More on Designing Fuzzy Controllers using Genetic Algorithms : Guided Constrained Optimisation

Gregory V. TAN and Xiheng HU
Department of Electrical Engineering,
University of Sydney, NSW 2006, Australia
gregt@ee.usyd.edu.au, hxh@ee.usyd.edu.au

Abstract

The design of fuzzy controllers using the Genetic Algorithm is appearing as a systematic method. This method easily provides an optimised design and forms the framework for further progress. This paper attempts to provide some methods to improve interpretability in Genetic Algorithm-designed fuzzy logic controllers as well as to reduce the amount of genetic material required to represent a complex fuzzy controller. This paper also introduces the concept of Guided Constrained Optimisation.

1. Introduction

Designing Fuzzy Logic Controllers (FLCs) manually requires expert knowledge and usually results in control rules and membership functions which model human thinking. This ensures that the design produced is easily understood and can further be modified to meet specific requirements and changes in environment. However, there is to-date no generalised systematic method beyond ad hoc trial and error ones. Systematic methods for designing and tuning FLCs using automatic methods is still a current research area [1, 2, 4, 8, 11, 21, 23]. One of the foremost of these methods is the Genetic Algorithm (GA) [15, 16]. This technique has been applied to fuzzy classifiers [9], fuzzy neural networks [13], fuzzy modeling [17], fuzzy expert systems [19], adaptive fuzzy controllers [12, 16] and optimised non-adaptive fuzzy controllers [6, 7, 15, 18, 20], showing a great potential.

However, we should be aware of the possible side effects of such techniques as any optimisation algorithm may have a strong impact on the system it is optimising. It may even alter the structure of the original system designed. With fuzzy systems being derived directly from human thinking and having strong physical meaning, this effect may be even more significant. We have posed the questions of what impact the GA may have on FLCs [24]. Our investigations have shown that in many cases FLCs designed by GAs are difficult for a human expert to understand. The membership functions often lose physical meaning, which is the fundamental nature of fuzzy sets theory. The rule base was disordered and seemed arbitrary, possibly resulting in rough control, as greatly differing inference were next to each other. The outcome of the design was difficult to interpret and analyse, let alone modify. A serious question would be : will industry accept such a design as reliable and the technique implementable[24]?

To overcome these problems, some special treatment must be introduced to the design procedure. A basic suggestion made by the authors [24] is to incorporate linguistic knowledge of the systems into the optimisation algorithm. This will restore the principle of fuzzy systems back into the design using GA technique. We may call this treatment *Guided Constrained Optimisation.* Constraints can be incorporated are system dependent and rely on how much the system is understood. Some general aspects may include ordering and symmetry of the membership functions, fixed partitions for certain membership functions, and, the control rule base can be pre-structured according to the heuristic knowledge, but allowing some degree of freedom for the GA to perform. These aspects will be discussed in section 3 and 4 of this paper. Before that, we shall review current works on Genetic Algorithms in the next section. An example of the implementation of the use of Guided Constrained Optimisation design on the inverted pendulum control system will also be studied in section 5. Lastly, we will discuss the method and results, conclude and provide some future plans.

2. Genetic Algorithms (GAs)

The genetic algorithm is a probabilistic computer-driven search and optimisation technique modelled after the mechanics of genetic evolution [3]. The algorithm works on the coding of parameters to be optimised, rather than the parameters themselves as traditional optimisation techniques, such as the simplex method or gradient

0-7803-3796-4/97/$10.00©1997IEEE

methods. The parameters coded together form what is called a chromosome. Thus, a population of chromosomes are formed, each representing a possible solution to the problem. Each chromosome is then evaluated on how good a solution it presents to the problem, giving a value called the fitness. In a simple GA [5], the population will then undergo operations similar to genetic evolution, namely 'reproduction', 'crossover' and 'mutation'. In 'reproduction', a new 'generation' of chromosomes are formed by randomly selecting chromosomes according to their fitness. These are operated on by 'crossover', which involve exchanging parts of two chromosomes structure, resulting in an exchange of information (Fig. 1). 'Mutation' is a local operator which is randomly applied with a low level of probability. It randomly flips bits from one generation to the next. 'Mutation' is usually associated with helping re-inject any information that may have been lost in previous generations and moving around local minima within the search space [3, 5].

