

# Design of Sophisticated Fuzzy Logic Controllers Using Genetic Algorithms

Kim Chwee Ng and Yun Li  
Department of Electronics and Electrical Engineering  
University of Glasgow, Rankine Building  
Glasgow G12 8LT, Scotland, U.K.

**Abstract** - Design of fuzzy logic controllers encounters difficulties in the selection of optimized membership functions and fuzzy rule base, which is traditionally achieved by a tedious trial-and error process. This paper develops genetic algorithms for automatic design of high performance fuzzy logic controllers using sophisticated membership functions that intrinsically reflect the nonlinearities encountered in many engineering control applications. The controller design space is coded in base-7 strings (chromosomes), where each bit (gene) matches the 7 discrete fuzzy value. The developed approach is subsequently applied to design of a proportional plus integral type fuzzy controller for a nonlinear water level control system. The performance of this control system is demonstrated higher than that of a conventional PID controller. For further comparison, a fuzzy proportional plus derivative controller is also developed using this approach, the response of which is shown to present no steady-state error.

## I. INTRODUCTION

Modern control theory has been successful for well defined, either deterministically or stochastically, systems. This approach, however, encounters problems in many engineering applications where systems to be controlled are difficult to model, have a strong nonlinearity or are embedded in a changing environment with uncertainty. With the development of modern information processing technology and computational intelligence, an alternative solution to these problems has been to incorporate human intelligence directly into automatic control systems. These intelligent control schemes tend to imitate the way of human decision making and knowledge representation and have received increasing attention widely across the control community in the world. It has been shown, in applications such as robot control, automotive systems, aircraft, spacecraft and process control, that they offer potential advantages over conventional control schemes in (a) less dependency on quantitative models; (b) natural decision making; (c) learning capability; (d) a greater degree of autonomy; (e)

ease of implementation and (f) friendly user interface [1-3].

Incorporating the uncertainty and abstract nature inherent in human decision making into intelligent control systems, a fuzzy logic controller (FLC) offers a more accurate and efficient approach, which tends to capture the approximate and qualitative *boundary conditions* of system variables (in contrast to the probability theory that deals with random *behaviour*) by fuzzy sets with a membership function. Such a system flexibly implements functions in near human terms, i.e. IF-THEN linguistic rules, with reasoning by fuzzy logic, which is a rigorous mathematical discipline. It has been demonstrated that fuzzy logic control systems are reliable and robust and are straightforward to implement [1-3].

The crux of designing an FLC lies, however, in the selection of high-performance membership functions that represents the human expert's interpretation of the linguistic variables, because different membership functions determine different extent to which the rules affect the action and hence the performance. The existing iterative approaches for choosing the membership functions are basically a manual trial-and-error process and lack learning capability and autonomy. Therefore, the more efficient and systematic genetic algorithm (GA) [4], which acts on the survival-of-the-fittest Darwinian principle for reproduction and mutation, has recently been applied to FLC design for searching the poorly understood, irregular and complex membership function space with improved performance. Successful application of this approach has been demonstrated in spacecraft rendezvous [5], cart-pole balancing [6], linear motion of cart [7], three-term control [8] and pH value control [9], where the FLC membership functions are represented by internally linear triangular/trapezoidal shapes which are mainly encoded in binary numbers for optimal searching by GAs.

This paper develops genetic algorithms for designing fuzzy logic controllers using sophisticated membership functions that intrinsically reflect the nonlinearity encountered in many engineering control problems. Since the coding parameters are increased in these FLCs and decision making of most FLCs are based

on seven discrete values in one dimension, the base-7 coding is used for the coding process. The developed approach is subsequently applied to design two FLCs for a nonlinear coupled tank water level control system and their performances are compared with conventional PID control schemes.

