An Intelligent Fuzzy Sliding Mode Control System with Application on Precision Table Positioning

Sinn-Cheng Lin,[1,*] Pai-Yi Huang,[2] Yung-Yaw Chen[2]
[1] *Department of Information and Library Science, Tamkang University, Taipei, Taiwan, Republic of China*
[2] *Department of Electrical Engineering, Taiwan University, Taipei, Taiwan, Republic of China*

In this paper, an intelligent fuzzy sliding mode control system, which cooperates with a new learning approach called modulus genetic algorithm, is proposed. Furthermore, it is applied to a high precision table positioning system for verifying its practicability. Fuzzy sliding mode controller (FSMC) is a special type of fuzzy controller with certain attractive advantages than the conventional fuzzy controller. The learning and stability issues of FSMC are discussed in the paper. Furthermore, to overcome the encoding/decoding procedure that leads to considerable numeric errors in conventional genetic algorithm, this paper proposes a new algorithm called modulus genetic algorithm (MGA). The MGA uses the modulus operation such that the encoding/decoding procedure is not necessary. It has the following advantages: (1) the evolution can be speeded up; (2) the numeric truncation error can be avoided; (3) the precision of solution can be increased. For verifying the practicability of the proposed approach, the MGA-based FSMC is applied to design a position controller for a high precision table. The experimental results show the proposed approach can achieve submicro positioning precision. © 2001 John Wiley & Sons, Inc.

I. INTRODUCTION

As we have found, without knowing the plant model, applying model-based strategies to design a controller is a hard job. However, by organizing human expertise into fuzzy IF-THEN rules, a fuzzy logic controller (FLC)[1-3] with linguistic rules, which works like human experts do, can be designed to control complex or ill-defined systems. The principle of FLC is based on fuzzy set theory[4] and approximation reasoning.[5] A lot of successful applications have been published to verify the capability of FLCs.[6-9] At present, the fuzzy logic

* Author to whom correspondence should be addressed; e-mail: sclin@mail.tku.edu.tw.

control has become a well-known powerful scheme for controlling modeless systems.

In this paper, a self-learning fuzzy sliding mode control system based on modulus genetic algorithm (called MGA-based FSMC) is proposed. Furthermore, the MGA-based FSMC is applied to a high precision table positioning system for verifying its practicability. In the proposed system, we use a fuzzy sliding mode controller instead of a conventional fuzzy logic controller, and in addition, we adopt the modulus genetic algorithm as a learning mechanism instead of the traditional genetic algorithm. One may ask: Why fuzzy sliding mode control? Why modulus genetic algorithm?

A. Why Fuzzy Sliding Mode Control?

The fuzzy sliding mode controller (FSMC) is a special type of fuzzy controller. In recent years, variant types of fuzzy sliding mode controller were proposed and studied.[10–15] Compared with a conventional FLC, as we have mentioned in our previous works, there are several attractive advantages,[8] such as: (1) The response of a fuzzy sliding mode control system can be specified in advance; (2) the fuzzy rules are simpler and the entire rule base is more compact, the speed of fuzzy inference of the FSMC therefore is faster than that of a conventional FLC; (3) by using genetic algorithm as a learning approach, the chromosome length of the FSMC is shorter than that of a FLC, consequently, the genetic evolution of the FSMC becomes more efficient.

B. Why Modulus Genetic Algorithm?

Knowledge acquisition is the most important task in the fuzzy controller design. However, extracting a set of rules, which can be sufficiently represent the skill human actions, is not an easy mission.[2,16] Recently, the self-learning approaches have become popular.[1,3,11,17–19] In particular, many researchers have focused on the topic of using a genetic algorithm (GA) to extract a fuzzy rule base.[20–22] Genetic algorithms were originally developed by Holland in 1962. The detailed principles, mathematical frameworks, and applications can be found in Goldberg's book.[23] The use of GAs for solving control problems was presented in Ref. 24. The conventional simple GA (SGA) encodes the searched parameters as binary strings. After applying the basic genetic operators such as reproduction, crossover, and mutation, a decoding procedure has to be used to convert the binary strings to the original parameter space. As a result, such an encoding/decoding procedure leads to considerable numeric errors. Hence, this paper proposes a new algorithm called modulus genetic algorithm (MGA) that uses the modulus operation to resolve this problem. In the MGA, the encoding/decoding procedure is not necessary. It has the following advantages: (1) the evolution can be speeded up; (2) the numeric truncation error can be avoided; (3) the precision of solution can be increased.

For verifying the practicability of this proposed approach, the MGA-based FSMC is applied to design a position controller for a high precision table. In most manufacturing systems, the precision of the positioning system will affect the quality of the products. Therefore, good technology that can increase the precision of the system would improve the manufacturing capability and the product value. For example, in an IC fabrication system, if the positioning accuracy has been increased, then the quality and density of the circuits that were implemented on the IC chips could also be increased. Presently, the positioning requirement in the level of "submicron" is always needed in the manufacturing systems of semiconductors, photoelectronic elements, and high-density magnetic memory devices. However, increasing the precision of positioning is not an easy task. It needs not only to satisfy the high manufacturing standard for the machine, but also needs the advanced control and sensing technologies. To apply the traditional model-based control strategies for controller design, a high precision plant model has to be derived. However, such a high precision model is not easy to obtain. Therefore, the fuzzy control strategy, which does not need an explicit mathematical model, is more suitable than the model-based approaches. In this paper, a precision table is used as a testing plant for demonstrating the proposed self-learning control approach.

This paper is organized as follows: after the Introduction in Section I, Section II describes the fundamentals of the FSMC. In Section III, the MGA is proposed and used to build an intelligent fuzzy sliding mode control system; optimal fitness function and system stability will also be considered. Section IV applies the MGA-based FSMC to a high precision plant. Conclusions are drawn in Section V.

II. FUNDAMENTALS OF FSMC

An FSMC, which is based on sliding mode control (SMC)[25,26] and fuzzy logic, has better properties than a conventional FLC. Consider a class of nonlinear systems with the following error dynamics:[3]

$$e^{(n)} = f(e, \dot{e}, \ldots, e^{(n-1)}) + g(e, \dot{e}, \ldots, e^{(n-1)})u \tag{1}$$

where $e \in \Re$ is the system error; $u \in \Re$ is the control input; $f(\cdot)$ is an unknown continuous function with known upper bound, i.e. $|f| \le f^U$; $g(\cdot)$ is an unknown positive definite function with known lower bound, i.e., $0 < g_L \le g$. Actually, Eq. (1) represents a general uncertain nonlinear dynamic system.

Define $x_i = e^{(i-1)}$, $i = 1, 2, \ldots, n$, then (1) can be represented by the following state representation:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= x_3 \\ &\vdots \\ \dot{x}_n &= f(\underline{x}) + g(\underline{x})u \end{aligned} \tag{2}$$

where $\underline{x} = [x_1 \;\; x_2 \;\; \cdots \;\; x_n]^T \in \Re^n$ is the state vector.

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

$$s(x) = \underline{c}^{\mathrm{T}}\underline{x} = \sum_{i=1}^{n} c_i x_i \tag{3}$$

where $\underline{c}^{\mathrm{T}} = [c_1 \ c_2 \ \cdots \ c_n]^{\mathrm{T}} \in \Re^n$ is a coefficient vector of sliding surface that has to be properly determined. Appendix A proposed an approach to obtain an optimal $\underline{c}$ based on the LQ technology.

