A method of Self-Generating Fuzzy Rule Base via Genetic Algorithm

Wen-June Wang, Tzu-Gaun Yen and **Chung-Hsun Sun**
Department of Electrical Engineering,
National Central University,
Chung-Li, 320, Taiwan, ROC
e-mail:wjwang@ee.ncu.edu.tw

Abstract

It is known that the fuzzy control rules for a control system is always built by designers with trial and error and based on their experience or some experiments. This paper introduces a Genetic Algorithm (GA) based method to generate a satisfactory fuzzy rule base spontaneously. With the specific structure of the chromosome, the special mutation operation and the adequate fitness function, the proposed method with GA produces a fuzzy rule base with small number of rules, suitable placement of the premise's fuzzy sets and proper location of the consequent singletons. The generated fuzzy rule base can be the controller in a closed loop system to achieve some control objective or can be a fuzzy model to approximate an unknown nonlinear system. Finally, two examples are illustrated to show the effectiveness of the proposed method on the fuzzy control design and fuzzy modeling respectively.

Keywords: Genetic algorithms, fuzzy controller, fuzzy model.

1. Introduction

In recent years, the fuzzy logic has been applied successfully in several of fields, such as image processing [1], VLSI design [2], power system, industry's control [3], *etc*. The fuzzy control has become an effective method in industry applications because it has the ability to solve difficult nonlinear control problems or it works without the extract model of the controlled plant. Trial-and-error always exists in building a satisfactory fuzzy rule base for controlling a nonlinear system or an un-modeled system. Designers usually cannot guarantee that the fuzzy control system designed with trial-and-error has a good performance. Furthermore, there has been a lot of literature [4-17] discussing about the fuzzy modeling for approaching an unknown system. However, their modeling methods are always complicated and/or hard to be implemented.

To avoid trial-and-error method and/or complex calculation, a number of papers have proposed some kinds of methods to build the fuzzy rule base by using GA [4-10], clustering method [11], or neural networks [12-15]. Obviously, GA has successfully been used in searching proper fuzzy rule bases with an assumed structure. But it is not well suited for evolving fuzzy rule base without any prior assumption. The reason is that two different structure rules will be inefficient in the crossover procedure of GA. In this case, some authors eliminate the crossover operation and only use the mutation of GA to design the fuzzy rule base for modeling and control [16]. Still, the consequents are determined by least square parameter estimation method [17]. In addition, [18] develops a single value decomposition (SVD) method to reduce the fuzzy rules.

This paper proposes a new and efficient GA based approach to construct a fuzzy rules base. In the method, we do not need to initialize the rules' number, the positions of the premise and the consequent fuzzy sets in the beginning of GA. The only we need is the setting of the length and the structure of the chromosome. With the specific structure of the chromosome, the special mutation operation and the adequate fitness function, GA will produces the fuzzy rule base spontaneously which has the small number of rules, arranges the suitable placement of the premise's fuzzy sets and choose the proper location of the consequent singletons. The generated fuzzy rule base can be the fuzzy controller of a closed loop system or be the fuzzy model to approaching an unknown system.

This paper is organized as follows. Problem formulation and some preliminary are stated in section 2. Generation of the fuzzy controller presented in section 3. Section 4 describes the work of fuzzy model generation. Simulation results and conclusion are given in section 5 and 6, respectively.

2. Problem formulation and some preliminary

In this paper, we are going to introduce a new GA-based method to generate a satisfactory fuzzy rule base to control a nonlinear system. This method can be also applied to the work of fuzzy modeling to approach a nonlinear unknown system. First, we will present the form of the fuzzy rule base which will be generated. Each rule of a complete fuzzy rule base with *m* inputs and *n* outputs is represented as follows:

$R(j_1, j_2, ..., j_m)$:
If x_1 is $A_{(1, j_1)}$ and x_2 is $A_{(2, j_2)}$ ... and x_m is $A_{(m, j_m)}$
Then y_1 is $B_1(j_1, j_2, ..., j_m)$
and y_2 is $B_2(j_1, j_2, ..., j_m)$ (1)
.....
and y_n is $B_n(j_1, j_2, ..., j_m)$
$j_i \in \{1,2,...,k\}, i \in \{1,2,...,m\}$.

