ELSEVIER

Fuzzy Sets and Systems 86 (1997) 139–153

FUZZY sets and systems

Design of self-learning fuzzy sliding mode controllers based on genetic algorithms

Sinn-Cheng Lin*, Yung-Yaw Chen

Lab. 202, Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan, ROC

Received March 1995; revised November 1995

Abstract

In this paper, genetic algorithms were applied to search a sub-optimal fuzzy rule-base for a fuzzy sliding mode controller. Two types of fuzzy sliding mode controllers based on genetic algorithms were proposed. The fitness functions were defined so that the controllers which can drive and keep the state on the user-defined sliding surface would be assigned a higher fitness value. The sliding surface plays a very important role in the design of a fuzzy sliding mode controller. It can dominate the dynamic behaviors of the control system as well as reduce the size of the fuzzy rule-base. In conventional fuzzy logic control, an increase in either input variables or the associated linguistic labels would lead to the exponential growth of the number of rules. The number of parameters or the equivalent length of strings used in the computations of genetic algorithms for a fuzzy logic controller are usually quite extensive. As a result, the considerable computation load prevents the use of genetic operations in the tuning of membership functions in a fuzzy rule-base. This paper shows that the number of rules in a fuzzy sliding mode controller is a linear function of the number of input variables. The computation load of the inference engine in a fuzzy sliding mode controller is thus smaller than that in a fuzzy logic controller. Moreover, the string length of parameters is shorter in a fuzzy sliding mode controller than in a fuzzy logic controller when the parameters are searched by genetic algorithms. The simulation results showed the efficiency of the proposed approach and demonstrated the applicability of the genetic algorithm in the fuzzy sliding mode controller design. © 1997 Elsevier Science B.V.

Keywords: Fuzzy control; Sliding mode; Fuzzy sliding mode; Genetic algorithms

1. Introduction

Recently, fuzzy logic controllers (FLCs) [8] based on approximate reasoning [20] have become an extensively researched topic. A number of applications have confirmed the capability of FLCs [12, 16, 17]. In most cases, the models of plants cannot be easily derived. However, skilled operators can control the plants successfully by simply following certain linguistic statements, such as "*IF condition A happens THEN take action B*". By incorporating human expertise into fuzzy IF–THEN rules, a linguistic FLC can be constructed to control complex or ill-defined systems. However, several well-known difficulties still exist in FLC design: (1) Fuzzy control rules are experience oriented. Suitable membership functions are usually

* Corresponding author.

SSDI 0165-0114(95)00402-5

obtained by time-consuming trial-and-error procedures. (2) Characteristics of fuzzy control systems cannot be pre-specified. (3) There is no criterion to obtain an optimal or at least sub-optimal FLC.

To overcome problems (1) and (2) discussed above, the fuzzy sliding mode control schemes have been extensively studied [3, 9–11, 13]. These approaches are based on the fuzzy logic control and the sliding mode control [19, 4], and have the advantages of both. By introducing the concept of sliding mode to the FLC and fuzzifying the sliding surface, a fuzzy sliding mode controller (FSMC) has a "fuzzy" sliding surface. The control actions in a fuzzy sliding mode controller are smoother than those in a conventional sliding mode controller (SMC). As a result, chattering in an FSMC is smaller than that in an SMC. In FSMC, characteristics of the closed-loop control system can be specified by a user-defined sliding surface. Moreover, establishing the control rules for FSMC is easier than that for conventional FLC. In this paper, we address the issue of using genetic algorithms (GAs) to solve problem (3). The goal is to find a sub-optimal rule-base as well as membership functions of FSMC. Such a sub-optimal FSMC can drive the system state to hit a pre-defined sliding surface as fast as possible, and keep the state sliding along the surface as close as possible.

Genetic algorithms were originally developed by Holland in 1962. The detailed principles, mathematical frameworks and applications can be found in Goldberg's recent book [2]. The use of GAs for solving control problems was proposed in [7]. In fuzzy control, GAs are always applied to search the string space, which is constructed by parameterizing and encoding the rules or membership functions of an FLC into a binary string. Consequently, a sub-optimal FLC can be obtained [5, 6]. However, these strategies have a major drawback: when the number of input variables or linguistic labels increases, the number of fuzzy rules increases exponentially, which further leads to the exponential increase of the encoded string length. This paper will show that the number of rules in an FSMC is in proportion to the number of input variables so that the corresponding string length does not increase exponentially, but linearly.

This paper is organized as follows. Section 2 is a brief description of GAs. Section 3 presents the fuzzy sliding mode control method. Section 4 transforms the design problem of an FSMC to a searching problem that can be solved by GAs. In Section 5, simulation results are presented to verify the efficiency of the proposed approach. Conclusions are given in Section 6.

2. Fundamentals of genetic algorithms

GAs are parallel and global search techniques which take the concepts from evolution theory and natural genetics. They emulate biological evolutions by means of genetic operations such as *reproduction*, *crossover* and *mutation*. Usually, GAs are used as optimization techniques. Although there is no necessary and sufficient condition on the functions which are optimizable by GAs, it has been shown that GAs perform well on *multimodal* functions, i.e. functions which have multiple local optimums. Moreover, various studies have shown that whenever GAs failed to derive the optimal solution on a function, other known techniques failed as well [2].

GAs work with a set of artificial elements (binary strings, e.g. 1 0 1 0 1 0 1 0), called a population. An individual (string) is referred to as a chromosome, and a single bit in the string is called a gene. GAs generate a new population (called offsprings) by applying the genetic operators to the chromosomes in the old population (called parents). Each iteration of genetic operations is referred to as a generation. A fitness function, i.e. the function to be maximized, is used to evaluate the fitness of an individual. One of the important purposes of GAs is to reserve the better schemata, i.e. the patterns of certain genes, so that the offsprings may yield higher fitness than their parents. Consequently, the value of fitness function increases from generation to generation. In most of the GAs, mutation is a random-work mechanism to avoid the local optimum trapping problem. As a result, GAs always can find the global optimal solution.

The basic operations (i.e. reproduction, crossover and mutation) of a simple genetic algorithm (SGA) are described as follows.

(1) *Reproduction*: The Darwinian "survival of the fittest" is the underlying spirit of reproduction. First, a fitness value F is assigned to each individual string in a population. A higher F value indicates a better fit (or larger benefit). Next, the old individual strings are probabilistically selected and copied into a mating pool according to their fit-Uness value. The arrangement allows the strings with a higher fitness to have a greater probability of contributing a larger amount of offsprings in the new population.

(2) *Crossover*: 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. First, two parents are randomly selected from the mating pool. Then, a random crossover point is picked up, on which the parents will exchange their genes. Finally, the parents' genetic codes are mixed by exchanging their codes following the crossover point. For example, consider two parent strings

10101010

01111100

The offsprings are in the following if the crossover point is 5:

10101100

01111010

This random process provides a highly efficient method to search the string space to find a better solution.

(3) *Mutation*: At each iteration, every gene is subject to a random change with probability of the pre-assigned mutation rate. In the binary-coded case, a mutation operator changes a bit from 0 to 1 or vice versa. Overall, the mutation operation introduces new genes into the populations such that the trapping in local optimal points may be avoided.

The offsprings are generated from the parents until the size of the new population is equal to that of the old population. This evolution procedure progresses until the fitness reaches the desired specifications.