In traditional SMC, a sliding surface that is denoted as Σ is a crisp set of states, on which (3) equals to zero, i.e.,

$$\Sigma = \{\underline{x} \mid s(\underline{x}) = 0\} \tag{4}$$

Keeping the state on Σ with an equivalent control law u_{eq} is the basic idea of classical SMC. Without loss of generality, let $c_n = 1$. By taking the derivative of (3), we have

$$\dot{s} = \underline{c}_r^{\mathrm{T}}\underline{x}_r + f + gu \tag{5}$$

where $\underline{c}_r = [c_1 \ c_2 \ \cdots \ c_{n-1}]^{\mathrm{T}}, \underline{x}_r = [x_2 \ x_3 \ \cdots \ x_n]^{\mathrm{T}}$. Then u_{eq} could be easily derived by setting (5) to zero. That is

$$u_{\mathrm{eq}} = u|_{\dot{s}=0} = -g^{-1}\left(f + \underline{c}_r^{\mathrm{T}}\underline{x}_r\right) \tag{6}$$

With the equivalent control obtained above, the state can be kept on Σ, the system is said to be in sliding mode, and its dynamics can be described by

$$c_1 e + c_2 \dot{e} + \cdots + e^{(n-1)} = 0 \tag{7}$$

Therefore, the characteristic polynomial of the equivalent control system is given by

$$p^{n-1} + c_{n-1}p^{n-2} + \cdots + c_1 = 0 \tag{8}$$

where p denotes the Laplace operator. With a suitable choice of coefficients c_is a stabile control system can be obtained if and only if (8) is Herwitz. As a result, the dynamic behavior of a sliding mode control system is determined by the predefined sliding surface.

The second stage in SMC design is to derive a discontinuous control law to drive the state to reach the sliding surface whenever $\underline{x} \notin \Sigma$.[25,26] that is,

$$u_d = \begin{cases} u^+ > 0 & \text{for } s < 0 \\ u^0 = 0 & \text{for } s = 0 \\ u^- < 0 & \text{for } s > 0 \end{cases} \tag{9}$$

Finally, the complete control law in a sliding mode control system is

$$u = u_{\mathrm{eq}} + u_d \tag{10}$$

However, there is no way to get an exact u_{eq} without knowing f and g. In the proposed FSMC, an alternative control law is used:

$$u = (1 - \alpha)u_f + \alpha u_h \tag{11}$$

where u_f is a fuzzy control law, which can be obtained from the following fuzzy sliding mode control rule base, and is used to drive the state toward the surface and then slide along it.

$$R_j\text{: IF } s \text{ is } S_j(m_j, \sigma_j) \text{ THEN } u_f \text{ is } U_j(\varphi_j) \tag{12}$$

in which $j = 1, 2, \dots, N$, and N is the number of rules; S_js are the input linguistic labels, and are simply assigned as Gaussian-shaped functions in this paper; i.e., $\mu_{s_j}(s) = \exp(-((s - m_j)/\sigma_j)^2)$; U_js are the output linguistic labels, especially, they are assigned to be fuzzy singleton in this paper, i.e., $\mu_{U_j}(u) = \begin{cases} 1, & u = \varphi_j \\ 0, & u \neq \varphi_j \end{cases}$. On the other hand, u_h is a hitting control to guarantee the stability of system, and α is a switch factor with the value of $\alpha = \begin{cases} 1, & \text{for } |s| \geq s_0 \\ 0 & \text{for } |s| < s_0 \end{cases}$. They will be discussed in the next section.

Suppose that the singleton fuzzification and the weighted average defuzzification methods are applied, then the output of (12) is given by

$$u_f(s) = \underline{\varphi}^{\mathrm{T}}\underline{\rho}(s) \tag{13}$$

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

Figure 1 shows a two-dimensional case of state plane to illustrate the concept of SMC and FSMC. The sliding surface in Figure 1(a) is a straight line that passes through the equivalent point (0, 0). The state plane has been divided into two parts by the sliding line. One is $s > 0$, and the other is $s < 0$. The sliding line is also called the switching line, because the control action switched at the opposite sides of the line. That is why the sliding mode control is also

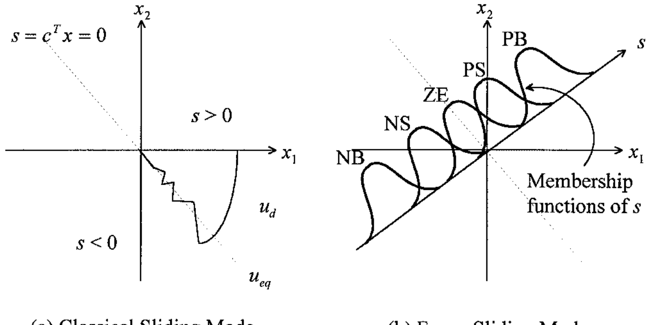

Figure 1. The basic concept of fuzzy sliding mode control.

known as the variable structure control (VSC). However, such a switching operation has many drawbacks. One of them is the chattering phenomenon due to the presentations of the system nonideality, such as hysteresis, delay, sampling, uncertainty, and so on. To reduce the system nonideality effects, the fuzzification operation was applied to convert the crisp sliding surface into a fuzzy one. The crisp value of s can be viewed as a generalized distance from a representative point to the sliding surface. Once the s value is fuzzified, a set of fuzzy rules based on the fuzzified distance can be constructed. This is the main idea of the FSMC. Figure 1(b) illustrates such a concept. Here, five linguistic labels (PB, PS, ZE, NS, and NB) are assigned to the sliding variable s.

III. LEARNING FSMC BY MGA

The fuzzy sliding mode control rule base (12) can be determined by certain strategies such as heuristic,[11] adaptive,[12] or self-organizing/self-learning[13,14] methods. In this section, a modulus genetic algorithm is proposed for the purpose of self-learning fuzzy sliding mode control rules. The block diagram of the proposed system is shown in Figure 2.

A. The Modulus Genetic Algorithm

Conventionally, a simple genetic algorithm (SGA) is also called the binary genetic algorithm since it works with a set of binary strings, e.g., 10110110. It is one of the disadvantages of SGA. First, SGA encodes the searched parameters to binary strings. Next, SGA applies the basic genetic operators such as reproduction, crossover, and mutation to generation offspring. Then, a decoding procedure is used to convert the binary strings to the original parameter space. As a result, such an encoding/decoding procedure leads to considerable numeric errors. The MGA, which works on parameters themselves instead of their binary codes, is described in this section to resolve the problem. In the MGA,

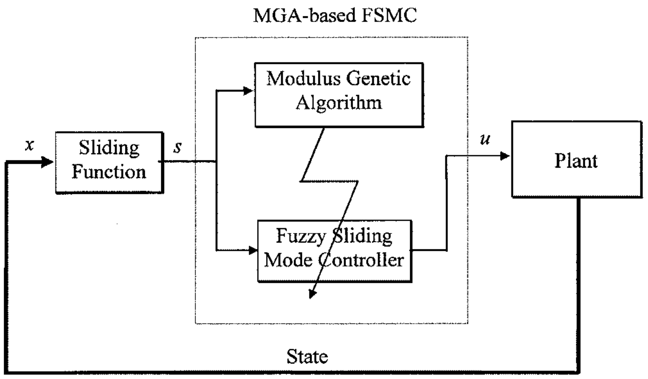

Figure 2. The block diagram of MGA-based FSMC.

the encoding/decoding procedure is not necessary. It has the following advantages: (1) the evolution can be speeded up; (2) the numeric truncation error can be avoided; (3) the precision of solution can be increased.

(1) Reproduction of MGA

The Darwinian "survival of the fittest" is the underlying spirit of reproduction. This operation, actually, is an artificial version of natural selection.

Let F be the fitness function, and F_i denote the value of fitness function associated with the individual string i in the current population. Reproduction is a process in which individual strings in the current population are copied according to their fitness function values F_i. A higher F value indicates a better fit (or larger benefit). To perform reproduction, first, F_is are calculated. Next, the current individual strings are probabilistically selected and copied into a mating pool according to their fitness value. The arrangement allows the strings with a higher fitness to have a greater probability of contributing a larger amount of offspring in the new population. The easiest way of implementing a reproduction operator is to create a biased roulette wheel. The slot size of it is in proportion to the fitness value of each individual in the current population. Let ps_i denote the probability of selection of the individual i, and M be the population size, then an individual string will get selected with the following probability:

$$ps_i = F_i \Bigg/ \sum_{j=1}^{M} F_j \tag{14}$$

(2) Crossover of MGA

Crossover provides a mechanism for individual strings to exchange information via a probabilistic process. Once the reproduction operator is applied, the members in the mating pool are allowed to mate with one another. The binary-coded GA takes the following step to accomplish the crossover: First, two parents are randomly selected from the mating pool. Then, a random crossover point is picked up. Finally, an exchange of the parents' genetic codes (binary digits) following the crossover point. This random process provides a highly efficient method to search the string space to find a better solution.