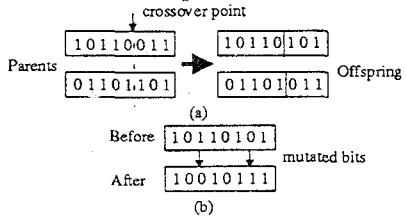

Fig. 1 Genetic operators : (a) crossover (b) mutation

3. Guided Constrained Optimisation

The aim of Guided Constrained Optimisation is to achieve an optimised GA-designed fuzzy logic controller which is incorporates human knowledge, which should result in the controller being interpretable. The structure of the proposed method is shown in Fig. 2. Section (3) of Fig. 2 is the standard GA-FLC design method. Section (4) describes the knowledge base, which can be built up off-line or interactively on-line. This knowledge base will 'guide' the GA optimisation procedure, usually through the coding procedure.

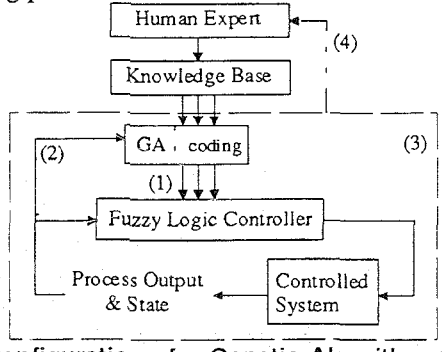

Fig. 2 Configuration of a Genetic Algorithm designed Fuzzy Logic Controller (FLC), with Guided Constrained Optimisation

Just as the usual trial and error method of designing a fuzzy controller, the expert gains knowledge from the performance of the GA in designing the FLC. Then he refines the knowledge base and the process is repeated until a satisfactory FLC is obtained (Fig. 2(4)). Thus the designer implements his knowledge in terms of the internal structural constraints of the controller and then allows the genetic algorithm to operate within these constraints.

The following few aspects can be used to implement such specific structure on the FLC; this list is not exhaustive :

a) Ordering of partitions : This should be implemented as linguistic labels are required to be able to define implications in fuzzy terminology and can be performed by a reconstruction operator during decoding of the chromosome. More comments on the coding will be given in a later section. This required less computer processing overheads then an ordered operator. With ordering, a FLC, for example, with seven partitions for its membership functions can be linguistically labelled as :

1- Large Negative
2- Medium Negative
3- Small Negative
4- Zero
5- Small Positive
6- Medium Positive
7- Large Positive

b) Using symmetry : By virtue of a problem being symmetrical, as most problems are, the genetic materia. required to represent each partition can be reduced by allowing the membership function to be reflected across the zero point. Thus this reduces the required representation by half and reduces convergence time.

c) Fixing of parts of partitions : As certain linguistic labels have stronger significance then others, it would be beneficial to fix certain points. As an example, the peak of the Zero (4) partition should be fixed at the zero point. In addition, the peaks of the Large Negative (1) and Large Positive (7) can be fixed at their respective ends of their universes of discourse. The bases for these partitions should then be movable to be optimised.

d) Pre-structuring of the rule base lookup table : This we perform as greatly differing neighbouring implications in the table possibly indicate rough control. Thus for smooth control, we ensure that change over between inferences do not move from or imply at the same time, say Large Negative and Large Positive or other largely differing implication. Using a 2-input 1-output FLC with 7 partitions per input/output membership function as an example, we show how this can be implemented. Suppose we know the system should have a result Large Negative when both inputs are Large Negative, and Large Positive what the inputs are Large Positive. In addition, the system is somewhat symmetrical and requires the steady state to be about zero. With this knowledge, we can pre-structure the rule base table allowing a limited range of possible

implication at different areas. A few possibilities for such pre-structuring can be shown in Fig. 3.