3. Fuzzy sliding mode control

3.1. Some useful definitions of fuzzy logic control

The membership functions adopted in this paper are all Gaussian-type functions, i.e. $\mu_A(x) = \exp[-((x-m)/\sigma)^2]$, where m is the center of the membership function on which the membership grade $\mu_A(m) =$, and σ denotes the spread of the membership function. For notational simplicity, we express a fuzzy set A as $A(m, \sigma)$ here and hereafter. For example, the fuzzy set "Positive Medium" with $m = 2$ and $\sigma = 0.7$ is represented as PM(2, 0.7)

Definition 1. A *fuzzy rule-base*, $R \triangleq \bigcup_{j=1}^{N} R_j$, is a union of N fuzzy rules. Each rule R_j in the fuzzy rule-base can be expressed as

R_j: **IF** x_1 *is* $A_{1j}(m_{1j}, \sigma_{1j})$ **and** x_2 *is* $A_{2j}(m_{2j}, \sigma_{2j})$ **and** ... **and** x_n *is* $A_{nj}(m_{nj}, \sigma_{nj})$

THEN u *is* $B_j(\varphi_j, \delta_j)$, $x_i \in X_i$, $u \in U$, (1)

where x_i, $i = 1, 2, \ldots, n$, are input variables and u is an output variable. X_i and U are the universe of inputs and ouput, respectively. A_{ij} and B_j are fuzzy sets called intput linguistic labels and output linguistic labels, respectively.

Remark. The representation of a fuzzy rule-base in (1) can be rewritten in the following compact form:

R_j: **IF** $\boldsymbol{x}$ *is* $\boldsymbol{A}_j(\boldsymbol{m}_j, \boldsymbol{\sigma}_j)$ **THEN** u *is* $B_j(\varphi_j, \delta_j)$, (2)

where $\boldsymbol{x} = [x_1 \ x_2 \ \ldots \ x_n]^{\mathrm{T}} \in \boldsymbol{X} \subset \mathbb{R}^n$ is the state vector, and $\boldsymbol{A}_j = [A_{1j} \ A_{2j} \ \ldots \ A_{nj}]^{\mathrm{T}}$ is called the fuzzy vector whose elements, A_{ij}, are all fuzzy sets.

Definition 2. The *firing strength of the jth rule* is defined as

$$\mu_{R_j}(\boldsymbol{x}) = \bigcap_{i=1}^{n} \mu_{A_{ij}}(x_i), \tag{3}$$

where $\cap$ denotes the "and" operator such as min, product or any T-norm [8].

Definition 3. A fuzzy rule-base R is said to be *complete* if and only if there is at least a rule R_k with nonzero firing strength for any input, i.e.

$$\forall \boldsymbol{x} \in \boldsymbol{X}, \exists k, \text{ s.t. } \mu_{R_k}(\boldsymbol{x}) \neq 0. \tag{4}$$

Definition 4. If a fuzzy rule-base R is complete, then the *jth fuzzy premise function of* R, denoted as $\rho_j(\boldsymbol{x})$, can be defined as

$$\rho_j(\boldsymbol{x}) \triangleq \frac{\mu_{R_j}(\boldsymbol{x})}{\sum_{j=1}^{N} \mu_{R_j}(\boldsymbol{x})}, \quad j = 1, 2, \ldots, N, \tag{5}$$

where $\mu_{R_j}(\boldsymbol{x})$ is the firing strength of the jth rule in R.

Definition 5. The *weighted average defuzzification* is defined as

$$u = \frac{\sum_{j=1}^{N} \varphi_j \mu_{R_j}(\boldsymbol{x})}{\sum_{j=1}^{N} \mu_{R_j}(\boldsymbol{x})} \triangleq \boldsymbol{\varphi}^{\mathrm{T}} \boldsymbol{\rho}(\boldsymbol{x}), \tag{6}$$

where $\boldsymbol{\rho} = [\rho_1 \ \rho_2 \ \ldots \ \rho_N]^{\mathrm{T}}$ is a vector of fuzzy premise functions; $\boldsymbol{\varphi} = [\varphi_1 \ \varphi_2 \ \ldots \ \varphi_N]^{\mathrm{T}}$ and φ_j are the centers of output linguistic labels, B_j.

By the above definitions, we have the following fact:

Fact 1. *If a fuzzy rule-base is complete, then all fuzzy premise functions* $\rho_j(x)$ *are well-defined, i.e.* $\sum_{j=1}^{N} \mu_{R_j}(\boldsymbol{x}) \neq 0$.

In FLC design, it is therefore very important to construct a complete fuzzy rule-base so that all fuzzy premise functions are well-defined. To obtain a complete fuzzy rule-base, a sufficient condition can be described as:

(a) for each individual input variable, overlap the adjacent membership functions of its corresponding linguistic labels;

(b) take all possible combinations of the input linguistic labels to form the premise part.

Fact 2. *Assuming that there are* L_i *corresponding linguistic labels defined on* X_i *for each input variable* x_i, *the total number of rules in a complete rule-base of a fuzzy logic controller described above is given by* $N = \prod_{i=1}^{n} L_i$, *if there are n input variables.*

Especially, if $L_i = L$ for all i, then $N = L^n$. As we can see, the number of rules will grow exponentially with the number of input variables.

3.2. Fuzzy sliding mode control and problem formulation

The dynamical equations of the nonlinear systems considered in this paper are assumed to have the following form [15]:

$$y^{(n)} = f(y, \dot{y}, \ldots, y^{(n-1)}) + b(y, \dot{y}, \ldots, y^{(n-1)})u, \tag{7}$$

where $f(\cdot)$ is an unknown continuous function with known upper bound, i.e. $|f| \leqslant f_{\max}$; $b(\cdot)$ is an unknown positive definite function with known lower bound, i.e. $0 < b_{\min} \leqslant b$; $u \in \mathbb{R}$ is the system input and $y \in \mathbb{R}$ is the system output. To realize (7) into a state space representation, the state can be defined as $x_i = y^{(i-1)} - r^{(i-1)} \triangleq e^{(i-1)}$, $i = 1, 2, \ldots, n$, where r is the reference input, $e = y - r$ is the error signal. Then the system can be represented by the following state equations:

$$\begin{cases} \dot{x}_i = x_{i+1}, & i = 1, 2, \ldots, n-1, \\ \dot{x}_n = f + bu - r^{(n)}. \end{cases} \tag{8}$$

In conventional SMC design, it is necessary to define a sliding surface first. Let us consider the following linear function:

$$s(\boldsymbol{x}) = \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x} = \sum_{i=1}^{n} c_i x_i, \tag{9}$$

where $\boldsymbol{x} = [x_1 \ x_2 \ \ldots \ x_n]^{\mathrm{T}}$ is the state vector and $\boldsymbol{c} = [c_1 \ c_2 \ \ldots \ c_n]^{\mathrm{T}}$ is the sliding surface coefficient vector which has to be properly selected. Then, a sliding surface, Σ, can be viewed as a *crisp* set of states on which (9) is zero, i.e.

$$\Sigma = \{\boldsymbol{x} \mid s(\boldsymbol{x}) = 0\}.$$

Without loss of generality, let us assume $c_n = 1$. One of the basic ideals of sliding mode control is to keep the system state on the sliding surface with an equivalent control law u_{eq} whenever $\boldsymbol{x} \in \Sigma$. By taking the derivative of (9), we have

$$\dot{s} = \bar{\boldsymbol{c}}^{\mathrm{T}}\boldsymbol{x} + [f + bu - r^{(n)}], \tag{10}$$

where $\bar{\boldsymbol{c}} = [0 \ c_1 \ c_2 \ \ldots \ c_{n-1}]^{\mathrm{T}}$. Then u_{eq} could be derived by setting (10) to zero:

$$u_{\mathrm{eq}} = u|_{\dot{s}=0} = -b^{-1}(f - r^{(n)} + \bar{\boldsymbol{c}}^{\mathrm{T}}\boldsymbol{x}). \tag{11}$$

With the equivalent control derived above, the state can be kept on Σ, the system is said to be

in sliding mode and its dynamics can be described by

$$c_1 e + c_2 \dot{e} + \cdots + e^{(n-1)} = 0. \tag{12}$$

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

$$p^{n-1} + c_{n-1} p^{n-2} + \cdots + c_1 = 0, \tag{13}$$

where p is the Laplace operator. With a suitable choice of coefficients c_i, a stabilized control system can be obtained if and only if (13) is Herwitz. As a result, *the dynamic behavior of sliding mode control system is determined by the pre-defined 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 $x \notin \Sigma$. That is [4],

$$u_d = \begin{cases} u^+ > 0 & \text{for } s < 0, \\ 0 & \text{for } s = 0, \\ u^- < 0 & \text{for } s > 0. \end{cases} \tag{14}$$

