A STABLE SELF-LEARNING OPTIMAL FUZZY CONTROL SYSTEM

Sinn-Cheng Lin and Yung-Yaw Chen

ABSTRACT

The issue of developing a stable self-learning optimal fuzzy control system is discussed in this paper. Three chief objectives are accomplished: **1) To develop a self-learning fuzzy controller based on genetic algorithms.** In the proposed methodology, the concept of a fuzzy sliding mode is introduced to specify the system response, to simplify the fuzzy rules and to shorten the chromosome length. The speed of fuzzy inference and genetic evolution of the proposed strategy, consequently, is higher than that of the conventional fuzzy logic control. **2) To guarantee the stability of the learning control system**. A hitting controller is designed to achieve this requirement. It works as an auxiliary controller and supports the self-learning fuzzy controller in the following manner. When the learning controller works well enough to allow the system state to lie inside a pre-defined boundary layer, the hitting controller is disabled. On the other hand, if the system tends to diverge, the hitting controller is turned on to pull the state back. The system is therefore stable in the sense that the state is bounded by the boundary layer. **3) To explore a fuzzy rule-base that can minimize a standard quadratic cost function.** Based on the fuzzy sliding regime, the problem of minimizing the quadratic cost function can be transformed into that of deriving an optimal sliding surface. Consequently, the proposed learning scheme is directly applied to extract the optimal fuzzy rule-base. That is, the faster the hitting time a controller has and the shorter the distance from the sliding surface the higher fitness it possesses. The superiority of the proposed approach is verified through simulations.

KeyWords: Fuzzy control, optimal control, fuzzy sliding mode control, genetic algorithms

I. INTRODUCTION

In the real world, there are many systems whose mathematical models can not be easily derived. Therefore, conventional model-based control technology can not be directly applied. Skilled operators, however, can control system operations successfully whether a model is available or not. In fact, the control actions of human operators simply follow certain linguistic rules, such as "**IF** condition *A* happens, **THEN** take action *B*". The fuzzy logic controller (FLC), a controller with linguistic rules that work like human thinking, has been extensively researched [1-5] in the last three decades. By incorporating human expertise into fuzzy IF-THEN rules, an FLC can be constructed to control a complex or ill-defined system [4-5] as human experts do. Knowledge acquisition, therefore, becomes the most important task in FLC design. However, extracting control rules from domain experts is not an easy mission [3]. Many researches have focused on self-learning or self-organizing the fuzzy rule-base [6]. These studies have provided various approaches to generating fuzzy rules, but few of them had dealt with the other major problems of FLC design, such as how to specify the characteristics of the fuzzy control system; how to guarantee the stability of the fuzzy control system and how to get an optimal fuzzy rule-base. This paper proposes a strategy for developing a stable self-learning optimal fuzzy control system so that the aforementioned problems can be solved. Three major goals are accomplished:

First, a self-learning fuzzy control system based on genetic algorithm (GA) is proposed. Genetic algorithms were originally developed by Holland in 1962. The

Manuscript received January 4, 1999; revised April 10, 1999; accepted April 14, 1999.

The authors are with Dept. of Electrical Engineering, National Taiwan University, Taipei, Taiwan.

detailed principles, mathematical frameworks and applications can be found in Goldberg's recent book [7]. The use of GAs for solving control problems was presented in [10]. GAs are applied to a fuzzy control system so as to self-extract a fuzzy rule-base, as proposed in [8-9]. In general, constructing the search space of GAs requires parameterizing the rules or membership functions of an FLC and encoding them into chromosomes. Such strategies, however, have a major drawback: the number of fuzzy rules grows exponentially as the number of either input variables or linguistic labels increases. This further leads to an exponential increase of the encoded chromosome length. To overcome this disadvantage, the fuzzy sliding mode controller (FSMC) [16] is introduced to replace the traditional FLC. The number of rules in an FSMC is in proportion to the number of input variables so that the corresponding chromosome length does not increase exponentially, but linearly [16]. In addition, the response of a fuzzy sliding mode control system is dominated by the sliding surface, which is carefully selected by the designer.

Secondly, a hitting controller is appended as an auxiliary controller so as to satisfy the stability requirement. It works with the self-learning fuzzy controller in the following way: when the learning controller works well enough to allow the system state to lie inside the prespecified boundary layer neighboring the sliding surface, the hitting controller is not activated; if the system tends to diverge, then as soon as the state hits the boundary of the layer, the hitting controller is turned on to pull the state back. Therefore, the hitting controller acts as a gatekeeper that works in a similar fashion to Wang's supervisory controller [19-20].

Finally, finding the optimal fuzzy rule-base which can minimize a quadratic cost function is the third objective in this paper. The optimization problem is solved by deriving an optimal sliding surface first and then applying the previous self-learning approach to find a suitable fuzzy rule-base which drives the state as close to the sliding mode as possible.

This paper is organized as follows: Following the introduction, Section 2 describes the problem we will deal with. Section 3 presents the details of the GA-based self-learning fuzzy controller. Design and analysis of the hitting controller are given in Section 4. Section 5 presents the procedure for transforming the problem of minimizing a quadratic cost function into one of deriving an optimal sliding surface. Section 6 presents simulation results and discussion. Conclusions are drawn in Section 7.