where $R(j_1, j_2,..., j_m)$ is the $(j_1, j_2,..., j_m)-$th rule, x_is stand for the input variables, y_is are the output variables, $A_{(i, j_i)}$ is the j_i-th fuzzy sets of input variable x_i, and $B_i(j_1, j_2, ..., j_m)$ is the $(j_1, j_2, ..., j_m)-$th rule singleton fuzzy set of the output variable y_i. The membership function of the fuzzy set $A_{(i,h)}$ is a triangular shape described as (2)

$$A_{(i,h)}(c_{(i,h)}; x_i) = \begin{cases} \dfrac{(c_{(i,h)} - c_{(i,l)}) - (c_{(i,h)} - x_i)}{c_{(i,h)} - c_{(i,l)}}, & c_{(i,l)} \le x_i \le c_{(i,h)} \\ \dfrac{(c_{(i,r)} - c_{(i,h)}) - (x_i - c_{(i,h)})}{c_{(i,r)} - c_{(i,h)}}, & c_{(i,h)} \le x_i \le c_{(i,r)} \\ 0, \quad \text{others} \end{cases} \quad (2)$$

where $c_{(i,h)}$ is the center of fuzzy set $A_{(i,h)}$, h, l, and r are some integers belonging to j_i. Hence, the shape of the membership function associated with the fuzzy set $A_{(i,k)}$ is determined by only one parameter $c_{(i,h)}$. In this paper, each terminal of a fuzzy set is a center of the adjacent one, in other words, two adjacent fuzzy sets are overlapping in a half as shown in Fig.1. It is seen from j_i of (1) that there are k fuzzy sets $A_{(i, j_i)}$ for the premise variable x_i. Except the beginning and final points of the universal set, the other $(k-2)$ points are set to be the centers of the other $(k-2)$ fuzzy sets, respectively.

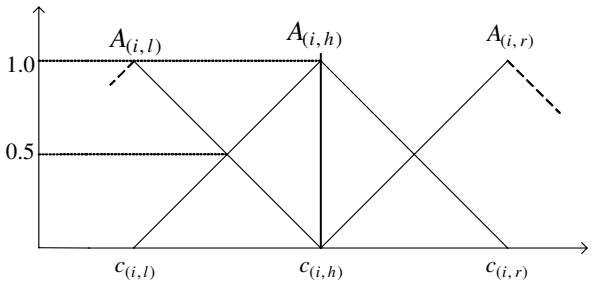


Fig. 1. Fuzzy sets $A_{(i,h)}$, $A_{(i,r)}$ and $A_{(i,l)}$.

For a rule base, a product inference engine and any defuzzification are chosen. We take the weighted average defuzzification to calculate the inference output value y as follows.

$$y = \frac{\sum_{j_1=1}^{k} \sum_{j_2=1}^{k} ... \sum_{j_i=1}^{k} \omega(j_1, j_2, ..., j_i) \cdot B(j_1, j_2 ,..., j_i)}{\sum_{j_1=1}^{k} \sum_{j_2=1}^{k} ... \sum_{j_i=1}^{k} \omega(j_1, j_2, ..., j_i)} \quad (3)$$

where $\omega(j_1, j_2, ..., j_i) = \prod_{i=1}^{q} A_{(i, j_i)}(x_i)$ (4)

The main task of this paper is as follows. Finding a GA-based fuzzy rule base (1) as the controller of a closed loop control system (as Fig. 2) to achieve some desired control objective. In Fig. 2, u is the output y of (3). By the way, applying the same method to build a fuzzy model for a nonlinear unknown system will be considered also. By using the given training data, we can establish a fuzzy model as (3) to approach the behavior of an unknown system.

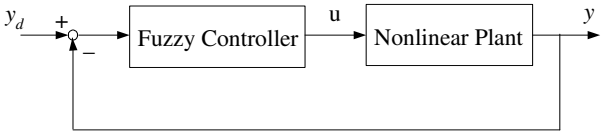


Fig. 2. A closed loop system with fuzzy controller.

In the method, GA plays a key role to search for the optimal parameters. These parameters include the number of fuzzy rules, the locations of all centers ($c_{(i, j_i)}$) (i.e., all membership function $A_{(i, j_i)}$) and the output of the consequent $B_h(j_1, j_2, ..., j_i)$ such that the controlled system or the fuzzy modeling has a satisfied performance.

3. Generation of the fuzzy controller

In this section, we focus on the generation problem of fuzzy controller. A fuzzy system with two inputs and one output will be illustrated to explain the control generation process.