In MGA, the parameters lie in the original space rather than binary space. Hence, the crossover operation has to be modified to work with parameters themselves rather than their binary codes.

Let $\{a, b\}$ and $\{a', b'\}$ be the parent and offspring parameter pair, respectively. Without loss of generality, suppose that their search space is in the range of $[0, X] \subseteq R$. The crossover of MGA is proposed as

$$\begin{aligned} a' &= (a - a_0 + b_0)\mathrm{MOD}\, X \\ b' &= (b - b_0 + a_0)\mathrm{MOD}\, X \end{aligned} \tag{15}$$

where MOD means the modulus operator. It is the reason why the proposed approach is called "modulus" genetic algorithm. The meanings of the other notations in (15) are

$$a_0 = a \,\mathrm{MOD}\, aX$$
$$b_0 = b \,\mathrm{MOD}\, aX$$

in which $\alpha \in [0, 1]$ is called the crossover factor.

The crossover of a binary-coded GA is a special case of (15) with the following a_0 and b_0:

$$a_0 = a \,\mathrm{MOD}\, 2^c$$
$$b_0 = b \,\mathrm{MOD}\, 2^c$$

where c denotes the bit number of crossover point.

(3) Mutation of MGA

In the genetic algorithm, the mutation operation introduces new genes into the populations such that the problem of trapping in local optimal points may be avoided. The gene of individual is subject to a random change with probability of the preassigned mutation rate. In the binary-coded case, a mutation operator changes a bit from 0 to 1 or vice versa. In MGA, mutation is a random work mechanism. It simply replaces a parameter, say a, with an arbitrary value, say a', in the search space $[0, X]$.

B. Learning Procedure

Recall the rule base of FSMC described by (12); the output of the rule base is uniquely determined by a set of parameters that is unionized by the parameters of the IF part, $\underline{m}$, $\underline{\sigma}$, and the THEN part, φ. Hence, the parameter vector to be learned by MGA, $\underline{\theta}$, can be defined as

$$\underline{\theta} = \left[\underline{m}^{\mathrm{T}} \;\; \underline{\sigma}^{\mathrm{T}} \;\; \underline{\varphi}^{\mathrm{T}}\right]^{\mathrm{T}} = [m_1 \;\; m_2 \;\; \cdots \;\; m_N \;\; \sigma_1 \;\; \sigma_2 \;\; \cdots \;\; \sigma_N \;\; \varphi_1 \;\; \varphi_2 \;\; \cdots \;\; \varphi_N]^{\mathrm{T}} \quad (16)$$

Assume that X_m, X_σ, X_φ are the search space of m_js, σ_js, φ_js, respectively; M is the population size; h is the number of generations. Then the details of learning procedures of MGA-based FSMC are described in the following.

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

$$\underline{m}^{(i)}(h) = \left[m_1^{(i)}(h) m_2^{(i)}(h) \cdots m_N^{(i)}(h)\right]^{\mathrm{T}}$$
$$\underline{\sigma}^{(i)}(h) = \left[\sigma_1^{(i)}(h) \sigma_2^{(i)}(h) \cdots \sigma_N^{(i)}(h)\right]^{\mathrm{T}}$$
$$\underline{\varphi}^{(i)}(h) = \left[\varphi_1^{(i)}(h) \varphi_2^{(i)}(h) \cdots \varphi_N^{(i)}(h)\right]^{\mathrm{T}}$$

where $m_j^{(i)}(h) \in X_m$, $\sigma_j^{(i)}(h) \in X_\sigma$, and $\varphi_j^{(i)}(h) \in X_\varphi$ $(i = 1, 2, \ldots, M, j = 1, 2, \ldots, N)$.

If the ith candidate of MGA-based FSMC is denoted by FSMC$^{(i)}$. Then the fuzzy rule base of FSMC$^{(i)}$ can be created as

$$\text{FSMC}^{(i)}: \begin{cases} R_1^{(i)}: \text{IF } s \text{ is } S_1^{(i)}\left(m_1^{(i)}, \sigma_1^{(i)}\right) \text{ THEN } u_f \text{ is } U_1^{(i)}\left(\varphi_1^{(i)}\right) \\ R_2^{(i)}: \text{IF } s \text{ is } S_2^{(i)}\left(m_2^{(i)}, \sigma_2^{(i)}\right) \text{ THEN } u_f \text{ is } U_2^{(i)}\left(\varphi_2^{(i)}\right) \\ \vdots \\ R_N^{(i)}: \text{IF } s \text{ is } S_N^{(i)}\left(m_N^{(i)}, \sigma_N^{(i)}\right) \text{ THEN } u_f \text{ is } U_N^{(i)}\left(\varphi_N^{(i)}\right) \end{cases}$$

where $S_j^{(i)}$s and $U_j^{(i)}$s are linguistic labels to be learned, and $m_j^{(i)}$s, $\sigma_j^{(i)}$s, and $\varphi_j^{(i)}$s are their parameters.

Step 2. Construct the parameter vector of the ith individual,

$$\begin{aligned} \underline{\theta}^{(i)}(h) &= \left[\theta_1^{(i)}(h) \cdots \theta_N^{(i)}(h) \vdots \theta_{N+1}^{(i)}(h) \cdots \theta_{2N}^{(i)}(h) \vdots \theta_{2N+1}^{(i)}(h) \cdots \theta_{3N}^{(i)}(h)\right]^{\mathrm{T}} \\ &= \left[\underline{m}^{(i)\mathrm{T}}(h) \vdots \underline{\sigma}^{(i)\mathrm{T}}(h) \vdots \underline{\varphi}^{(i)\mathrm{T}}(h)\right]^{\mathrm{T}} \end{aligned}$$

Step 3. Establish the population in the generation $h, \mathbf{P}(h)$,

$$\mathbf{P}(h) = \left\{\theta^{(1)}(h), \theta^{(2)}(h), \ldots, \theta^{(n)}(h)\right\}$$

Step 4. Applied fuzzy reasoning to get u_f, apply it to the plant. Evaluate the fitness value of each individual.

The fitness function F can be defined in many ways. The following is one way, with intuition: a controller which drives the state to reach the sliding surface as fast as possible without consuming too much control energy and then keep the state onto the surface as close as possible gets a higher score. For example,

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

$$F = \frac{1}{(J_s + \varepsilon_0)}$$

Here 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$(\cdot)$ is the round-off operator, $K = \text{int}(t_{\max}/\Delta t)$ denotes the number of iterations during one run, $t_{\max}$ is the running time during one run, and w and v are positive weights. The norm $\|\cdot\|$ 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 of dividing by zero. Appendix A shows a suboptimal method that determines F by LQ technology.

Step 5. Apply the modulus genetic operators, i.e., reproduction of MGA, crossover of MGA, and mutation of MGA, to generate a new population $\mathbf{P}(h+1)$, which is the offspring of $\mathbf{P}(h)$.

Step 6. Keep the elitist. That is, (1) pick up the best fitted individuals in $\mathbf{P}(h)$ and $\mathbf{P}(h+1)$; (2) compare their fitness; (3) if the best individual of $\mathbf{P}(h)$ has a better fitness value than that of $\mathbf{P}(h+1)$, then randomly replace an individual in $\mathbf{P}(h+1)$ with the elitist.

Step 7. Use the parameters to calculate the output u_f of the FSMC.

Step 8. Set $h = h + 1$; go to Step 2 and repeat the procedure until $F \geq F_M$ or $h \geq H$. F_M and H denote an acceptable fitness value and a stop generation number, respectively, as specified by the designer.

C. System Stability

During the learning phase described previously, the unfit individuals (the controllers which have unsuitable parameters) may cause unstable behavior. Hence, we propose a stabilizer called a hitting controller, such that

$$|x_i| \leq \delta_i, \qquad i = 1, 2, \ldots, n \tag{17}$$

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

The hitting control law u_h works in the following manner: If the state of the control system is inside a prespecified boundary layer s_0, i.e., $|s| < s_0$, then the hitting controller is turned off. In such a situation, the self-learning fuzzy control u_f provides the control action only. On the other hand, if the state tends to diverge, i.e., $|s| \geq s_0$, then the learning controller is turned off and the hitting controller is turned on to pull the state back. In this situation, the hitting control u_h provides the control action only. Therefore, the hitting controller acts as a gatekeeper that works in a fashion similar to Wang's supervisory controller.[27]

Achieving the goal described above requires the hitting control law to satisfy the so-called sliding condition[28] when $|s| \geq s_0$, that is,

$$s\dot{s} \leq -\eta |s| \tag{18}$$

where η is a positive constant.

