A GA-Based Fuzzy Controller with Sliding Mode

Sinn-Cheng Lin[1], *Student Member, IEEE* and Yung-Yaw Chen[1,2], *Member, IEEE*

[1]Department of Electrical Engineering, Lab. 202, National Taiwan University, Taipei, Taiwan, R.O.C.
[2]Electronic Research Laboratory, University of California, Berkeley, CA 94720, USA

Keywords: fuzzy logic control, fuzzy sliding mode control, genetic algorithm.

Abstract — In this study, the genetic algorithms are applied to find out a nearly optimal fuzzy rule-base for fuzzy sliding mode controller in the sense of fitness. In conventional fuzzy logic controllers (FLC), linearly increasing in either input variables or input linguistic labels would lead the number of rules grow up exponentially. Since the larger size of rule base would cause the longer string length and higher computing load, it becomes one of the difficulties of realizing genetic algorithms to search the suitable rules or membership functions for fuzzy logic controllers. This paper will show that the number of rules in fuzzy sliding mode controller (FSMC) is a linear function of input variables, such that the inferring load of the inference engine in FSMC is more light than that of FLC, and the string length of unknown parameters in FSMC is shorter than that in FLC. Therefore, using genetic algorithms to search fuzzy rules or membership functions for FSMC becomes more economical and applicable. The simulation results verify the efficiency of proposed approach.

I. Introduction

Fuzzy logic controller (FLC) [1] based on approximate reasoning [2] were widely applied to various situation where the plants' models can not be easily developed, but the skilled operators may satisfactorily control them merely follow some experimental statements such as "IF something happens THEN do some actions". By transferring human's expertise into fuzzy IF-THEN rules, one can construct a linguistic FLC to control complex or ill-defined systems. In the last two decades, FLC has become an extensively researched topic. There are various applications confirm the capability of FLC [3]-[5]. However, in FLC design, it still has several well-known difficulties: 1) The fuzzy control rules are experience oriented and the suitable membership functions should be obtained by time-consuming trial and error procedure; 2) The characteristics of control system can not be pre-specified; 3) There is no criterion to get an optimal or at least sub-optimal FLC.

To overcome the problems 1) and 2) discussed above, we had proposed a fuzzy sliding mode control (FSMC) scheme [6] which has the advantages of both FLC and SMC (sliding mode control) [8][9]. By introducing the sliding mode to the FLC and fuzzifying the sliding surface, the FSMC has a fuzzy sliding surface which is similar to a crisp one but more smooth such that the chattering in FSMC is smaller than that in the crisp SMC. In the FSMC, the characteristics of 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 adopted the genetic algorithms (GAs) to search the parameter space of membership functions for FSMC and tried to fine out a nearly optimal rule base which can drive the state to *hit* a pre-defined sliding surface and then do its best to keep the state *sliding* along the surface.

Genetic algorithms were originally developed by Holland in 1962, The detail principle, mathematical framework and applications of GAs can be found in the Goldberg's recent book [10]. In fuzzy control area, GAs have been used to search the string space, which is constituted by parameterizing and encoding the rules or membership functions of FLC into a binary string, to find out the nearly optimal FLC [12][13]. However, this strategy has a main drawback: While input variables or linguistic labels increased, the fuzzy rules will grow up exponentially such that the encoded string length also grow up exponentially. This problem can be avoided by our fuzzy sliding mode control method.

This paper is organized as follows. Section II is a brief description of GAs. In Section III, the fuzzy sliding mode control method is presented. Section IV formulates how to map the design problem of FSMC into string space for search by GAs. In Section V, some simulation results verify the efficiency of proposed ap-

0-7803-2461-7/95/$4.00

proach. Conclusions are given in Section VI.

II. Description of Genetic Algorithms

A. What are GAs ?

GAs are parallel, global search techniques which take the concepts from evolution theory and natural genetics. They emulates biological evolution by means of genetic operators such as reproduction, crossover and mutation *etc.*. GAs work with a set of artificial creatures (string) called population. In every generation, GAs generate a set of offsprings from old population according to a user-defined fitness function. By exchanging the information between every individual, GAs keep the better schemata, which may yield higher fitness, from generation to generation such that the performance can be improved.