3.1 Coding for the premise variable

Since GA is the main tool in this paper for establishing the fuzzy control rule base. First, the chromosome coding should be defined. We set two code strings for each x_i. The upper string is consisting of $d_{(i,j_i)}$s (called d-string) which binary codes are denoting the index of $c_{(i,j_i)}$s ; the lower string is consist of $c_{(i,j_i)}$s (called c-string) which are real values denoting the center location of $A_{(i,j_i)}$. When $d_{(i,j_i)}=1$ means the usefulness of $c_{(i,j_i)}$, but $d_{(i,j_i)}=0$ means $c_{(i,j_i)}$ is useless. It is noted that $c_{(i,j_i)}$ is the center of $A_{(i,j_i)}$, if $c_{(i,j_i)}$ is useless, $A_{(i,j_i)}$ is useless either. For example, suppose input x_1 has seven fuzzy sets represented in Fig. 3(a). Assume, { $d_{(1,1)}=1$, $d_{(1,2)}=1$, $d_{(1,3)}=0$, $d_{(1,4)}=1$, $d_{(1,5)}=0$, $d_{(1,6)}=1$, $d_{(1,7)}=1$ }, and { $c_{(1,1)}=1.0$, $c_{(1,2)}=5.937$, $c_{(1,3)}=0$, $c_{(1,4)}=1.875$, $c_{(1,5)}=0$, $c_{(1,6)}=3.812$, $c_{(1,7)}=7.0$ }. Fig. 3(b) shows the fuzzy sets of x_1 realized from the code strings of Fig. 3(a). It is seen that all fuzzy sets exist (useful) except $A_{(1,3)}$ and $A_{(1,5)}$, and these existing (useful) fuzzy sets locate with the center at positions $c_{(i,j_i)}$. The similar setting can be presented for x_2 to set the fuzzy sets $A_{(2,j_2)}$.

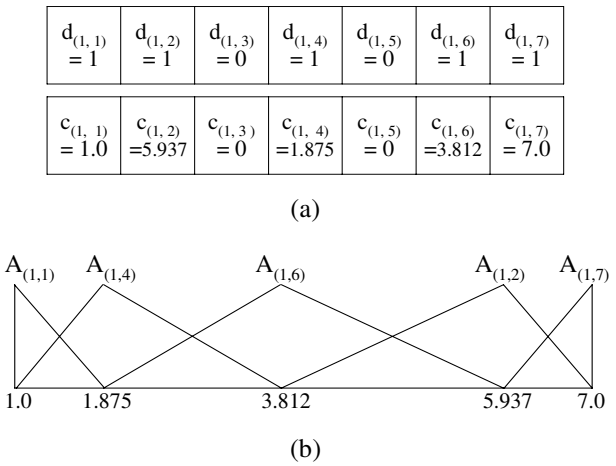


Fig. 3. (a) Two code strings of x_1 . (b) The realization of $A_{(1,j_1)}$ by the codes in (a).

3.2 Determination of the singleton output

For convenient calculation, the consequent of each rule in (1) is set as a singleton; therefore, the location of each singleton should be determined. The number of singletons depends on the number of rules. The rule number is associated with the status of strings $d_{(i,j_i)}$ and $c_{(i,j_i)}$. For instance, suppose each input x_i has seven $d_{(i,j_i)}$ s, two inputs at most have $7\times 7=49$ fuzzy rules; therefore there are 49 corresponding consequent singletons. Fig. 4 shows the code string determined by two code strings $d_{(1,j_1)}$ and $d_{(2,j_2)}$, where $j_i=1,2,...,7.$ The consequent $B(j_1,j_2)$ is a real value denoting the position of singleton which depends on the value of $d_{(1,j_1)}$ and $d_{(2,j_2)}$. If any $d_{(i,j_i)}=0$ ($i=1$ or 2), the corresponding output, that is $B(j_1,j_2)$, loses its efficacy (i.e., it is useless and is ignored). That is, the rule $R(j_1,j_2)$ is useless. Only both $d_{(1,j_1)}=1$ and $d_{(2,j_2)}=1$, $B(j_1,j_2)$ will be useful and the value will be determined by GA. In other words, only on the case that both $d_{(1,j_1)}=1$ and $d_{(2,j_2)}=1$, $B(j_1,j_2)$ will be changed in the GA's generation alternating; otherwise, the value $B(j_1,j_2)$ will be remained (be ignored).

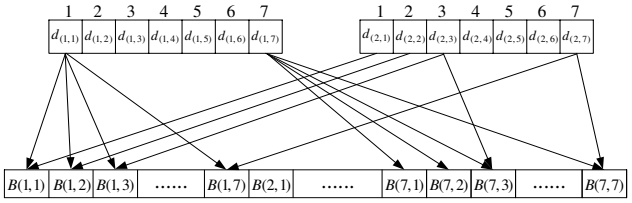


Fig. 4. The code string of consequent determined by two code strings $d_{(1,j_1)}$ and $d_{(2,j_2)}$.

After setting the premise and consequent codes, a complete individual chromosome is produced. A complete individual chromosome for the illustrated fuzzy system is represented in Fig. 5. The upper string is the d-string and the lower string is the combination of c-string and consequent B-string.

