$y^j = \Pi_{i=1}^{n} \mu_{A_i^j}(x_i^p); \tag{28}$$

hence, using the chain rule, we have

$$\frac{\partial e}{\partial \bar{z}^j} = (f - \hat{f}) \frac{\partial f}{\partial a} \frac{\partial a}{\partial \bar{z}^j} = (f - \hat{f}) \frac{1}{b} y^j. \tag{29}$$

Substituting (29) into (25), we obtain the training algorithm for $\bar{z}^j$:

$$\bar{z}^j(k+1) = \bar{z}^j(k) - \alpha \frac{f - \hat{f}}{b} y^j, \tag{30}$$

where $j = 1, 2, ..., K_0$, and $k = 0, 1, 2, ...$

To train $\bar{x}_i^j$, we use

$$\bar{x}_i^j(k+1) = \bar{x}_i^j(k) - \alpha \frac{\partial e}{\partial \bar{x}_i^j}|_{\bar{x}_i^j = \bar{x}_i^j(k)}, \tag{31}$$

where $i = 1, 2, ..., n, j = 1, 2, ..., K_0$, and $k = 0, 1, 2, ...$ We see from Fig. 3 that f (and hence e) depends on $\bar{x}_i^j$ only through y^j; hence, using the chain rule, we have

$$\frac{\partial e}{\partial \bar{x}_i^j} = (f - \hat{f}) \frac{\partial f}{\partial y^j} \frac{\partial y^j}{\partial \bar{x}_i^j}. \tag{32}$$

Now

$$\frac{\partial f}{\partial y^j} = \frac{\partial}{\partial y^j} [\frac{\sum_{l=1}^{K_0} \bar{z}^l y^l}{\sum_{l=1}^{K_0} y^l}] = \frac{\bar{z}^j - f}{b} \tag{33}$$

and (using the Gaussian form of the membership functions)

$$\frac{\partial y^j}{\partial \bar{x}_i^j} = y^j \frac{x_i^p - \bar{x}_i^j}{\sigma_i^{j2}}. \tag{34}$$

Substituting (32)-(34) into (31), we obtain the training algorithm for $\bar{x}_i^j$:

$$\bar{x}_i^j(k+1) = \bar{x}_i^j(k) - \alpha \frac{f - \hat{f}}{b}(\bar{z}^j - f) y^j \frac{x_i^p - \bar{x}_i^j(k)}{\sigma_i^{j2}(k)}, \tag{35}$$

where $i = 1, 2, ..., n, j = 1, 2, ..., K_0$, and $k = 0, 1, 2, ...$

Using the same method as above, we obtain the following training algorithm for σ_i^{j2}:

$$\begin{aligned} \sigma_i^{j2}(k+1) &= \sigma_i^{j2}(k) - \alpha \frac{\partial e}{\partial \sigma_i^{j2}}\Big|_{\sigma_i^{j2} = \sigma_i^{j2}(k)} \\ &= \sigma_i^{j2}(k) - \alpha \frac{f - \hat{f}}{b}(\bar{z}^j - f) y^j \frac{(x_i^p - \bar{x}_i^j(k))^2}{2(\sigma_i^{j2}(k))^2}, \end{aligned} \tag{36}$$

where $i = 1, 2, ..., n, j = 1, 2, ..., K_0$, and $k = 0, 1, 2, ...$

The training algorithm of (30), (35) and (36) is an error back-propagation procedure: to train $\bar{z}^j$, the "normalized" error $(f - \hat{f})/b$ is back-propagated to the layer of $\bar{z}^j$, then $\bar{z}^j$ is updated using (30) in which y^j is the input to $\bar{z}^j$ (see Fig. 3); to train $\bar{x}_i^j$ and σ_i^{j2}, the "normalized" error $(f - \hat{f})/b$ times $(\bar{z}^j - f)$ and y^j is back-propagated to the processing unit of Layer 1 whose output is y^j; then, $\bar{x}_i^j$ and σ_i^{j2} are updated using (35) and (36) respectively in which the remaining variables $\bar{x}_i^j, x_i^p$ and σ_i^{j2} (i.e., the variables on the right-hand sides of (35) and (36), except the back-propagated error $\frac{f - \hat{f}}{b}(\bar{z}^j - f) y^j$) can be obtained locally.

The training procedure for the FLC of Fig. 3 is a two-pass procedure: first, for a given input $\underline{x}^p$, compute forward along the network (i.e., the FLC) to obtain y^j $(j = 1, 2, ..., K_0)$, a, b and f; then, train the network parameters $\bar{z}^j, \bar{x}_i^j$ and σ_i^{j2} $(i = 1, 2, ..., n, j = 1, 2, ..., K_0)$ backward using (30),

(35) and (36), respectively. The purpose of training is for the FLC to match the given input-output pairs $(\underline{x}^p, \hat{f}^p)$ for $p = 1, 2, ..., N$. The back-propagation training algorithm of (30), (35) and (36) can only train the FLC for one input-output pair at a time; hence, we need to train the FLC in turn for all the N pairs. We call the FLC of Fig. 3, equipped with the error back-propagation training algorithm (30), (35) and (36), a "Back-Propagation Fuzzy Logic Control", or "BP FLC".

The error back-propagation training algorithm is a steepest descent algorithm; hence, a good choice of initial parameters $\bar{z}^j(0), \bar{x}_i^j(0)$ and $\sigma_i^{j2}(0)$ is essential to its performance (i.e., convergence, rate of convergence, etc.). Now we propose a method of choosing the initial parameters. Let the given input-output pairs be $(\underline{x}^p, \hat{f}^p)$, $p = 1, 2, ..., N$, and suppose that $N = m \times K_0$, where K_0 is the number of rules in the FLC and m is an integer. First, rearrange the pairs such that $\hat{f}^i \leq \hat{f}^{i+1}$, $i = 1, 2, ..., N-1$, for the sequence of pairs $[(\underline{x}^1, \hat{f}^1), (\underline{x}^2, \hat{f}^2), ..., (\underline{x}^N, \hat{f}^N)]$. Then, choose

$$\bar{z}^j(0) = \frac{1}{m} \sum_{k=1}^{m} \hat{f}^{k+(j-1)m}, \tag{37}$$

and,

$$\bar{x}_i^j(0) = \frac{1}{m} \sum_{k=1}^{m} x_i^{k+(j-1)m}, \tag{38}$$

where $j = 1, 2, ..., K_0$, and $i = 1, 2, ..., n$. Finally, let $x_{imax} = max(x_i^1, x_i^2, ..., x_i^N)$ and $x_{imin} = min(x_i^1, x_i^2, ..., x_i^N)$, $i = 1, 2, ..., n$, and choose

$$\sigma_i^{j2}(0) = (\frac{x_{imax} - x_{imin}}{K_0})^2, \tag{39}$$

where $j = 1, 2, ..., K_0$ and $i = 1, 2, ..., n$.