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

$$u = u_{eq} + u_d. \tag{15}$$

However, there is no way to get an exact u_{eq} with unknown f and b. In this paper, an alternative control law is proposed:

$$u = (1 - \alpha)u_f + \alpha u_h, \tag{16}$$

where u_f is a fuzzy control law obtained from the following fuzzy sliding mode control rule-base:

$$R_j: \quad \textbf{IF } s \text{ is } S_j(m_j, \sigma_j) \textbf{ THEN } u_f \text{ is } U_j(\varphi_j, \delta_j), \tag{17}$$

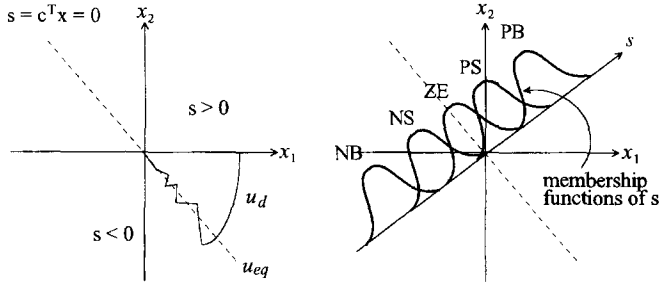
in which S_j and U_j are unknown fuzzy sets; in the second term of (16), u_h is a hitting control law that guarantees the stability of the control system. The switching factor α is defined as

$$\alpha = \begin{cases} 1 & \text{for } |s| \geqslant s_0, \\ 0 & \text{for } |s| < s_0, \end{cases} \tag{18}$$

in which s_0 is a pre-specified bound of s.

Fact 3. *According to Fact* 2, *without loss of generality, assume that* $L_i = L, \forall i$. *Then the number of rules in a conventional FLC with complete rule-base* (1) *is given by* $N = L^n$. *On the other hand, if the states are combined into a single variable s, the number of rules in an FSMC with complete rule-base* (17) *is given by* $N = L$.

Fig. 1 shows a 2-dimensional case of state plane to illustrate the concept of SMC and FSMC. The sliding surface in Fig. 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 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, etc. To reduce the system nonideality effects, the fuzzification operation was applied to convert the *crisp* sliding surface into a *fuzzy* one



(a) Classical Sliding Mode (b) Fuzzy Sliding Mode Control

Fig. 1. The concept of fuzzifying a crisp sliding surface to a fuzzy one.

[9, 10]. 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. Fig. 1(b) illustrates such a concept, in which five linguistic labels (PB, PS, ZE, NS, and NB) are assigned to the sliding variable s.

The input and output linguistic labels of the fuzzy sliding mode control rule-base (17) can be determined by certain strategies such as heuristic [3, 10, 13], adaptive [9] or self-organizing [11] methods. However, what is the optimal selection of the membership functions of S_j and U_j? It remains an open problem. In the next section, the parameters of these fuzzy sets are tuned by means of GAs to derive a sub-optimal FSMC in the sense that the fitness functions are maximized. The main purposes of this paper are:

(1) Apply GAs to search the parameter space of linguistic labels, S_j and U_j, in the fuzzy sliding mode control rule-base (17) for suitable fuzzy control law, u_f. The purpose of such a fuzzy control law is to draw the state to hit and slide along the sliding surface without consuming too much control energy.

(2) Derive the hitting control law, u_h, to guarantee the stability. During the learning process, if the GA-learned fuzzy control law, u_f, is not suitable, such that the state travels toward the direction of divergence, i.e. $|s| \geqslant s_0$, then u_f is turned off, and u_h is turned on to pull the state back to the safe region, $|s| < s_0$.

4. GA-based fuzzy sliding mode controller and stability consideration

In this section, the FSMC with the hitting control law and the fuzzy control law are described in detail. The hitting control law is derived for guaranteeing the system stability; the fuzzy control law is learned by the genetic algorithms. Such kinds of FSMCs are denoted as GA-based FSMCs. Two types of GA-based FSMCs are proposed. In the Type-I GA-based FSMC, only the membership functions in the THEN-part of the fuzzy control rule-base (17) are adjusted; on the other hand, the Type-II GA-based FSMC tunes all membership functions in the fuzzy control rule-base. The hitting control law in both types of FSMC are identical.

To improve the performance of GA-based FSMC, the genetic algorithm with elitist model [2] was adopted. Its basic principle is to always include the most fitted individual in the population. The operations are: if the best individual generated up to time t is not in the population of the new generation of time $t + 1$, then select one member from the new population and replace it with the elitist of the old population.

4.1. Hitting control law u_h

The hitting control law can be used to drive the state to *hit* the sliding surface wherever the initial state is. To derive the hitting control law, let us set $\alpha = 1$ and consider a Lyapunov function

$$V = \tfrac{1}{2}s^2. \tag{19}$$

The sliding condition [15] which guarantees the stability of a sliding mode control system is

$$\dot{V} = s\dot{s} < -\eta|s|, \tag{20}$$

where $\eta > 0$. Suppose that we know the upper bound of $f(\cdot)$ and the lower bound of b, i.e. $|f| \leqslant f_{\max}$, $0 < b_{\min} \leqslant b$. Substituting (10) into (20), we have

$$\dot{V} \leqslant |sb|\,|b^{-1}(f - r^{(n)} + \bar{c}^{\mathrm{T}}\boldsymbol{x})| + sbu_h. \tag{21}$$

To satisfy (20) and (21), the hitting control law can be selected as

$$u_h = -\operatorname{sign}(s)[b_{\min}^{-1}(f_{\max} + |r^{(n)}| + |\bar{c}^{\mathrm{T}}\boldsymbol{x}| + \eta)]. \tag{22}$$

By substituting (22) into (21), the stability condition of (20) can be verified.

Remark. Estimating the suitable values of $f_{\max}$ and $b_{\min}$ is an important job for deriving u_h. However, the precision of the estimation values of these two bounds is dependent on how well the designer understands the system. For example, to control

the famous inverted pendulum system,

$$\begin{cases} \dot{x}_1 = x_2, \\ \dot{x}_2 = \dfrac{g\sin x_1 - (ml/(m_c+m))x_2^2\cos x_1 \sin x_1}{\frac{4}{3}l - (ml/(m_c+m))\cos^2 x_1} \\ \qquad + \dfrac{(1/(m_c+m))\cos x_1}{\frac{4}{3}l - (ml/(m_c+m))\cos^2 x_1}\, u, \end{cases}$$

where x_1 denotes the angle of the pendulum with respect to the vertical line, and x_2 denotes the angular velocity of the pendulum. In a real system, the limit values of x_1 and x_2 are always specified. For example, $|x_1| < 1$ rad and $|x_2| < 4$ rad/s. If we know the structure of the system and the variant range of parameters (e.g. the gravity constant: $g = 9.8\,\mathrm{m/s^2}$; the mass of the cart: $0.8\,\mathrm{kg} < m_c < 1.2\,\mathrm{kg}$; the mass of the pendulum: $0.08\,\mathrm{kg} < m < 0.12\,\mathrm{kg}$, the length of the pendulum: $0.9\,\mathrm{m} < l < 1.1\,\mathrm{m}$), then we can calculate the bounds of f and b as follows:

$$\begin{aligned} |f(x_1,x_2)| &= \left|\frac{g\sin x_1 - (ml/(m_c+m))x_2^2\cos x_1\sin x_1}{\frac{4}{3}l - (ml/(m_c+m))\cos^2 x_1}\right| \\ &\leqslant \left|\frac{9.8 + ((0.12\times 1.1)/(0.8+0.08))\times 4^2}{\frac{4}{3}\times 0.9 - (0.12\times 1.1)/(0.8+0.08)}\right| \\ &\leqslant 10.31 \triangleq f_{\max}, \\ |b(x_1,x_2)| &= \left|\frac{(1/(m_c+m))\cos x_1}{\frac{4}{3}l - (ml/(m_c+m))\cos^2 x_1}\right| \\ &\geqslant \left|\frac{(1/(1.2+0.12))\cos 1}{\frac{4}{3}\times 1.1 - ((0.08\times 0.9)/(1.2+0.12))\cos^2 1}\right| \\ &\geqslant 0.278 \triangleq b_{\min}. \end{aligned}$$

