Fuzzy Control of pH Using Genetic Algorithms

Charles L. Karr and Edward J. Gentry

***Abstract*— Establishing suitable control of pH, a requirement in a number of mineral and chemical industries, poses a difficult problem because of inherent nonlinearities and frequently changing process dynamics. Researchers at the U.S. Bureau of Mines have developed a technique for producing adaptive fuzzy logic controllers (FLC's) that are capable of effectively managing such systems. In this technique, a genetic algorithm (GA) alters the membership functions employed by a conventional FLC, an approach that is contrary to the tactic generally used to provide FLC's with adaptive capabilities in which the rule set is altered. GA's are search algorithms based on the mechanics of natural genetics that are able to rapidly locate near-optimal solutions to difficult problems. The Bureau-developed technique is used to produce an adaptive GA-FLC for a laboratory acid–base experiment. Nonlinearities in the laboratory system are associated with the logarithmic pH scale (pH is proportional to the logarithm of H_3O^+ ions) and changing process dynamics are introduced by altering system parameters such as the desired set point and the concentration and buffering capacity of input solutions. Results indicate that FLC's augmented with GA's offer a powerful alternative to conventional process control techniques in the nonlinear, rapidly changing pH systems commonly found in industry.**

I. Introduction

FUZZY logic controllers (FLC's) are being used successfully in an increasing number of application areas, including cement kiln control, task scheduling, and robot arm manipulation [1], [2]. These rule-based systems incorporate *fuzzy linguistic variables* into their rule set to model a human's "rule-of-thumb" approach to problem solving. FLC's include rules to direct the decision process and membership functions to convert linguistic variables into the precise numeric values required in most application areas. Since both the rule set and the membership functions play an integral role in determining the control action prescribed, an FLC can be provided with adaptive capabilities by altering either the rule set or the membership functions. Adaptive controllers are becoming more and more necessary in today's industry because of the economizing and streamlining of plant systems being forced on industry by current market conditions.

Attaining and maintaining control of pH is a problem that exemplifies the need for an adaptive controller. The problem of effectively manipulating pH is apparent in a number of diverse industries including mineral processing [3], wastewater treatment [4], and chemical processing [5]. Adaptive controllers are well suited for managing pH because of the nonlinearities and changing process dynamics associated with pH environments. The nonlinearities occur because of the logarithmic pH scale (pH is proportional to the logarithm of H_3O^+ ions) while the changing process dynamics occur through frequent alterations in the concentrations and buffering capacities of the input streams. Despite the difficult nature of the problem, adaptive FLC's have been successfully used for the efficient manipulation of pH systems [6], [7].

The approach that has generally been adopted for producing adaptive FLC's is to alter the associated rule set. Procyk and Mamdani [8] developed an adaptive FLC that utilizes the system Jacobian (composed of derivatives) to alter its rule set. Unfortunately, determining the system Jacobian for complex industrial systems can be extremely difficult if not impossible. Galluzzo *et al.* [6] developed an adaptive pH FLC that altered its rule set via a set of metarules that take into account information concerning the performance of their controller. These systems, and systems like them, have performed well and show great promise in the area of industrial process control.

Researchers at the U.S. Bureau of Mines have developed a technique for producing adaptive FLC's in which a genetic algorithm (GA) is employed to alter membership functions. GA's are search algorithms based on the mechanics of natural genetics that are capable of rapidly locating near-optimal solutions to difficult problems [9]. They have a demonstrated ability to alter membership functions in response to changes in the problem environment to produce more efficient FLC performance [10].

In this paper, the Bureau-developed technique is used to produce an adaptive GA FLC for a laboratory acid–base system that is an extension of the system studied by Galluzzo and his coworkers [6]. Nonlinearities in the laboratory system are associated with the logarithmic pH scale, and the changing process dynamics are introduced by altering system parameters such as the desired set point and the concentration and buffering capacity of input solutions. Although the current paper is not meant as a direct comparison of the two controllers, it will demonstrate the effectiveness of this alternative approach to developing adaptive FLC's while demonstrating the potential of adaptive FLC's in the volatile environment associated with pH systems.