Since the output of the FLC is a weighted sum of $\bar{z}^j$, it is reasonable to think that a good choice of initial $\bar{z}^j(0)$ is to make $\bar{z}^j(0)$ $(j = 1, 2, ..., K_0)$ uniformly cover the range of $\hat{f}^p$ $(p =$

$1, 2, ..., N$). (37) and (38) are just such a choice. The purpose of choosing $\sigma_i^{j2}(0)$ from (39) is to make the initial Gaussian membership functions neither too flat nor too sharp. The resulting initial FLC can be viewed as an optimal FLC for matching the K_0 pairs of data: $(\underline{x}^q, \hat{f}^q)$, where $\hat{f}^q = \frac{1}{m}\sum_{k=1}^{m} \hat{f}^{k+(q-1)m}$, $x_i^q = \frac{1}{m}\sum_{k=1}^{m} x_i^{k+(q-1)m}$, and $q = 1, 2, ..., K_0$.

It is interesting to compare the BP FLC with the popular back-propagation feedforward neural network (BP FNN) [31]. They are similar in the following aspects:

1) It was proven in [3,7] that the BP FNN can approximate any real continuous function on a compact set to arbitrary accuracy, provided that there are sufficient hidden-layer neurons; similarly, we proved in Section 3 that the BP FLC can also approximate any real continuous function on a compact set to arbitrary accuracy, provided that there are sufficient fuzzy rules. Therefore, from the system capability point of view, BP FLC and BP FNN are isomorphic.

2) Both BP FLC and BP FNN are multi-layer parallel networks, and, their operations are quite similar. They first compute forward along the network; then the network parameters are trained backward. This cycle continues until the given input-output pairs are matched to a certain degree. After training is completed, both the BP FLC and BP FNN are used as nonlinear approximators. Consequently, from system architecture and operation points of view, BP FLC and BP FNN are isomorphic.

There is a fundamental advantage of the BP FLC over the BP FNN, namely: the BP FLC can use both numerical information (in the form of input-output pairs) and human linguistic information (in the form of fuzzy IF-THEN rules), whereas the BP FNN can only use numerical information. A way of incorporating human linguistic information into the trained BP FLC is given next.

Suppose we want to use a BP FLC to approximate a nonlinear (static) system. We have two kinds of information about the system: one is some input-output pairs of the system which are obtained by measuring the outputs of the system for some typical input signals; the other

is some linguistic rules (in the form of (1)) collected from human experts who are familiar with the behavior of the system. We build a BP FLC to approximate the system, using both kinds of information. First, we choose an appropriate K_0 (the number of fuzzy rules in the BP FLC) by compromising between accuracy and complexity (the larger the K_0 is, the more accurate is the approximation, and the more complex is the BP FLC). Then, we train the BP FLC to match the input-output pairs using the error back-propagation algorithm developed in this section, with the initial parameters given by (37)-(39). The parameters of the trained BP FLC give us the specific forms of the membership functions defined for each coordinate of the input space and the centers of the membership functions defined for the output space. (From (3) we see that only the center values of the output membership functions enter the computation; hence, the shape of the output membership functions has no influence on the BP FLC, i.e., we only need to consider the center values of the output membership functions.) Now we ask the human experts who provide the linguistic rules to view these input membership functions and center values of the output membership functions, and to determine which input membership function corresponds to their linguistic term A_i^j (see (1)) and which output center corresponds to their linguistic term B^j. If the experts can find appropriate input membership functions and center values of the output membership functions for all their linguistic terms A_i^j and B^j in their fuzzy rules, then we just add these linguistic rules to the trained BP FLC to form the final FLC which has (at most) $K_0 + L_0$ rules, where L_0 is the number of expert linguistic rules (since some linguistic rules may agree with the rules in the trained BP FLC, the final FLC has at most $K_0 + L_0$ rules). If the experts cannot find the appropriate membership functions for some linguistic terms in their rules, then we add new membership functions for these linguistic terms, and combine the linguistic rules into the trained BP FLC to form the final FLC.

Now we illustrate the above procedure of combining both numerical and linguistic information

through a very simple example. Suppose that we want to build a FLC to approximate the system $f : [0,3] \rightarrow R$, where

$$f(x) = \begin{cases} 1, & 0 \leq x < 1 \\ 0, & 1 \leq x < 2 \\ 1, & 2 \leq x \leq 3 \end{cases} \tag{40}$$

which is shown in Fig. 4. We have two sets of information: one is a set of 10 input-output pairs: $[(0.1,1),(0.3,1),(0.5,1),(0.7,1),(0.9,1),\ (1.1,0),(1.3,0),(1.5,0),(1.7,0),(1.9,0)]$; and, the other is a set of two linguistic rules: (Rule 1: IF x is medium, THEN $f(x)$ is small), (Rule 2: IF x is large, THEN $f(x)$ is large), where by "x is medium" and "x is large" the experts mean that x is somewhere around 1.5 and 2.5, respectively, and, by "$f(x)$ is small" and "$f(x)$ is large" the experts mean that $f(x)$ is somewhere around 0 and 1, respectively. We see that the 10 input-output pairs only cover the responses of the system for inputs in the interval $[0,2]$; hence, we may expect that the pure BP FLC (without the linguistic rules) will have large approximation error for inputs in the interval [2,3]. On the other hand, the linguistic rules only describe the behavior of the system for inputs somewhere around 1.5 and 2.5; hence, it will have large approximation error for small input (somewhere around 0.5) if we just use the linguistic rules to build the FLC. Since Rule 2 describes the response of the system for input in the interval [2,3], we may expect that the final FLC which combines the trained BP FLC and the linguistic rules will more correctly approximate the system.

Let $K_0 = 5$. Define *a sweep* of training as one forward computation along the BP FLC plus one backward training for one input-output pair, and define *a cycle* of training as the training for all the given input-output pairs in turn with each pair trained for one sweep. We trained the BP FLC for two cycles for the 10 input-output pairs (longer training gave no improvement

for the approximation accuracy; this showed that the training for BP FLC was very fast), with the initial parameters determined from (37)-(39). We chose $\alpha = 0.02$ for the training. The final membership functions for the input space are shown in Fig. 5 as S2, S1, CE, B1 and B2, where the centers of S2, S1, CE, B1 and B2 are at 0.1986, 0.6005, 1.0024, 1.4045 and 1.8070, respectively, and , the "standard deviations $\sigma^{j"}$ of S2, S1, CE, B1 and B2 are 0.3590, 0.3606, 0.3541, 0.3369 and 0.3001, respectively. The centers $\bar{z}^j$ ($j = 1, 2, ..., 5$, for the five rules) of the final output membership functions are at 1.0025,1.0040, 0.4993, -0.0049 and -0.0027, respectively. The input-output response of the trained BP FLC is shown in Fig. 6 which shows that the approximation for the inputs in the interval [0,2] is acceptable (considering that we only have five rules), but the approximation for the inputs in the interval [2,3] is totally wrong.

Next we see how to combine the two linguistic rules into the trained BP FLC. By observing the input membership functions S2, S1 CE, B1 and B2, the experts (in this case, the present authors) felt that they could accept B1 as their "medium", but that there is no acceptable membership function among S2, S1, CE, B1 and B2 that corresponds to their "large"; hence, they added a new input membership function, shown in Fig. 5 as B3, that corresponds to their "large". For the output membership functions, the experts agreed that the membership functions with centers at 1.0040 and -0.0049 correspond to their "large" and "small", respectively. If we use B2, B1, CE, S1 and S2 to denote the output membership functions with centers at 1.0025, 1.0040, 0.4993, -0.0049 and -0.0027, respectively, then Rule 1 becomes: "IF x is B1, THEN $f(x)$ is S1", and Rule 2 becomes: "IF x is B3, THEN $f(x)$ is B1". We see that Rule 1 already exists in the trained BP FLC; hence, we only added Rule 2 to the trained BP FLC to form the final FLC which now has six rules. The input-output response of this final FLC is shown in Fig. 7. Comparing Figs. 6 and 7 we see that great improvement was obtained by adding this new rule. To obtain more accurate approximations we need more data and rules; [44] shows some related examples. We also trained

BP FNN's for these 10 input-output pairs, and the input-output response of the best trained BP FNN was, as expected, about the same as Fig. 6.