B. Why GAs ?

Although there is no given necessary and sufficient conditions under which a function can be optimizable by GAs, it has been shown that GAs behave well on multimodal functions. Moreover, various studies have shown that functions on which GAs failed to get the optimal solution, any other known techniques failed, too [10].

C. How does GAs work ?

There are three basic operators in a simple GA: Reproduction, Crossover, and Mutation. We describe them as follows:

1) *Reproduction:* The Darwinian survival of the fittest is the underlying spirit of reproduction. First, a fitness value (performance index), F, is assigned to each individual string in the population on which higher value of F indicates better fit (or larger benefit). Secondly, the old individual strings are selected and copied into a mating pool according to their fitness value, such that the stings witch own higher fitness than the others have higher probability of contributing 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 resulting members in the mating pool are allowed to mate with others. First, two parents are randomly selected from the mating pool. Secondly, pick up a random crossover point on which the parents will exchange their genes. Finally, mix the parent's genetic codes by crossing their codes following the crossover point. For example, the two parent strings with a crossover point at 5 will get two offsprings

10101<u>010</u> 10101<u>100</u>

-->

01111<u>100</u> 01111<u>010</u>

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

3) *Mutation:* mutation introduce new information into the population. In the binary coding case, mutation operator just changes a bit from 0 to 1 or vice versa.

III. Principles of Fuzzy Sliding Mode Control

A. Some definitions of fuzzy logic control

In this study, the membership functions we adopted are all in the form of Gaussian-like shape, *i.e.*, $\mu_A(x) = \exp[-(\frac{x-m}{\sigma})^2]$, where m is the center of the membership functions where the membership grade is equal to 1, and σ is used to denote the spread of the membership functions. For the convenience of notational simplicity, we express a fuzzy set A as $A(m, \sigma)$ here and hereafter, *e.g.*, the fuzzy set *"Positive Medium"* can be 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, say R_j, in the fuzzy rule base can be expressed as

$$R_j : \textbf{IF } x_1 \textit{ is } A_{1j}(m_{1j}, \sigma_{1j}) \textbf{ and } \ldots \textbf{ and } x_n \textit{ is } A_{nj}(m_{nj}, \sigma_{nj}) \textbf{ THEN } u \textit{ is } B_j(\theta_j, \delta_j), \quad x_i \in X_i, u \in U \tag{1}$$

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

$$R_j : \textbf{IF } \mathbf{x} \textit{ is } \mathbf{A}_j(\mathbf{m}_j, \boldsymbol{\sigma}_j) \textbf{ THEN } u \textit{ is } B_j(\theta_j, \delta_j)$$

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

Definition 2: The *firing strength of the j-th rule* is defined as

$$\mu_{R_j}(\mathbf{x}) = \bigcap_{i=1}^{n} \mu_{A_{ij}}(x_i) \tag{2}$$

where $\cap$ denotes the "*and*" operator such as *min*, *product* or any T-*norm* [1].

Definition 3: A fuzzy rule base, R, is said to be *complete* if and only if there at least exist a rule, say k, in R, such that its firing strength is not equal to zero, that is

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

Definition 4: If a fuzzy rule base R is complete, then the *j-th fuzzy premise function of R*, $p_j(\mathbf{x})$, can be determined by the corresponding premise part of the *j-th* rule R_j, and can be defined as

$$p_j(\mathbf{x}) \triangleq \frac{\mu_{R_j}(\mathbf{x})}{\sum_{j=1}^{N} \mu_{R_j}(\mathbf{x})}, \ j = 1, 2, ..., N \tag{3}$$