## II. FUZZY LOGIC CONTROLLER

A schematic of a fuzzy control system for a regulation control task is shown in fig 1. The fuzzy logic controller relates the control variables (error and change\_in\_error) being translated by a component known as a condition interface into fuzzy linguistic terms which are specified by the membership functions of the fuzzy sets. In order to reflect the fuzzy nature of the seven linguistic classifications and to allow for the best options for high performance design, this paper uses the symmetrical exponential membership functions given by :

$$\mu_{\pm i}(x) = \exp\left(-\frac{|x \mp \alpha_i|^{\beta_i}}{\sigma_i}\right) \quad (1a)$$

$$\forall x \in [-\text{large}, +\text{large}]$$

where

$$i = \{\text{zero}, \pm \text{small}, \pm \text{medium}, \pm \text{large}\},$$

and

$$\mu_{+\text{large}} = 1 \quad x > \alpha_{+\text{large}} \quad (1b)$$

$$\mu_{-\text{large}} = 1 \quad x < \alpha_{-\text{large}} \quad (1c)$$

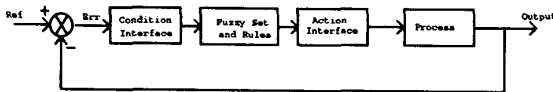


Fig. 1 A fuzzy controller

An example of the shapes of the membership functions of the error and change\_in\_error-error variable is shown in fig 2. Here,  $\alpha_i$  is the position parameter which describes the centre point of the membership function along the universe of discourse,  $\beta_i \in [1.5, 5.0]$  is the "shape parameter" which resembles the shapes from a triangular to a trapezoidal, and  $\sigma_i \in [0.1, 3.0]$  is the "scale parameter" which modifies the base-length of the membership functions and determines the amount of overlapping. Note that a small overlapping is necessary for a distinctive fuzzy decision making. Note also that in (1a),  $\beta_i$  is not included in the power of  $\sigma_i$  and this will allow for a gradual and consistent change in the base-length.

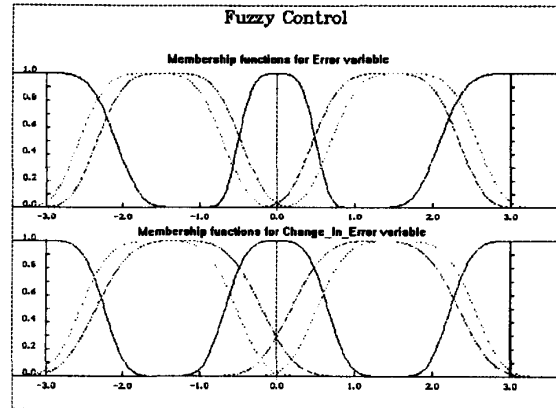


Fig. 2 Symmetrical exponential membership functions

In fuzzy control, the linguistic terms which are defined on the appropriate universe of discourse evaluate the control rules using the compositional rules of inference. The result of application of the rules is a fuzzy set defined on the universe of possible control actions and in turn, an appropriately computed control action is determined and then reconverted to the crisp value to regulate the process. Therefore, the essential design steps in designing fuzzy controllers include 1) defining input and output variables, 2) specifying all the fuzzy sets and their membership functions defined for each input and output variable, 3) converting the input variables to fuzzy sets, 4) compilation of an appropriate and complete set of heuristic control rules that operate on these fuzzy sets, i.e. formulating the fuzzy rule-base, 5) designing the computational unit that accesses the fuzzy rules and computes for the fuzzy control action, 6) devising a transformation method for converting fuzzy control action to crisp value.

The major task in the design of a fuzzy controller lies in the optimal choice of the membership functions, or the  $\alpha$ ,  $\beta$  and  $\sigma$  parameters in the case of membership functions given in (1). In manual design, these functions and the rule-sets are usually obtained by a tedious trial-and-error process which after does not go through the entire possible solution space and therefore does not result in an optimal design. This is also the reason that manually design FLCs compromise accuracy with simplicity by using pure triangular and trapezoidal membership functions.