In addition to the fundamental advantage of BP FLC over BP FNN (i.e., BP FLC can use both numerical and linguistic information), BP FLC has two other advantages over BP FNN, namely: (1) there is now a very good strategy for choosing the initial parameters for BP FLC, whereas the initial parameters for the BP FNN are usually chosen randomly; due to this advantage, we may expect that the training for BP FLC is much faster than that for BP FNN; our limited simulation experiences (for the examples in this and the next section) support this conjecture; and, (2) the parameters of the trained BP FLC have a very clear physical meaning, i.e., they are the centers and "standard deviations" of the membership functions, while the parameters (weights) of the trained BP FNN have no physical meaning; we may use the parameters of the trained BP FLC to analyse the training data, e.g., the centers and "standard deviations" of the membership functions may give us a sense of how the training data are distributed, etc..

A (possible) disadvantage of the BP FLC is that for some very complicated problems the mapping determined by (3) may not be robust, in the sense that for some input $\underline{x}$ the weights, $\Pi_{i=1}^{n}\mu_{A_i^j}(x_i)$, of all the rules may be very small. In this case we may face near 0/0 operations in a computer program that implements the BP FLC, i.e., underflow may occur in the implementation of BP FLC due to limited word length in computers. In principle, this is not a serious problem because it is always possible to increase the "standard deviations σ_i^j" of the membership functions such that some weights $\Pi_{i=1}^{n}\mu_{A_i^j}(x_i)$ will not be too small. However, in order to achieve higher resolution, we often require smaller σ_i^j ([44] shows some related examples); hence, there seems to be an "uncertainty principle": the larger the σ_i^j's, the more robust the BP FLC, but the poorer the performance of the BP FLC; on the other hand, the smaller the σ_i^j's, the better the performance of the BP FLC, but the less robust the BP FLC.

6 APPLICATION OF THE BP FLC TO NONLINEAR SYSTEM CONTROL

In this section, we apply the BP FLC developed in the last section to approximate a controller for the nonlinear ball and beam system in [6]. Our purpose is to use the BP FLC to control the system to track a desired trajectory for a certain range of initial conditions. We use the input-output linearization algorithm developed in [6] to generate state-control pairs for a few typical initial conditions in the initial condition range, and then train the BP FLC to match these state-control pairs. The trained BP FLC is then used as a controller for the ball and beam system for arbitrary initial conditions in the initial condition range.

The ball and beam system is shown in Fig. 8. The beam is made to rotate in a vertical plane by applying a torque at the center of rotation and the ball is free to roll along the beam. We require that the ball remain in contact with the beam. Our goal is to track a trajectory.

Let $\underline{x} = (r, \dot{r}, \theta, \dot{\theta})^T$ be the state of the system, and $y = r$ be the output of the system. Then, from [6], the system can be represented by the state-space model

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} x_2 \\ B(x_1 x_4^2 - G sin x_3) \\ x_4 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u \tag{41}$$

$$y = x_1, \tag{42}$$

where the control u and parameters B and G are defined in [6]. The purpose of control is to determine $u(\underline{x})$ such that the system output y will track a desired trajectory y_d.

The input-output linearization algorithm of [6] determines the control law $u(\underline{x})$ as follows: for

state $\underline{x}$, compute $v(\underline{x}) = y_d^{(4)} + \alpha_3(y_d^{(3)} - \phi_4(\underline{x})) + \alpha_2(\ddot{y}_d - \phi_3(\underline{x})) + \alpha_1(\dot{y}_d - \phi_2(\underline{x})) + \alpha_0(y_d - \phi_1(\underline{x}))$, where $\phi_1(\underline{x}) = x_1$, $\phi_2(\underline{x}) = x_2$, $\phi_3(\underline{x}) = -BGsinx_3$, $\phi_4(\underline{x}) = -BGx_4cosx_3$, and the α_i are chosen so that $s^4 + \alpha_3 s^3 + \alpha_2 s^2 + \alpha_1 s + \alpha_0$ is a Hurwitz polynomial; compute $a(\underline{x}) = -BGcosx_3$ and $b(\underline{x}) = BGx_4^2 sinx_3$; then, $u(\underline{x}) = (v(\underline{x}) - b(\underline{x}))/a(\underline{x})$.

In our simulations, we chose the desired trajectory to be $y_d = cos(\pi t/5)$ (which is the same as in [6]). We considered the system with initial conditions: $\underline{x}_0^T = (1, 0, \theta_0, 0)$, where $\theta \in [0.0872, 0.2617] (\simeq [5^o, 15^o])$, i.e., the ball was initially fixed (which means that $\dot{r} = 0, \dot{\theta} = 0$) at $r = 1$, with the angle θ in the range $[0.0872, 0.2617]$. We generated the desired state-control pairs by applying the above control law to three typical initial conditions: $\underline{x}_0^T = (1, 0, 0.0872, 0), (1, 0, 0.1744, 0)$, and $(1, 0, 0.2617, 0)$. We discretized the system with sample interval $T = 0.01$ sec. and ran the closed-loop system with each initial condition for 25 sec.; hence, we had $3 \times 25/0.01 = 7500$ state-control pairs. The errors $e = y_d - y$ using the above control law for the three initial conditions are shown in Fig. 9, where the error curves from smaller error to larger error correspond to the initial conditions $(1, 0, 0.0872, 0)$, $(1, 0, 0.1744, 0)$, and $(1, 0, 0.2617, 0)$, respectively. We simulated three BP FLC's with $K_0 = 500, 250$, and 100, respectively. We trained the three BP FLC's for three cycles over the 7500 pairs using the BP training algorithm and the initial parameter choosing method of the last section (more training gave no improvement for the performance of the trained BP FLC used as a controller). We chose $\alpha = 0.01$ for all the training. We used the trained BP FLC to control the system with initial conditions $(1, 0, 0.1221, 0)$ and $(1, 0, 0.2268, 0)$, which were arbitrarily chosen from the initial condition range. The error curves using the trained BP FLC with $K_0 = 500, 250$ and 100 are shown in Figs. 10, 11 and 12, respectively, where the smaller error curves correspond to the initial condition $(1, 0, 0.1221, 0)$. From Figs. 10-12 we see that: (1) the BP FLC with $K_0 = 500$ showed very good performance (when we used the control law of [6] for these two initial conditions, the error curves were almost the same as Fig. 10); (2) the BP FLC with $K_0 = 250$

showed poorer performance initially, but finally the system tracked the desired trajectory quite well; and, (3) the BP FLC with $K_0 = 100$ could control the system to track the desired trajectory for the initial contition $(1, 0, 0.1221, 0)$, but the control system was unstable for the initial condition $(1, 0, 0.2268, 0)$.