II. PROBLEM FORMULATION

Consider a class of n-th order nonlinear systems which is expressed by the following state equation [20]:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = x_3 \\ \quad\vdots \\ \dot{x}_n = f(\underline{x}) + g(\underline{x})u\,, \end{cases} \tag{1}$$

where $\underline{x} = [x_1\ x_2\ \ldots\ x_n]^T \in \Re^n$ is the state vector; $u \in \Re$ is the control input; f(.) is an unknown continuous function with a known upper bound, i.e., $|f| \le f^U$; $g(.)$ is an unknown positive definite function with a known lower bound, i.e., $0 < g_L \le g$. Actually, equation (1) represents a general uncertain nonlinear dynamic system.

The chief objectives of this paper are:

1) Develop a self-learning fuzzy control system based on genetic algorithms.
2) Guarantee the stability of the learning control system:

$$|x_i| \le \delta_i\,,\ i = 1, 2, \ldots, n\,. \tag{2}$$

3) Find the optimal fuzzy rule-base that can minimize the following quadratic cost function:

$$J = \frac{1}{2}\int_0^\infty \left(\underline{x}^T(t)\,\underline{Q}\,\underline{x}(t) + u^T(t)Ru(t)\right)dt\,, \tag{3}$$

where $Q \in \Re^{n\times n}$ are $R \in \Re$ are two positive definite weighting matrices.

III. GA-BASED SELF-LEARNING FUZZY CONTROLLER

The fuzzy sliding mode control approach has been widely studied in recent years [14-18,21]. Fuzzy sliding mode control is based on fuzzy logic and sliding mode control (SMC) [11-12]. An FSMC has many properties that make it superior to a conventional FLC, so it is more suitable for genetic learning, as shown in our previous studies [14-17]. In this paper, therefore, an FSMC is used in the proposed self-learning fuzzy control system instead of an FLC. This section summarizes the procedure for developing a GA-based FSMC [14].

The first step in FSMC design is to define a sliding function:

$$s(x) = \underline{c}^T\underline{x} = \sum_{n=1}^{\infty} c_i x_i\,, \tag{4}$$

where $\underline{c}^T = [c_1\ \ c_2\ \ \ldots\ \ c_n]^T \in \Re^n$ is a coefficient vector that has to be properly determined. Each rule in a fuzzy sliding mode control system is represented as

$$R_j\text{: } IF\ s\ is\ S_j\ (m_j, \sigma_j)\ THEN\ u_f\ is\ U_j\ (\varphi_j), \tag{5}$$

where $j = 1, 2, \ldots, N$, and N is the number of rules; S_j's are the input linguistic labels, which are simply assigned as

Gaussian-shaped in this paper, i.e.., $\mu_{S_j} = \exp\left(-\left(\frac{s-m_j}{\sigma_j}\right)^2\right)$; U_j's are the output linguistic labels. Furthermore, they are all fuzzy singletons, *i.e.*, $\mu_{U_j}(u) = \begin{cases} 1, & u = \varphi_j \\ 0, & u \neq \varphi_j \end{cases}$.

Suppose that the singleton fuzzification and the weighted average defuzzification methods [20] are adopted; then, the control output of (5) is given by

$$u_f(s) = \underline{\varphi}^T \underline{\rho}(s), \tag{6}$$

where and in which $\underline{\varphi} = [\varphi_1, \varphi_2, \ldots, \varphi_N]^T$ and $\underline{\rho} = [\rho_1, \rho_2, \ldots, \rho_N]^T$, in which $\rho_j(s) = \dfrac{\mu_{S_j}(s)}{\sum_{j=1}^{N} \mu_{S_j}(s)}$.

Constructing a parameter space to be searched by GAs requires transforming the fuzzy rule-base (5) into a parameter representation. Clearly, the output of the rule-base (5) is uniquely determined by a parameter vector θ as

$$\underline{\theta} \equiv [\underline{m}^T \ \underline{\sigma}^T \ \underline{\varphi}^T]^T$$
$$= [m_1 \ m_2 \ \ldots \ m_N \ \sigma_1 \sigma_2 \ \ldots \ \sigma_N \ \varphi_1 \varphi_2 \ \ldots \ \varphi_N]^T.$$

Assume that X_m, X_σ and X_φ are the search spaces of m_j, σ_j and φ_j, respectively; M is the population size; h is the number of generations. The following steps describe the learning procedure for the fuzzy rules:

Step 1. Initially, set $h = 0$ and randomly generate $3M$ initial parameter vectors:

$$\underline{m}^{(i)}(h) = [m_1^{(i)}(h) \ m_2^{(i)}(h) \ \ldots \ m_N^{(i)}(h)]^T,$$
$$m_j^{(i)}(h) \in X_m,$$
$$\underline{\sigma}^{(i)}(h) = [\sigma_1^{(i)}(h) \ \sigma_2^{(i)}(h) \ \ldots \ \sigma_N^{(i)}(h)]^T, \ \sigma_j^{(i)}(h) \in X_\sigma,$$
$$\underline{\varphi}^{(i)}(h) = [\varphi_1^{(i)}(h) \ \varphi_2^{(i)}(h) \ \ldots \ \varphi_N^{(i)}(h)]^T, \ \varphi_j^{(i)}(h) \in X_\varphi,$$

where $i = 1, 2, \ldots, M$.