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

$$u = \frac{\sum_{j=1}^{N} \theta_j \mu_{R_j}(\mathbf{x})}{\sum_{j=1}^{N} \mu_{R_j}(\mathbf{x})} \triangleq \theta^{\mathrm{T}} \mathbf{p}(\mathbf{x}) \tag{4}$$

where $\mathbf{p} = [p_1 \ p_2 \ ... \ p_N]^{\mathrm{T}}$ is a vector of fuzzy premise function; $\theta = [\theta_1 \ \theta_2 \ ... \ \theta_N]^{\mathrm{T}}$ in which θ_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

i). All fuzzy premise functions, $p_j(x)$, are well-defined, and

ii). $0 \leq p_j(x) \leq 1$, such that the fuzzy premise functions can be viewed as *normalized firing strength.*

In the FLC design procedure, according to the above fact, the most important thing is to construct a complete fuzzy rule base such that all fuzzy premise functions are well-defined. There are two criteria to obtain a complete fuzzy rule base:

1) For each individual input variable, overlapping the membership functions of its corresponding linguistic labels;

2) Take all possible combinations between every input linguistic labels to form the premise part. However, the cost we need to pay is that the amount of rules will be grown up exponentially.

FACT 2: If a fuzzy logic controller has n input variables. Moreover, for each input variable x_i, there are l_i corresponding linguistic labels defined on X_i, then the total number of rules in a complete rule base is given by $N = \prod_{i=1}^{n} l_i$.

B. Fuzzy sliding mode control

In conventional SMC design, the designer must choose a sliding function first, *i.e.*

$$s(\mathbf{x}) : R^n \rightarrow R \tag{5}$$

for example, $s(.)$ can be selected as a linear combination of state, $s = \mathbf{c}^{\mathrm{T}}\mathbf{x}$. In this paper, we will adopt the linear sliding function for simplicity. The sliding surface can be viewed as a set of states, in which, the mapping (5) is equal to zero, *i.e.*

$$S \triangleq \{\mathbf{x} \mid s(\mathbf{x}) = 0\} \tag{6}$$

Fig. 1 shows an example of 2-dimensional state plane, the sliding surface in this case is a straight line goes through the equivalent point (0, 0). The state plane has been partitioned into two parts by the sliding line, one is $s > 0$, and the other one is $s < 0$. The sliding line also called as the switching line, that because the control action would be switched at the opposite sides of the line. That is why the sliding mode control also known as the variable structure control (VSC). Such a switching operation has many drawbacks, the one of them is chattering phenomenon which is due to the presentations of the system nonideality, such as hysteresis, delay, sampling, uncertainty ... *etc.* To reduce the system nonideality effects, we had applied the fuzzification operation to convert the crisp sliding surface into a fuzzy one [8]. This is the main idea of FSMC. Fig. 1(b) illustrate such concept. In this example, there are five linguistic labels, *PB*, *PS*, *ZE*, *NS*, and *NB*, are assigned to the sliding variable s. The crisp function value of s can be viewed as the *distance* from a state point, $\mathbf{x}$, to the sliding surface, $s = 0$. Once the sliding variable be fuzzified, the complete rule base for fuzzy sliding mode control may look like

$$R_j: \textbf{IF } s \textit{ is } S_j \textbf{ THEN } u \textit{ is } U_j \tag{7}$$

where $\{S_j \mid j = 1, 2, ..., 5\} = \{PB, PS, ZE, NS, NB\}$; U_j are unknown fuzzy sets that should be determined by some strategies such as experienced [6], adaptive [7], and learning methods. In the next section, the parameters of consequence fuzzy sets, U_j, are self-generated by means of GAs. The fitness function is defined in such a way that the controller which can drive the state to *hit* the sliding surface faster than others and then keep the state *slide* along the surface with less chattering yields higher fitness.