II. Physical System

The laboratory pH system considered here was selected to be representative of pH systems present in a number of mineral and chemical industries [3], [11]. It contains both nonlinearities and changing process dynamics and is an extension of the system studied by Galluzzo and his coworkers [6]. The

Manuscript received May 15, 1992; revised August 22, 1992.

C. L. Karr is with the Tuscaloosa Research Center, U.S. Bureau of Mines, Tuscaloosa, AL 35486-9777.

E. J. Gentry was with the Tuscaloosa Research Center, U.S. Bureau of Mines, Tuscaloosa, AL. He is now with SEER Technologies, Cary, NC.

IEEE Log Number 9205332.

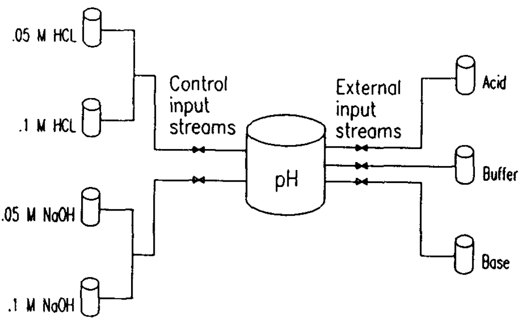

Fig. 1. The physical pH system includes nonlinearities and changing process dynamics.

nonlinearities are due to the fact that the output of pH sensors is proportional to the logarithm of concentration, while the changing process dynamics can occur in three ways. First, there is a mechanism in place for introducing a buffer into the system which significantly alters the manner in which the pH responds as acid or base is added. Second, the concentrations of the acid and the base that the controller uses to manipulate the pH of the system can be altered. Third, the set point of the system, the desired pH, can be altered. The system studied by Galluzzo and coworkers included a mechanism for the addition of a buffer. However, the inclusion of a means for changing the concentration of the input acid and base manipulated by the controller and for changing the system set point makes the system considered here considerably more difficult to control.

A schematic of the particular pH system considered is shown in Fig. 1. The system consists of a beaker initially containing a given volume of a solution having some known pH. There are five valved input streams into the beaker. Only the valves on the two control input streams can be adjusted by the controller. These two control input streams can be changed (by an "external agent") to be either 0.1 M HCl or 0.05 M HCl and 0.1 M NaOH or 0.05 M NaOH. The other three valved input streams are used to manipulate external streams which are altered by the same external agent that can manipulate the concentration of the control input streams. Of the three external streams, one is 0.05 M HCl, one is 0.05 M CH_3COONa, and one is a buffer (a combination of 0.1 M CH_3COOH and 0.1 M CH_3COONa). Additionally, the aforementioned external agent is capable of changing the desired set point to which the system pH is to be driven.

The objective of the control problem is to drive the pH of the solution to the desired set point in the shortest time possible by adjusting the valves on the two control input streams. Furthermore, the valves on the input streams are to be fully closed after the solution reaches the desired pH. As a constraint on the control problem, the valves can only be adjusted a limited amount (0.5 mL/s/s, which is 20% of the maximum flow rate of 2.5 mL/s), thereby restricting the pressure transients in the associated pumping systems.

The pH system was designed on a small scale so that experiments could be performed in a laboratory of limited space. Titrations were performed in a 1000 mL beaker with a magnetic stirring bar in the solution. Computer-driven peristaltic pumps were used for the five input streams. An industrial pH electrode and transmitter sent signals through an analog-to-digital board to a 33 MHz 386 personal computer, which controlled the entire system.