Step 2. Construct the parameter vector of the i-th individual:

$$\underline{\theta}^{(i)}(h) = [\theta_1^{(i)}(h) \ \ldots \ \theta_N^{(i)} \vdots \ \theta_{N+1}^{(i)}(h) \ \ldots \ \theta_{2N}^{(i)}(h) \vdots \ \theta_{2N+1}^{(i)}(h) \ \ldots \ \theta_{3N}^{(i)}(h)]^T$$
$$= [\underline{m}^{(i)T}(h) \vdots \ \underline{\sigma}^{(i)T}(h) \vdots \ \underline{\varphi}^{(i)T}(h)]^T.$$

Step 3. Encode each parameter in binary code:

$$B_l^{(i)}(h) = enc(\theta_l^{(i)}(h)), \ l = 1, \ldots, 3N,$$

where $enc(.)$ denotes the encoding operator which encodes the real values into the corresponding binary codes.

Step 4. Cascade the binary codes to form the i-th chromosome:

$$P^{(i)}(h) = B_1^{(i)}(h) \ \ldots \ B_N^{(i)}(h) \vdots \ B_{N+1}^{(i)}(h) \ \ldots \ B_{2N}^{(i)}(h) \vdots \ B_{2N+1}^{(i)}(h) \ \ldots \ B_{3N}^{(i)}(h).$$

Step 5. Establish the population in the generation h, $\boldsymbol{P}(h)$:

$$\boldsymbol{P}(h) = \{P^{(1)}(h), P^{(2)}(h), \ldots, P^{(M)}(h)\}.$$

Step 6. Evaluate the fitness value F for each individual using the following equations:

$$J_s = t_s + \sum_{k=1}^{K} \left(w \| s(k) \| + v \| u(k) \| \right), \tag{7a}$$

$$F = 1/(J_s + \varepsilon_0), \tag{7b}$$

where t_s denotes the reach time of the sliding mode; $k = \text{int}(t/\Delta t)$ denotes the iteration instance; Δt is the sampling period; int(.) is the round-off operator; $K = \text{int}(t_{\max}/\Delta t)$ denotes the number of iterations in one run; $t_{\max}$ is the running time in one run; w and v are positive weights. The norm $\| . \|$ can be viewed as a generalized energy measure of a signal. Finally, ε_0 is a small positive constant used to avoid the numerical error "divided by zero".

Based on (7), a controller gets a higher fitness score if it 1) drives the state to reach the sliding surface faster; 2) consumes less control energy; and 3) keep the state on the sliding surface.

Step 7. Apply the genetic operators, *i.e.*, reproduction, crossover and mutation, to generate a new population $\boldsymbol{P}(h + 1)$, which is always known as the offspring of $\boldsymbol{P}(h)$.

Step 8. Keep the elitist. That is, 1) pick up the best fitted individuals in $\boldsymbol{P}(h)$ and $\boldsymbol{P}(h + 1)$; 2) compare their fitness; 3) if the best individual in $\boldsymbol{P}(h)$ has a better fitness value than does that in $\boldsymbol{P}(h +1)$, then randomly replace an individual in $\boldsymbol{P}(h +1)$ with the elitist.

Step 9. Decode each binary code to obtain its real value and use it to calculate the fuzzy control output u_f. Then, apply u_f to the system (1), *i.e.*, $u = u_f$.

Step 10. Set $h = h + 1$; go to Step 2 and repeat the procedure until $F \geq F_M$ or $h \geq H$, where F_M and H denote an acceptable fitness value and a stop generation number, respectively, which are specified by the designer.

Based on the above learning procedure, the genetic algorithm is used to search the parameter space of fuzzy

rules. The controllers which get higher fitness values have a higher probability of generating offspring. Consequently, the fitness value becomes higher and higher from generation to generation. Thus, the state moves closer and closer to the sliding surface. The dynamic behavior of the final control system is, therefore, dominated by the sliding surface.

IV. SYSTEM STABILITY: A HITTING CONTROLLER

During the learning phase described previously, the inferior individuals (the controllers which have unsuitable parameters) may cause unstable behavior. This section, hence, introduces a *hitting controller* as a stabilizer so that

$$|x_i| \le \delta_i, \quad i = 1, 2, \ldots, n. \tag{8}$$

Consequently, the system becomes stable in the sense that the state is bounded.

As mentioned earlier, the sliding function is defined in (4), and the states are defined in (1). Let p denote the Laplace operator. If $X_1(p)$ and $S(p)$, respectively, represent the Laplace transform of x_1 and s, then we have the following filter equation:

$$X_1(p) = \frac{1}{p^{n-1} + c_{n-1}p^{n-2} + \cdots + c_1} S(p) \equiv T(p)S(p). \tag{9}$$

Undoubtedly, the values of c_i's can be carefully assigned so that the poles of the transfer function $T(p)$ are all negative real , *i.e.*,

$$T(p) = 1 \Big/ \prod_{i=1}^{n} (p + \lambda_i), \tag{10}$$

where $\lambda_i \in \Re^+$, $i = 1, 2, \ldots, n-1$. Without loss of generality, let $\lambda_1 \ge \lambda_2 \ge \ldots \ge \lambda_{n-1}$. From (9) and (10), s and x_1 can be, respectively, viewed as the input and output signals of an (n-1)th order filter, which is cascaded by (n-1) first order law-pass filters, as shown in Fig. 1.