The following theorem provides the methodology for designing the hitting controller so that (18) is satisfied, and that all states are bounded by (17).

THEOREM. *Consider the system* (2) *with the control law*

$$u = (1 - \alpha) u_f + \alpha u_h \tag{19}$$

where u_f denotes the fuzzy control law given by the self-learning fuzzy control rule base (12)*; u_h is the hitting control law and α is a switching factor that is used to determine whether the fuzzy controller or the hitting controller is to be activated.*

If the hitting control law is given by

$$u_h = -\operatorname{sign}(s)\left[g_L^{-1}\left(f^U + |\underline{c}_r^{\mathrm{T}}\underline{x}_{\mathrm{r}}| + \eta\right)\right] \tag{20}$$

and the switching factor is defined as

$$\alpha = \begin{cases} 1 & \text{for } |s| \geq s_0 \\ 0 & \text{for } |s| < s_0 \end{cases} \tag{21}$$

where $\underline{c}_r = [c_1\ c_2\ \cdots\ c_{n-1}]^{\mathrm{T}}$ and $\underline{x}_r = [x_2\ x_3\ \cdots\ x_n]^{\mathrm{T}}$, then

(1) The sliding condition (18) is satisfied when $|s| \geq s_0$.
(2) The control system is stable in the sense that all system states x_i $(i = 1, 2, \ldots, n)$ are bounded by

$$|x_i(t)| \leq \left(2^{i-1}\Big/\prod_{j=1}^{n-i}\lambda_j\right)s_0 := \delta_i \tag{22}$$

Proof. See Appendix B.

Figure 3 shows a two-dimensional case of the hitting control theorem in which $s = \lambda x_1 + x_2$. Hence, from (22) we have $|x_1| \leq s_0/\lambda$ and $|x_2| \leq 2s_0$.

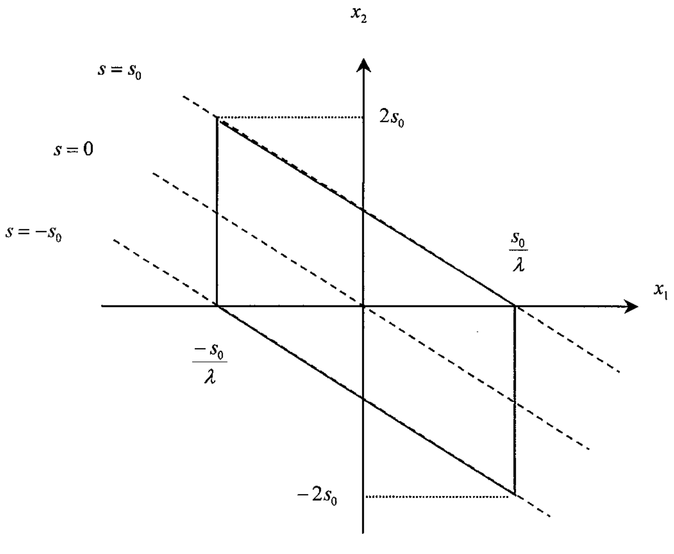

Figure 3. A two-dimensional case of hitting control theorem.

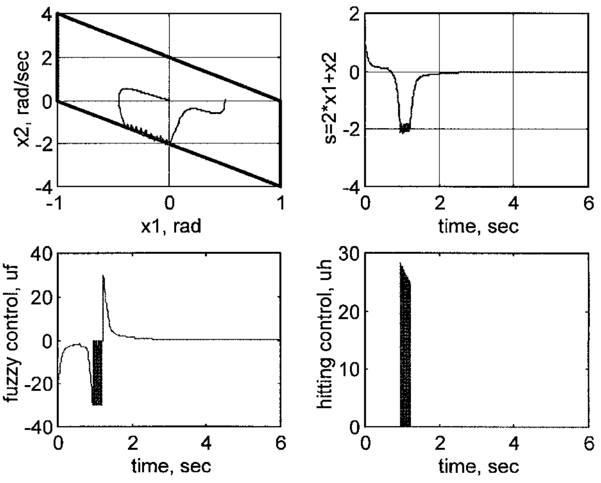

Figure 4. Demonstration of the action of hitting control.

Figure 4 is a computer simulation case of the action of hitting control. If the fuzzy control is "poor" then the state goes to diverge. As soon as the state hits the boundary of s_0, the fuzzy control turns off, and the hitting controller is activated to pull the state back. While the state is back inside s_0, the "poor" fuzzy control turns on again, and pushes the state to hit the boundary again, then the hitting control turns on,.... Such a push–pull process goes progress and keeps the state "sliding" on the boundary until the fuzzy control "well-done" by learning.

According to the above theorem, the hitting controller can guarantee the stability of system in the sense that all states are bounded. In general, the hitting controller can be applied not only to the proposed MGA-based FSMC system, but also to almost all fuzzy sliding mode control systems. For example, during the design phase, the fuzzy rule-base (12) may be constructed based on certain learning/tuning approaches. If the fuzzy control does not work well, the membership functions and/or the rules should be adjusted. In such a design/tuning stage, the unsuitable adjustments may cause the system to be unstable. Thus, the hitting controller is applied to protect the system so that all system states are inside the safe regions.

IV. EXPERIMENTAL RESULTS

The high precision table employed in the experiment is an *XY* table, as shown in Figure 5. In our experiment, only one axis is used to demonstrate the proposed control scheme. This table is made by NSK Inc. The specifications of the tables are listed in Table I. The table is driven by a dc motor with a low noise linear amplifier. There is an encoder attached to the motor. The rotation angle of the motor can be obtained by reading the encoder. Since the table is a

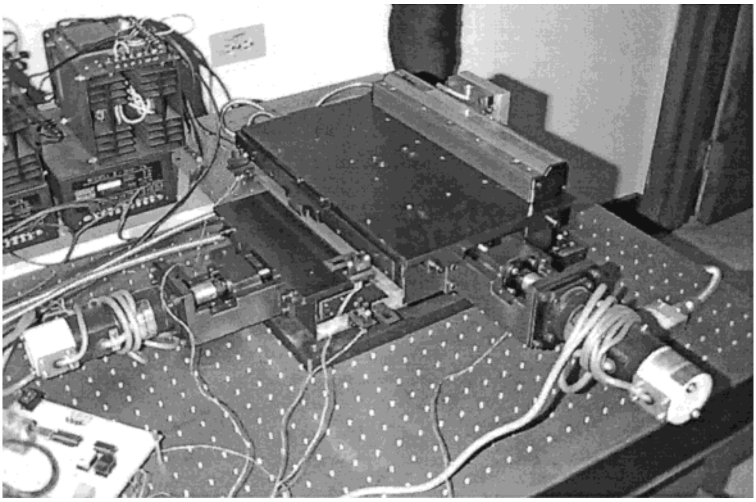

Figure 5. The experimental environment of the high precision control system.

ball-screw type, the position of the table is in proportion to the counter. Whereas the required precision is low, one can obtain the position of the table by the encoder directly. However, as the required precision becomes higher and higher, the error between encoder and exact table position becomes considerable. Therefore, a laser scale, which is a direct measurement device, is mounted on the table for the sake of providing the actual position of the table. The laser scale, also shown in Figure 5, is manufactured by FUTABA—its resolution is 100 nm. With the help of such a measurement device, the resolution of the table has been promoted to a "submicron" level.

The main difficulty of such a precision positioning control problem is that the mathematical model cannot easily be derived accurately. For example, the friction that is caused by the sliding motion of the table will dominate the system characteristics. The contacts between the steel balls and screw lead and between the moving plate and its guides are two major sources of friction. Whereas the required precision becomes higher and higher, the moving speed of the controlled plate becomes slower and slower. Consequently, the friction force is larger and larger. The friction is not only highly nonlinear and uncertain but also hard to model. There are other different types of unmodeled factors that

Table I. Specifications of the experimental precision table.