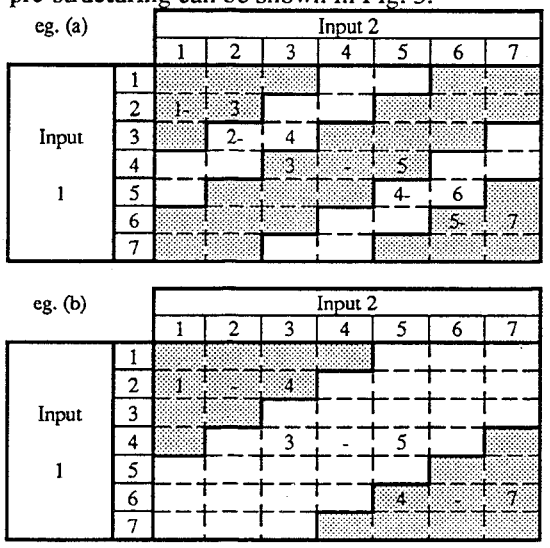

Fig. 3. Possible pre-structures for rule base lookup table. Each implication has linguistic labels as suggested in 3(a) Ordering partitions.

4. Application of GA to design of Fuzzy Logic Controllers (FLCs), with Guided Constrained Optimisation

A general structure for a FLC would include a fuzzification interface, a knowledge base, a rule base, which contain decision making logic, and a defuzzification interface [18]. When the GA is used to design a FLC, the GA would usually replace the knowledge base. However, with Guided Constrained Optimisation, the knowledge base is not removed. Instead, the GA is simply added to the system, being interfaced with the knowledge base at the coding point (Fig. 2 (1)). This point of interface is the decoding of a single chromosome into a FLC. This decoding provides the pre-structuring of the FLC into a format determined by the expert, which comply which his heuristic knowledge of the system. Another interface would be at the output of the controlled system (Fig. 2 (2)). This takes the form of an objective function which determines the fitness of particular solution.

4.1 Decoding

For simplicity, we shall use triangular membership functions. As both the fuzzification and defuzzification interfaces are similar, they were coded in the same way. A single byte was used to represent each point. For each membership function, the byte determines the position of the peak by the quantised level it represents between the beginning and the end of its particular universe of discourse. Two other bytes are used to represent the distance of the two bases of the triangular partition of membership functions. Thus each partition in the membership function can be represented by 3 bytes, as shown in Fig. 4(a). Then on decoding from the chromosome into the FLC, each partition would be ordered in ascending order depending on each peak of each partition. Also, some points, based on their linguistic meanings, may be fixed, such as the partition Zero, Large Negative and Large Positive. If the system under control is symmetrical, we may just use half the number of partitions and reflect the membership function about the zero point, giving us a symmetrical membership function. The rule base was coded using integers to represent an implication between a particular input and output. A zero in the rule base indicates that there is no implication between those inputs and outputs. This is shown in Fig 4(b). As for pre-structuring, certain areas on the rule base table would be limited to allows only certain possible implications. For examples of such pre-structuring, one can refer to Fig 3(a) and (b). Each membership function is then concatenated to form a chromosome. For a system with 2 inputs and one output with 7 partitions, this is shown in Fig. 4(c). For ease of compacting, each possible implication may be represented by half a byte. Thus the 2-input, 1-output, 7 partition fuzzy controller, requires at most $3 \times 3 \times 7 + \frac{1}{2} \times 7 \times 7 = 87\frac{1}{2}$ or 88 bytes, to round it off.

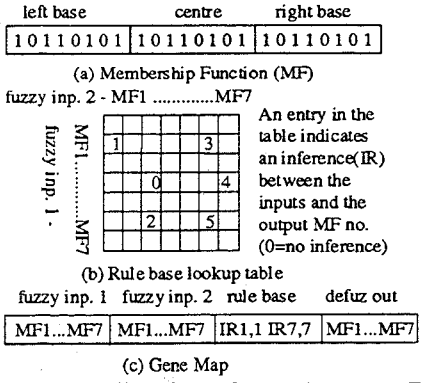

Fig 4. Maximal Coding for 2-input 1-output 7-partition FLC.