7 CONCLUSIONS

In this report, we: (1) proved that the set of FLC's with product inference, centroid defuzzification, and Gaussian membership function is dense in $C(U)$ for the sup-norm (Theorem 1) and dense in $L_2(U)$ for the L_2-norm (Corollary 1) for compact $U \subset R^n$; (2) proposed an optimal FLC design which can match N given input-output pairs to arbitrary accuracy; (3) developed a new error back-propagation training algoritm for the FLC; (4) proposed a method of combining both numerical and linguistic information into the BP FLC design; and, (5) applied the BP FLC to a nonlinear control problem.

By comparing the BP FLC with the back-propagation feedforward neural network (BP FNN), we showed that: (1) a BP FLC can be used for problems which are suited for a BP FNN; (2) a BP FLC can use both numerical and linguistic information in a very efficient way, whereas a BP FNN can only use numerical information; and, (3) even if only numerical information is available, the BP FLC may be preferred over the BP FNN because the training for a BP FLC may be much faster than that for a BP FNN. Of course, to achieve the same approximation accuracy, the complexities of the BP FLC and BP FNN are problem-dependent; but, if linguistic information is available and essential, then the BP FLC is always preferred.

8 ACKNOWLEDGMENTS

The authors would like to thank Dr. J. Hauser for providing us with the ball and beam example.

9 REFERENCES

[1] Bernard, J. A. "Use of rule-based system for process control," *IEEE Contr. Syst. Mag.*, Vol.8, No.5, pp.3-13, 1988.

[2] Ciliz, K., J. Fei, K. Usluel and C. Isik, "Practical aspects of the knowledge-based control of a mobile robot motion," *Proc. 30th Midwest Symp. on Circuits and Systems*, Syracuse, NY, 1987.

[3] Cybenko, G. "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals, and systems*, 1989.

[4] Fujitec, F. "FLEX-8000 series elevator group control system," Fujitec Co., Ltd., Osaka, Japan, 1988.

[5] Goodwin, G. C. and R. L. Payne, *Dynamic System Identification: Experiment Design and Data Analysis*, Academic Press, Inc., 1977.

[6] Hauser, J., S. Sastry and P. Kokotovic, "Nonlinear control via approximate input-output linearization: the ball and beam example," *IEEE Trans. on Automatic Control*, 1991.

[7] Hornik, K., M. Stinchcombe and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, Vol.2, pp.359-366, 1989.

[8] Isik, C., "Identification and fuzzy rule-based control of a mobile robot motion," *Proc. IEEE Int. Symp. Intelligent Control*, Philadelphia, PA, 1987.

[9] Itoh, O., K. Gotoh, T. Nakayama and S. Takamizawa, "Application of fuzzy control to activated sludge process," *Proc. 2nd IFSA Congress*, Tokyo, Janan, pp.282-285, 1987.

[10] Kasai, Y. and Y. Morimoto, "Electronically controlled continuously variable transmission," *Proc. Int. Congress on Transportation Electronics*, Dearborn, MI, 1988.

[11] Kickert, W. J. M. and H. R. Van Nauta Lemke, "Application of a fuzzy controller in a warm water plant," *Automatica*, Vol.12, No.4, pp.301-308, 1976.

[12] Kinoshita, M., T. Fukuzaki, T. Satoh and M. Miyake, "An automatic operation method for

control rods in BWR plants," *Proc. Specialists' Meeting on In-Core Instrumentation and Reactor Core Assessment*, Cadarache, France, 1988.

[13] Kosko, B., *Neural Networks and Fuzzy Systems*, Prentice-Hall: Englewood Cliffs, NJ, 1991.

[14] Lapedes, A. and R. Farber, "Nonlinear signal processing using neural networks: prediction and system modeling," *LA-UR-87-2662*, 1987.

[15] Larkin, L. I., "A fuzzy logic controller for aircraft flight control," in *Industrial Applications of Fuzzy Control*, M.Sugeno, Ed., Amsterdam: North-Holland, pp.87-104, 1985.

[16] Larsen, P. M., "Industrial application of fuzzy logic control," *Int. J. Man Mach. Studies*, Vol.12, No.1, pp.3-10, 1980.

[17] Lee, C. C., "Fuzzy logic in control systems: fuzzy logic controller, part I," *IEEE Trans. on Syst., Man, and Cybern.*, Vol.SMC-20, No.2, pp.404-418, 1990.

[18] Lee, C. C., "Fuzzy logic in control systems: fuzzy logic controller, part II," *IEEE Trans. on Syst., Man, and Cybern.*, Vol.SMC-20, No.2, pp.419-435, 1990.

[19] Ljung, L., *System Identification — Theory for the User*, Prentice-Hall: Englewood Cliffs, NJ, 1987.

[20] Mamdani, E. H., "Applications of fuzzy algorithms for simple dynamic plant," *Proc. IEE*, Vol.121, No.12, pp.1585-1588, 1974.

[21] Mamdani, E. H. and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *Int. J. Man Mach. Studies*, Vol.7, No.1, pp.1-13, 1975.

[22] Mamdani, E. H., "Advances in the linguistic synthesis of fuzzy controllers," *Int. J. Man Mach. Studies*, Vol.8, No.6, pp.669-678, 1976.

[23] Murakami, S., "Application of fuzzy controller to automobile speed control system," *Proc. IFAC Symp. on Fuzzy Information, Knowledge Representation and Decision Analysis*, Marseille, France, pp.43-48, 1983.

[24] Murakami, S. and M. Maeda, "Application of fuzzy controller to automobile speed control system," in *Industrial Applications of Fuzzy Control*, M.Sugeno, Ed., Amsterdam: North-Holland, pp.105-124, 1985.

[25] Narendra, K. S. and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. on Neural Networks*, Vol.1, No.1, pp.4-27, 1990.

[26] Naylor, A. W. and G. R. Sell, *Linear Operator Theory in Engineering and Science*, Springer-Verlag, New York, 1982.

[27] Nguyen, D. and B. Widrow, "The truck backer-upper: an example of self-learning in neural networks," *IEEE Cont. Syst. Mag.*, Vol.10, No.3, pp.18-23, 1990.

[28] Ostergaad, J. J., "Fuzzy logic control of a heat exchange process," in *Fuzzy Automata and Decision Processes*, M.M.Gupta, G.N.Saridis and B.R.Gaines, Eds., Amsterdam: North-Holland, pp.285-320, 1977.

[29] Pappis, C. P. and E. H. Mamdani, "A fuzzy logic controller for a traffic junction," *IEEE Trans. Syst. Man Cybern.*, Vol.SMC-7, No.10, pp.707-717, 1977.

[30] Rudin, W., *Principles of Mathematical Analysis*, McGraw-Hill, Inc., 1976.

[31] Rumelhart, D. E. and J. L. McCleland (eds.), *Parallel Distributed Processing I, II*, Cambridge: MIT Press, 1986.

[32] Sakai, Y., "A fuzzy controller in turning process automation," in *Industrial Applications of Fuzzy Control*, M.Sugeno, Ed., Amsterdam: North-Holland, pp.139-152, 1985.

[33] Scharf, E. M. and N. J. Mandic, "The application of a fuzzy controller to the control of a multi-degree-freedom robot arm," in *Industrial Applications of Fuzzy Control*, M.Sugeno, Ed., Amsterdam: North-Holland, pp.41-62, 1985.

[34] Sugeno, M. and K. Murakami, "Fuzzy parking control of model car," *Proc. 23rd IEEE Conf. on Decision and Control*, Las Vegas, 1984.