Stroke	200 mm $\times$ 200 mm
Pitch	5 mm
Repeatability	0.003 mm
Backlash	0.001 mm
Resolution of the laser scale	100 nm

would dominate in the tiny motion. It is the reason why we utilize the proposed GA-based FSMC as a control strategy.

Before the MGA learning stage, the hitting control, which is considered as a safety guard to monitor the system states, should be designed properly. Recall the hitting control law (20); two design parameters f^U and g_L are required. In the following, an experimental methodology is developed to estimate these two values.

Define the state vector $\underline{x} = [e, \dot{e}]^{\mathrm{T}}$, where e and $\dot{e}$ represent the error and the error derivative (velocity) of the plant. Hence, the hitting control becomes

$$u_h = -\operatorname{sign}(s)\left\lfloor g_L^{-1}\left(f^U + |\lambda \cdot \dot{e}| + \eta\right)\right\rfloor$$

The main ideal of the proposed estimation method is shown in Figure 6. In this motor-driven table, by experimenting, we find that when the input u keeps constant, the moving velocity of the table almost remains constant. In other words, the moving velocity will not change abruptly unless the control input changes. The dashed-line represents the sliding function of the hitting controller. The upper part $|s| > s_0$ is a forbidden zone that any system state should not enter. The design of the hitting controller is to choose a sliding function with different slopes and s_0. A sliding function with a larger slope would require a powerful motor driver, so it should be considered with system capabilities.

The estimation procedure is the following. First, if the table moves with constant output voltage u_1, it will move at constant velocity **V1**. As time progresses, it will reach the cross point **P**. The hitting control, if designed properly, should generate a suitable force and pull the state trajectory back to the $|s| < s_0$ area. If the pushing force is not strong enough, like trajectories **P ~ P1** and **P ~ P2**, then it should be increased until the trajectory is pulled back to $|s| < s_0$.

Hence, if we know the smallest force of u_h, say $u_{h,\min}$, which can scarcely bring the state trajectory back to $|s| < s_0$, then we can solve two unknown

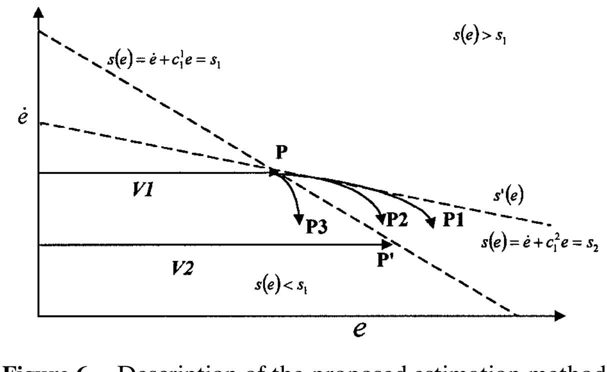
$s(e) > s_1$
$s(e) = \dot{e} + c_1^1 e = s_1$
$\dot{e}$
P
V1
$s'(e)$
P3
P2
P1
$s(e) = \dot{e} + c_1^2 e = s_2$
P'
V2
$s(e) < s_1$
e

Figure 6. Description of the proposed estimation method.

variables f^U and g_L by the following two equations:

$$u_{h,\min}^{(1)} = -\operatorname{sign}(s)\left[g_L^{-1}\left(f^U + |\lambda^{(1)} \cdot \dot{e}^{(1)}| + \eta\right)\right]$$
$$u_{h,\min}^{(2)} = -\operatorname{sign}(s)\left[g_L^{-1}\left(f^U + |\lambda^{(2)} \cdot \dot{e}^{(2)}| + \eta\right)\right]$$

$\lambda^{(1)}$ and $\lambda^{(2)}$ are the slopes of two distinct sliding functions $s^{(1)}$ and $s^{(2)}$, respectively, while $u_{h,\min}^{(1)}$ and $u_{h,\min}^{(2)}$ are the smallest forces to bring the state trajectory back to $|s^{(1)}| < s_0^{(1)}$ and $|s^{(2)}| < s_0^{(2)}$, respectively. But, without knowing $u_{u,\min}^{(1)}$, the values of f^U and g_L can be estimated by finding the local values in various conditions by experimental testing. That is,

$$u_h^{(i)} = -\operatorname{sign}(s)\left[g_i^{-1}\left(f_i + |\lambda^{(i)} \cdot \dot{e}| + \eta\right)\right]$$

where $i = 1, 2, \ldots, N$, and N is the number of the testing points. Then the upper bound of f, i.e., f^U, and the lower bound of g, i.e., g_L, can be obtained by $f^U = \max(f_i)$ and $g_L = \min(g_i)$. In our experimental testing, a fixed initial velocity with many different voltage changes has been made. The data are collected and by arbitrarily choosing two sets of them, then values f_i and g_i can be calculated. If we repeat the above procedure with other initial velocities, the other set of f_is and g_is can be evaluated. Finally, the calculated f_is and g_is are drawn in Figure 7. The x axis in Figure 7 is the value of f_i, the y axis is g_i. Each point represents an estimation for f_i and g_i. Three groups with different notations ($\circ$, $+$, and $*$) indicate three different initial velocities of 39.4 mm/sec,

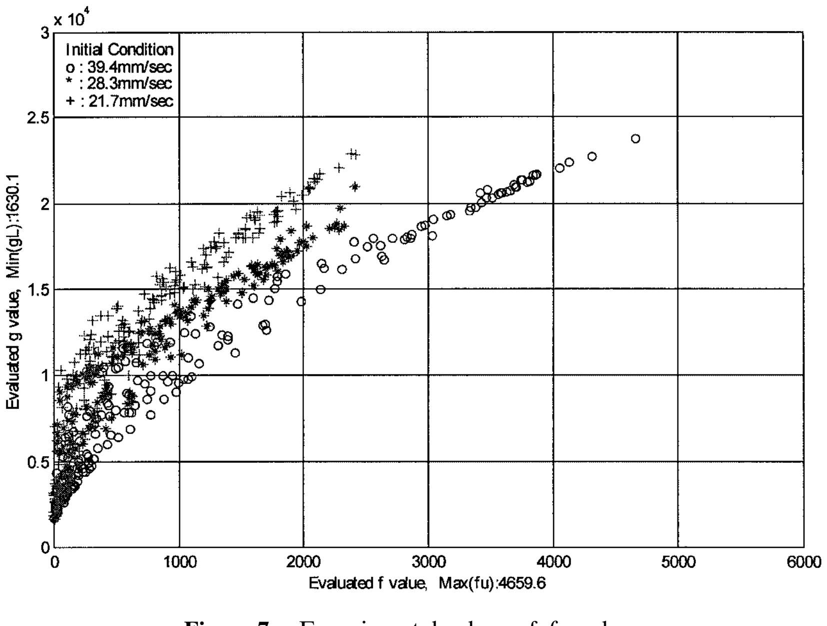

Figure 7. Experimental values of f_i and g_i.

28.3 mm/sec, and 21.7 mm/sec, respectively. Finally, the estimation values of f^U and g_L are given as

$$f^U = \max(f_i) \cong 4659.6$$

and

$$g_L = \min(g_i) \cong 1630.1$$

After this design stage, the MGA learning scheme can be applied in the guarded bound. Once the table goes to touch the bound, the hitting control will be responsible for taking the state to a safe place.

A compatible IBM PC is used to implement the GA-based self-learning fuzzy controller. An interface card is equipped to receive the position feedback and a D/A converter is used to send a control command to the dc servo. The control program is implemented as an interrupt that has been triggered by a timer. In every sample period, the interrupt subroutine will read the position from the laser scale, and evaluate the control command. The experimental results are shown in Figures 8 to 11. The fitness function, Figure 8, indicates that the GA-based fuzzy controller performs better and better from generation to generation.

In Figure 9, the position error seems to be reduced to zero. To view the effect in slow motion, Figure 10 plots the error from 0.15 to 2 sec. We can see the steady state error is -7.5×10^{-4} mm, or equivalently, -0.75 μm. Figure 11 is the state trajectory.