Now, our main job is to develop a hitting controller which works in the following way: if the state of the control system is inside a pre-specified boundary layer $\overline{s}$, *i.e.*, $|s| < \overline{s}$ then the hitting controller is turned off. In this situation, the main control action is given by the self-learning fuzzy control law u_f. On the other hand, if the state hits or tends to travel outside the boundary layer, i.e., $|s| \ge \overline{s}$, then the hitting controller is turned on to drive the state back inside the boundary layer. Therefore, the hitting controller acts as a gatekeeper that works in a similar fashion to Wang's supervisory controller [19-20].

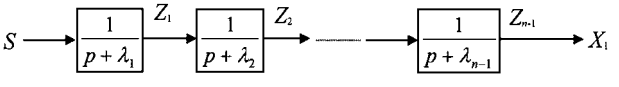

Fig. 1. The relationship between S and X_1.

Achieving the goal described above requires that the hitting control law satisfy the so-called sliding condition [13] when $|s| \ge \overline{s}$, that is,

$$s\dot{s} \le -\eta|s|, \tag{11}$$

where η is a positive constant.

The following theorem provides the methodology for designing the hitting controller so that (11) is satisfied, and so that all the system states are bounded by (8).

Theorem. Consider the system (1) with the control law

$$u = (1-\alpha)u_f + \alpha u_h, \quad \alpha = \begin{cases} 1, & \text{for } |s| \ge \overline{s} \\ 0, & \text{for } |s| < \overline{s}, \end{cases} \tag{12}$$

where u_f is obtained from the self-learning fuzzy control rule-base (5), and the hitting control law u_h is given by

$$u_h = -\text{sign}(s)\left[g_L^{-1}\left(f^U + |\overline{c}^T\overline{x}| + \eta\right)\right], \tag{13}$$

in which $\overline{c} = [c_1\, c_2 \ldots c_{n-1}]^T$ and $\overline{x} = [x_1\, x_2 \ldots x_{n-1}]^T$. Then,
1) the sliding condition (11) is satisfied when $|s| \ge \overline{s}$;
2) the control system is stable in the sense that all the system states x_i (i = 1, 2, …, n) are bounded by

$$|x_i(t)| \le \left(2^{i-1} \Big/ \prod_{j=1}^{n-i} \lambda_j\right)\overline{s} \equiv \delta_i. \tag{14}$$

Proof. 1) Consider the case of $|s| \ge \overline{s}$ (i.e. $\alpha = 1$). Define a Lyapunov function

$$V = \frac{1}{2}s^2. \tag{15}$$

Then, from (1) and (4), we have

$$\dot{V} = s[f + \overline{c}^T\overline{x} + gu_f] + sgu_h, \tag{16}$$

that is,

$$\dot{V} \le |s|\left[|f| + |\overline{c}^T\overline{x}| + g|u_f|\right] + sgu_h. \tag{17}$$

Substituting (13) into (17), we get

$$\dot{V} \le |s|\left[\left(|f| - mf^U\right) + (1-m)|\overline{c}^T\overline{x}| + g|u_f|\right] - m\eta|s|, \tag{18}$$

where $m \equiv gg_L^{-1} \geq 1$, implying $\dot{V} = s\dot{s} \leq -\eta|s|$.

Therefore, the hitting controller in (13) guarantees that $|s|$ decreases if $s \geq \overline{s}$. If the initial condition $|s(\underline{x}(0))| < \overline{s}$, then we always have $|s| < \overline{s}$. On the other hand, even if the initial condition $|s(\underline{x}(0))| \geq \overline{s}$, $|s|$ will still be forced into the boundary layer (i.e. $|s| < \overline{s}$) by the hitting controller in finite time.

2) With the hitting controller in (13), the representative point will be kept inside the pre-specified boundary layer, $|s| < \overline{s}$. From Fig. 1, if $Z_1(p)$ denotes the output of the first filter, then we have

$$Z_1(p) = \frac{1}{p + \lambda_1} S(p), \tag{19}$$

that is,

$$z_1(t) = \int_0^t e^{-\lambda_1(t-\tau)} s(\tau) d\tau. \tag{20}$$

Since $|s| < \overline{s}$, we have

$$|z_1(t)| \leq \overline{s} \int_0^t e^{-\lambda_1(t-\tau)} d\tau = (\overline{s}/\lambda_1)(1 - e^{-\lambda_1 t}) \leq (\overline{s}/\lambda_1) \equiv \zeta_1. \tag{21}$$

Similarly, we can apply the same procedure to the 2nd, 3rd, …, *i*th filters and directly get

$$z_i(t) \leq \zeta_{i-1}/\lambda_i = \overline{s} \Big/ \prod_{j=1}^{i} \lambda_j \equiv \zeta_i, \tag{22}$$

Since $|x_1(t)| = |z_{n-1}(t)|$, we have

$$|x_1(t)| \leq \overline{s} \Big/ \prod_{i=1}^{n-1} \lambda_j \equiv \delta_i. \tag{23}$$