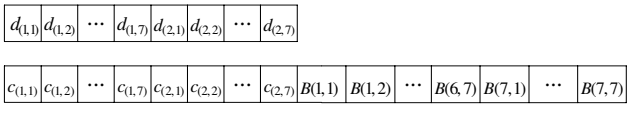


Fig. 5. A complete chromosome.

3.3 GA operations

Generally, there are three genetic operations in GA: reproduction, crossover, and mutation. The three operators offer the genes a fine searching mechanism. First, we initialize randomly P individuals with real numbers on $c_{(i,\,j_i)}$ and $B(j_1, j_2, ..., j_m)$ and binary codes on $d_{(i,\,j_i)}$. The reproduction here includes two strategies, one is the elitist strategy and the other is roulette wheel strategy. The elitist strategy [19,20] is applied to ensure the survival of the best two strings in each generation. The other better offspring are produced by the conventional roulette wheel principle. Then the populations with the highest fitness values will be copied to the next generation. To do this we set the generation gap less than one.

Next, a single crossover point is chosen with uniform probability on any position for a pair of chromosomes and then the genes behind the crossover point are exchanged each other. Fig. 6 shows that the genes behind the crossover point exchanges between parent 1 and parent 2 to generate two new offspring 1 and offspring 2.

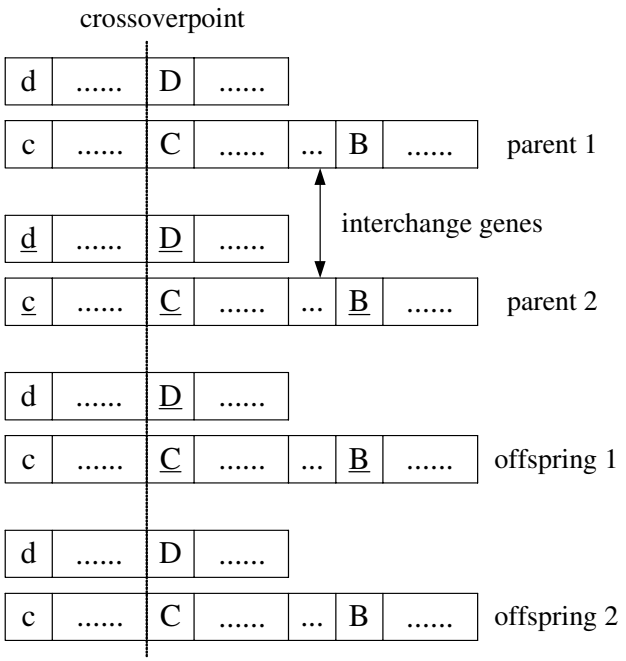


Fig. 6. One-point crossover operation.

After the crossover, mutation will be followed. However it should be noted that there are two strings (upper and lower) in each chromosome (see Fig. 5). If we mutate the lower front string (c-string) early and mutate the upper string (d-string) lately the realization of the membership function $A_{(i,j_i)}$ will be confusing. In order to avoid this confusion, a hierarchical mutation is introduced. That is, we first mutate d-string and then mutate c-string and B-string. The mutation of $d_{(i,\,j_i)}$ will influence the usefulness of $c_{(i,\,j_i)}$ and then the value of $B(j_1, j_2)$ will be varied, too. For instance, in Fig. 7(a), $d'_{(1,\,j_1)}$ or $d'_{(2,\,j_2)}$ is the code string after mutation from $d_{(1,\,j_1)}$ or $d_{(2,\,j_2)}$, respectively. ($c_{(1,\,j_1)}$ or $c_{(2,\,j_2)}$) and ($c'_{(1,\,j_1)}$ or $c'_{(2,\,j_2)}$) depend on the original string ($d_{(1,\,j_1)}$ or $d_{(2,\,j_2)}$) and mutated string ($d'_{(1,\,j_1)}$ or $d'_{(2,\,j_2)}$), respectively. In Fig. 7(b), $B(j_1, j_2)$ and $B'(j_1, j_2)$ are the strings associated with original string pair $\{ d_{(1,\,j_1)}, d_{(2,\,j_2)} \}$ and mutated string $\{ d'_{(1,\,j_1)}, d'_{(2,\,j_2)} \}$, respectively. It should be noted that in order to fix the number of fired rules and avoid decreasing the searching space, the beginning point and the final point are avoided in mutation operation. The above mutation process is called a "Hierarchical mutation".