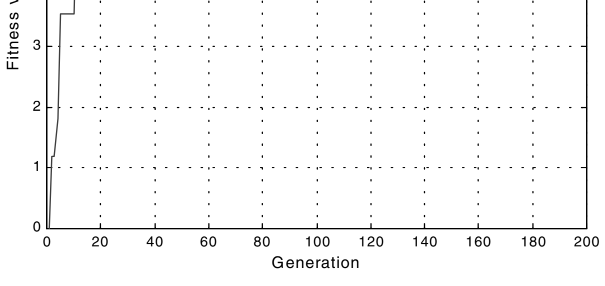

Figure 8. The fitness function.

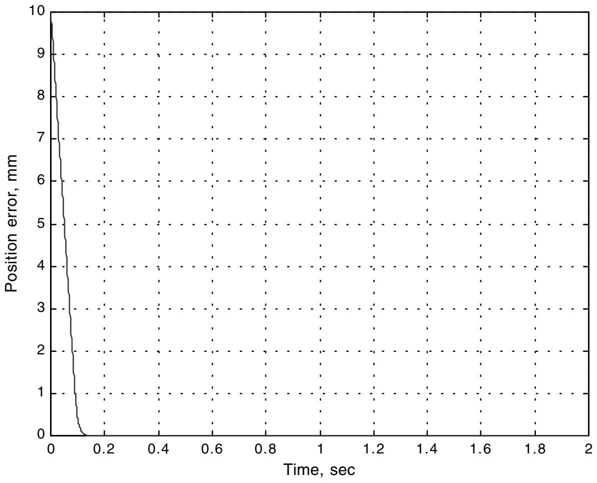

Figure 9. The position error.

V. CONCLUSIONS

In this paper, a new approach called modulus genetic algorithm is described. The numeric error, which arises by the encoding/decoding procedure in conventional GAs, was avoided. In the modulus genetic algorithm, the encoding/decoding procedure is not necessary. It has the following advantages: (1) The evolution can be speeded up; (2) the numeric truncation error can be avoided; (3) the precision of solution can be increased. The MGA was applied to

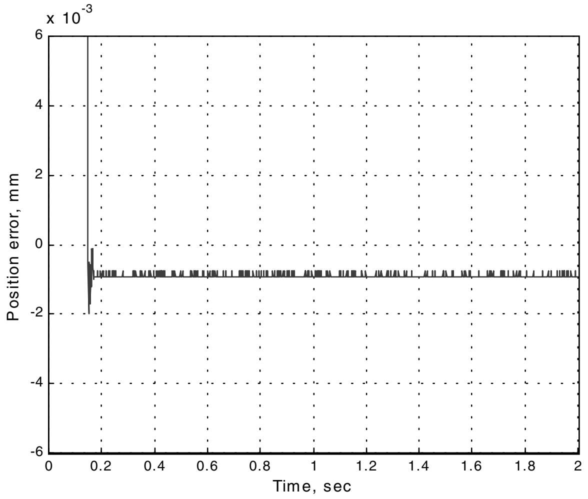

Figure 10. The position error from 0.15 to 2 sec.

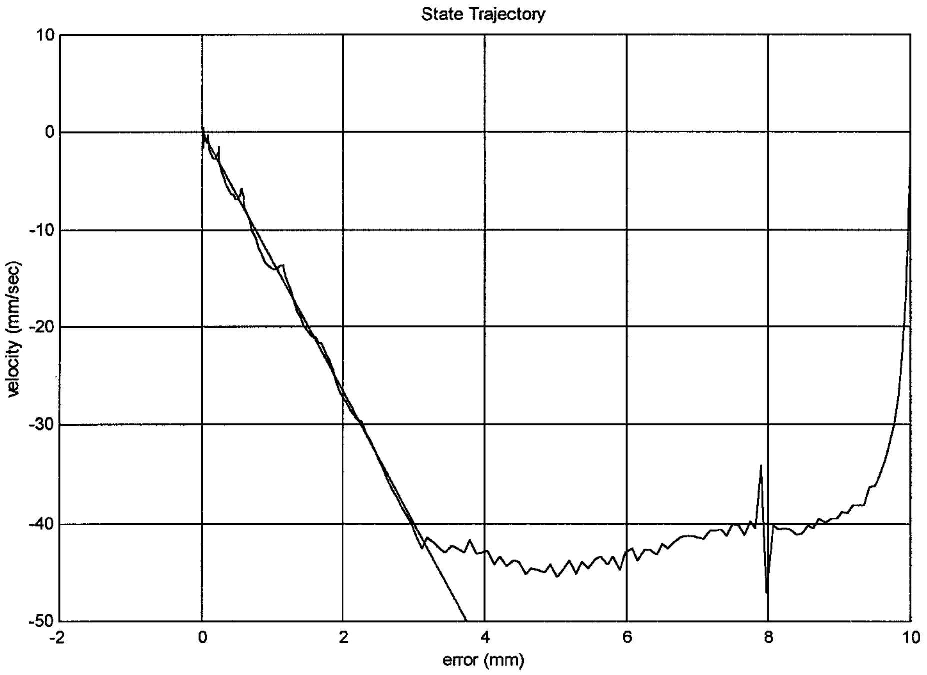

Figure 11. The state trajectory.

resolve the learning problem for fuzzy sliding mode control systems. Furthermore, with a properly designed hitting controller, the stability of the learning system can be guaranteed.

Finally, the MGA-based FSMC was applied to a high precision positioning system. An *XY* table with a high-resolution laser scale (0.1 μm) was used as a demonstrated plant. The experimental results indicated the practicability of a proposed self-learning control strategy.

References

1. Lee CC. A self-learning rule-based controller employing approximation reasoning and neural net concepts. Int J Intel Syst 1991;6:71–93.
2. Lee CC. Fuzzy logic in control systems: Fuzzy logic controller, part I. IEEE Trans Systems Man Cybernet 1990;20(2):404–418.
3. Wang LX. Adaptive fuzzy systems and control. Englewood Cliffs, NJ: Prentice Hall; 1994.
4. Zadeh LA. Fuzzy sets. Inform Control 1965;8:338–353.
5. Zadeh LA. Outline of a new approach to the analysis of complex systems and decision processes. IEEE Trans Systems Man Cybernet 1973;3:28–44.
6. Mamdani EH. Application of fuzzy algorithms for control of simple dynamic plant. Proc IEE 1974;121(13):1585–1588.
7. Sugeno M, Nishida M. Fuzzy control of model car. Fuzzy Sets and Systems 1985;16:103–113.