IV. Genetic Algorithms for Fuzzy Sliding Mode Control

A. General description

A class of *n-th* order dynamic nonlinear systems considered in this paper have the form of

$$y^{(n)} = f(y, \dot{y}, ..., y^{(n-1)}) + bu, \quad b > 0 \tag{8}$$

where $f(.)$ is an unknown continuous function with known upper bound, $|f| \le f_{max}$; b is an unknown positive constant that bounded below by a positive value b_{min}, *i.e.* $0 < b_{min} \le b$; $u \in R$ is the system input and $y \in R$ is the system output [15]. To transfer this differential equation to a state space realization, let r be the reference input, and let $e = y - r$ be the error signal. Define the state $x_i = y^{(i-1)} - r^{(i-1)} \triangleq e^{(i-1)}$, $i=1, 2, ..., n$, then the system (8) can be represented as the following state equations

$$\begin{cases} \dot{x}_i = x_{i+1}, & i = 1, 2, ..., n-1 \quad (9a) \\ \dot{x}_n = f + bu - r^{(n)} & (9b) \end{cases}$$

To get a sliding mode control, it is necessary to choose a sliding surface firstly. Let's define the sliding function

$$s = \mathbf{c}^{\mathrm{T}}\mathbf{x} \quad = \sum_{i=1}^{n} c_i x_i \tag{10}$$

where $\mathbf{e} = [x_1\ x_2\ ...\ x_n]^{\mathrm{T}}$ is the state vector and $\mathbf{c} = [c_1\ c_2\ ...\ c_{n-1}\ c_n]^{\mathrm{T}}$ is the sliding surface coefficient vector which has to be properly selected by the designer. Without loss of generality, let $c_n = 1$, then

$$\dot{s} = \bar{\mathbf{c}}^{\mathrm{T}}\mathbf{x} + [f + bu - r^{(n)}] \tag{11}$$

where $\bar{\mathbf{c}} = [0\ c_1\ c_2 \cdots c_{n-1}]^{\mathrm{T}}$. In design of the sliding mode control system, one has to find the equivalent control law, u_{eq}, which will keep the system state staying on the sliding surface whenever $\mathbf{x} \in S$. So, u_{eq} could be derived from setting (11) equals to zero, that is

$$u_{eq} = u\big|_{\dot{s}=0} \doteq -b^{-1}(f - r^{(n)} + \bar{\mathbf{c}}^{\mathrm{T}}\mathbf{x}) \tag{12}$$

Moreover, in the sliding mode, $s = 0$, we obtain an equivalent control system

$$c_1 e + c_2 \dot{e} + ... + e^{(n-1)} = 0 \tag{13}$$

thus, the system dynamic can be characterized by the following characteristic polynomial

$$p^{n-1} + c_{n-1}p^{n-2} + ... + c_1 = 0 \tag{14}$$

where $p \triangleq \frac{d}{dt}$ is the Laplace operator. With suitable choice of the coefficients c_i, one can obtain a stabilized control system if and only if (14) is Herwitz. So, *the dynamic behavior of sliding mode control system can be dominated by the predefined sliding surface.*

However, without knowing f and b, there is no way to get an exactly u_{eq}. In this study, the control law is in the form of

$$u = u_f + \alpha u_h \tag{15}$$

where $u_f = \boldsymbol{\theta}^{\mathrm{T}}\mathbf{p}(s)$ is called fuzzy control part and obtained from a complete fuzzy rule base, R, with rule format of "**IF** s *is* ... **THEN** u_f *is* ..."; u_h is a hitting control part that guarantees the stability of control system, and

$$\alpha = \begin{cases} 1, \text{for } |s| \ge s_0 \\ 0, \text{for } |s| \le s_0 \end{cases} \tag{16}$$

is a switching constant used to enable/disable u_h. In which s_0 is a user-specified bound of s. The main purpose of this work is applied GAs to search parameter

space Θ, and find a suitable parameter vector θ such that the performance index is minimal.

B. Fuzzy control part u_f

To improve the performance of FSMC, the GA with elitist model [10] was adopted in this paper. That is, if the best individual generated up to time t, is not in the population of new generation $t+1$, then replace one new member by the elitist. Moreover, as [14] pointed out that in many more complex optimization problems, the low mutation implementations are not always ideal. Hence, a higher mutation rate was used in this work to avoid the solution trapped in some local optimum. The procedure of applying GAs to obtain the fuzzy control part of FSMC, u_f, is summarized as follows.