Unfortunately, if we do not have enough knowledge about a system, we cannot estimate the bounds precisely. However, even under the situation of lacking system information, we can always assume that $f_{\max}$ is very large and $b_{\min}$ is very small. Then, the hitting control u_h can be calculated without any difficulty. In such a situation, the poorer esimations of $f_{\max}$ and $b_{\min}$ we take up, the larger u_h we will get. If the estimation is too poor so that the amplitude of the hitting control is too large, then u_h will be saturated. That is, u_h becomes a high gain bang-bang control: $u_h = -\mathrm{sign}(s)\bar{U}$, where $\bar{U}$ *denotes the maximal control input.*

From the control law (16), we see that the hitting control u_h plays the role of a protector. That is, when the state lies in the boundary layer, $|s| < s_0$, the GA-learned fuzzy control u_f is active and the hitting control u_h is disabled; on the other hand, if u_f unfortunately does not behave well such that the state tends to go outside the boundary layer, as soon as the state hits the boundary of the layer, the hitting control becomes enabled for driving the state back toward the layer.

4.2. *Fuzzy control law* u_f

The fuzzy control law of (16) is given by the following fuzzy sliding mode control rule-base:

$$R_j\text{: } \textbf{IF } s \textit{ is } S_j(m_j,\sigma_j) \textbf{ THEN } u_f \textit{ is } U_j(\varphi_j), \quad j = 1,2,\ldots,N. \tag{23}$$

Remark. Since the defuzzification method adopted in this paper is the weighted-average, the values of δ_j (the second parameters of U_j) are never used, the search for suitable δ_j is not necessary. Hence, for notational simplicity, we omit δ_j here and hereafter. However, the values of δ_j are required in the center-of-area defuzzification. The optimization of δ_j and their effects are reserved for future research.

From (5) and (6), the fuzzy control law u_f of (23) is given by

$$u_f = \boldsymbol{\varphi}^T\boldsymbol{\rho}(s) \tag{24}$$

with $\boldsymbol{\varphi} = [\varphi_1\ \varphi_2\ \ldots\ \varphi_N]^T$, $\boldsymbol{\rho}(s) = [\rho_1(s)\ \rho_2(s)\ \ldots\ \rho_N(s)]^T$ and

$$\rho_j(s) = \frac{\mu_{S_j}(s)}{\sum_{j=1}^{N}\mu_{S_j}(s)}. \tag{25}$$

The details of learning procedures of the fuzzy control law in both types of GA-based FSMC are described as follows:

4.2.1. Type-I *GA-based FSMC*

The fuzzy control law of Type-I GA-based FSMC is given by (24) with adjustable consequence part, $\boldsymbol{\varphi}$, and fixed premise part, $\boldsymbol{\rho}(s)$. That is, m_j and σ_j of the membership functions in the IF-part of

(23) are pre-determined and fixed during the learning period. Only the parameters in the THEN-part, φ_j, are adjusted by GAs. Hence the parameter vector, $\boldsymbol{\theta}$, for the genetic algorithm is defined as

$$\boldsymbol{\theta} \triangleq \boldsymbol{\varphi} = [\varphi_1 \ \varphi_2 \ \dots \ \varphi_N]^{\mathrm{T}}. \tag{26}$$

Every string in a population corresponds to an FSMC candidate which has its parameters encoded into binary form. The population size is heuristically selected to be M. The learning procedure for Type-I GA-based FSMC is summarized as follows:

Step 1: Define a sliding surface function $s = \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}$ by carefully selecting the parameter vector $\boldsymbol{c}$ such that (13) is stable and satisfies the desired specifications.

Step 2: Define a suitable performance index (cost function), J. For example, it can be defined as

$$J = \sum_{k=1}^{K} (w \| s(k) \| + v \| u(k) \|), \tag{27}$$

where $k = \mathrm{int}(t/\Delta t)$ denotes the iteration instance; Δt is the sampling period; $\mathrm{int}(\cdot)$ is the round-off operator; $K = \mathrm{int}(t_{\max}/\Delta t)$ is the number of iterations in one run; $t_{\max}$ denotes the running time in one run; w and v are positive weights. The norm $\| \cdot \|$ can be viewed as a generalized energy measure of a signal. Then the fitness function can be defined as

$$F = 1/(J + \varepsilon_0), \tag{28}$$

where ε_0 is a small positive constant used to avoid the numerical error of dividing by zero. If the goal is to drive the state to hit the sliding surface as fast as possible regardless of the amount of control effort required, then the value of w selected would be significantly larger than that of v. On the other hand, the consumption of control energy will be punished by heavy weighting in u (select a larger v value).

Step 3: Determine: (a) the number of rules N; (b) the population size M; (c) the crossover and mutation rates and (d) the bit length d.

Step 4: Determine the parameter values of the IF-part (m_j and σ_j) to form a number of Gaussian-type membership functions for s. Heuristics and practical considerations will be helpful for deciding the values. In Section A.1, we propose an algorithm that can be used to determine m_j and σ_j mathematically.

Step 5: Determine the search space of the parameters of the THEN-part. If the specified range of control signal is $[-\bar{U}, +\bar{U}]$, then take this interval as the search space of φ_j.

Step 6: Randomly generate M parameter vectors

$$\boldsymbol{\varphi}^{(i)} = [\varphi_1^{(i)} \ \varphi_2^{(i)} \ \dots \ \varphi_N^{(i)}] \tag{29}$$

for $i = 1, 2, \dots, M$. Then the fuzzy control rules of the ith Type-I GA-based FSMC (denoted by FSMC-I$^{(i)}$) with complete premise parts and random consequence parts can be constructed, e.g.

FSMC-I$^{(i)}$:

$$\begin{cases} R_1^{(i)}\text{: } \textbf{IF } s \textit{ is } \mathrm{PB}(3, 1.2) \textbf{ THEN } u_{\mathrm{f}} \textit{ is } U_1^{(i)}(\varphi_1^{(i)}), \\ R_2^{(i)}\text{: } \textbf{IF } s \textit{ is } \mathrm{PM}(1.5, 1) \textbf{ THEN } u_{\mathrm{f}} \textit{ is } U_2^{(i)}(\varphi_2^{(i)}), \\ \qquad \vdots \\ R_N^{(i)}\text{: } \textbf{IF } s \textit{ is } \mathrm{NB}(-3, 1.2) \textbf{ THEN } u_{\mathrm{f}} \textit{ is } U_N^{(i)}(\varphi_N^{(i)}), \end{cases} \tag{30}$$

where $U_j^{(i)}$ are unknown linguistic labels for control output u_{f}; $\varphi_j^{(i)}$ are adjustable parameters of $U_j^{(i)}$ and will be encoded into the binary strings for genetic operations. Then the output of FSMC-I$^{(i)}$ can be obtained by (24).