[35] Sugeno, M. and M. Nishida, "Fuzzy control of model car," *Fuzzy Sets Syst.*, Vol.16, pp.103-113, 1985.

[36] Sugeno, M. and K. Murakami, "An experimental study on fuzzy parking control using a model car," in *Industrial Applications of Fuzzy Control*, M.Sugeno, Ed., Amsterdam: North-Holland, pp.125-138, 1985.

[37] Tanscheit, R. and E. M. Scharf, "Experiments with the use of a rule-based self-organising controller for robotics applications," *Fuzzy Sets Syst.*, Vol.26, pp.195-214, 1988.

[38] Tong, R. M., M. B. Beck and A. Latten, "Fuzzy control of the activated sludge wastewater treatment process," *Automatica*, Vol.16, No.6, pp.695-701, 1980.

[39] Togai, M. and H. Watanabe, "Expert system on a chip: an engine for real-time approximate reasoning," *IEEE Expert Syst. Mag.*, Vol.1, pp.55-62, 1986.

[40] Togai, M. and S. Chiu, "A fuzzy accelerator for a programming environment for real-time fuzzy control," *Proc. 2nd IFSA Congress*, Tokyo, Japan, pp.147-151, 1987.

[41] Umbers, I. G. and P. J. King, "An analysis of human-decision making in cement kiln control and the implications for automation," *Int. J. Man Mach. Studies*, Vol.12, No.1, pp.11-23, 1980.

[42] Uragami, M., M. Mizumoto and K. Tananka, "Fuzzy robot controls," *Cybern.* Vol.6, pp.39-64, 1976.

[43] Wang, L. X. and J. M. Mendel, "Generating fuzzy rules by learning from examples," *Proc. 6th IEEE International Symposium on Intelligent Control*, 1991.

[44] Wang, L. X. and J. M. Mendel, "Generating fuzzy rules from numerical data, with applications," USC SIPI Report, No. 169, 1991.

[45] Wang. L. X. and J. M. Mendel, "Structured trainable networks for matrix algebra," *Proc. 1990 International Joint Conf. on Neural Networks*, Vol. 2, pp. II125-II132, 1990.

[46] Wang, L. X. and J. M. Mendel, "Matrix computations and equation solving using structured

networks and training," *Proc. IEEE 1990 Conf. on Decision and Control*, pp. 1747-1750, 1990.

[47] Wang, L. X. and J. M. Mendel, "Parallel structured network for solving a wide variety of matrix algebra problems," submitted to *Journal of Parallel and Distributed Processing*, 1991.

[48] Wang, L. X. and J. M. Mendel, "Three-dimensional structured networks for matrix equation solving," *IEEE Trans. on Computers*, to appear, Dec., 1991.

[49] Wang, L. X. and J. M. Mendel, "Cumulant-based parameter estimation using structured networks," *IEEE Trans. on Neural Networks*, Vol. 2, No. 1, pp. 73-83, 1991.

[50] Watanabe, H. and W. Dettloff, "Reconfigurable fuzzy logic processor: a full custom digital VLSI," *Proc. Int. Workshop on Fuzzy System Applications*, Iizuka, Japan, pp.49-50, 1988.

[51] Yamakawa, T. and T. Miki, "The current mode fuzzy logic integrated circuits fabricated by standard CMOS process," *IEEE Trans. Computer*, Vol.C-35, No.2, pp.161-167, 1986.

[52] Yamakawa, T. and K. Sasaki, "Fuzzy memory device," *Proc. 2nd IFSA Congress*, Tokyo, Japan, pp.551-555, 1987.

[53] Yamakawa, T., "Fuzzy microprocessors – rule chip and defuzzification chip," *Proc. Int. Workshop on Fuzzy System Applications*, pp.51-52, 1988.

[54] Yagishita, O., O. Itoh and M. Sugeno, "Application of fuzzy reasoning to the water purification process," in *Industrial Applications of Fuzzy Control*, M.Sugeno, Ed., Amsterdam: North-Holland, pp.19-40, 1985.

[55] Yasunobu, S., S. Miyamoto and H. Ihara, "Fuzzy control for automatic train operation system," *Proc. 4th IFAC/IFIP/IFORS Int. Congress on Control in Transportation Systems*, Baden-Baden, 1983.

[56] Yasunobu, S. and S. Miyamoto, "Automatic train operation by predictive fuzzy control," in *Industrial Application of Fuzzy Control*, M.Sugeno, Ed., Amsterdam: North-Holland, pp.1-18, 1985.

[57] Yasunobu, S., S. Sekino and T. Hasegawa, "Automatic train operation and automatic crane operation systems based on predictive fuzzy control," *Proc. 2nd IFSA Congress*, Tokyo, Japan, pp.835-838, 1987.

[58] Yasunobu, S. and T. Hasegawa, "Evaluation of an automatic container crane operation system based on predictive fuzzy control," *Control Theory Adv. Technol.*, Vol.2, No.3, pp.419-432, 1986.

[59] Yasunobu, S. and T. Hasegawa, "Predictive fuzzy control and its application for automatic container crane operation system," *Proc. 2nd IFSA Congress*, Tokyo, Japan, pp.349-352, 1987.

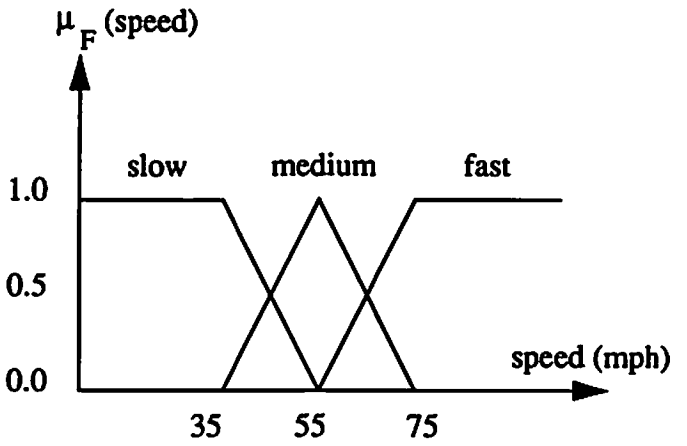

Figure 1: Membership functions of three fuzzy sets, namely, "slow", "medium", and "fast", defined for the speed of a car.