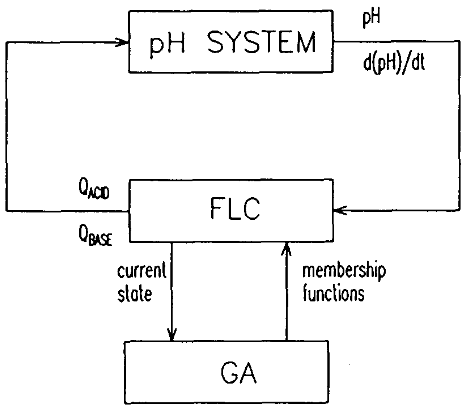

Fig. 2. A GA searches for high-performance membership functions while the FLC manipulates the pH system.

To develop an adaptive FLC using GA's, a computer model of the physical system is required. This is due to the way in which the adaptive FLC is designed. While the FLC is actually manipulating the laboratory pH system, a GA searches for improved membership functions by performing simulations of the pH system using a computer model. The GA's search is ideally performed in parallel with the FLC's manipulation of the laboratory system. However, because of limitations in computer hardware, this parallelism must often be simulated. Fig. 2 shows a schematic of the basic design of an adaptive FLC that uses a GA for membership function selection. Fortunately, the dynamics of the pH system are well understood and can be modeled using conventional techniques for buffered reactions [12]. The result is the following cubic equation that must be solved for $[H_3O^+]$ ions, which directly yields the pH of the solution:

$$x^3 + Ax^2 + Bx + c = 0, \tag{1}$$

where

$x = [H_3O^+]$
$A = k_a + [CH_3COONa] + [NaOH] - [HCl]$
$B = k_a[NaOH] - k_a[HCl] - k_a[C_H3OOH] - k_w$
$C = -k_a k_w$
$k_a = 1.8 \times 10^{-5}$ is the equilibrium constant for CH_3COOH
$k_w = 1.0 \times 10^{-14}$ is the equilibrium constant for H_2O.

with the terms in brackets representing molar concentrations. Further details of the computer model appear in a paper by Karr and Gentry [7].

In the pH system considered here, the development of a model of the physical system does not present an insurmountable obstacle. However, it should be pointed out that for many complex industrial systems, the development of an accurate computer model is, at the very least, an imposing task. Although this requisite model can be a hindrance to the application of the technique in which a GA is used to locate membership functions for use in an adaptive FLC, the use of this technique circumvents the problems associated with certain other techniques devised for developing adaptive FLC's. For example, according to Galluzzo *et al.* [6], the writing of metarules for their adaptive pH FLC "is a complex process requiring a long and difficult trial-and-error adjustment procedure...". As will be seen shortly, once a computer model of the physical system has been developed and a crude FLC has been written, a GA is employed to produce an efficient, robust, adaptive controller. Before the details of the adaptive GA FLC are presented, some of the special characteristics of GA's are discussed.

III. Genetic Algorithms Differ from Other Search Procedures

Later in this paper the details of using a GA to produce an adaptive FLC for the pH system are described. But first, consideration is given to the ways in which GA's differ from more conventional search techniques. In this section, some of these differences are discussed, and the following section presents a gentle introduction to the operation of GA's.

GA's are powerful search algorithms based on the mechanics of natural genetics. They ensure the proliferation of quality solutions while investigating new solutions via a systematic information exchange that utilizes probabilistic decisions. It is this combination which allows GA's to exploit historical information to locate new points in the search space with expected improved performance.

GA's are unlike many conventional search algorithms in the following ways:

1) GA's consider many points in the search space simultaneously, not a single point;
2) GA's work directly with strings of characters representing the parameter set, not the parameters themselves;
3) GA's use probabilistic rules to guide their search, not deterministic rules.