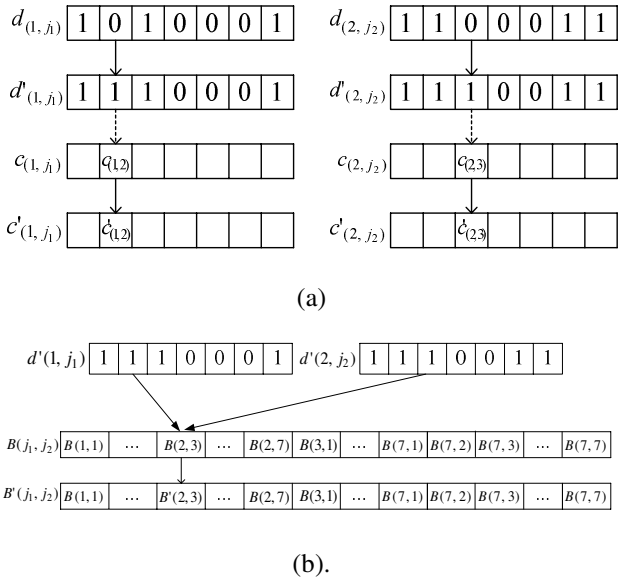


Fig. 7. Hierarchical mutation. (a) Mutation between $d_{(i,j_i)}$ and $c_{(i,j_i)}$, $i \in \{1, 2\}$. (b) Before and after mutation on $B(j_1, j_2)$

After introducing the above operations in GA, the remaining most important thing is the setting of the "Fitness function". The fitness function is used to evaluate the performance of each chromosome in a generation. It is set as below:

$$Fit = \frac{1}{(w_p \cdot \|e_i\|_2 + w_r \cdot R^i)}, \quad w_p + w_r = 1 \tag{5}$$

where $e_i = y_i - y_d$ is the output error. y_i is the output of the closed loop system with the controller generated by the $i-$th individual chromosome. It is noted that each chromosome represents one fuzzy rule base. y_d is the desired output and R^i is the rules number of the $i-$th chromosome. In general, there is a trade-off relation between the output error e_i of the system and fuzzy rules' number; therefore, we set two weights w_p (for the output error) and w_r (for the rules' number) in the fitness function. The weights can adjust the significance on the output error or on the rules' number, respectively.

The above whole process is summarized into the following procedure.

Step 1: Initialize P chromosomes randomly and each chromosome has two strings as Fig. 5.
Step 2: Evaluate fitness function for each individual and rank the individuals according to their fitness.
Step 3: The new generation is generated by using reproduction, crossover and mutation operations of GA, and then go to step 2 and repeat.

This procedure is repeated until the fitness of the best chromosome is satisfied by the designers.

Finally, we point out the advantages of the proposed method in the following.

1. The fuzzy rule number, the locations of the premise fuzzy sets and the position of the consequent singleton of each rule will be generated automatically, in other words, the GA-based optimal fuzzy rules table will be generated automatically.
2. Trial and error always used by designers in the conventional fuzzy control design can be avoided.
3. This method is also applicable to multi-input and multi-output system, in this case, the code strings of $c_{(.,.)}$, $d_{(.,.)}$ and the consequent $B(j_1, j_2)$ are extended to multiple $c_{(.,.)}$, $d_{(.,.)}$, and consequents $B_n(j_1, j_2, ..., j_m)$, respectively.

4. Generation of the Fuzzy Model

Eventually, the algorithm in section 3 can be applied completely to the work of fuzzy modeling. The only difference is that when we compute the fitness function (5), y_d is the training data from the unknown system to be modeled (see Fig. 8).

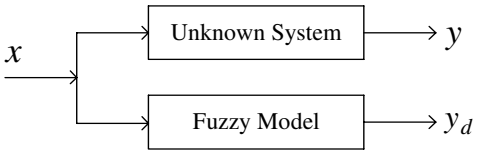


Fig. 8. Fuzzy modeling for the unknown system.

5. Two Examples and their simulations

Example 1: *Fuzzy Controller for the Inverted Pendulum*

In this example, we try to design a fuzzy controller for poising an inverted pendulum system. The inverted pendulum is shown in Fig. 9 and its model is described as below.

$$\dot{\theta}_1 = \theta_2 \tag{6}$$

$$\dot{\theta}_2 = \frac{g\sin\theta_1 - \cos\theta_1\left(\frac{\mathrm{m}L}{\mathrm{M}+\mathrm{m}}\theta_2^2\sin\theta_1 + \frac{1}{\mathrm{M}+\mathrm{m}}u\right)}{\frac{4}{3}L - \frac{\mathrm{m}L}{\mathrm{M}+\mathrm{m}}\cos^2\theta_1} \tag{7}$$