Step 7: The parameter vectors to be tuned by GA for FSMC-I$^{(i)}$ ($i = 1, 2, \dots, M$) are assigned to be

$$\begin{aligned} \boldsymbol{\theta}^{(i)} &= [\theta_1^{(i)} \ \theta_2^{(i)} \ \dots \ \theta_N^{(i)}]^{\mathrm{T}} \\ &:= [\varphi_1^{(i)} \ \varphi_2^{(i)} \ \dots \ \varphi_N^{(i)}]^{\mathrm{T}} = \boldsymbol{\varphi}^{(i)}. \end{aligned} \tag{31}$$

Step 8: Encode $\theta_j^{(i)}$ ($i = 1, 2, \dots, M, j = 1, 2, \dots, N$) into d-bit binary strings $B_j^{(i)}$, i.e.

$$B_j^{(i)} = (b_j^1 \ b_j^2 \ \dots \ b_j^d)^{(i)}, \tag{32}$$

where $b_j^1, \dots, b_j^d \in \{0, 1\}$. Then, an individual chromosome can be constructed by cascading N d-bit binary strings into a single string as

$$P^{(i)} = B_1^{(i)} : B_2^{(i)} : \cdots : B_N^{(i)}. \tag{33}$$

Step 9: A population can be established as

$$\boldsymbol{P} = \{P^{(1)}, P^{(2)}, \dots, P^{(M)}\}, \tag{34}$$

in which an individual $P^{(i)}$ corresponds to a set of binary-coded parameters of an FSMC-I$^{(i)}$ which is a candidate of the optimal fuzzy sliding mode controller.

Step 10: Use (24) to obtain the fuzzy control part of FSMC-I$^{(i)}$, $i = 1, \ldots, M$, and apply the control law (16) to the plant, where the hitting control law u_h is given in (22).

Step 11: Evaluate the performance index and the fitness function of every FSMC by (27) and (28), respectively.

Step 12: Apply genetic operators (reproduction, crossover, mutation) to the old population $\boldsymbol{P}$.

Step 13: Generate a set of offsprings to form a new population $\boldsymbol{P}'$. Then apply the elitist model, i.e. compare the best fitted strings in the old and the new populations; then randomly replace a string in $\boldsymbol{P}'$ with the elitist in $\boldsymbol{P}$ if it has a better fitness value than all individuals in $\boldsymbol{P}'$.

Step 14: Replace the old population $\boldsymbol{P}$ by the new population $\boldsymbol{P}'$.

Step 15: Decode every new individual $P^{(i)}$ back to the original parameter space and obtain a set of new parameter $\boldsymbol{\theta}^{(i)}$, $i = 1, 2, \ldots, M$. Totally, M new Type-I FSMCs are generated by GAs.

Step 16: Take $\boldsymbol{\varphi}^{(i)} = \boldsymbol{\theta}^{(i)}$ and repeat Step 10 to Step 16 until the performance of the best individual satisfies the desired specifications.

4.2.2. Type-II *GA-based FSMC*

The fuzzy control law of Type-II GA-based FSMC is given by (24) with both adjustable consequence part, $\boldsymbol{\varphi}$, and premise part, $\boldsymbol{\rho}(s)$. Both m_j, σ_j in the IF-part and φ_j in the THEN-part of (23) are adjusted during the learning period. Hence, the parameter vector, $\boldsymbol{\theta}$, of the genetic algorithm can be written as

$$\begin{aligned}\boldsymbol{\theta} &= [\boldsymbol{m}^{\mathrm{T}}\boldsymbol{\sigma}^{\mathrm{T}}\boldsymbol{\varphi}^{\mathrm{T}}]^{\mathrm{T}} \\ &= [m_1\ m_2\ \ldots\ m_N\ \sigma_1\ \sigma_2\ \ldots\ \sigma_N\ \varphi_1\ \varphi_2\ \ldots\ \varphi_N]^{\mathrm{T}}.\end{aligned} \tag{35}$$

Basically, the learning procedure of Type-II GA-based FSMC is similar to that of Type-I. It is described as follows:

Step 1–*Step* 3: Same as Step 1–Step 3 in the learning procedure of Type-I GA-based FSMC.

Step 4: Determine the search space of m_j and σ_j. See Section A.2.

Step 5: Same as Step 5 in the learning procedure of Type-I GA-based FSMC.

Step 6: Randomly generate $3M$ parameter vectors:

$$\begin{cases}\boldsymbol{m}^{(i)} = [m_1^{(i)}\ m_2^{(i)}\ \ldots\ m_N^{(i)}]^{\mathrm{T}} \\ \boldsymbol{\sigma}^{(i)} = [\sigma_1^{(i)}\ \sigma_2^{(i)}\ \ldots\ \sigma_N^{(i)}]^{\mathrm{T}} \\ \boldsymbol{\varphi}^{(i)} = [\varphi_1^{(i)}\ \varphi_2^{(i)}\ \ldots\ \varphi_N^{(i)}]^{\mathrm{T}}\end{cases} \tag{36}$$

for $i = 1, 2, \ldots, M$. Then the fuzzy control rules of the ith Type-II GA-based FSMC (denoted by FSMC-II$^{(i)}$) with random premise and consequence parts can be constructed as

FSMC-II$^{(i)}$:

$$\begin{cases}R_1^{(i)}\text{: } \textbf{IF } s \textit{ is } S_1^{(i)}(m_1^{(i)}, \sigma_1^{(i)}) \textbf{ THEN } u_{\mathrm{f}} \textit{ is } U_1^{(i)}(\varphi_1^{(i)}), \\ R_2^{(i)}\text{: } \textbf{IF } s \textit{ is } S_2^{(i)}(m_2^{(i)}, \sigma_2^{(i)}) \textbf{ THEN } u_{\mathrm{f}} \textit{ is } U_2^{(i)}(\varphi_2^{(i)}), \\ \qquad\vdots \\ R_N^{(i)}\text{: } \textbf{IF } s \textit{ is } S_N^{(i)}(m_N^{(i)}, \sigma_N^{(i)}) \textbf{ THEN } u_{\mathrm{f}} \textit{ is } U_N^{(i)}(\varphi_N^{(i)}),\end{cases} \tag{37}$$

where $S_j^{(i)}$ and $U_j^{(i)}$ are unknown linguistic labels for input variables s and control output u_{f}, respectively; $m_j^{(i)}$, $\sigma_j^{(i)}$ and $\varphi_j^{(i)}$ are adjustable parameters of $S_j^{(i)}$ and $U_j^{(i)}$, and will be encoded into the binary strings for genetic operations. Then the fuzzy control part of FSMC-II$^{(i)}$ can be obtained by (24).

Step 7: The parameter vectors to be tuned by GA for FSMC-II$^{(i)}$ ($i = 1, 2, \ldots, M$) are assigned as

$$\begin{aligned}\boldsymbol{\theta}^{(i)} &= [\theta_1^{(i)}\ \ldots\ \theta_N^{(i)} : \theta_{N+1}^{(i)}\ \ldots\ \theta_{2N}^{(i)} : \theta_{2N+1}^{(i)}\ \ldots\ \theta_{3N}^{(i)}]^{\mathrm{T}} \\ &:= [m_1^{(i)}\ \ldots\ m_N^{(i)} : \sigma_1^{(i)}\ \ldots\ \sigma_N^{(i)} : \varphi_1^{(i)}\ \ldots\ \varphi_N^{(i)}]^{\mathrm{T}} \\ &= [\boldsymbol{m}^{(i)\mathrm{T}} : \boldsymbol{\sigma}^{(i)\mathrm{T}} : \boldsymbol{\varphi}^{(i)\mathrm{T}}]^{\mathrm{T}}.\end{aligned} \tag{38}$$

Step 8: Encode $\theta_j^{(i)}$ ($i = 1, 2, \ldots, M, j = 1, 2, \ldots, 3N$) into d-bit binary strings $B_j^{(i)}$ as shown in (32), and then cascade $3N$ d-bit binary strings into an individual string

$$\begin{aligned}P^{(i)} = B_1^{(i)} : \cdots : B_N^{(i)} : B_{N+1}^{(i)} : \cdots : B_{2N}^{(i)} : \\ B_{2N+1}^{(i)} : \cdots : B_{3N}^{(i)}.\end{aligned} \tag{39}$$

Step 9: A population can be established as

$$\boldsymbol{P} = \{P^{(1)}, P^{(2)}, \ldots, P^{(M)}\}, \tag{40}$$

in which an individual $P^{(i)}$ corresponds to a set of binary-coded parameters of an FSMC-II$^{(i)}$ which is a candidate of the optimal fuzzy sliding mode controller.