From (1) and Fig. 1, we can write

$$X_2(p) = \left[\frac{1}{\prod_{i=1}^{n-2}(p + \lambda_i)}\right]\left(\frac{p}{p + \lambda_{n-1}}\right) S(p) = \left(\frac{p}{p + \lambda_{n-1}}\right) Z_{n-2}(p) = \left(1 - \frac{\lambda_{n-1}}{p + \lambda_{n-1}}\right) Z_{n-2}(p); \tag{24}$$

thus,

$$|x_2(t)| \leq \left(1 + \frac{\lambda_{n-1}}{\lambda_{n-1}}\right)\zeta_{n-1} = 2\zeta_{n-1}. \tag{25}$$

Therefore, by applying the similar procedure to all states, it is easy to obtain

$$|x_i(t)| \leq 2^{i-1}\zeta_{n-i} = \left(2^{i-1} \Big/ \prod_{j=1}^{n-i} \lambda_j\right)\overline{s} \equiv \delta_i. \tag{26}$$

■

According to the above theorem, the hitting controller can guarantee the stability of a self-learning fuzzy control system in the sense that all the states are bounded. In general, the hitting controller can be applied not only to the genetic learning fuzzy system, but also to almost all fuzzy control systems. For example, during the design phase, the fuzzy rule-base (5) can be constructed based on certain approaches [15-16][18]. If the fuzzy controller does not work well, the membership functions and/or the rules should be adjusted. In such a design/tuning stage, unsuitable adjustments may cause the system to be unstable. Thus, the hitting controller is applied to protect the system so that all the system states are inside the safe regions.

V. OPTIMIZATION

Define a quadratic cost function [22]:

$$J = \frac{1}{2}\int_0^\infty \left(\underline{x}^T(t)\, \underline{Q}\, \underline{x}(t) + u^T(t) R u(t)\right) dt, \tag{27}$$

where $Q \in \Re^{n \times n}$ and $R \in \Re$ are two positive definite weighting matrices. Given the system in (1) and the controller in (12), and based on the learning strategy described previously, the objective in this section is to extract a fuzzy rule-base so that (27) is minimized.

In learning step 6, the fitness function F is evaluated from J_s instead of J. Therefore, finding an alternative cost function J_s for J, so as to use GA to minimize J by maximizing the fitness function F, is the major aim of this section.

Separating the cost function (27) into two parts yields

$$J_1 = \frac{1}{2}\int_0^\infty \left[\underline{x}^T(t)\, \underline{Q}\, \underline{x}(t)\right] dt,$$

$$J_2 = \frac{1}{2}\int_0^\infty \left[u^T(t) R u(t)\right] dt. \tag{28}$$

To minimize J_2, define an alternative cost function $J_2 = \sum_{k=1}^{K} Ru^2$. On the other hand, to minimize J_1, derive the optimal sliding surface.

Consider again the sliding surface defined in (4). Without loss of generality, let $c_n = 1$, i.e.,

$$s(x) = x_n + \sum_{i=1}^{n-1} c_i x_i \equiv x_n + \overline{c}^T \overline{x} = 0, \tag{29}$$

where $\overline{c} = [c_1\, c_2 \ldots c_{n-1}]^T$ and $\overline{x} = [x_1\, x_2 \ldots x_{n-1}]^T$. Let there be a control u^* which drives the state to reach the sliding mode in finite time, *i.e.*, $s = 0$ and $\dot{s} = 0$ as $t \geq t_s$. Then, the original system (1) is linearlized by u^*, and the equivalent system is given as

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \vdots \\ \dot{x}_{n-1} \\ -- \\ \dot{x}_n \end{bmatrix} = \left[\begin{array}{ccccc|c} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \cdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & 1 & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 1 \\ \hline 0 & -c_1 & -c_2 & \cdots & -c_{n-2} & -c_{n-1} \end{array}\right] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_{n-1} \\ -- \\ x_n \end{bmatrix} \equiv Ax \cdot \tag{30}$$

Rewriting (30) yields

$$\underline{\dot{x}} \equiv \begin{bmatrix} \dot{\underline{\varepsilon}} \\ \dot{\xi} \end{bmatrix} = \begin{bmatrix} \underline{A}_{11} & \underline{A}_{12} \\ \underline{A}_{21} & \underline{A}_{22} \end{bmatrix} \begin{bmatrix} \underline{\varepsilon} \\ \xi \end{bmatrix}, \tag{31}$$

in which the state, $\underline{x}$, is partitioned into two parts, $\underline{\varepsilon} \in \Re^{(n-1)\times 1}$ and $\xi \in \Re$; the system matrix, A, is divided into four sub-matrices, $\underline{A}_{11} \in \Re^{(n-1)\times(n-1)}$, $\underline{A}_{12} \in \Re^{(n-1)\times 1}$, $\underline{A}_{21} \in \Re^{1\times(n-1)}$ and $\underline{A}_{22} \in \Re$. Obviously, $\xi = x_n$ and $\underline{\varepsilon} = \overline{x}$. Hence, from (29) we have

$$\xi + \overline{c}^T \underline{\varepsilon} = 0 . \tag{32}$$

Partitioning the weighting matrix of J_1, *i.e.*, Q, in the same way in which A was partitioned, we get

$$Q = \begin{bmatrix} \underline{Q}_{11} & \underline{Q}_{12} \\ \underline{Q}_{21} & \underline{Q}_{22} \end{bmatrix}. \tag{33}$$