STEP 1: Define a sliding function $s = \mathbf{c}^T\mathbf{x}$ by carefully selecting the parameter vector $\mathbf{c}$, such that Eq. (14) is stable and satisfies the desired behavior.

STEP 2: Fuzzify the sliding variable, s, by defining a number of Gaussian-like membership functions such as $PB(3, 1.2)$, $PS(1.5, 1)$, ... *etc.* Make sure that each two membership functions have overlapping.

STEP 3: Constructing an initial fuzzy rule base with complete premise part and random consequence, *e.g.*

R_1: **IF** s *is* $PB(3, 1.2)$ **THEN** u_f *is* $U_1(\theta_1, \delta_1)$

R_2: **IF** s *is* $NB(-3, 1.2)$ **THEN** u_f *is* $U_2(\theta_2, \delta_2)$

.....

where U_j are unknown linguistic labels for control output u_f ; θ_j are the center of U_j given by random number generator, they will lately be coding into some parameter space which can be searched by GAs. δ_j denote the spread of U_j, designer can randomly selecte any positive constants form them. Then, the fuzzy control part, u_f, can be obtained by the Eq. (4).

Remark: Since the defuzzification method adopted in this paper is weighted-average, the values of δ_j are never used, such that searching suitable δ_j is not necessary in this time. However, in the center-of-area defuzzification, they are necessary for computing the area of U_j. The ways of finding suitable δ_j and their effects are left as future reach topics.

STEP 4: Encoding θ_j as d-bit binary strings

$$b_j^1 b_j^2 \cdots b_j^d \tag{17}$$

Cascading θ_j into a row vector, θ

$$\theta = [\theta_1\ \theta_2\ \ldots\ \theta_N] \tag{18}$$

then, form an individual string by cascading N n-bit binary strings into a single string

$$P_i = b_1^1 b_1^2 \ldots b_1^d \mid b_2^1 b_2^2 \ldots b_2^d \mid \cdots \mid b_N^1 b_N^2 \ldots b_N^d \tag{19}$$

where the vertical lines are added to separate the parameter strings for clear notation.

STEP 5: Repeat STEP 3 and STEP 4 for M times to establish an initial population

$$\boldsymbol{P} = \{P_1, P_2, \ldots, P_M\} \tag{20}$$

where M is population size. Each individual, P_i, in the population, $\boldsymbol{P}$,corresponds to an FSMC, denoted as FSMC_i.

STEP 6: Define a performance index, J, for example

$$J = \sum_{k=1}^{K} \left(w|s(k)| + v|u_f(k)| \right) \tag{21}$$

where $k = \text{int}(t/\Delta t)$ denotes the iteration instance, Δt is sampling period, int(.) is a round-off operator, $K = \text{int}(t_{max}/\Delta t)$ is the iteration times in one run, t_{max} denotes the running-period in one run; w and v are positive weights. If one want to drive the state to quickly hit the sliding surface no matter how large control force is, the value of w would be selected larger than that of v. Otherwise, consuming too more control energe will be punish by heavy weighting in u_f (select larger value of v). Then the fitness function can be defined as

$$F = 1 / (J + \varepsilon_0) \tag{22}$$

where ε_0 is any small positive constant used to avoid the numerical error of divided by zero.

STEP 7: Apply GAs (Reproduction, Crossover, Mutation) to the old population $\boldsymbol{P}$ for the purpose of maximizing the fitness function described in STEP 6.

STEP 8: Generate a set of offsprings and used them to form a new population $\boldsymbol{P'}$, except replacing an indi-

vidual, which is selected randomly, of the offsprings by the elite individual of the old population.

STEP 9: Replacing the old population $\boldsymbol{P}$ by the new population $\boldsymbol{P}'$.