Step 10: Use (24) to obtain the fuzzy control law of FSMC-II$^{(i)}$, $i = 1, \ldots, M$. Then apply the control law (16) to the plant, where the hitting control law u_h is given by (22).

Step 11–*Step* 15: Same as Step 11–Step 15 in the learning procedure of Type-I Ga-based FSMC.

Step 16: Take $\boldsymbol{m}^{(i)} = [\theta_1^{(i)} \ \ldots \ \theta_N^{(i)}]^T$, $\boldsymbol{\sigma}^{(i)} = [\theta_{N+1}^{(i)} \ \ldots \ \theta_{2N}^{(i)}]^T$ and $\boldsymbol{\varphi}^{(i)} = [\theta_{2N+1}^{(i)} \ \ldots \ \theta_{3N}^{(i)}]^T$, and repeat Step 10 to Step 16 until the performance of the best individual satisfies the desired specifications.

Comparing with Type-I and Type-II approaches, Type-II is more flexible than Type-I, but Type-I is more efficient than Type-II. First, the search space is merely constructed by the parameter strings of the THEN-part in the Type-I approach, while it is constructed by the IF-part and the THEN-part parameter strings in the Type-II approach. Obviously, the former is much smaller than the latter. Secondly, the encoded length of bit strings in the Type-I approach is much shorter than that in Type-II, such that the computation load of the Type-I approach is less heavy than that of Type-II. Therefore, if what we want is just an *acceptable* result instead of a best one, the Type-I approach will be a better choice.

5. Simulation results and discussions

In this section, we examine the performance of our methods by a number of examples. Consider a second-order nonlinear system with the dynamical equation in the following:

$$m\ddot{y} + \beta(y^2 - 1)\dot{y} + ky = u, \tag{41}$$

where u and y denote the system input and output, respectively, m, c and k are positive constants. Eq. (41) can be regarded as the description of a mass–spring–damper system with a position-dependent damping coefficient $\beta(y^2 - 1)$, or, equivalently, an RLC electrical circuit with a nonlinear resistor. With $m = 1$ and $k = 1$, the unforced system ($u = 0$) becomes the famous Ven der Pol oscillator

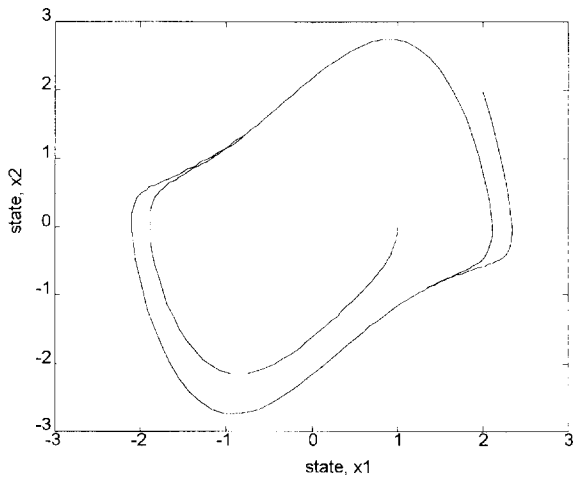


Fig. 2. The unforced system behaves as a Ven der Pol oscillator.

[15]. If the states are defined to be $x_1 = y$, and $x_2 = \mathrm{d}y/\mathrm{d}t$, then the state trajectory of the unforced system with $\beta = 1$ has a limit cycle, as shown in Fig. 2. Assume that the specified input voltage (force) is between $[-30, 30]$. In our demonstration, the control goal is to drive the state toward the sliding surface, and slide along it so that x_1 and x_2 approach zero as t approaches infinity.

In the following simulations, the sliding surface function is selected as $s = x_1 + x_2$, the boundary layer is selected as $[-3, 3]$, and the initial conditions are $x_1(0) = 3$ and $x_2(0) = 0$. The prototype of fuzzy control rule-base has six rules ($N = 6$), defined as follows:

$$R_j\text{: } \textbf{IF } s \textit{ is } S_j(m_j, \sigma_j) \textbf{ THEN } u_f \textit{ is } U_j(\varphi_j),$$
$$j = 1, 2, \ldots, 6. \tag{42}$$

The population size $M = 10$; the crossover and mutation rate are selected as 0.8 and 0.01, respectively.

According to (27), select $w = T \triangleq k \cdot \Delta t$, $v = 2$, and use the l_1-norm. If $t_{max} = 6\,\mathrm{s}$ and $\Delta t = 5\,\mathrm{ms}$, then $K = 1200$ and the cost function becomes

$$J = \sum_{k=1}^{1200} (T|s(k)| + 2|u(k)|). \tag{43}$$

Comparing with the equal-weighted case ($w = v = 1$), in (43) while T is small, the weight of the control

signal is larger than that of the sliding function, so that the controller which can reserve more control energy gets higher fitness. However, as time progresses, T is larger and larger. Hitting and sliding become more and more important than energy conservation. Consequently, the controller which can pull the state closer to the sliding surface gets a higher score. In conclusion, the controller which can drive the state toward the surface quickly and does not dissipate too much control energy will yield higher fitness.

The following two examples are presented to illustrate the learning results of Type-I and Type-II GA-based FSMC, respectively.

Example 1 (Type-I *GA-based FSMC*). By the EPA (Equal Partition Algorithm, see Section A.1), it is easy to determine the value of the IF-part parameters. Assume that the membership grade of crossover point is $\mu = 0.6$, then we have $\{S_j(m_j, \sigma_j) \mid j = 1, 2, \ldots, 6\} = \{\mathrm{NB}(-3, 0.84), \mathrm{NS}(-1.8, 0.84), \mathrm{NZ}(-0.6, 0.84), \mathrm{PZ}(0.6, 0.84), \mathrm{PS}(1.8, 0.84), \mathrm{PB}(3, 0.84)\}$.

Since the range of system input is $[-30, 30]$, that implies the amplitude of controller cannot exceed the interval. It is reasonable to select the universe of the discourse of φ_j to be within the interval $[-30, 30]$. In the Type-I approach, the parameters to be learned by GAs are $\theta_j := \varphi_j$ $(j = 1, 2, \ldots, 6)$. The initial population is generated by a random number generator. If every parameter is encoded to an 8-bit $(d = 8)$ binary string, then cascading six strings of $(\theta_1, \theta_2, \ldots, \theta_6)$ forms a 48-bit chromosome.

Fig. 3 shows the evaluation process of the cost function of the best individual. We can see that the cost function indeed decreased from generation to generation. The decoded values of the best individual in the generation 500 are shown in Table 1. It is the final parameter set of Type-I GA-based FSMC that we want. If each output linguistic label U_j is assigned an appropriate physical meaning such as NB and PS based on the values of φ_j, then the final fuzzy rule-base and membership functions

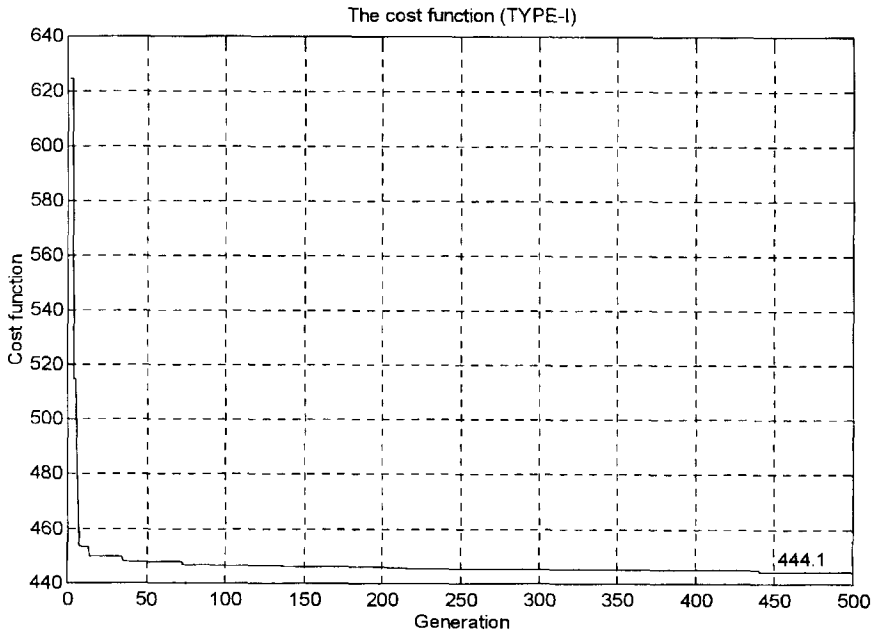


Fig. 3. The cost function (Type-I).