At this time, the original optimization problem can be transformed into a standard LQ one. According to (28), (31), (32) and (33), we have a pseudo system for J_1:

$$\dot{\underline{\varepsilon}} = \underline{A}_0 \varepsilon + A_{12}\beta \tag{34}$$

and

$$J_1 = \frac{1}{2}\int_s^\infty \left(\underline{\varepsilon}^T \underline{Q}_0 \underline{\varepsilon} + \beta^T Q_{22} \beta \right) dt , \tag{35}$$

where

$$\begin{cases} \underline{A}_0 = \underline{A}_{11} - \underline{A}_{12}\underline{Q}_{22}^{-1}\underline{Q}_{21} \\ \underline{Q}_0 = \underline{Q}_{11} - \underline{Q}_{12}Q_{22}^{-1}\underline{Q}_{21} \\ \beta = Q_{22}^{-1}\underline{Q}_{21}\underline{\varepsilon} + \xi . \end{cases} \tag{36}$$

Obviously, (34) and (35) form a standard LQR system [22] with a pseudo state ε and a pseudo control β. Therefore, the optimal control of (34) for minimizing (35), β^*, can be solved using the famous LQR technique [22]

$$\beta^* = -\underline{\psi}^{*T} \underline{\varepsilon} \tag{37}$$

with the optimal gain

$$\underline{\psi}^* = \left(Q_{22}^{-1} \underline{A}_{12} \underline{P}_0 \right)^T, \tag{38}$$

where P_0 is the solution of the following Riccati equation:

$$\underline{A}_0^T \underline{P}_0 + \underline{P}_0 \underline{A}_0 - \underline{P}_0 \underline{A}_{12} Q_{22}^{-1} \underline{A}_{12}^T \underline{P}_0 + \underline{Q}_0 = 0. \tag{39}$$

Consequently, the coefficients of the optimal sliding surface are given by

$$\overline{c}^* = \left(\underline{\psi}^{*T} + Q_{22}^{-1} \underline{Q}_{21} \right)^T . \tag{40}$$

After deriving the optimal sliding surface, the primary mission of the fuzzy rules is to drive the state to hit the optimal sliding surface ($s^* = 0$) as quickly as possible and to then keep the state as close to $s^* = 0$ as possible. To achieve this goal, define an alternative cost function for J_1:

$$J_{1s} = t_s + \sum_{k=1}^{K} s^2 . \tag{41}$$

The complete alternative cost function for J is defined as

$$J_s = J_{1s} + J_{2s} = t_s + \sum_{k=1}^{K} s^2 + Ru^2 . \tag{42}$$

It is similar to the cost function defined in learning step 6, *i.e.*, equation (7). Therefore, the fitness function for the optimal self-learning fuzzy controller is directly defined as

$$F = 1/(J_s + \varepsilon_0) . \tag{43}$$

VI. SIMULATION RESULTS AND DISCUSSION

6.1 System description

Consider an underwater vehicle whose simplified model is represented as [13]

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -\dfrac{d}{m} x_2 |x_2| + \dfrac{1}{m} u , \end{cases} \tag{44}$$

where x_1, x_2 define the position and velocity, respectively; u is the control force; m is the mass of the vehicle; d denotes the drag coefficient. The parameter values used in [13] were also adopted in the following simulations, *i.e.*, $m = 3 + 1.5 \sin(|x_2|t)$ and $d = 1.2 + 0.2 \sin(|x_2|t)$.

In the following simulations, the weighting matrices were selected as $Q = \begin{bmatrix} 2 & 0.5 \\ 0.5 & 1 \end{bmatrix}$, R = 0.1; the population

size M = 10, the crossover rate = 0.8 and the mutation rate = 0.03.

To demonstrate the superiority of the proposed methodology over the traditional one, two examples are presented in this section. In the first example, the plant described earlier is controlled by a conventional FLC with a supervisory controller [19]. In the second example, the architecture of an FSMC with a hitting controller is adopted. In both cases, the learning mechanisms are realized by means of genetic algorithms. We call the former *GA-based Fuzzy Logic Control* (GA-FLC) and the latter GA-based *Fuzzy Sliding Mode Control* (GA-FSMC).

6.2 Simulation examples

Example 1. GA-based Fuzzy Logic Control

In general, each rule in the rule-base of a conventional FLC is slightly different from (5) and is always represented as

$$R_j : IF\ x_1\ is\ X_{1j}\ (m_{1j}, \sigma_{1j})\ \text{and}\ x_2\ is\ X_{2j}\ (m_{2j}, \sigma_{2j})$$
$$and\ \ldots\ x_n\ is\ X_{nj}\ (m_{nj}, \sigma_{nj})\ THEN\ u_f\ \text{is}\ U_j(\varphi_j), \qquad (45)$$

where j = 1, 2, …, N_l and N_l is the number of rules; X_{ij} and U_j are fuzzy sets called input linguistic labels and output linguistic labels, respectively.

Similarly, the control output of FLC depends on the parameters of its membership functions, *i.e.*, m_{ij}, σ_{ij} and φ_{ij}. Hence, the genetic algorithms are applied to adjust the parameter set $\{m_{ij}, \sigma_{ij}, \varphi_{ij} | i = 1, 2, \ldots, n, j = 1, 2, \ldots, N_l\}$ of (45). The fitness function for GA-FLC can be directly defined as $F = 1/(J + \varepsilon_0)$. To guarantee stability in the learning phase, the supervisory control proposed by Wang [19] is used.