4.2 Objective Function

In designing FLCs using GAs, usually a simulation of the system to be controlled is executed. The objective function can be designed based on standard control parameters such as rise time, overshoot and steady-state zero error. Another important parameter which influences the resulting design is the starting state of the simulation, which should be varied sufficiently.

5. Design of Inverted Pendulum FLC

In this section, we first present the system and common objective function used, then the GA-designed FLC for the inverted pendulum system. Then the resulting FLC designed by GA with Guided Constrained Optimisation and the methods used.

5.1 Brief Description of System

The classic inverted pendulum was chosen as the sample control system. A schematic of the system is shown in Fig. 5. The FLC is a two input (θ,$\dot{\theta}$), one output (force) controller. The input and output membership functions for simplicity were chosen to be triangular.

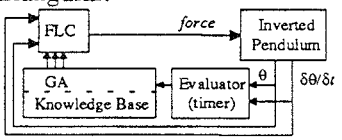

Fig. 5 Complete System Schematic

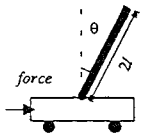

Fig. 6 Inverted Pendulum

The inverted pendulum system can be liken to a pole on a movable cart, the pole restricted to the vertical plane (Fig. 6). The state of the system is described by the pole's angle, θ, and its angular velocity, $\dot{\theta}$, allowing for infinite movement of the cart. The system was simulated using differential equations and approximated by a two-step forward Euler integration. A more complete and mathematical description can be found in the references [10].

5.2 Objective Function of Inverted Pendulum

The objective function used in the simulation was similar to Lee and Takagi [14]. The same possible outcomes still apply : the pole balances, the time expires or the pole falls. Thus each trial was evaluated using the following modified function :

$$fitness = \begin{cases} a_1(t_{\max} - t_{end}) + a_2 reward & (a) \\ reward & (b) \\ b \cdot t_{end} & (c) \end{cases}$$

where a_1, a_2, b and *reward* are constants, and (a) pole balanced, t_{zero} is time required to achieve steady state zero error, (b) $t_{max} = t_{end}$, and (c) pole fell over ($|\theta| \geq 90°$). If the controller balances the pole, a shorter time is better than a longer one. In addition, if the pole fell over, the controller was credited according the time it kept the pole from falling. However, unlike Lee and Takagi, there were no penalties for the number of rules in the system [14].

5.3 Raw GA-designed FLC

5.3.1 Procedure. The GA used was a generational one based on the Simple GA[7], with three-point crossover, mutation and elitism. The population was set to 30 and crossover and mutation probabilities to 0.8 and 0.01 respectively. All members were initialised with random values. The GA was given complete freedom to place membership functions anywhere within the available range and any implication between input and output membership functions was allowed. Initial pole angles were set at both positive and negative angles, thus completing two trials for every evaluation. The GA was allowed to execute for 1000 generations, allowing the solution to converge and stabilise. The fittest controller was then selected as the final design result. The inference operator used was the max-product rule and the de-fuzzification method used was the centre of area method. The FLC had 7 partitions per input and output, and this, as well as the rule base, were coded as for a general case. Thus each possible design was represented by 88 bytes.