STEP 10: Decoding every new individual P_i back to the original parameter sapce and get a set of new parameter $\boldsymbol{\theta}_i$. Note that i = 1, 2, ..., M. So, we have M new FSMC's generated by GA.

STEP 11: Using $\boldsymbol{\theta}_i$ to obtain the fuzzy control part of FSMC$_i$ by Eq. (4), i = 1, ..., M, and apply control law (15) to the plant, where the hitting control part u_h is derived in the next paragraph.

STEP 12: Evaluate the performance index and fitness function of every FSMC by Eqs. (21) and (22), respectively.

STEP 13: Repeat STEP 7 to STEP 11 until the performance of the best individual satisfies the desired specification.

STEP 14: Take the best individual as an appropriate fuzzy sliding mode controller.

FACT 3: Without loss of generality, assume that for each input of the fuzzy system, the number of input linguistic labels are all equal to L. According to FACT 2, the number of rules in a conventional FLC with complete rule base (1) is $N = L^n$. However, by combining the states into a single variable s, then the number of rules in an FSMC with complete rule base (7) is given by $N = L$.

C. Hitting control part u_h

To draw the state to *hit* the sliding surface no matter where the initial state is, a hitting control law should be applied. Let's define a Lyapunov function

$$V = \frac{1}{2}s^2 \tag{23}$$

the *hitting* condition [15] which guarantees the stability of sliding mode control system is

$$\dot{V} = s\dot{s} < -\eta|s| \tag{24}$$

Suppose that we know the upper bound of $f(.)$ and the lower bound of b, *i.e.* $|f| \le f_{max}$, $0 < b_{min} \le b$, substitute (11) into (19), we have

$$\dot{V} \le |sb|\left[\left|b^{-1}\left(f - r^{(n)} + \bar{\mathbf{c}}^{\mathrm{T}}\mathbf{x}\right)\right| + \left|u_f\right|\right] + sbu_h \tag{25}$$

To guarantee (20) less then or equal to $-\eta|s|$, substitute (12) into (20), the hitting control law must be selected as

$$u_h = -sign(s)\left[b_{\min}^{-1}\left(f_{\max} + \left|r^{(n)}\right| + \left|\bar{\mathbf{c}}^{\mathrm{T}}\mathbf{x}\right| + \eta\right) + \left|u_f\right|\right] \tag{26}$$

Substitute (21) into (20), one can get $\dot{V} \le -\eta|s|$, and the hitting condition can be satisfied, therefore the system is stable. From the control law (15), one can see that the hitting part, u_h, play a role like a watchdog. That is, when the state lies in the boundary layer, s_0, the hitting controller is disable; On the other hand, as soon as the state goes outside the layer, the hitting controller becomes active for driving the state back toward the layer.

IV. Results and Discussions

We now look at how the GAs can be used in a fuzzy sliding mode controller. Consider the plant to be controlled is a second order nonlinear system:

$$m\ddot{y} + \beta(y^2 - 1)\dot{y} + ky = u \tag{27}$$

where m, c and k are positive constants. The unforced system, *i.e.* u = 0, with m = 1 and k = 1 is the famous Ven der Pol oscillator [15]. If we define two states $x_1 = y$, and $x_2 = dy / dt$, then the state trajectory of unforced system with β = 1 is shown in Figure 2. One can see there exist a limit cycle. The control problem is applying the control signal, u, to drive the state vector $\mathbf{x}$ going toward the sliding surface, and slide along it such that x_1 and x_2 approximate to zero as t goes to infinity.

In this simulation, the sliding function is selected as $s = x_1 + x_2$, the initial condition are $x_1(0) = 2$, $x_2(0) = 2$. The prototype of fuzzy control rule base has 6 rules, and defined as follows.

$$R_j\text{: IF } s \text{ is } S_j(m_j, \sigma_j) \text{ THEN } u_f \text{ is } U_j(\theta_j, \delta_j) \tag{28}$$

for j = 1, 2, ..., 6. Where $\{S_j(m_j,\sigma_j) \mid j = 1,2, ...,6\}$ = {*NB*(-3, 1.2), *NS*(-1.5, 1), *NZ*(-.5, 1), *PZ*(.5, 1), *PS*(1.5, 1), *PB*(3, 1.2)}, the parameters m_j and σ_j are selected by designer in such a way that the premise part of (28) is complete.