8. Takagi T, Sugeno M. Fuzzy identification of systems and its applications to modeling and control. IEEE Trans Systems and Cybernet 1985;15(1):116–132.
9. Tanscheit R, Scharf EM. Experiments with the use of a rule based self-organizing controller for robotics applications. Fuzzy Sets and Systems 1988;26:195–214.
10. Hwang GC, Lin SC. A stability approach to fuzzy control design for nonlinear systems. Fuzzy Sets and Systems 1992;48:279–287.
11. Lin SC, Kung CC. A linguistic fuzzy-sliding mode controller. In: Proc A.C.C. 1992, p 1904–1905.
12. Lin SC, Chen YY. Design of adaptive fuzzy sliding mode for nonlinear system control. Proc IEEE Int Conf Fuzzy Systems, Orlando 1994, p 35–39.
13. Lin SC, Chen YY. Design of self-learning fuzzy sliding mode controller based on genetic algorithms. Fuzzy Sets and Systems 1997;86:139–153.
14. Lu YS, Chen JS. A self-organizing fuzzy sliding-mode controller design for a class of nonlinear servo systems. IEEE Trans Industrial Electron 1994;41:492–502.
15. Palm R. Sliding mode fuzzy control. In: Proc IEEE Int Conf Fuzzy Systems, San Diego, 1992, p 709–715.
16. Lee CC. Fuzzy logic in control systems: Fuzzy logic controller, part II. IEEE Trans Systems Man Cybernet 1990;20(2):419–435.
17. Berenji HR, Khedkar P. Learning and tuning fuzzy logic controllers through reinforcements. IEEE Trans Neural Networks 1992;3(5):724–740.
18. Jang JSR. Self-learning fuzzy controllers based on temporal backpropagation. IEEE Trans Neural Networks 1992;3(5):714–723.
19. Lin CT, Lee CSG. Reinforcement structure/parameter learning for neural-network-based fuzzy logic control systems. IEEE Trans Fuzzy Systems 1994;2(1):46–63.
20. Karr CL. Applying genetics to fuzzy logic. AI Expert 1991;Mar:38–43.
21. Karr CL. Genetic algorithms for fuzzy controller. AI Expert 1991;Feb:26–33.
22. Lin SC, Chen YY. Application of self-learning methodologies to the fuzzy inference systems. J Chinese Fuzzy Systems Association 1995;1:7–26.
23. Goldberg DE. Genetic algorithms in search, optimization, and machine learning. Reading, MA: Addison-Wesley; 1989.
24. Kristinsson K, Dumont GA. System identification and control using genetic algorithms. IEEE Trans Systems Man Cybernet 1992;22:1033–145.
25. Itkis U. System of variable structure. New York: Wiley Press; 1976.
26. Utkin VI. Sliding modes and their application in variable structure systems. Moscow: Mir Press; English translation; 1978.
27. Wang LX. A supervisory controller for fuzzy control systems that guarantees stability. IEEE Trans Automatic Control 1994;39(10):1845–1847.
28. Slotine JJE, Li W. Applied nonlinear control. Englewood Cliffs, NJ: Prentice-Hall; 1991.
29. Lewis FL. Applied optimal control and estimation. Englewood Cliffs, NJ: Prentice Hall; 1992.
30. Wang LX, Mendel JM. Generating fuzzy rules by learning from examples. IEEE Trans Systems Man Cybernet 1992;22(6):1414–1427.

APPENDIX A: AN OPTIMAL APPROACH TO DETERMINE THE FITNESS FUNCTION

Given the system of (2) and the controller of (12), we define a quadratic cost function:[29]

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

where $Q \in \Re^{n\times n}$, $R \in \Re$ are two positive definite weighting matrices. The objective in this section is to determine a suitable fitness function F such that the genetic-learning algorithm can extract a fuzzy rule-base to minimize J.

Separating the cost function (A.1) into two parts yields

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

$$J_2 = \frac{1}{2}\int_0^\infty \left[u^{\mathrm{T}}(t)Ru(t)\right] dt \tag{A.3}$$

To minimize J_2, we define an alternative cost function:

$$J_{2s} = \sum_{k=1}^{K} Ru^2 \tag{A.4}$$

On the other hand, to minimize J_1, an optimal sliding surface is derived.

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

$$s(\underline{x}) = x_n + \sum_{i=1}^{n-1} c_i x_i := x_n + \underline{c}_r^{\mathrm{T}}\underline{\varepsilon} = 0 \tag{A.5}$$

where $\underline{c}_r = [c_1 \ c_2 \ \cdots \ c_{n-1}]^{\mathrm{T}}$ and $\underline{\varepsilon} = [x_1 \ x_2 \ \cdots \ x_{n-1}]^{\mathrm{T}}$. Assume there is a control u^* which can drive the system to reach the sliding mode in a finite time t_s, i.e., $s = 0$ and $\dot{s} = 0$ as $t \geq t_s$. Then, the original system (2) can be linearized by u^*, and the equivalent system is given as

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \vdots \\ \dot{x}_{n-1} \\ \hline \dot{x}_n \end{bmatrix} = \left[\begin{array}{ccccc|c} 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \cdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & 1 & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 1 \\ \hline 0 & -c_1 & -c_2 & \cdots & -c_{n-2} & -c_{n-1} \end{array}\right] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_{n-1} \\ \hline x_n \end{bmatrix} := \underline{A}\underline{x} \tag{A.6}$$

Defining $\xi = x_n$, and rewriting (A.6), yields

$$\dot{\underline{x}} := \begin{bmatrix} \dot{\underline{\varepsilon}} \\ \dot{\xi} \end{bmatrix} = \begin{bmatrix} \underline{A}_{11} & \underline{A}_{12} \\ \underline{A}_{21} & A_{22} \end{bmatrix} \begin{bmatrix} \underline{\varepsilon} \\ \xi \end{bmatrix} \tag{A.7}$$

in which the state vector $\underline{x}$ is partitioned into two parts; $\underline{\varepsilon} \in \Re^{(n-1)\times 1}$, $\xi \in \Re$; the system matrix $\underline{A}$ is divided into four submatrices; $\underline{A}_{11} \in \Re^{(n-1)\times(n-1)}$, $\underline{A}_{12}$

$\in \Re^{(n-1)\times 1}$, $\underline{A}_{21} \in \Re^{1\times(n-1)}$, and $A_{22} \in \Re$. Hence, from (A.5) we have

$$\xi + \bar{c}^{\mathrm{T}}\underline{\varepsilon} = 0 \tag{A.8}$$

Partitioning the weighting matrix of J_1, i.e., $\underline{Q}$, in the same way as partitioning $\underline{A}$, we get

$$\underline{Q} = \begin{bmatrix} \underline{Q}_{11} & \underline{Q}_{12} \\ \underline{Q}_{21} & \underline{Q}_{22} \end{bmatrix} \tag{A.9}$$

where $\underline{Q}_{11} \in \Re^{(n-1)\times(n-1)}$, $\underline{Q}_{12} \in \Re^{(n-1)\times 1}$, $\underline{Q}_{21} \in \Re^{1\times(n-1)}$, and $Q_{22} \in \Re$. Since $\underline{Q}$ is symmetric, so $\underline{Q}_{12} = \underline{Q}_{21}^{\mathrm{T}}$.

Now, the original optimization problem is ready to be transferred to a standard LQ one. Replace $\underline{x}$ in (A.2) with $[\underline{\varepsilon}^{\mathrm{T}} \;\; \xi]^{\mathrm{T}}$, and substitute (A.9) into (A.2) and we have

$$\begin{aligned} \underline{x}^{\mathrm{T}}\underline{Q}\underline{x} &= \begin{bmatrix} \underline{\varepsilon}^{\mathrm{T}} & \xi \end{bmatrix} \begin{bmatrix} \underline{Q}_{11} & \underline{Q}_{12} \\ \underline{Q}_{21} & Q_{22} \end{bmatrix} \begin{bmatrix} \underline{\varepsilon} \\ \xi \end{bmatrix} \\ &= \underline{\varepsilon}^{T}\underline{Q}_{11}\underline{\varepsilon} + \underline{\varepsilon}^{\mathrm{T}}\underline{Q}_{12}\,\xi + \xi\underline{Q}_{21}\underline{\varepsilon} + \xi Q_{22}\,\xi \\ &= \underline{\varepsilon}^{\mathrm{T}}\left(\underline{Q}_{11} - \underline{Q}_{12}\underline{Q}_{22}^{-1}\underline{Q}_{21}\right)\underline{\varepsilon} \\ &\quad + \left(\underline{Q}_{22}^{-1}\underline{Q}_{21}\underline{\varepsilon} + \xi\right)^{\mathrm{T}} Q_{22}\left(\underline{Q}_{22}^{-1}\underline{Q}_{21}\underline{\varepsilon} + \xi\right) \end{aligned} \tag{A.10}$$

From (A.7), we have

$$\begin{aligned} \dot{\underline{\varepsilon}} &= \underline{A}_{11}\underline{\varepsilon} + \underline{A}_{12}\,\xi \\ &= \left(\underline{A}_{11} - \underline{A}_{12}Q_{22}^{-1}\underline{Q}_{21}\right)\underline{\varepsilon} + \underline{A}_{12}\left(Q_{22}^{-1}\underline{Q}_{21}\underline{\varepsilon} + \xi\right) \end{aligned} \tag{A.11}$$

Define

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

Substitute $\underline{A}_0, \beta$ into (A.11) and we get a pseudo linear system:

$$\dot{\underline{\varepsilon}} = \underline{A}_0\underline{\varepsilon} + A_{12}\,\beta \tag{A.13}$$