The universe of discourse of each state is partitioned into 6 subspaces in this example. Therefore, there are 6 corresponding linguistic labels for each input variable. Since the system has two states, the fuzzy rule-base has 36 rules, i.e., N_l = 36.

The dot curves in Fig. 1. and Fig. 2 show the state response and the evaluation result of the cost function of GA-FLC, respectively.

Example 2. GA-based Fuzzy Sliding Mode Control

Based on the control methodology and the learning procedure proposed in Sections 3, 4 and 5 of this paper, we designed a GA-FSMC to control the system (44).

Similar to the previous example, 6 linguistic labels are defined for input variables. Since the FSMC merely has one input s, the fuzzy rule-base only needs 6 rules, that is, N = 6.

The curves in Fig. 1. and Fig. 2 show the state response and the evaluation result of the cost function of GA-FSMC, respectively.

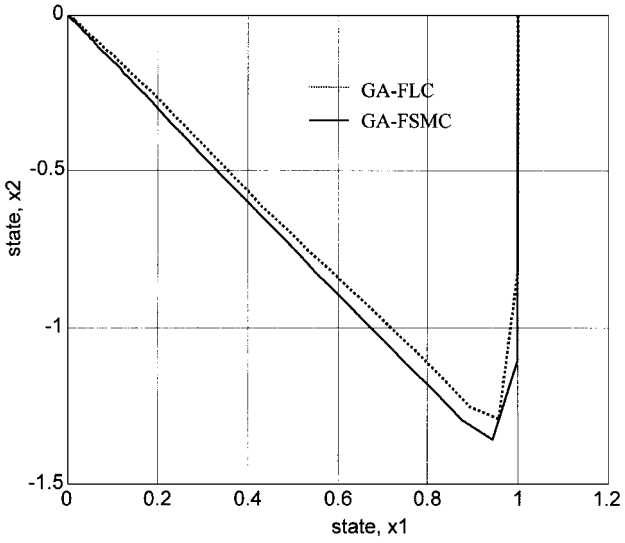

Fig. 1. State space response of GA-FLC and GA-FSMC.

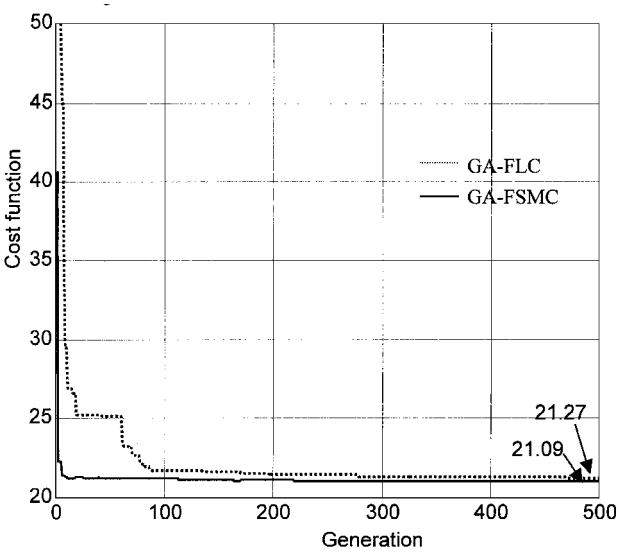

Fig. 2. Cost function values of GA-FLC and GA-FSMC.

6.3 Comparisons and discussions

One of the most important advantages of FSMC is that the size of its rule-base is significantly smaller than that of FLC. As a result, the length of the chromosome and the size of the search space of GA-FSMC can be efficiently reduced, as we have pointed out in [16]. There fore, we expect that the convergent rate of GA-FSMC may be faster than that of GA-FLC. The simulation result shown in Fig. 2 agrees with our expectation. Figure 1 shows the final state space responses of GA-FLC and GA-FSMC, respectively. They demonstrate that although the architectures in both cases are very different, their state trajectories, are very close. Comparison results between these two strategies are summarized in Table 1. We can see the superiority of the proposed approach (GA-FSMC) over the conventional one (GA-FLC).

Table 1. Comparison results between GA-FSMC and GA-FLC.

Comparison terms	GA-FSMC	GA-FLC
Number of linguistic labels associated with each input variable	6	6
Number of input variables	1	2
Total number of rules	6	36
Number of parameters to be learned in each rule	3	3
Total number of parameters to be learned	18	108
Encoded bit length of each parameter	8	8
Total length of chromosome	146	864
Number of floating point operations in each generation (running in MATLAB)	116780	525243
Value of cost function J at generation 50	21. 24 (within 0.7% of final value)	25. 60 (within 25.6% of final value)
Final value of cost function J at generation 500	21.09	21.27

VII. CONCLUSIONS

In this paper, we have developed a stable self-learning method for optimal fuzzy control system design. In the proposed approach, the fuzzy sliding mode control technology has been introduced to: 1) reduce the size of the fuzzy rule-base; 2) shorten the chromosome length of GA; 3) specify the response of the control system by defining a sliding surface; 4) transform the optimization problem into a standard LQR problem by deriving an optimal sliding surface. The hitting controller has been used to stabilize the learning system. Simulation results have confirmed the superiority of the proposed methodology.