GA's consider many points in the search space simultaneously and therefore have a reduced chance of converging to local optima. In most conventional search techniques, a single point is considered based on some decision rule. These methods can be dangerous in multimodal (many peaked) search spaces because they can converge to local optima. However, GA's generate entire populations of points (coded strings), test each point independently, and combine qualities from existing points to form a new population containing improved points. Aside from producing a more global search, the GA's simultaneous consideration of many points makes it highly adaptable to parallel processors since the evaluation of each point requires an independent computation.

GA's require the natural parameter set of the problem to be coded as a finite length string of characters. The parameter sets in this study, which consist of an entire set of fuzzy membership functions describing both condition and action variables, are coded as strings of 0's and 1's. For example, one of the parameters considered in this study is a point that defines the upper limit of a particular trapezoidal membership

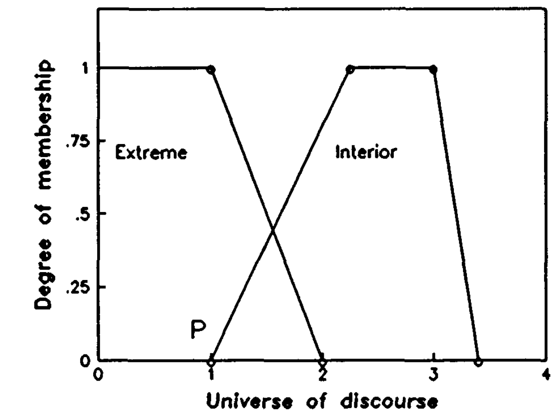

Fig. 3. Four points such as the one marked P must be defined for each trapezoidal membership function describing both condition and action variables.

function, P, as shown in Fig. 3. Note that four points are required to completely define a single trapezoidal membership function. This parameter may easily be represented as a binary string. Seven bits are allotted for defining each of the four points needed to completely define each trapezoidal membership function. The choice of seven bits was not arbitrary; in the current problem, seven bits allowed for adequate "resolution" while preventing the problem from becoming too large to comfortably handle [9]. These seven bits are interpreted as a binary number (0000011 being the binary number 3). This value is mapped linearly between some user-determined minimum ($P_{\min}$) and maximum ($P_{\max}$) values according to the following:

$$P = P_{\min} + \frac{b}{(2^m - 1)}(P_{\max} - P_{\min}), \tag{2}$$

where P is the value of the parameter being coded and b is the integer value represented by an m bit string.

This coding technique provides a convenient means of representing a single parameter: one of four points needed to define a single membership function. (Four points are needed to represent each membership function when trapezoidal membership functions are used, while only three points are needed to represent individual triangular membership functions.) However, each string must represent an entire set of membership functions: each membership function describing all of the condition and action variables. This multiple parameter coding may be accomplished by coding each of the necessary parameters in the manner described above (the string lengths for individual parameters may vary) and concatenating the bit strings. Thus, the first seven bits in the binary strings may represent the lowermost limit on the trapezoidal membership function defining pH as LOW, while the second seven bits may represent the second point defining the same fuzzy membership function, and so on until all of the membership functions used in the FLC are represented. This type of coding is relatively common and is called a concatenated, mapped, unsigned binary coding. The number of parameters that must be coded is mentioned in a later section.

In other problems, the creation of appropriate finite string codings may require more complicated mappings, but with a little creativity the possibilities are endless. Since GA's work directly with a coding of the parameter set and not the parameters themselves, they are difficult to fool because they are not dependent upon continuity of the parameter space.

A GA requires only information concerning the quality of the solution produced by each parameter set (objective function values). This differs from many optimization methods which require derivative information or, worse yet, complete knowledge of the problem structure and parameters. Since GA's do not require such problem-specific information, they are more flexible than most search methods.

Last, GA's differ from a number of search techniques in that they use random information to guide their search. Although random choices are used to define their decision rules, GA's are not random walks through the search space. They use random choice efficiently in their exploitation of prior knowledge to rapidly locate near-optimal solutions.