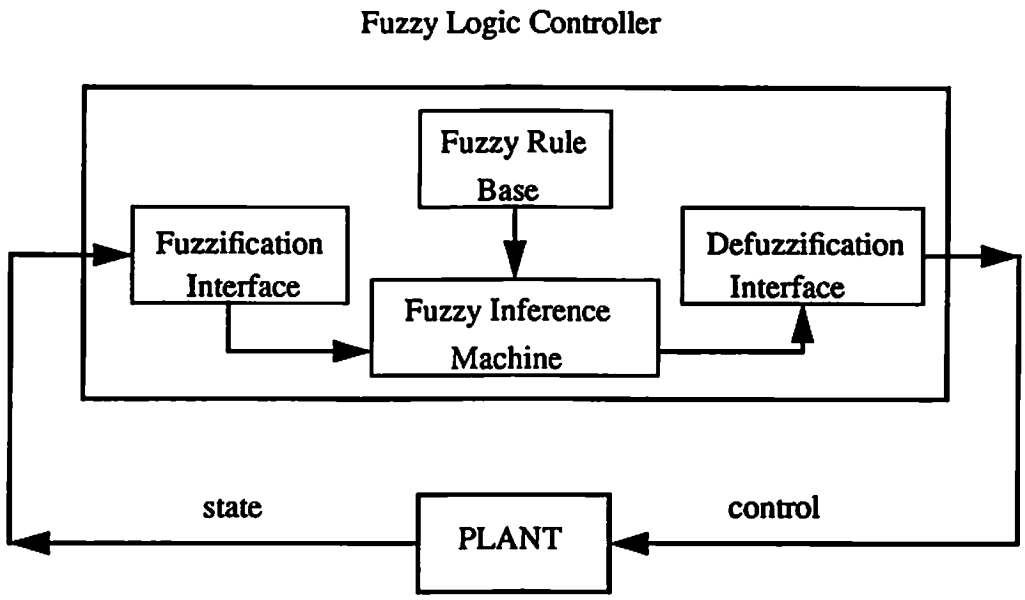

Figure 2: Basic configuration of fuzzy logic controller (FLC).

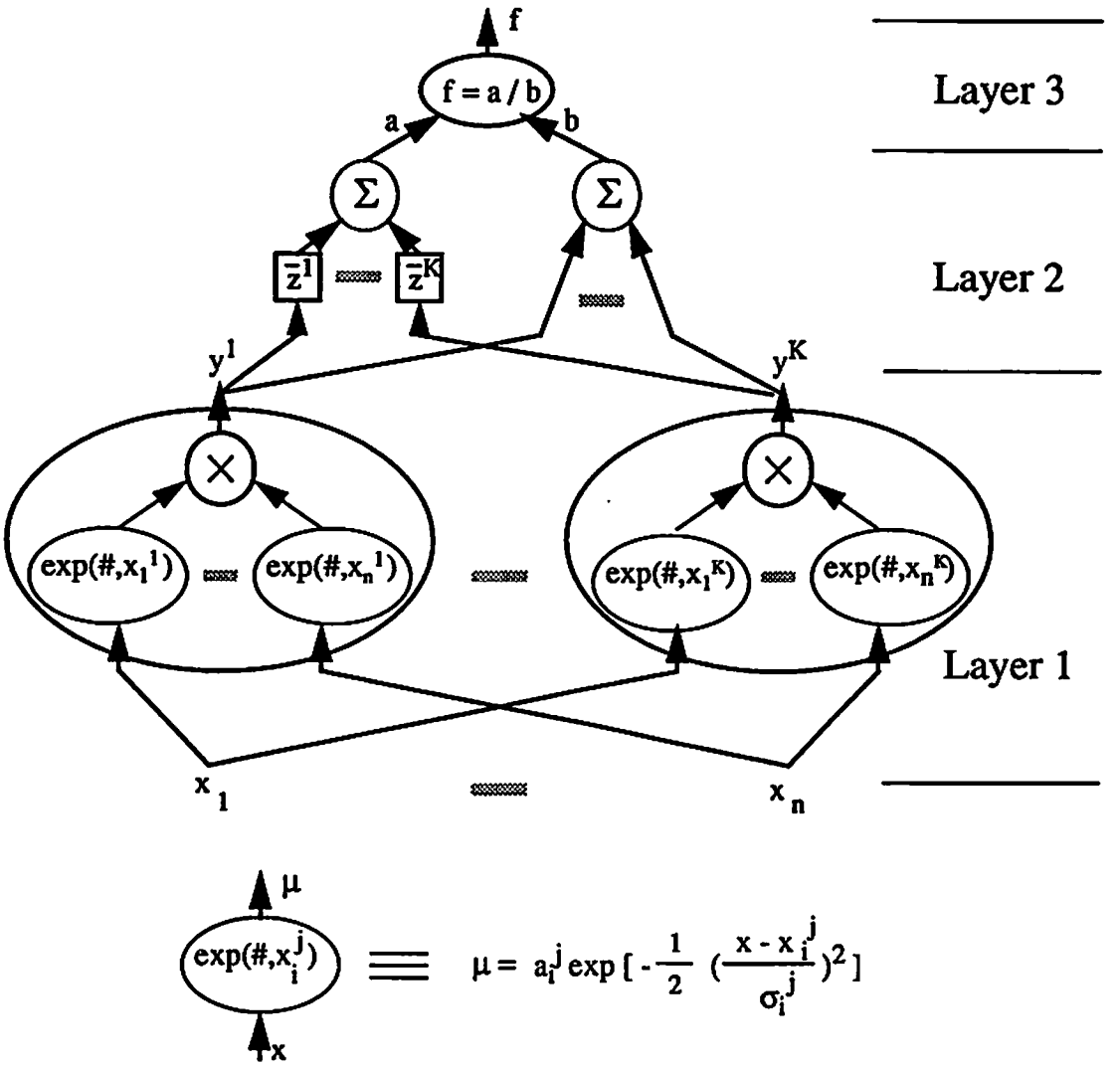

Figure 3: A parallel processing architecture for the fuzzy system.

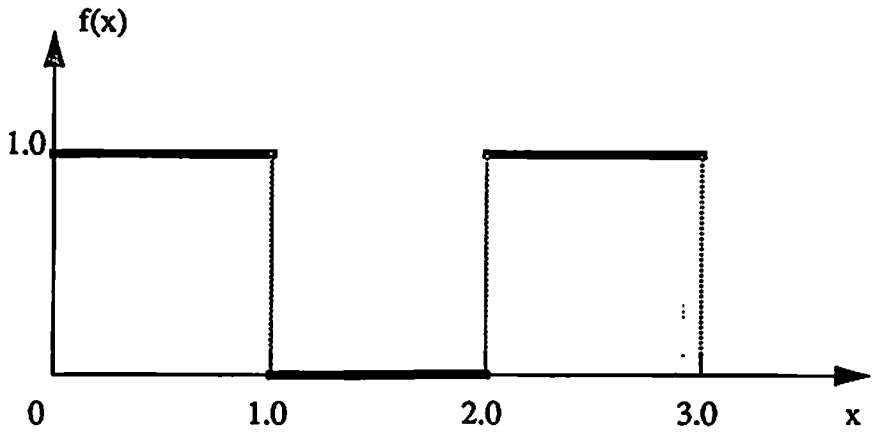

Figure 4: A nonlinear system.