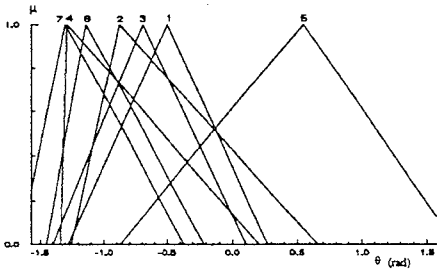

Fig. 7(a) Input membership functions : input 1 - θ

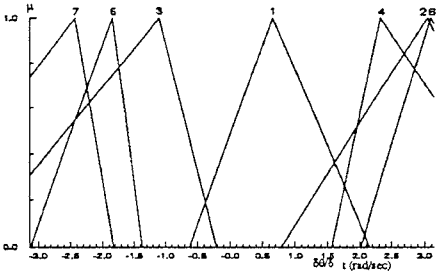

Fig. 7(b) Input membership functions : input 2 - $\dot{\theta}$

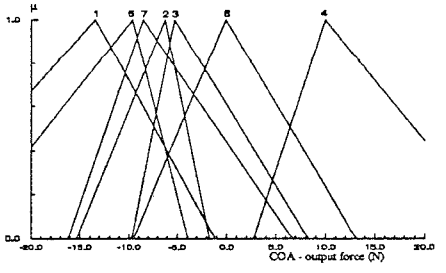

Fig. 8 Output membership functions

5.3.2 Results. The resulting design could balance the pole in less than 0.4 seconds. The membership functions of the input and the output are presented in Fig. 7 and

Fig. 8 respectively, while rule base in Fig. 9 and the control surface in Fig. 10.

		Input 2 - δθ/δt						
		1	2	3	4	5	6	7
Input 1 - θ	1	2	3	0	4	2	4	0
	2	4	1	3	2	0	0	6
	3	1	2	2	4	6	0	7
	4	1	3	4	0	4	6	7
	5	4	4	2	4	6	7	6
	6	4	3	4	6	6	7	4
	7	4	4	5	6	6	4	5

Fig. 9 Rule base

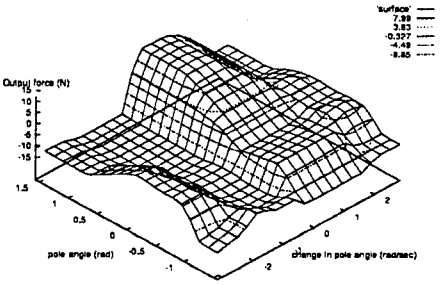

Fig. 10 Control surface for GA-designed FLC (Inverted Pendulum)

5.4 Design by Guided Constrained GA

5.4.1 Procedure. The GA used here was the same one as use for the raw GA-designed FLC, with the same parameters for crossover, mutation and no. of generations. Inference operators and well as selection of the final design result was identical. With what knowledge we have of the system and to improve interpretability, ordering of partitions in the membership function as per 3(a) , the use of symmetry (3(b)), as well as the fixing of the peak of the Zero (4) partition, and of Large Negative (1) and Large Positive (7). The rule base table was pre-structured as in Fig. 3(a).

5.4.2 Resulting Design. The resulting design could balance the pole in less than 0.4 seconds. The membership functions of the input and the output are presented in Fig. 11 and Fig. 12 respectively, while the rule base is in Fig. 13 and the control surface in Fig. 14.

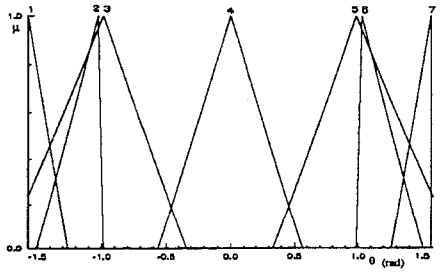

Fig. 11(a) Input membership functions : input 1 - θ

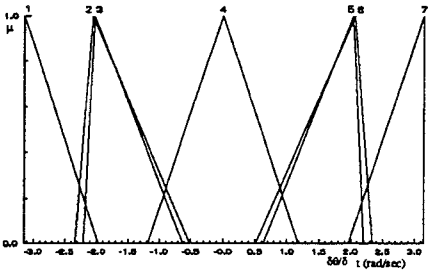

Fig. 11(b) Input membership functions : input 2 - $\dot{\theta}$

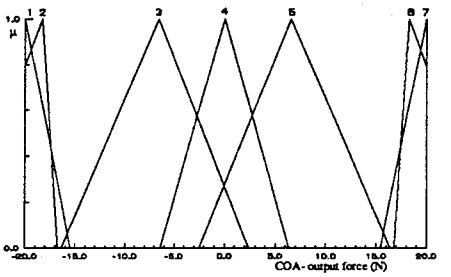

Fig. 12 Output membership functions

		Input 2 - dθ/dt						
		1	2	3	4	5	6	7
Input 1 - θ	1	3	0	3	2	3	5	4
	2	3	3	4	0	3	5	0
	3	1	0	3	5	0	0	5
	4	2	0	5	4	3	5	0
	5	0	5	3	3	4	4	7
	6	0	3	3	5	6	0	7
	7	0	4	4	4	7	7	0