Again, substitute $\underline{Q}_0, \beta$ into (A.10), and rewrite the cost function J_1 as a standard quadratic form

$$J_1 = \frac{1}{2}\int_s^{\infty}\left(\underline{\varepsilon}^{\mathrm{T}}\underline{Q}_0\underline{\varepsilon} + \beta^{\mathrm{T}}Q_{22}\,\beta\right)dt \tag{A.14}$$

Obviously, Eqs. (A.13) and (A.14) represent a standard LQ system[29] with pseudo state $\underline{\varepsilon}$ and pseudo control β. From Ref. 28, if $(\underline{A}_0, \underline{A}_{12})$ is stabilizable, then the pseudo optimal control law of (A.13) for minimizing (A.14), β^*, can be solved by the famous LQ technique. That is,

$$\beta^* = -\underline{\psi}^{*\mathrm{T}}\underline{\varepsilon} \tag{A.15}$$

where the optimal gain $\underline{\psi}^*$ is given by

$$\underline{\psi}^* = \left(Q_{22}^{-1}\underline{A}_{12}^{\mathrm{T}}\underline{P}_0\right)^{\mathrm{T}} \tag{A.16}$$

and P_0 is the solution of the following algebraic Riccati equation (ARE):

$$\underline{A}_0^{\mathrm{T}}\underline{P}_0 + \underline{P}_0\underline{A}_0 - \underline{P}_0\underline{A}_{12}Q_{22}^{-1}\underline{A}_{12}^{\mathrm{T}}\underline{P}_0 + \underline{Q}_0 = 0 \tag{A.17}$$

If $(\underline{A}_0, \underline{A}_{12})$ is not stabilizable, the designer has to select a new weighting matrix $\underline{Q}$, and solve the ARE again. Compare the definition of β in (A.12) with the optimal value of β obtained by (A.15) and we have

$$\xi = -\left(\underline{\psi}^{*\mathrm{T}} + Q_{22}^{-1}\underline{Q}_{21}\right)\underline{\varepsilon} = -\underline{Q}_{22}^{-1}\left(\underline{A}_{12}^{\mathrm{T}}\underline{P}_0 + \underline{Q}_{21}\right)\underline{\varepsilon} \tag{A.18}$$

Recall (A.8), the optimal sliding surface can be written as $s^* = \xi + \bar{c}^{*\mathrm{T}}\underline{\varepsilon} = 0$. By definition, the coefficient vector of the optimal sliding surface $\underline{c}^*$ can be easily obtained from (A.18):

$$\underline{c}^* := \left[\bar{c}^{*\mathrm{T}} \;\; c_n\right]^{\mathrm{T}} = \left[\left(Q_{22}^{-1}\left(\underline{A}_{12}^{\mathrm{T}}\underline{P}_0 + \underline{Q}_{21}\right)\right)^{\mathrm{T}} \;\; 1\right]^{\mathrm{T}} \tag{A.19}$$

After deriving the optimal sliding surface, the mission left is to extract a fuzzy rule base that can drive the state to achieve the optimal sliding mode as good as possible. To accomplish such a goal, define an alternative cost function for J_1:

$$J_{1s} = t_s + \sum_{k=1}^{K} s^2 \tag{A.20}$$

Then, the complete alternative cost function for J is defined as

$$J_s = J_{1s} + J_{2s} = t_s + \sum_{k=1}^{K} s^2 + Ru^2 \tag{A.21}$$

Finally, the fitness function for a GA-based optimal self-learning fuzzy sliding mode controller is directly defined as

$$F = \frac{1}{(J_s + \varepsilon_0)} \tag{A.22}$$

APPENDIX B: PROOF OF THEOREM

Let p denote the Laplace operator. If $X_1(p)$ and $S(p)$ represent the Laplace transform of x_1 and s, respectively, then we have the following filter equation:

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

Undoubtedly, the value of c_is can be carefully assigned so that the poles of the transfer function $T(p)$ are all negative real, i.e.,

$$T(p) = 1 \Big/ \prod_{i=1}^{n-1} (p + \lambda_i) \tag{B.2}$$

where $\lambda_i \in \Re^+$ $i = 1, 2, \ldots, n-1$. Without loss of generality, let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{n-1}$. From (B.1) and (B.2), s and x_1 can be viewed, respectively, 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 Figure B.1.

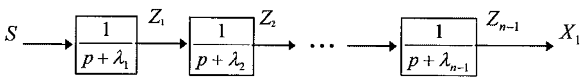

Figure B.1. The relationship of S and X_1.

(1) Consider the case of $|s| \geq s_0$ (i.e., $\alpha = 1$). Define a Lyapunov function

$$V = \tfrac{1}{2}s^2 \tag{B.3}$$

then, from (3) and (5), we have

$$\dot{V} = s\left[f + \underline{c}_r^{\mathrm{T}}\underline{x}_r\right] + sgu_h \tag{B.4}$$

That is,

$$\dot{V} \leq |s|\left[|f| + |\underline{c}_r^{\mathrm{T}}\underline{x}_r|\right] + sgu_h \tag{B.5}$$

Substitute (2.1) into (B.5) and define $m = gg_L^{-1} \geq 1$; we have

$$\dot{V} \geq |s|\left[\left(|f| - mf^U\right) + (1-m)|\underline{c}_r^{\mathrm{T}}\underline{x}_r|\right] - m\eta|s| \tag{B.6}$$

$m \geq 1$ implies $\dot{V} \leq -\eta|s|$, or equivalently, $s\dot{s} \leq -\eta|s|$. Therefore, the sliding condition (18) is satisfied.

The hitting control law (20) guarantees $|s|$ to be decreased whenever $s \geq s_0$. If the initial condition $|s(\underline{x}(0))| < s_0$, we always have $|s| < s_0$. On the other hand, even if the initial condition $|s(\underline{x}(0))| \geq s_0$, $|s|$ will be forced into the boundary layer (i.e., $|s| < s_0$) by the hitting controller in the finite time.

(2) With the hitting control law (20), the representative point will be kept inside the prespecified boundary layer, $|s| < s_0$. From Figure B.1, let $Z_1(p), Z_2(p), \ldots, Z_{n-1}(p)$ denote the output of the 1st, 2nd, …, $(n-1)$th filter, respectively. Then we have

$$Z_1(p) = \frac{1}{p + \lambda_1} S(p) \tag{B.7}$$

that is,

$$z_1(t) = \int_0^t e^{-\lambda_1(t-\tau)} s(\tau)\, d\tau \tag{B.8}$$

Since $|s| < s_0$, we have

$$|z_1(t)| \leq s_0 \int_0^t e^{-\lambda_1(t-\tau)}\, d\tau = (s_0/\lambda_1)(1 - e^{-\lambda_1 t}) \leq s_0/\lambda_1 := \zeta_1 \tag{B.9}$$

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

$$z_i(t) \leq \zeta_{i-1}/\lambda_i = s_0 \Big/ \prod_{j=1}^{i} \lambda_j := \zeta_i \tag{B.10}$$

From Figure B.1, we have $|x_1(t)| = |z_{n-1}(t)|$ and

$$|x_1(t)| \leq s_0 \Big/ \prod_{j=1}^{n-1} \lambda_j := \delta_1 \tag{B.11}$$

From (3.2) and Figure 5, we can write

$$\begin{aligned} X_2(p) &= \left[\frac{1}{\prod_{i=1}^{n-2}(p + \lambda_i)} \right] \left(\frac{p}{p + \lambda_{n-1}} \right) S(p) \\ &= \left(\frac{p}{p + \lambda_{n-1}} \right) Z_{n-2}(p) = \left(1 - \frac{\lambda_{n-1}}{p + \lambda_{n-1}} \right) Z_{n-2}(p) \end{aligned} \tag{B.12}$$

Thus,

$$|x_2(t)| \leq \left(1 + \frac{\lambda_{n-1}}{\lambda_{n-1}} \right) \zeta_{n-2} = 2\zeta_{n-2} \tag{B.13}$$

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

$$|x_i(t)| \leq 2^{i-1} \zeta_{n-i} = \left(2^{i-1} \Big/ \prod_{j=1}^{n-i} \lambda_j \right) s_0 := \delta_i \tag{B.14}$$