Eq. (27) can be regarded as describing a mass-spring-damper system with a position-dependent damping coifficient $\beta(y^2-1)$, or, equivalently, an RLC electri-

cal circuit with a nonlinear resistor. Suppose that the output amplitude of fuzzy controller, u_f, is between [-30, 30], it is reasonably that assume the universe of discourse of θ_j be within the interval [-30, 30]. Before applying GAs, the parameters of consequence part, θ_j, were encoded into a 4-bit binary string. Therefore, cascading 6 strings (θ_1, ..., θ_6) forms a 24-bits individual. Selecting the population size $N = 10$, the initial population was generated by random number generator. Fig. 2 shows the responses of the best and worst individuals in the initial population, respectively. One can see that neither of them achieve the control goal. To view how GAs work, we define two types of cost functions as follows.

CASE I: According to Eq. (21), select w =1, v = 0.5, that is

$$J = \sum_k \left(|s(k)| + 0.5|u(k)| \right) \tag{29}$$

where $k = \text{int}(t/\Delta t)$ is the iteration instance, and $\Delta t = 5$ *ms* is the sampling period. In this case, the weight of $|s(k)|$, w, is larger than that of $|u(k)|$, v, such that the goal of driving the state to hit the surface as soon as possible and then keeping sliding it is more important than that of reserving the control energe. Fig. 4 shows the state response and the control signal of the best individual in the generation 50. It's the final FSMC we want. The output labels of this FSMC is shown in Fig. 6.

CASE II: Take $w = T$ and $v = 2$, the performanc index becomes

$$J = \sum_k \left(T|s(k)| + 2|u(k)| \right) \tag{30}$$

where $T = k \cdot \Delta t$. Fig. 5 shows the state response and control signal of the best individuals in generation 50. In this case, while T is small, the control signal, u, was weighted by a larger constant than that sliding function, s, such that the controller which can reserve more control energe will get higher fitness; However, as time progress, T is more and more large, such that, hitting and sliding become more and more important than energe saving, such that the controller which can pull the state more near the sliding surface will get the award. In conclusion, the controller which can not only quickly drive the state toward the surface but also without dissipating too much control energy will yield higher fitness. The best individual in generation 50 is the final FSMC we want. The output labels of this FSMC is shown in Fig. 7.

Comparing Fig. 5 with Fig. 4, one can see that the control signal of final FSMC in CASE II, is smaller than that in CASE I, but the hitting speed in CASE II is slower than that in CASE I. Therefore, before applying GA to generate rules or membership functions for FSMC, the designer have to define a suitable performance index (fitness function) by carefully selecting the weights w and v, based on the purposes of minimizing the sliding function or control energe.

VI. Conclusions

In this study, one can see that the GA indeed behave as an efficient searching tool for finding an nearly optimal fuzzy sliding mode controller. The performance of final control system highly depends on the user-defined sliding surface and the fitness (cost) functions. One of advantages of introducing the sliding mode into a fuzzy control system is that the number of control rules could be reduced to a linear function of the number of input variables, and then the corresponding string length of each individual in this GA-based FSMC is shorter than that in the conventional FLC.

References

[1] C. C. Lee, "Fuzzy logic in control systems: Fuzzy logic controller, parts I and II," *IEEE Trans. Syst., Man, Cybern.*, vol. 20, no. 2, pp. 404-435, 1990.

[2] L. A. Zadeh, "Outline of a new approach to the analysis of complex systems and decision processes," *IEEE Trans. on Syst. Man and Cyber.*, vol. SMC-3, no. 1, pp.28-44, 1973.