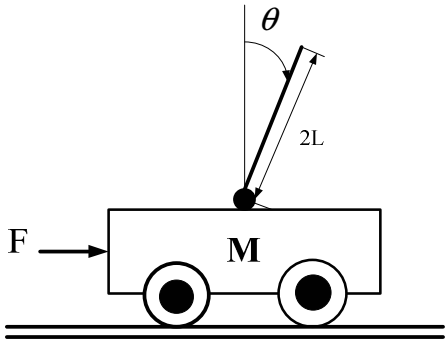
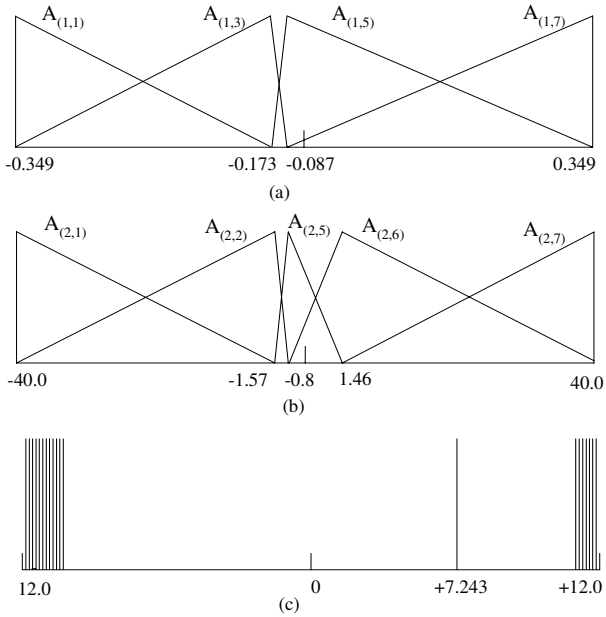


Fig. 9. The inverted pendulum

where θ is the angular displacement of the pole, $\dot{\theta}$ is the angular velocity of the pole, g (acceleration due to the gravity) is $9.8\,m/s^2$, M (mass of the cart) is $1.0\,kg$, m (mass of the pole) is $0.1\,kg$, L (half length of the pole) is $0.5\,m$, and F is the application force in Newton. In this simulation, the boundary conditions of the θ, $\dot{\theta}$ and F are [-20, +20], [-40, +40] and [-12, +12], respectively. Then initial parameters for GA are as follows: Population size $P = 25$, sample rate 0.01sec, crossover probability 1.0, mutation probability 0.005, number of generation 3000, $w_p = 0.995$ and $w_r = 0.005$. After the iteration of GA, Fig. 10 is the generated fuzzy rule base for the controller, in which $4 \times 5 = 20$ fuzzy rules are generated with the premise fuzzy sets in Fig. 10(a) and in Fig. 10(b) and with the consequent singletons in Fig. 10(c). Fig. 11 shows the simulation result of the system with the

designed fuzzy controller in Fig. 10. Fig. 12 shows the output of the designed fuzzy controller for the inverted pendulum.



$B(1,1) = -11.963$, $B(1,2) = -11.936$, $B(1,5) = -11.984$, $B(1,6) = -11.730$, $B(1,7) = 7.243$, $B(3,1) = -11.958$, $B(3,2) = -11.842$, $B(3,5) = -11.886$, $B(3,6) = 11.960$, $B(3,7) = 11.9763$, $B(5,1) = -11.921$, $B(5,2) = -11.985$, $B(5,5) = -11.855$, $B(5,6) = 11.898$, $B(5,7) = 11.9761$, $B(7,1) = -11.959$, $B(7,2) = -11.896$, $B(7,5) = 11.897$, $B(7,6) = 11.963$, $B(7,7) = 11.967$.

Fig. 10. Fuzzy control for the inverted pendulum system. (a). The membership functions of θ . (b). The membership functions of $\dot{\theta}$. (c). The singleton output of consequents.

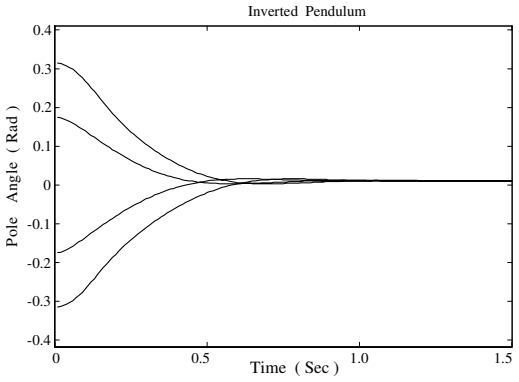


Fig. 11. The output of the inverted pendulum with the designed fuzzy controller.