IV. The Mechanics of a Simple GA

A simple GA that has given good results in a variety of engineering problems is composed of three operators: (1) reproduction, (2) crossover, and (3) mutation. These operators are implemented by performing the basic tasks of copying strings, exchanging portions of strings, and generating random numbers—tasks that are easily performed on a computer. Before looking at the operators, consider the overall processing of a GA during a single generation. The GA begins by randomly generating a population of N strings, each of length m. Keep in mind that each string represents one possible solution to the problem, one particular set of all of the fuzzy membership functions employed by the FLC. Each of these strings is decoded so that the character strings yield the actual parameters (the fuzzy membership functions). The parameters are sent to a mathematical model of the pH system, evaluated with some objective function (told how good an FLC the parameters produce), and assigned a fitness value. This fitness value is a reward based on the quality of the solution represented by the string, or it can be thought of as a measure of how well a FLC using the associated membership functions actually controls the pH system. Fitness values are then used when employing the three operators that produce a new population of strings (a new generation); this procedure will be explained shortly. On average, this new generation will contain more efficient membership functions. The new strings are again decoded, evaluated, and transformed using the basic operators. The process continues until convergence is achieved or a suitable solution is found.

Reproduction is simply a process by which strings with large fitness values, good solutions to the problem at hand, receive correspondingly large numbers of copies in the new population. In this study use is made of tournament selection [13]. In tournament selection, pairs of strings compete with each other on a head-to-head basis for the right to be reproduced in the next generation. The participants in these competitions are selected based on the relative fitnesses of the strings, manifesting a "survival-of-the-fittest" atmosphere. The fitness values of two strings that are adjacent in the population (adjacent positions which are determined randomly) are compared. The string with the "best" fitness value is selected. An advantage of tournament selection is that "best" can be defined as "highest" for a maximization problem or as "lowest" for a minimization problem. Thus, tournament selection yields a linear rank-based selection. Actually, the particulars of the reproduction scheme are not critical to the performance of the GA; virtually any reproduction scheme that biases the population toward the fitter strings works well. Once the strings are reproduced, or copied for possible use in the next generation, they are placed in a mating pool where they await the action of the other two operators.

Strings exchange information via probabilistic decisions by the second operator, crossover. Crossover provides a mechanism for strings to mix and match their desirable qualities through a random process. After reproduction, simple crossover proceeds in three steps. First, two newly reproduced strings are selected from the mating pool produced by reproduction. Second, a position along the two strings is selected uniformly at random. This is illustrated below where two binary coded strings, A and B, of length 16 are shown aligned for crossover:

$$A = \overline{1\,1\,1\,1\,1\,1\,1\,1\,1}\mid 1\,1\,1\,1\,1\,1\,1$$
$$B = 0\,0\,0\,0\,0\,0\,0\,0\,0\mid \underline{0\,0\,0\,0\,0}\,0\,0.$$

Notice how crossing site 9 has been selected in this particular example through random choice, although any of the other 14 positions were just as likely to have been selected. The third step is to exchange all characters following the crossing site. The two new strings following this crossing are shown below as A' and B':

$$A' = \overline{1\,1\,1\,1\,1\,1\,1\,1\,1}\mid 0\,0\,0\,0\,0\,0\,0$$
$$B' = 0\,0\,0\,0\,0\,0\,0\,0\,0\mid \underline{1\,1\,1\,1\,1}\,1\,1.$$

String A' is made up of the first part of string A and the tail of string B. Likewise, string B' is made up of the first part of string B and the tail of string A. Although crossover uses random choice, it should not be thought of as a random walk through the search space. When combined with reproduction, it is an effective means of exchanging information and combining portions of high-quality solutions.