## III. DESIGN OF SOPHISTICATED FLCs USING GAS

The genetic algorithms developed by Holland [4] is to simulate the natural evolution process that operates on chromosomes. The simple genetic algorithm that yields satisfactory results in many practical problems is

done by using three operators: (1) reproduction; (2) crossover; and (3) mutation in the following process.

```

Make initial population
REPEAT
  Choose parents from the population;
  Selected parents produce children with the
  number weighted by their individual
  fitness;
  Extend the population with the children;
  Select fittest elements of the extended
  population to survive for the next cycle
UNTIL satisfactory generation found
Decode the optimum population (and form the final
FLC)
  
```

By coding the coefficients ( $\alpha$ ,  $\beta$  &  $\sigma$ ) of the membership functions, the fuzzy logic rule-set and the gains of error and change\_in\_error into a decimal-number string, FLC design can be developed and optimised by using GAs. The coded FLC design population can be found by the entire space of the strings termed "chromosomes", each of which was randomly generated "bits", termed "genes". Then the GA process is used to reproduce and select the "fittest" individual, i.e., the "optimal" solution to designing FLCs. A FLC design process is usually complicated, nonlinear and poorly understood and GAs has been proven to be an extremely efficient searching tool for such a process. It is also shown that such a searching technique is robust and convergies faster than conventional searching alprithms.

Small alterations had been made to Goldberg's general-purpose GA [10] to include an adaptive mutation method and the selection of the fittest elements from the new and previous generations to survive. In the adaptive mutation method, an identical string of chromosome is prevented in the new generation by increasing the mutation rate based on the similarity compared to their parents after the process of crossover. With this conception, the maximum mutation rate is limited to 0.2 and the lowest is at 0.03.

With a concatenated, mapped, unsigned coding method, the coefficients ( $\alpha$ ,  $\beta$  and  $\sigma$ ) and the gains of the error and change\_in\_error ( $K_1$  and  $K_2$ ) are coded with values mapped form a minimum value  $C_{min}$  to a maximum value  $C_{max}$  using an  $n$ -bit, unsigned base-7 integer starting from 0. The decoding mapping is given by

$$C = C_{min} + (C_{max} - C_{min}) \times \frac{string\_val}{7^n - 1} \quad (2)$$

where the *string\_val* is the base-7 value represented by an  $n$ -bit string, and  $C$  is the decimal (real) value being coded. The choice of the specific number of bits  $n$  used

to represent each sub-string variable is dependent on the resolution required in its variation. An example of a complete chromosome string is shown in fig 3, where sub-string groups A, B, C, D and E represent the rule-set for the fuzzy rule-base, scaling parameters ( $\sigma$ ), position parameters ( $\alpha$ ), gains ( $K$ ) and shape parameters ( $\beta$ ).

```

0530123550125300021060123456313466362352553456656 | 3235350114516310 | 41053506 | 0403102 | 66666655
|-----A-----|-----B-----|-----C-----|-----D-----|-----E-----|
  
```

Fig. 3 A coded chromosome string and it's partitioned sub-strings

The fuzzy rule-set focuses  $7 \times 7$  possible control actions corresponding to values in input error and change\_in\_error and therefore 49 bits are used in string group A to form the look-up table, where a single bit represents each control action. This illustrated in Fig 4.

In sub-string B, each 2 bit group represents the value of  $\sigma_{\pm B}$ ,  $\sigma_{\pm M}$ ,  $\sigma_{\pm S}$  and  $\sigma_{ZO}$  to be used for modifying the scaling factors of the error and change\_in\_error membership functions, respectively. The next sub-string of 8 integers are coded for defining the positions of the fuzzy sets "small" and "medium" along the universe of discourse, whilst the positions of "Big" and "Zero" are fixed, i.e.  $\alpha_B = 3.0$  and  $\alpha_{ZO} = 0.0$ . Again, each  $\alpha$  parameter will require two bits. Sub-string D represents  $K_1$  and  $K_2$  used as gains of the error and change\_in\_error, with three bits assigned to each. The final group of 8 integer characters is coded for the shape coefficients  $\beta$  of both the error and change\_in\_error variables, requiring one bit for each parameter.