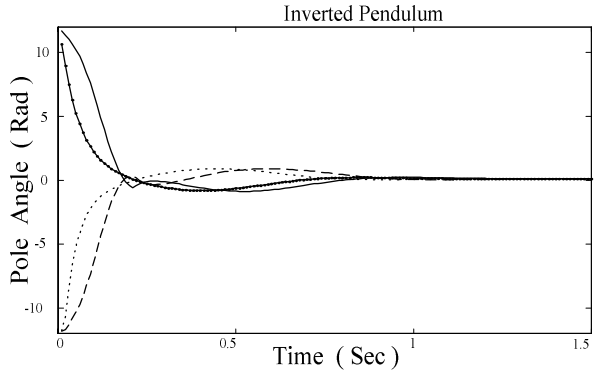


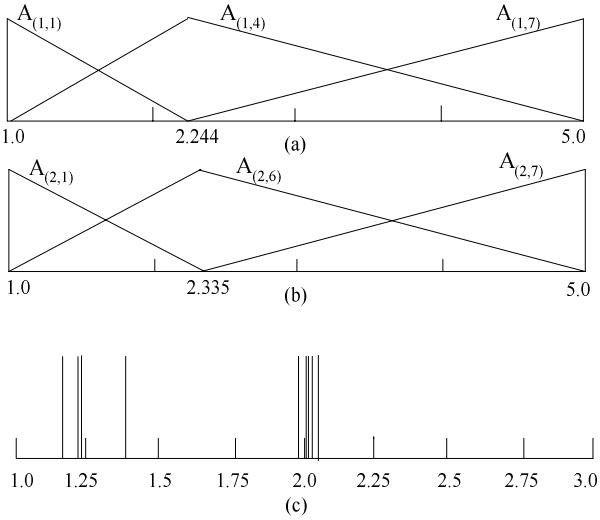
Fig. 12. The output of the designed fuzzy controller for the inverted pendulum.

Example 2: Nonlinear system fuzzy modeling

Consider the nonlinear system as in (8).

$$y = (1 + x_1^{-1.5} + x_2^{-2}) \text{ , } 1 \le x_1, x_2 \le 5 \text{ , } \quad (8)$$

Suppose the system is unknown. We uniformly divide the domain space to obtain 100 training data (input/output data). Initial parameters for GA are as follows: population size $P = 25$, generation number = 2000, mutation probability = 0.005, crossover probability = 1, $w_p = 0.995$, and $w_r = 0.005$. After the iteration of GA, finally, $3 \times 3 = 9$ rules is the trained fuzzy model combined by Fig. 13(a), 13(b) and Fig. 13(c) for the unknown system (8). The comparison of model output and the nonlinear function output is shown in Fig. 14. To test the validity of the desired model, we randomly take additional 25 data to be the testing data. From simulation result, we get the mean square error and the error mean between the testing data and the output of the built fuzzy model is 0.0056 and 0.054, respectively.



$B(1,1) = 2.081$, $B(1,6) = 2.070$, $B(1,7) = 2.003$,
$B(4,1) = 2.035$, $B(4,6) = 1.344$, $B(4,7) = 1.216$,
$B(7,1) = 1.956$, $B(7,6) = 1.243$, $B(7,7) = 1.165$.

Fig. 13. Optimal fuzzy model for nonlinear system. (a) The membership functions of x_1. (b) The membership functions of x_2. (c) The singleton output of consequents.

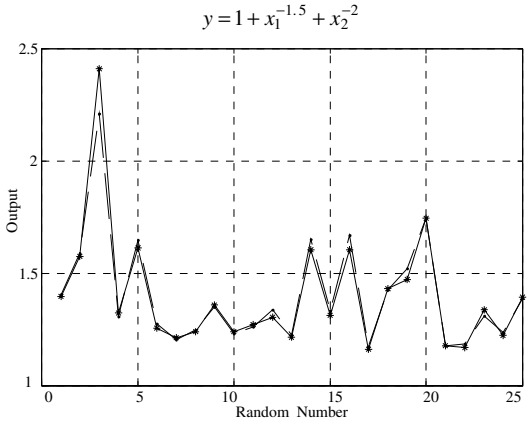


Fig. 14. Comparison of the target output (.) and the fuzzy model output (*) for testing data.

6. Conclusion

This paper has developed a method to generate satisfactory GA-based fuzzy rules. With the aids of GA, trial and error is avoided in the fuzzy rules establishment. In the GA, we have designed the specific structure of the chromosome, the special mutation operation and the adequate fitness function such that the fuzzy rule base, which is with small number of rules, suitable placement of the premise's fuzzy sets and proper location of the consequent singletons, can be generated spontaneously. This paper also has presented the generated fuzzy rule base can be the controller in a closed loop system to achieve some control objective or can be a fuzzy model to approximate a unknown nonlinear system. Finally, two examples have been illustrated to show the effectiveness of the proposed method.