Reproduction and crossover give GA's the majority of their search power. The third operator, mutation, enhances the ability of the GA to find near-optimal solutions. Mutation is the occasional alteration of a value at a particular string position. It is an insurance policy against the permanent loss of any simple bit. A generation may be created that is void of a particular character at a given string position. For example, a generation may exist that does not have a 1 in the third string position when, due to the chosen coding, a 1 at the third position may be critical to obtaining a quality solution. Under these conditions, neither reproduction nor crossover will ever produce a 1 in this third position in subsequent generations. Mutation, however, causes a 0 in the third position to occasionally be changed to a 1. Thus, the critical piece of information can be reinstated into

TABLE I
THE ENTIRE RULE SET CONSISTS OF FOURTEEN RULES THAT ACCOUNT FOR ALL OF THE POSSIBLE CONDIDITIONS AS SET FORTH BY THE SELECTED MEMBERSHIP FUNCTIONS

		ΔpH	
		S	L
pH	VA	Z	Z
	A	Z	Z
	MA	Z	Z
	N	Z	Z
	MB	S	VS
	B	M	S
	VB	L	M

		ΔpH	
		S	L
pH	VA	L	M
	A	M	S
	MA	S	VS
	N	Z	Z
	MB	Z	Z
	B	Z	Z
	VB	Z	Z

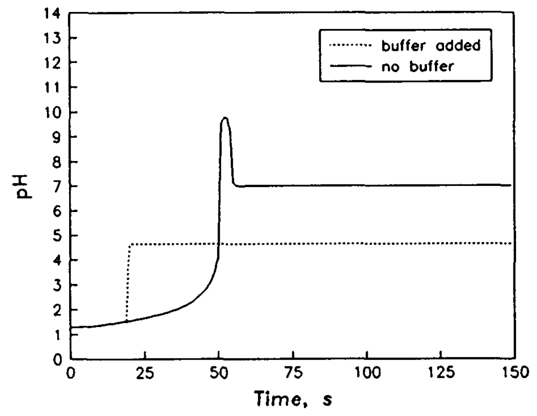

Fig. 4. A nonadaptive FLC performs well when no buffer is added, but is unable to accommodate the changing process dynamics associated with buffer addition.

the population. Although mutation can serve a vital role in a GA, it should be noted that it occurs with a small probability (on the order of one mutation per thousand string positions), and is secondary to reproduction and crossover.

This has been a brief overview of a simple three-operator GA. For details of the processing power of GA's and fitness assignment in GA's, reference should be made to [14]. Goldberg [9] presents a thorough overview of the operation of GA's including discussions of higher-order operators that have been used to enhance the searching power of GA's. In the next section, the application of a GA to the problem of selecting membership functions for the pH FLC "on-line" is presented.

V. AN ADAPTIVE pH GA FLC

At this point, most of the groundwork has been laid for the development of an adaptive pH GA FLC. The mechanics of a GA have been introduced, and the basic coding scheme that will be used has been described. Before the details of using a GA to produce an adaptive FLC are presented, the characteristics of the rudimentary pH FLC are expounded.

The FLC used rules of the form

IF ⟨*condition*⟩ THEN ⟨*action*⟩

where the *condition* portion of the rules was composed of the current difference between the set point and the actual pH value (*Error*), and the current time rate of change of this difference (δ*Error*). The *action* portion of the rules was composed of the change in the volumetric flow rates of the input acid and base (Q_{ACID} and Q_{BASE}).

Seven linguistic terms were used to characterize *Error*, two linguistic terms were used to characterize δ*Error*, and five terms were used to characterize both Q_{ACID} and Q_{BASE}. Thus, the FLC was composed of 14 rules, which allowed for the inclusion of all possible combinations of the condition variables as described by the chosen linguistic variables. The complete rule set is included in Table I. A standard center of area method was used for defuzzification.

The nonadaptive pH FLC produced using the above guidelines performs well on the pH system—until the system becomes dynamic. Fig. 4 shows the performance of the FLC for a situation in which the pH system is "settled," i.e., the system dynamics are not changing, and for a situation for which buffer solution is added to the system via the external agent. Certainly, the FLC performs well when the process dynamics remain constant. However, when the system dynamics are altered, when the buffer is added, the performance of the nonadaptive controller deteriorates rapidly. This is due to the fact that the nonadaptive FLC has no mechanism to accommodate the changes in system response caused by the addition of the buffer.