Code Equivalent:		Change-in-error						
		NB	NM	NS	ZO	PS	PM	PB
Negative Big	- NB - 0	0	0	2	0	0	0	0
Negative Medium	- NM - 1	1	2	0	1	5	4	5
Negative Small	- NS - 2	6	1	3	2	2	4	6
Zero	- ZO - 3	0	1	2	3	4	5	6
Positive Small	- PS - 4	3	4	6	4	4	2	4
Positive Medium	- PM - 5	5	3	1	5	6	5	2
Positive Big	- PB - 6	6	6	6	6	4	4	6

Fig. 4 Fuzzy rule-set look-up table for control actions

For each individual chromosome (a complete string) in the population, it is necessary to establish a measure of its fitness,  $f(x)$ , that is often used to accurately evaluate the performance of the controller and will be used to generate a probability according to which the individual in question will be selected for reproduction. However, the task of defining a fitness function is always application specific. In this paper, the objective

of the controller is to drive the output of the process to the desired set-point in the shortest time possible and to maintain the output at the desired set-point, which is evaluated by:

$$x = \sum_{n=0}^{finish\_time} \{n e_n^2 + n(\Delta e_n)^2\}$$

$$f(x) = \exp\left(-\sqrt{\frac{x}{finish\_time}}\right) \quad (3)$$

where  $n$  is the time index,  $e$  the error,  $\Delta e$  the change\_in\_error and the  $finish\_time$  in the following implementation is 300.

For a given set of membership functions, error energies were calculated with the intent of using the GA to minimise it. This fitness function provides a mean for evaluating the performance of each FLC using different fuzzy membership functions and rules-base being selected, so that an optimised FLC would be developed.

#### IV. IMPLEMENTATION ON A NONLINEAR SYSTEM

The approach developed in the previous section is programmed in Pascal and is then used to design an FLC (of proportional plus integral type) for a nonlinear twin-tank coupled water level control system. The membership functions designed by the GA are shown in Fig. 2, whose parameters are given below:

	For error	For change in error
Position	$\alpha_S = 1.40$	$\alpha_S = 1.27$
	$\alpha_M = 1.63$	$\alpha_M = 1.53$
Shape	$\beta_{ZO} = 4.0$	$\beta_{ZO} = 3.85$
	$\beta_S = 4.0$	$\beta_S = 3.58$
	$\beta_M = 4.0$	$\beta_M = 4.0$
	$\beta_B = 4.0$	$\beta_B = 4.0$
Scale	$\sigma_{ZO} = 0.09$	$\sigma_{ZO} = 0.33$
	$\sigma_S = 1.11$	$\sigma_S = 1.88$
	$\sigma_M = 1.11$	$\sigma_M = 1.51$
	$\sigma_B = 0.98$	$\sigma_B = 0.50$

It can be inferred that manually designed membership functions could not be as sophisticated as those shown in Fig. 2 and would not ultimately lead to optimised results by trial-and-error. The response of the water level of one tank to a step with amplitude 75mm is given by curve (1) in Fig. 5, using this PI type FLC designed automatically by a genetic algorithm. For comparison, the performance of a conventional PID controller is shown in curve (2), whose gains were

initially determined by the Ziegler-Nichols rule and further manually tuned to their best performance. As can be seen, the performance achieved by the genetic FLC is apparently superior to that obtained from this PID controller, with both the overshoot and decay rate being improved.