Table 1
The final parameter sets of the Type-I GA-based FSMC

φ_1	φ_2	φ_3	φ_4	φ_5	φ_6
−2.9412	−30	−21.5294	21.7647	29.2941	27.1765

for the FSMC may look like:

R_1: **IF** s *is* NB(−3, 0.84) **THEN** u_f *is* NS(−2.9412),

R_2: **IF** s *is* NS(−1.8, 0.84) **THEN** u_f *is* NB(−30),

R_3: **IF** s *is* NZ(−0.6, 0.84) **THEN** u_f *is* NM(−21.5294),

R_4: **IF** s *is* PZ(0.6, 0.84) **THEN** u_f *is* PZ(21.7647),

R_5: **IF** s *is* PS(1.8, 0.84) **THEN** u_f *is* PB(29.2941),

R_6: **IF** s *is* PB(3, 0.84) **THEN** u_f *is* PM(27.1765).

Finally, the state trajectory and the control signal of the fuzzy sliding mode control system are shown in Fig. 4.

Example 2 (Type-II *GA-based FSMC*). According to the strategy in Section A.2, the search space of

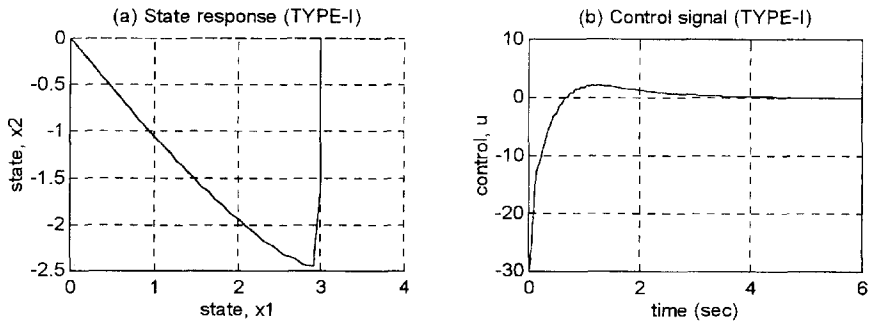


Fig. 4. The state response and the control signal of the Type-I GA-based FSMC.

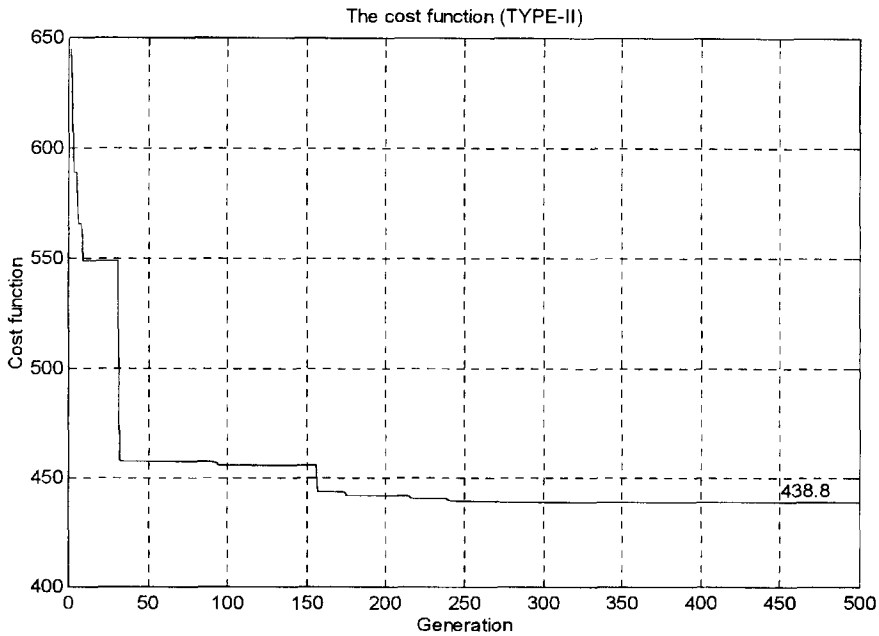


Fig. 5. The cost function (Type-II).

m_j, σ_j and φ_j are selected as $[-3,3]$, $[0.5,1.5]$ and $[-30,30]$, respectively. The parameters of premise and consequence parts, m_i, σ_i, φ_j, were encoded to 8-bit binary strings, respectively. Therefore, cascading 18 strings of $(m_1, \ldots, m_6, \sigma_1, \ldots, \sigma_6, \varphi_1, \ldots, \varphi_6)$ forms a 144-bit individual. Because the number of adjustable parameters in Type-II is larger than that in Type-I, we may expect that the final controller learned by Type-II would be better than that by Type-I.

Fig. 5 shows the evaluation process of the cost function of the best individual. Again, the cost function decreased from generation to generation. Comparing Fig. 5 with Fig. 3 we see that the minimum of the cost function in Type-II (438.8) is smaller than that in Type-I (444.1), but the convergence rate of Type-II is slower than that of Type-I. The decoded values of the best individual in the generation 500 are shown in Table 2. It is the final parameter set of Type-II GA-based FSMC that we want. If each linguistic label is assigned with an appropriate physical meaning such as NB and PS based on its value, then the final fuzzy rule-base and membership functions for the FSMC may look like:

R_1: **IF** s *is* PM(1.9882, 1.1784) **THEN** u_f *is* PM(29.5294),

R_2: **IF** s *is* NS(−1.2353, 1.4725) **THEN** u_f *is* NS(−11.4118),

R_3: **IF** s *is* NB(−2.9765, 1.1275) **THEN** u_f *is* NB(−29.7647),

R_4: **IF** s *is* PB(2.0824, 1.0059) **THEN** u_f *is* PS(22.2353),

R_5: **IF** s *is* NM(−1.6118, 0.5863) **THEN** u_f *is* NM(−28.1176),

R_6: **IF** s *is* PS(1.9412, 1.3353) **THEN** u_f *is* PB(30.0000).

Finally, the state trajectory and the control signal of the fuzzy sliding mode control system are shown in Fig. 6.

We had also simulated the design examples under the conditions of different GA parameters. In general, mutation rate $\in [0.005, 0.01]$ and crossover rate $\in [0.75, 0.95]$ (see [14]). However, we cannot include the complete results due to limited space. The GA parameters selected in this paper are typical values as seen in many related researches. In our simulations, we found that a higher mutation

Table 2
The final parameter sets of the Type-II GA-based FSMC

parameter \ index j	1	2	3	4	5	6
m_j	1.9882	−1.2353	−2.9765	2.0824	−1.6118	1.9412
σ_j	1.1784	1.4725	1.1275	1.0059	0.5863	1.3353
φ_j	29.5294	−11.4118	−29.7647	22.2353	−28.1176	30

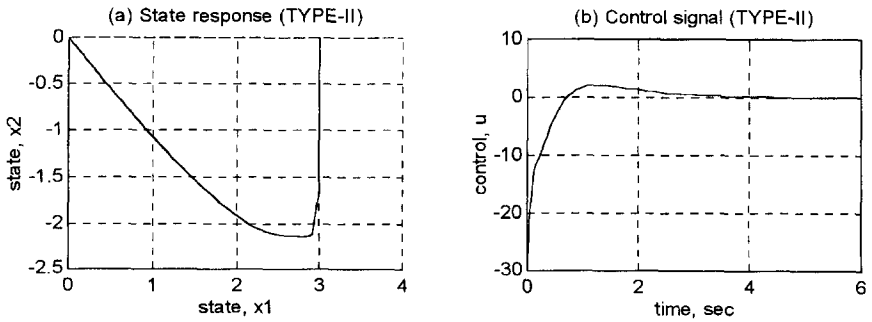


Fig. 6. The state response and the control signal of the Type-I GA-based FSMC.

rate may yield a higher probability to escape the local minimum but may break the better schemata. The larger population size contains sufficient information for finding the solution but takes a very long time to evaluation and causes heavy computation load. More details of the effects of GA parameters can be found in [1, 14].