Fig. 13 Rule base

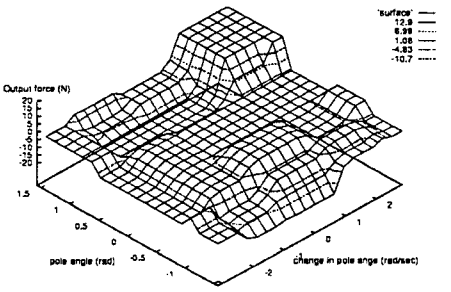

Fig. 14 Control surface for GA-designed FLC (Inverted Pendulum)

6. Discussion

We can see that by using the method of Guided Constrained Optimisation, we can improve the interpretability of a GA-designed FLC immensely. However, the idea of this method is to use what little we do know of the system to be controlled and to incorporate it into the design process. The methods proposed, apart from ordering, are actually derived from heuristic knowledge of the system, ie. the inverted pendulum system. This method is not rigid, but also involves a little trial and error, however not as much as used when

designing a FLC by hand. Also, this shows how a complex controller may be designed by a GA, while retaining it inherent linguistic value. The constraining of the rule base lookup table performed here is by no means to only way to do it. The constrains placed on the table relate to the how coarse or fine the control requires, and the knowledge of what general implications work for the system under control. Lastly, another way to improve interpretability would be to set a penalty on the number of implications in the rule base. The more rules, the higher the penalty. this would reduce the number of rule in the rule base. However, it is aim of the authors that the method be applied to a reasonably complex controller in order to gauge how well the method works.

7. Conclusions & Future Work

From the example, we can see that Guided Constrained Optimisation has achieved a Fuzzy Logic Controller which, although complex, can be understood and interpreted. This also suggest that the resulting design could also be modified by humans. Thus with the new method of Guided Constrained Optimisation, it is hoped that some of the problems faced by the implementation of GA-designed FLCs would be reduced. This would hopefully allow the method of using GAs to design fuzzy controller to gain wider acceptance not only among the academic community, but in industry as well. As this is but a theoretical foundation for such a method, we are continuing to seek to implement the method practically.

REFERENCES

[1]Burkhardt, D. G. and P. P. Bonissone : *Automated fuzzy knowledge base generation and tuning*. Proc. IEEE Int'l. Conf. on Fuzzy Systems, pp. 179-188, 1992.

[2]Chen, Y. Y., K. Z. Lin and S. T. Hsu : *A self-learning fuzzy controller*. Proc. IEEE Int'l. Conf. on Fuzzy Systems, pp. 189-196, 1992.

[3]Davis, L. : *A handbook of genetic algorithms*. Van Nostrand Reinhold, New York, 1990.

[4]Esogbue, A. O., W. E. Hearnes and Q. Song : *A Reinforcement Learning Fuzzy Controller for Set-Point Regulator problems*. Proc. IEEE Int'l. Conf. on Fuzzy Systems, pp. 2136-2142, 1996.

[5]Goldberg, D. E. : *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley, 1989.

[6]Homaifar, A. and E. McCormick : *Simultaneous design of membership functions and rule sets for fuzzy controllers using genetic algorithms*. IEEE Trans. Fuzzy Systems, vol. 3, no. 2, May 1995.

[7]Hwang, W. R. and W. E. Thompson : *Design of intelligent fuzzy logic controllers using genetic algorithms*. Proc. IEEE Int'l. Conf. on Fuzzy Systems, pp. 1383-1388, 1994.