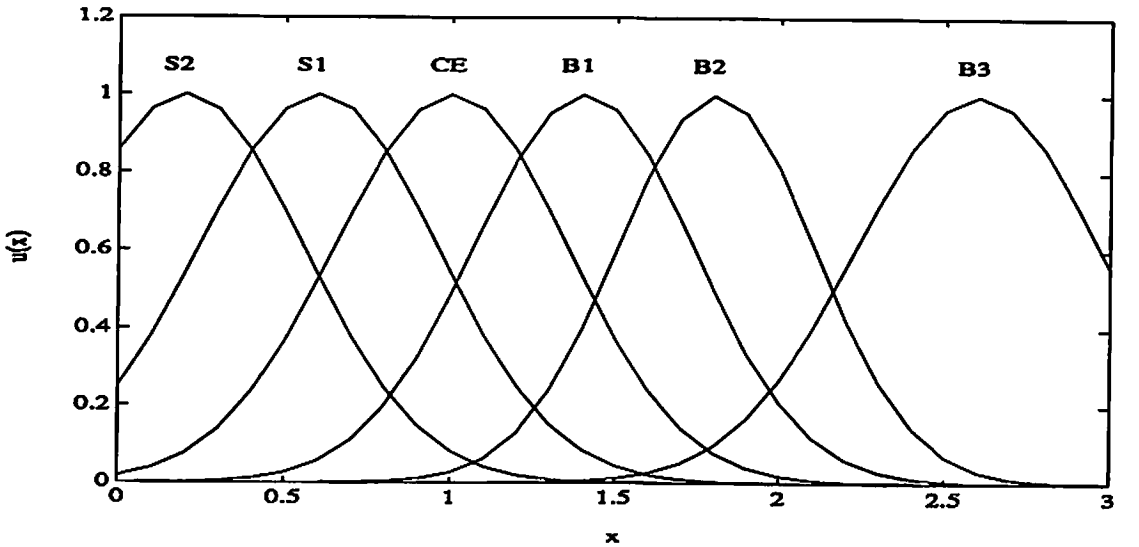

Figure 5: The input membership functions of the trained BP FLC for approximating the function of Fig. 4.

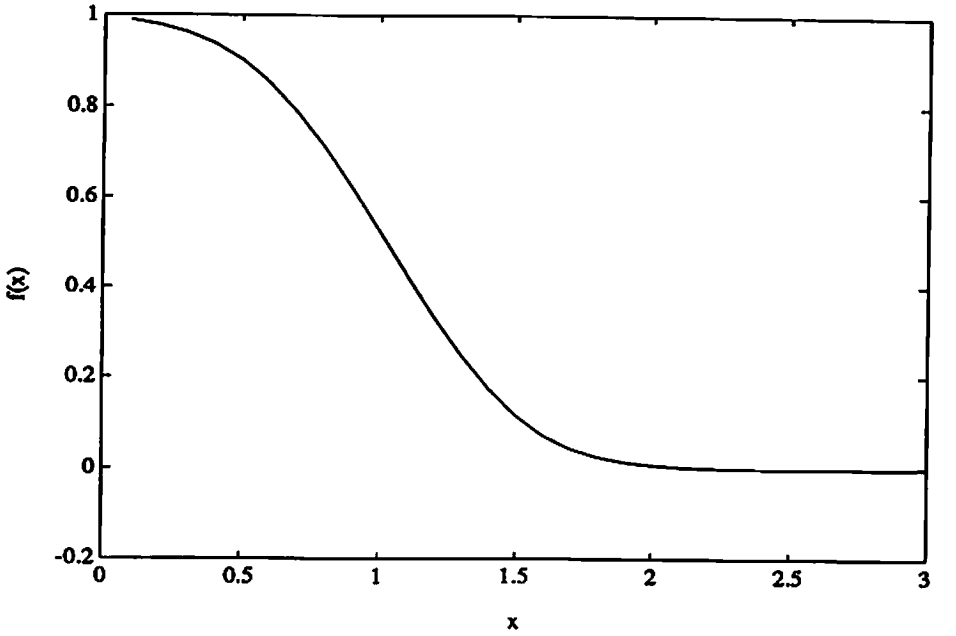

Figure 6: The input-output response of the trained BP FLC (for approximating the function of Fig. 4) using only numerical pairs.

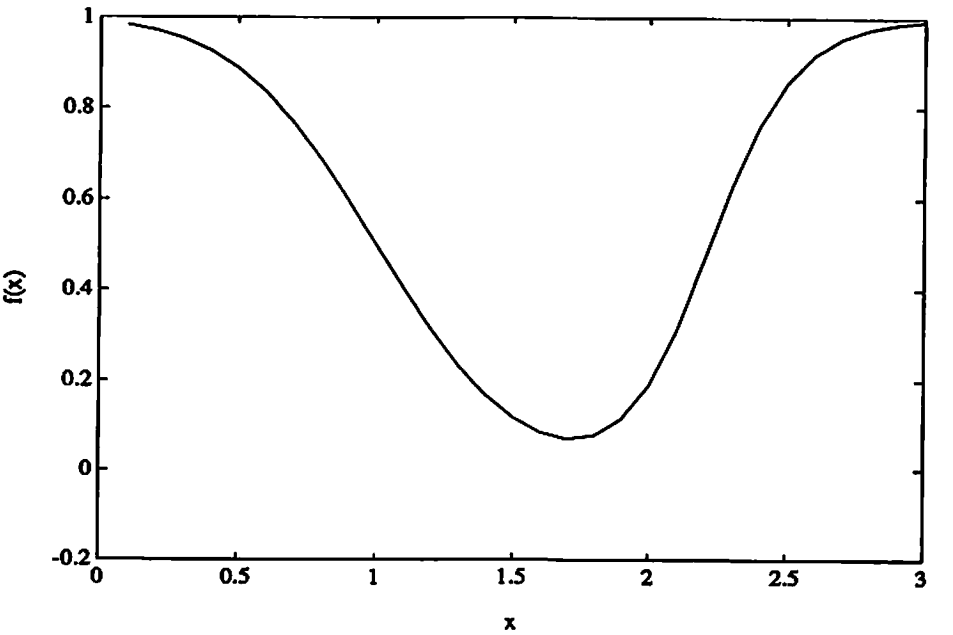

Figure 7: The input-output response of the trained BP FLC (for approximating the function of Fig. 4) using both numerical pairs and linguistic rules.

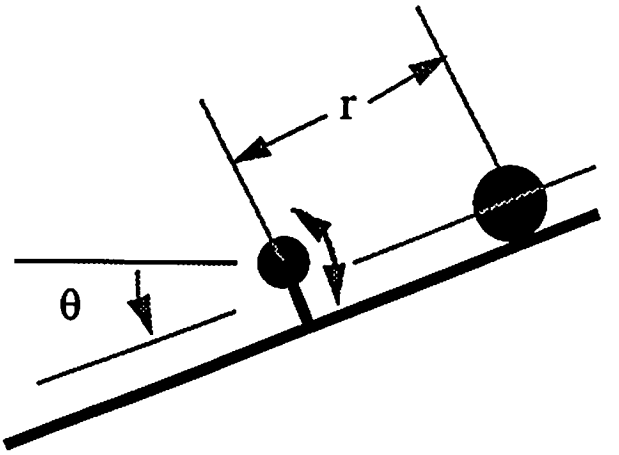

Figure 8: The ball and beam system.