6. Conclusions

In this paper, we have shown a genetic algorithm with the elitist model that indeed behaves as an efficient searching tool for finding a sub-optimal fuzzy sliding mode controller. Two types of GA-based learning algorithms for FSMCs are discussed. In the Type-I approach, only the parameters in the THEN-part are learned, while all the parameters in both the IF-part and the THEN-part are considered in the Type-II approach. The Type-I approach has the advantages of a shorter string length and a smaller search space such that its convergence rate is faster than Type-II. The shortcoming is that the IF-part of the rule-base has to be pre-defined heuristically. However, an Equal Partition Algorithm is proposed to determine the IF-part mathematically. On the other hand, the Type-II approach has a longer string length by including more parameters to be learned in the rule-base. The IF-part need not be defined a priori and can be tuned by the genetic algorithm. It leads the Type-II approach to have a greater possibility of finding the global minimum than Type-I, but the convergence rate may be slower.

The most important advantage of introducing the sliding mode into fuzzy control is that the dynamic behavior of a fuzzy control system can be specified and dominated by the user-defined sliding surface. Moreover, an MISO fuzzy controller can be transformed into an SISO fuzzy sliding mode controller by combining the system states into a single sliding variable. The number of control rules can be reduced to a linear function of the number of input variables. Hence, the string length used to learn an FSMC is shorter than that used to learn a conventional FLC.

The proposed algorithms are currently under hardware implementation and will also be tested experimentally in hard disk servo control as well as XY table precision positioning in the near future.

Appendix A

This appendix gives one of the possible methods to determine the universe of discourse as well as the values of the parameters m_j and σ_j in the IF-part.

A.1. Equal Partition Algorithm for Type-I GA-based FSMC

In the Type-I GA-based FSMC, one of the most important jobs is to decide how to partition the IF-part and then determine the values of m_j and σ_j.

First, we have to determine the range of m_j. From (9), we have

$$|s| \leqslant \sum_{i=1}^{n} |c_i||x_i| \leqslant \sum_{i=1}^{n} |c_i|\bar{x}_i \triangleq s_0,$$

where the c_i's are design parameters, given by the designer; $\bar{x}_i$ denotes the upper (safe) bound of $|x_i|$ and can be obtained from the system specification. Therefore, the boundary layer $[-s_0, s_0]$ is easy to calculate and the universe of discuss of m_j can be set to be $[-s_0, s_0]$ directly. Next, for simplicity, the values of m_j's are obtained by equally dividing the interval into $N-1$ partitions, that is

$$m_j = (-s_0) + (j-1)\Delta,$$

where N is the number of membership functions and $\Delta = 2s_0/(N-1)$ is the width of the partition.

Secondly, to determine the value of σ_j, let us consider the simplest case that every Gaussian-type membership function in the IF-part has equal variant, i.e. $\sigma_j \triangleq \sigma$, $\forall j$. By definition, we can get the value of σ by solving any two equations of adjacent membership functions:

$$\mu_i = \exp\left(-\left(\frac{s-m_i}{\sigma}\right)^2\right)$$

and

$$\mu_j = \exp\left(-\left(\frac{s-m_j}{\sigma}\right)^2\right).$$

Assume that the membership grade of the crossover point of these two adjacent membership functions is μ, then we have

$$\sigma = \sqrt{-\left(\frac{\Delta}{2}\right)^2 \Big/ (\ln \mu)}.$$

A.2. Determine the search space of m_j and σ_j in Type-II *GA-based FSMC*

In the Type-II GA-based FSMC, we have to determine the search space of m_j and σ_j such that GAs can search on. It is reasonable to choose the search space as $m_j \in [-s_0, s_0]$ and $\sigma_j \in [\underline{\sigma}, \bar{\sigma}]$, where

$$\underline{\sigma} = \sqrt{-\left(\frac{\Delta}{2}\right)^2 \Big/ \ln \underline{\mu}},$$

$$\bar{\sigma} = \sqrt{-\left(\frac{\Delta}{2}\right)^2 \Big/ \ln \bar{\mu}},$$

in which $\underline{\mu}$ and $\bar{\mu}$ are the lower and upper membership grades of the crossover points. For example, $\underline{\mu} = 0.1$ and $\bar{\mu} = 0.9$.

References

[1] T. Back, Optimal mutation rates in genetic search, *Proc. 5th Internat. Conf. on Genetic Algorithms* (1993) 2–8.

[2] D.E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning* (Addison-Wesley, Reading, MA, 1989).

[3] G.C. Hwang and S.C. Lin, A stability approach to fuzzy control design for nonlinear systems, *Fuzzy Sets and Systems* **48** (1992) 279–287.

[4] U. Itkis, *Control System of Variable Structure* (Wiley, New York, 1976).

[5] C.L. Karr, Genetic algorithms for fuzzy controller, *AI Expert* **6** (Feb. 1991) 26–33.

[6] C.L. Karr, Applying genetics to fuzzy logic, *AI Expert* **6** (Mar. 1991) 38–43.

[7] K. Kristinsson and G.A. Dumont, System identification and control using genetic algorithms, *IEEE Trans. Systems Man Cybernet.* **SMC-22** (1992) 1033–1046.

[8] C.C. Lee, Fuzzy logic in control systems: fuzzy logic controller, Parts I and II, *IEEE Trans. Systems Man Cybernet.* **SMC-20** (1990) 404–435.

[9] S.C. Lin and Y.Y. Chen, Design of adaptive fuzzy sliding mode for nonlinear system control, *Proc. IEEE Internat. Conf. on Fuzzy Systems*, Orlando (1994) 35–39.

[10] S.C. Lin and C.C. Kung, A linguistic fuzzy-sliding mode controller, *Proc. ACC* **3** (1992) 1904–1905.

[11] Y.S. Lu and J.S. Chen, A self-organizing fuzzy sliding-mode controller design for a class of nonlinear servo systems, *IEEE Trans. Industrial Electronics* **41** (1994) 492–502.

[12] E.M. Mamdani, Application of fuzzy algorithms for control of simple dynamic plant, *Proc. IEE* **121** (1974) 1585–1588.

[13] R. Palm, Sliding mode fuzzy control, *Proc. IEEE Internat. Conf. on Fuzzy Systems*, San Diego (1992) 709–715.

[14] J.D. Schaffer, R.A. Caruana, L.J. Eshelman and R. Das, A study of control parameters affecting online performance of genetic algorithms for function optimization, in: J.D. Schaffer, Ed. (1989) 51–60.

[15] J.J.E. Slotine and W. Li, *Applied Nonlinear Control* (Prentice-Hall, Englewood Cliffs, NJ, 1991).

[16] M. Sugeno and M. Nishida, Fuzzy control of model car, *Fuzzy Sets and Systems* **16** (1985) 103–113.

[17] R. Tanscheit and E.M. Scharf, Experiments with the use of a rule-based self-organizing controller for robotic applications, *Fuzzy Sets and Systems* **26** (1988) 195–214.

[18] D.M. Tate and A.E. Smith, Expected allele coverage and the role of mutation in genetic algorithms, *Internat. Conf. on Genetic Algorithms* (1993) 31–37.

[19] V.I. Utkin, *Sliding Modes and Their Application in Variable Structure System* (Mir Press, Moscow, English translation, 1978).

[20] L.A. Zadeh, Outline of a new approach to the analysis of complex systems and decision processes, *IEEE Trans. Systems Man Cybernet.* **SMC-3** (1973) 28–44.