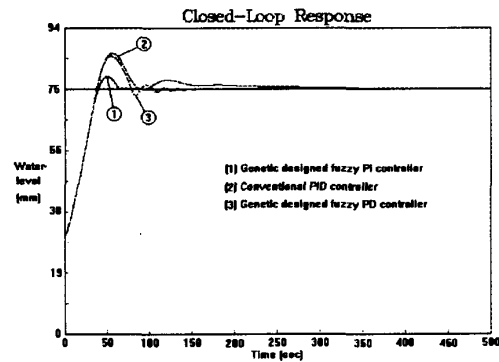


Figure 5 Performance comparison between GA designed PI and PD FLCs and manually designed PID controller

The GA approach developed in the previous section can also be extended to other designs. For comparison purpose, it has been used to design an FLC in a traditional and simplest way, where the control action generated obeys the PD instructions, rather than the PI instructions. The performance is shown in curve (3) of Fig. 5, which is slightly better than that of the manually tuned PID controller. It is interesting to note that there is no steady-state error resulting from this fuzzy PD controller, whilst this is not the case for a conventional PD controller (results not shown in Fig. 5).

#### V. CONCLUSION AND FUTURE WORK

GAs have been proven to be an extremely efficient and robust searching tool for complicated and poorly understood processes. It is also shown in the literature that such a searching technique converges intelligently and much faster than conventional learning means.

Genetic algorithms for automatic design of fuzzy logic controllers have been developed, using sophisticated membership functions that intrinsically reflect the nonlinearity encountered in many engineering applications. Utilising these sophisticated membership functions, the control laws can be implemented in a simple format, such as PI or PD scheme. The sophistication obtained by the machine based automatic design could not be reached by manual design which is exclusively based on a painstaking trial-and-error process. The genetic design approach

discussed in this paper offers a convenient and complete way to design a fuzzy controller in the shortest time.

Further work underway includes on-line design of adaptive FLCs. For such a genetic-fuzzy controllers, parallel architectures are currently studied in order to provide a high throughput rate for the control signals with short system latency, whilst performing the adaptation tasks.

#### ACKNOWLEDGEMENT

Mr. Ng is grateful for the University of Glasgow and CVCP for their support in the form of a Postgraduate Scholarship and Overseas Research Scheme award. The authors would like to thank their colleague, Dr. Ken Sharman, for useful discussions on evolutionary algorithms.

#### REFERENCES

1. Special Issue on Intelligent Control, *IEEE Control Systems*, vol.13, no.3, June 1993.
2. K.C. Ng and Y. Li, *Application of Genetic Algorithms to Design of Fuzzy Logic Controllers*, Internal Report, Department of Electronics and Electrical Engineering, University of Glasgow, Aug. 1993.
3. E. Rogers and Y. Li., Eds., *Parallel Processing in a Control Systems Environment*, London: Prentice Hall International, May 1993.
4. J.H. Holland, "Genetic algorithms," *Scientific American*, pp.44-50, July 1992.
5. C.L. Karr, L.M. Freeman and D.L. Meredith, "Genetic algorithm based fuzzy control of spacecraft autonomous rendezvous," *Proc. 5th Conf. on Artificial Intelligence for Space Applications*, 1990, ch62 3073, pp.43-51.
6. C.L. Karr, "Design of an adaptive fuzzy logic controller using a genetic algorithm," *Proc. 4th Int. Conf. on Genetic Algorithms*, 1991, pp.450-457.
7. A. Homaifar and E. McCormick, "Full design of fuzzy controllers using genetic algorithms," *Proc. SPIE Conf. on Neural and Stochastic Methods in Image and Signal Processing*, 1992, vol.1766, pp.393-404.
8. P. Wang and D.P. Kwok, "Optimal fuzzy PID control based on genetic algorithm," *Proc. 1992 Int. Conf. on Industrial Electronics, Control, Instrumentation and Automation*, 1992, ch286, vol.3, pp.977-981.
9. C.L. Karr and E.J. Gentry, "Fuzzy control of pH using genetic algorithms," *IEEE Trans. Fuzzy Systems*, vol.1, no.1, pp.46-53, Jan. 1993.
10. D.E. Goldberg, *Genetic algorithms in search, optimization and machine-learning*, Reading MA: Addison-Welsey, 1989.