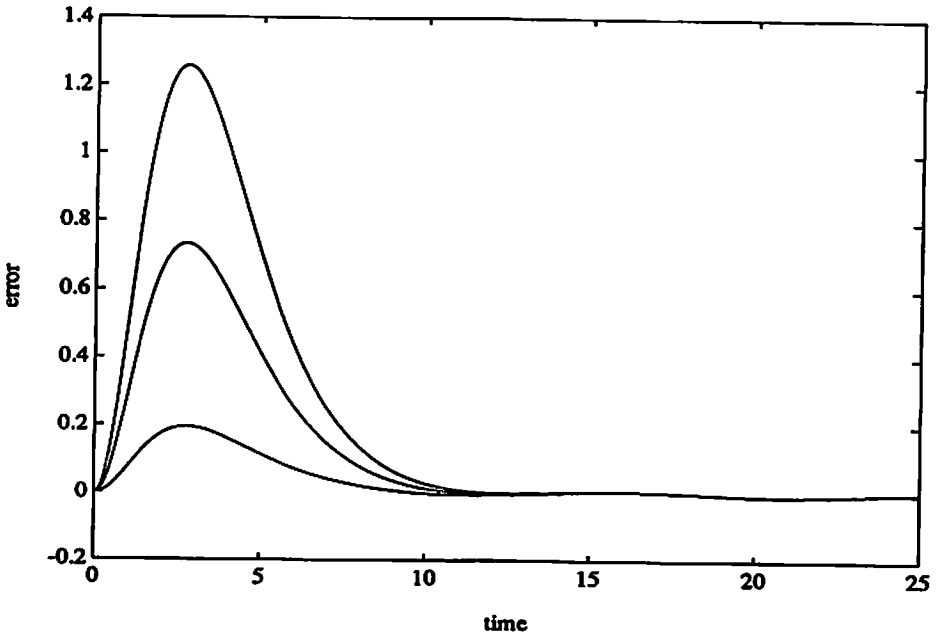

Figure 9: Error curves using the control law of [6] .

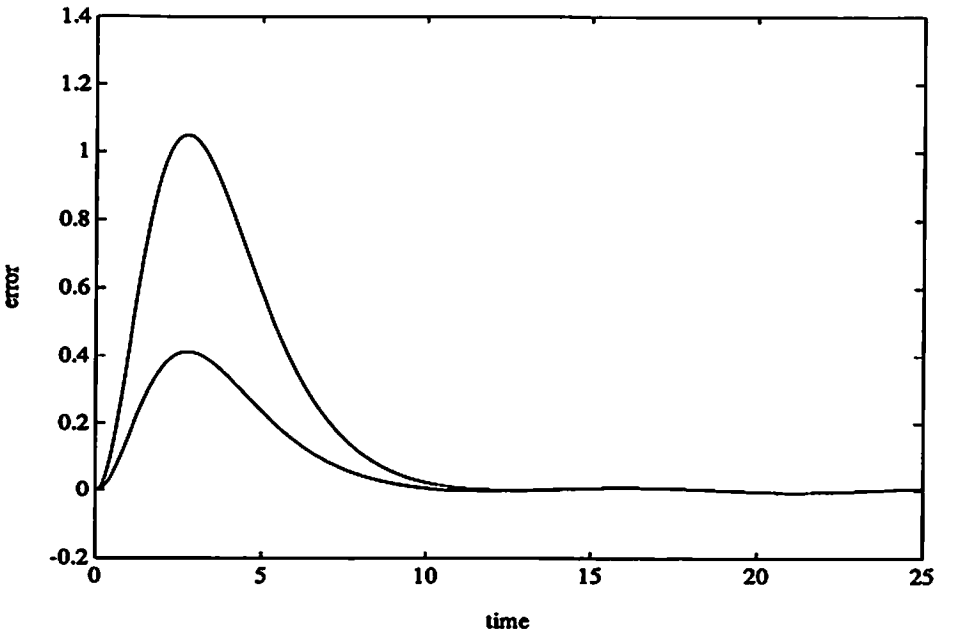

Figure 10: Error curves using the trained PB FLC with $K_0 = 500$.

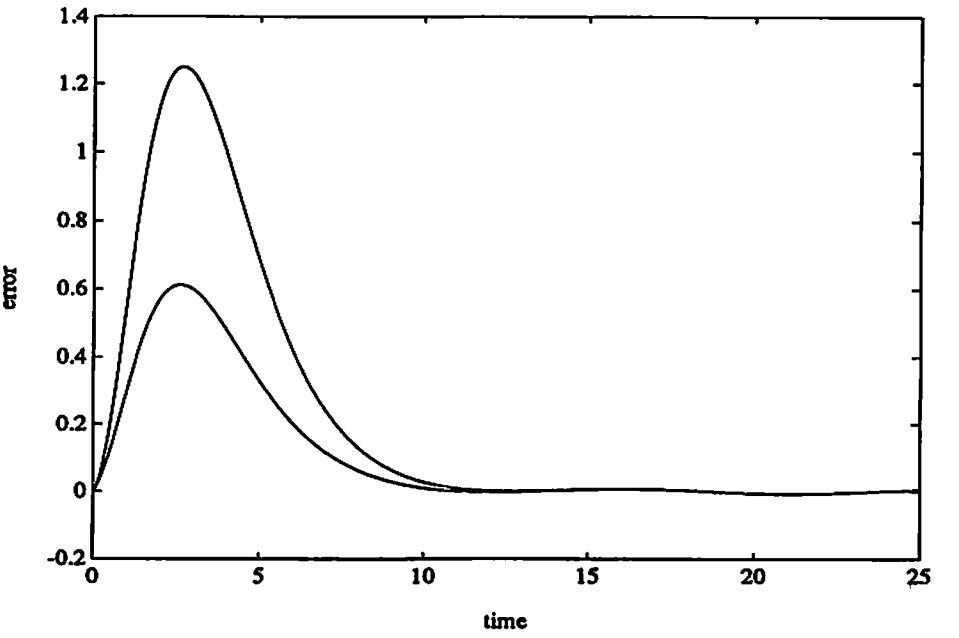

Figure 11: Error curves using the trained PB FLC with $K_0 = 250$.

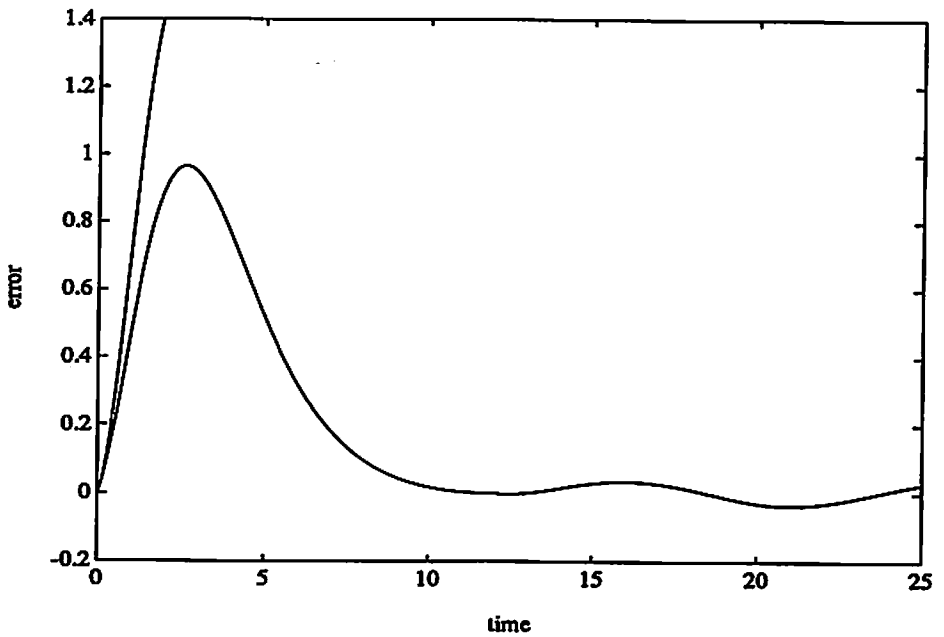

Figure 12: Error curves using the trained PB FLC with $K_0 = 100$.