Acknowledgment

This paper is supported by National Science Council of Taiwan under the Grant NSC-92-2213-E-008-020.

References

[1] K. Arakawa, Fuzzy rule-based signal processing and its application to image restoration, *IEEE J. Slect, Areas Commun.,* vol. 12, pp. 1495-1502, Dec. 1994.

[2] E. Shragowitz, J. Y. Lee, Q. Kang, Application of fuzzy in computer-aided VLSI design, *IEEE Trans. Fuzzy Syst.,* vol. 6, pp. 163-172, Feb. 1998.

[3] C. L. Karr, E. J. Gentry, Fuzzy control of PH using genetic algorithms, *IEEE Trans. Fuzzy Syst.,* vol. 1, pp. 46-53, Jan. 1993.

[4] K. C. C. Chan, V. Lee, H. Leung, Generating fuzzy rules for target tracking using a steady-state genetic algorithm, *IEEE Trans. Evol. Comput.,* vol. 1, pp. 189-200, Sept. 1997.

[5] C. C. Wong, C. S. FAN, Rule mapping fuzzy control design, *Fuzzy Sets Syst.,* 108, pp. 253-261, 1999.

[6] C. C. Cheng, C. C. Wong, Self-generating rule-mapping fuzzy controller design using a genetic algorithm, *IEE proc. –Control Theory Appl.,* vol. 149, no. 2, Mar. 2002.

[7] A. Homaifar, E. Mccormick, Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithm, *IEEE Trans. Fuzzy Syst.,* vol. 3, pp. 129-139, May 1995.

[8] W. L. Tung, C. Quek, GenSoFNN: A Genatic Self-Organizing Fuzzy Neurak Network, *IEEE Trans. Neural Networks,* vol. 13, no. 5, pp. 1075-1086, Sep. 2002.

[9] M. Russo, FuGeNeSys–A Fuzzy Genetic Neural System for Fuzzy Modeling. *IEEE Trans. Fuzzy Syst.* vol. 6, no. 3, pp. 373-388, Aug. 1998.

[10] Y. Wang, Gang Rong, A self-organizing neural-network-based fuzzy system, *Fuzzy Sets Syst.,* 103, pp 1-11, 1999.

[11] C. T. Lin, C. S. G. Lee, Neural-network-based fuzzy logic control and decision system, *IEEE Trans. Comput.,* vol. 40, pp. 1320-1336, Dec. 1991.

[12] C. J. Lin, C. T. Lin, Reinforcement learning for an ART-based fuzzy adaptive learning control network, *IEEE Trans. Neural Network,* vol. 7, pp. 709-731, May 1996.

[13] T. L. Seng, M. B. Khalid, R. Yusof, Tuning of a neural-fuzzy controller by genetic algorithm, *IEEE Trans. Syst. Man Cybernet.* vol. 29, no. 2, pp. 226-236, Apr. 1999.

[14] J.S.R. Jang, Fuzzy controller design without domain expert, *IEEE Internat. Conf. Fuzzy Syst,* pp. 289-296, 1992.

[15] J.S.R. Jang, ANFIS: adaptive-network-based inference system, *IEEE Trans. Syst. Man Cybernet.* vol. 23, no. 3, pp. 665-685, 1992.

[16] S.J. Kang, C.H. Woo, H. S. Hwang, K. B. Woo, Evolutionary design of fuzzy rule base for nonlinear system modeling and control, *IEEE Trans. Fuzzy Syst.,* vol. 8, no. 1, pp.37-45, Feb. 2000.

[17] J.B. Gomm, D. L. Yu, Selecting radial basis function network centers with recursive orthogonal least square training, *IEEE Trans. Neural Network,* vol. 11, no. 2, pp. 306-314, Mar. 2000.

[18] Y. Yam, P. Baranyi, C-T Yang., Reduction of fuzzy rule base via singular value decomposition, *IEEE Trans. Fuzzy Syst,* vol. 7, no. 2, pp. 120-132, Apr. 1999.

[19] D.A. Linkens, H.O. Nyogesa., Genetic algorithms for fuzzy control part 1: Offline system development and application, *IEE Proc,-Control Theory Appl.,* vol. 142, no. 3, pp. 161-176, May 1995.

[20] D.A. Linkens, H.O. Nyogesa, Genetic algorithms for fuzzy control part 2: Online system development and application, *IEE Proc,-Control Theory Appl.,* vol. 142, no. 3, pp.177-185, May 1995.