[3] E. M. Mamdani, "Application of fuzzy algorithms for control of simple dynamic plant," *Proc. IEE*, vol. 121, no. 12, pp. 1585-1588, 1974.

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

[5] M. Sugeno and M. Nishida, "Fuzzy control of model car," *Fuzzy Sets and Systems*, vol. 16, pp. 103-113, 1985.

[6] S. C. Lin and C. C. Kung, "A linguistic fuzzy-sliding mode controller," *Proc. of 1992 A.C.C.*, pp. 1904-1905.

[7] S. C. Lin and Y. Y. Chen, "Design of adaptive

fuzzy sliding mode for nonlinear system control," *3rd IEEE International Conference on Fuzzy Systems*, Orlando, 1994, pp. 35-39.

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

[9] U. Itkis, *Control system of variable structure.* New York: Wiley, 1976.

[10] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine learning*, Addison-Wesley, 1989.

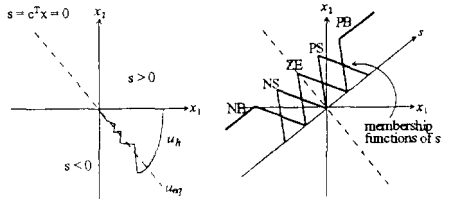
[11] K. Kristinsson and G. A. Dumont, "System identification and control using genetic algorithms," *IEEE Trans. on Syst. Man and Cyber.*, vol. 22, no. 5, Sep./Oct. 1992, pp.1033-145.

[12] C. L. Karr, "Genetic algorithms for fuzzy controller," *AI Expert*, Feb. 1991, pp.26-33.

[13] C. L. Karr, "Applying genetics to fuzzy logic," *AI Expert*, Mar. 1991, pp.38-43.

[14] D. M. Tate and A. E. Smith, "Expected allele coverage and the role of mutation in genetic algorithms", *International Conference on Genetic Algorithms*, 1993, pp.31-37.

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



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

Fig. 1 The basic concept of fuzzy sliding mode

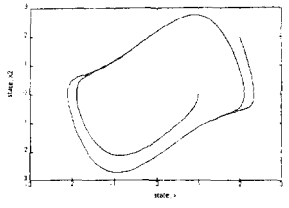


Fig. 2 The unforced system behaves as a Ven der Pol oscillator

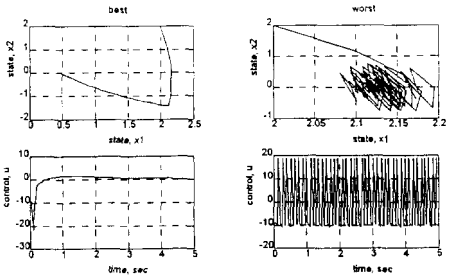


Fig. 3 The responses of the best and the worst individuals in the initial population

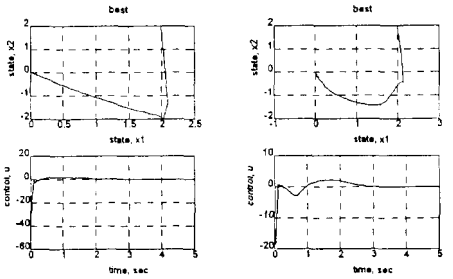


Fig. 4 CASE I

Fig. 5 CASE II

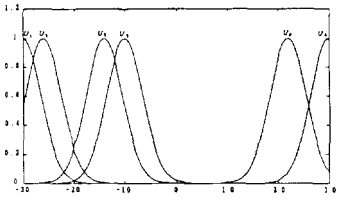


Fig. 6 The final output labels for FSMC in CASE I

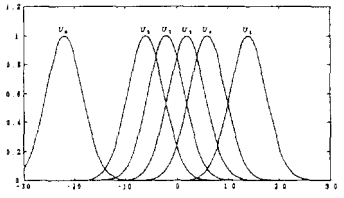


Fig. 7 The final output labels for FSMC in CASE II