[8]Ishibuchi, H. and M. Nu : *Learning of Fuzzy Connection Weights in Fuzzified Neural Networks*. Proc. IEEE Int'l. Conf. on Fuzzy Systems, pp. 373-379, 1996.

[9]Ishibuchi, H., K. Nozaki, N. Yamamoto and H. Tanaka : *Acquisition of fuzzy classification knowledge using genetic algorithms*. Proc. IEEE Int'l. Conf. on Fuzzy Systems, pp. 1963-1968, 1994.

[10]Jang, J.-S. R. : *Fuzzy controller design without domain experts*. Proc. IEEE Int'l. Conf. on Fuzzy Systems, pp. 1963-1968, 1994.

[11]Kang, H. and G. Vachtsevanos : *Adaptive fuzzy logic control*. Proc. IEEE Int'l. Conf. on Fuzzy Systems, pp. 407-414, 1992.

[12]Karr, C. L. : *Design of an adaptive fuzzy logic controller using a genetic algorithm*. Proc. 4th Int'l. Conf. on Genetic Algorithms, pp. 450-457, 1991.

[13]Krishnamraju, P. V., J. J. Buckley, K. D. Reilly and Y Hayashi : *Genetic learning algorithm for fuzzy neural nets*. . Proc. IEEE Int'l. Conf. on Fuzzy Systems, pp. 1969-1974, 1994.

[14]Lee, M. A. and H. Takagi : *Integrating design stages of fuzzy systems using genetic algorithms*. Proc. IEEE Int'l. Conf. on Fuzzy Systems, pp. 612-617, 1993.

[15]Linkens, D. A. and H. O. Nyongesa : *Genetic algorithms of fuzzy control Part 1 : Offline system development and application*. IEE Proc.-Control Theory Appl., Vol. 142, No. 3, pp. 161-176, May 1995.

[16]Linkens, D. A. and H. O. Nyongesa : *Genetic algorithms of fuzzy control Part 2 : Online system development and application*. IEE Proc.-Control Theory Appl., Vol. 142, No. 3, pp. 177-185, May 1995.

[17]Liska, J. and S. S. Melsheimer : *Complete design of fuzzy logic systems using genetic algorithms*. Proc. IEEE Int'l. Conf. on Fuzzy Systems, pp. 1377-1382, 1994.

[18]Ng, K. C. and Y. Li : *Design of sophisticated fuzzy logic controllers using genetic algorithms*. Proc. IEEE Int'l Conf. on Fuzzy Systems, pp. 1708-1712, 1994.

[19]Perneel, C., J. Themlin, J. Renders, and M. Acheroy : *Optimisation of fuzzy expert systems using genetic algorithms and neural networks*. IEEE Trans. Fuzzy Systs., Vol. 3, No. 3, pp. 300-312, August 1995.

[20]Satyadas, A. and K. Kishna Kumar : *GA-optimised fuzzy controller for spacecraft attitude control*. Proc. IEEE Int'l Conf. on Fuzzy Systems, pp. 1979-1984, 1994.

[21]Shi, T., M. Mizumoto, N. Yubazaki and M. Otani : *A Learning Algorithm for Tuning Fuzzy Rules Based on the Gradient Descent Method*. Proc. IEEE Int'l. Conf. on Fuzzy Systems, pp. 55-61, 1996.

[22]Shieh, C. Y. and S. S. Nair : *A new self tuning fuzzy controller design and experiments*. Proc. IEEE Int'l. Conf. on Fuzzy Systems, pp. 309-314, 1993.

[23]Smith, S. M. and D. J. Comer : *An algorithm for automated fuzzy logic controller tuning*. Proc. IEEE Int'l Conf. on Fuzzy Systems, pp. 615-622, 1992.

[24] Tan, G. V. and X. Hu : *On Designing Fuzzy Controllers Using Genetic Algorithms*. Proc. IEEE Int'l. Conf. on Fuzzy Systems, pp. 905-911, 1996.