REFERENCES

1. Zadeh, L.A., "Fuzzy Sets," *Inf. Contr.*, Vol. 8, pp. 338-353 (1965).
2. Zadeh, L.A., "Outline of a New Approach to the Analysis of Complex Systems and Decision Processes," *IEEE Trans. Syst. Man Cybern.*, Vol. SMC-3, pp. 28-44 (1973).
3. Lee, C.C., "Fuzzy Logic in Control Systems: Fuzzy Logic Controller, Parts I and II," *IEEE Trans. Syst. Man Cybern.*, Vol. SMC-20, pp. 404-435 (1990).
4. Mamdani, E.H., "Application of Fuzzy Algorithms for Control of Simple Dynamic Plant," *Proc. IEE.*, Vol. 121, No. 13, pp. 1585-1588 (1974).
5. Sugeno, M. and M. Nishida, "Fuzzy Control of Model Car," *Fuzzy Sets Syst.*, Vol. 16, pp. 103-113 (1985).
6. Tanscheit, R. and E.M. Scharf, "Experiments with the Use of a Rule-based Self-organizing Controller for Robotics Applications," *Fuzzy Sets Syst.*, Vol. 26, pp. 195-214 (1988).
7. Goldberg, D.E., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, New York (1989).
8. Karr, C.L., "Applying Genetics to Fuzzy logic," *AI Expert*, pp. 38-43 (1991).
9. Karr, C.L., "Genetic Algorithms for Fuzzy Controller," *AI Expert*, pp. 26-33 (1991).
10. Kristinsson K. and G.A. Dumont, "System Identification and Control Using Genetic Algorithms," *IEEE Trans. Syst. Man Cybern.*, Vol. SMC-22, pp. 1033-145 (1992).
11. Itkis, U., *Control System of Variable Structure*, Wiley Press, New York (1976).
12. Utkin, V.I., *Sliding Modes and Their Application in Variable Structure System*, Mir Press, Moscow, English translation (1978).
13. Slotine, J.J.E. and W. Li, *Applied Nonlinear Control*, Prentice Hall, Englewood-Cliffs, New Jersey (1991).
14. Lin, S.C. and C.C. Kung, "A Linguistic Fuzzy-sliding Mode Controller," *Proc. Amer. Contr. Conf.*, pp. 1904-(1992).
15. Lin, S.C. and Y.Y. Chen, "Design of Adaptive Fuzzy Sliding Mode for Nonlinear System Control," *3rd IEEE Int. Conf. Fuzzy Syst.*, Orlando, pp. 35-39 (1994).
16. Lin, S.C. and Y.Y. Chen, "Design of Slef-learning Fuzzy Sliding Mode Controller Based on Genetic Algorithms," *Fuzzy Sets Syst.*, Vol. 86, No. 2 (1997).
17. Lin, S.C. and Y.Y. Chen, "Design a Hitting Controller to Stabilize the Fuzzy Sliding Mode Control Systems," *Proc. Int. Joint Conf. CFSA/IFIS/SOFT Fuzzy Theory Appl.*, Taipei, Taiwan, pp. 374-379 (1995).
18. Lu, Y.S. and J.S. Chen, "A Self-organizing Fuzzy Sliding-mode Controller Design for a Class of Nonlinear Servo Systems, *IEEE Trans. Ind. Electron.*, Vol. 41, pp. 492-502 (1994).
19. Wang, L.X., "A Supervisory Controller for Fuzzy Control Systems that Guarantees Stability," *IEEE Trans. Automat.Contr.*, Vol. 39, No. 10, pp. 1845-1847 (1994).

20. Wang, L.X. *Adaptive Fuzzy Systems and Control* Printice-Hall, Englewood-Cliffs, New Jersey (1994).
21. Hwang, G.C. and S.C. Lin, "A Stability Approach to Fuzzy Control Design for Nonlinear Systems," *Fuzzy Sets Syst.*, Vol. 48, pp. 279-287 (1992).
22. Lewis, F.L., *Applied Optimal Control and Estimation*, Prentice Hall, Englewood Cliffs, New Jersey (1992).

Sinn-Cheng Lin received the B.S. degree in electronic engineering from Tamkang University, Taiwan, R.O.C., in 1990 and the M.S. degree in electrical engineering from Tatung Institute of Technology, Taiwan, R.O.C., in 1992 and the Ph.D. degree in electrical engineering from National Taiwan University, Taiwan, R.O.C., in 1997.

In 1997-1998, he taught at Lunghwa Institute of Technology. Since August 1998, he has been with the Department of Educational Media and Library Science, Tamkang University, Taipei, Taiwan, R.O.C. His current research interests are intelligent systems, soft computing, fuzzy systems, self-learning algorithms, artificial intelligence and multimedia applications.

Yung-Yaw Chen received the B.S. degree in electrical engineering from National Taiwan University in 1981, and the Ph.D. degree in electrical engineering and computer science from University of California, Berkeley, in 1989.

He was with the Artificial Intelligence branch in NASA-Ames in 1989. He joined National Taiwan University as an associate professor in 1989, and is currently a professor in the Department of Electrical Engineering. From 1993 to 1994, he was a visiting scholar at the University of California, Berkeley. His research interests include fuzzy control, precision motion control, intelligent systems, and ultrasound hyperthermia.