The changing system dynamics can be accounted for by allowing the FLC to alter its conception of the terms used in its rule set. This is accomplished by altering the membership functions associated with the action and control variables in response to changes in the pH environment such as the addition of the buffer or changing concentration of the control input streams. In this work, a GA is used for this task.

As was pointed out earlier, there are basically two decisions to be made when using a GA in a search problem. The issue of coding has been addressed. Thus, consider the second decision: how to evaluate the merit of each string (each membership function set). This task of defining a fitness function is always application specific; it always comes down to accurately describing the goal of the controller. In this case, the objective of the controller is to drive the pH to the desired set point in the shortest time possible and to maintain the system pH at the desired setpoint. The ability of the FLC to achieve these objectives can be represented by a fitness function that specifies how well the controller reduces the error over some finite time period. Additionally, to ensure that a robust FLC is obtained, the fitness function should reflect the controller's ability to reach the set point from a number of initial condition cases. Mathematically, this fitness function is expressed as

$$f = \sum_{i=\text{case }1}^{\text{case }4} \sum_{j=0\text{ s}}^{30\text{ s}} (Error)^2.$$

The initial condition cases were selected from different regions of the state space of the pH control system.

Using the aforementioned coding and fitness function, an adaptive pH GA FLC was produced. Each time a change was detected in the pH system, a GA was used to locate a new, better performing set of membership functions. The performance of this adaptive pH GA FLC is demonstrated for three different situations. First, the pH system is perturbed with the external addition of an acid, a base, and a buffer. In this case, the process dynamics are dramatically altered by the addition of the buffer. Second, the desired set point is altered. This, for all intents and purposes, changes the objective of the controller. Third, the concentrations of the acid and base the FLC uses to control pH are changed, which causes the system to handle differently. For example, if the 0.1 MHCl is the control input, the pH falls a certain amount when this acid is added. However, all other factors being the same, the pH will not fall as much when the same volume of the 0.05 M HCl is added. These three scenarios provide a challenging test-bed for any control system.

Consider first a situation where a buffer is added to the pH system by an external agent. The adaptive pH GA FLC alters the membership functions it uses to enact its production rules (which do not change) because the process dynamics are altered when the buffer is added. Fig. 5 compares the performance of the adaptive GA FLC with a nonadaptive FLC that does not employ a GA. The adaptive controller is able to achieve the objective much more efficiently than the nonadaptive FLC because the adaptive controller accounts for the fact that the dynamics of the problem have changed.

Next, consider a situation where the set point is changed by the external agent. Again, the adaptive pH GA FLC alters its membership functions in response to an "environmental" change. It is important to realize that declaring a new set point is actually changing the objective of the FLC. Changing the objective of the controller often requires a modification of the FLC rule set. However, the technique of using a GA to alter membership functions is powerful enough to allow the FLC to maintain a suitable level of control over the pH system despite the demanding environment in which it must operate. Fig. 6 compares the performance of an adaptive GA FLC with a nonadaptive FLC. As in the previous example, the adaptive pH GA FLC outperforms the nonadaptive FLC.

Finally, consider a very disruptive change to the pH system. Consider a case where the concentrations of the acid and base the FLC is using to manipulate the pH are altered. This is perhaps the most severe change in process dynamics that could be implemented. The response of the system is now completely different: additions of acid or base induce changes in the pH of the system that are far different from the changes in pH that the very same additions of acid or base induced before the concentrations were changed. Fig. 7 compares the performance of the adaptive pH GA FLC with the performance of a nonadaptive FLC. The adaptive GA FLC is able to maintain a high degree of control over the pH system despite the dramatic changes in the environment.

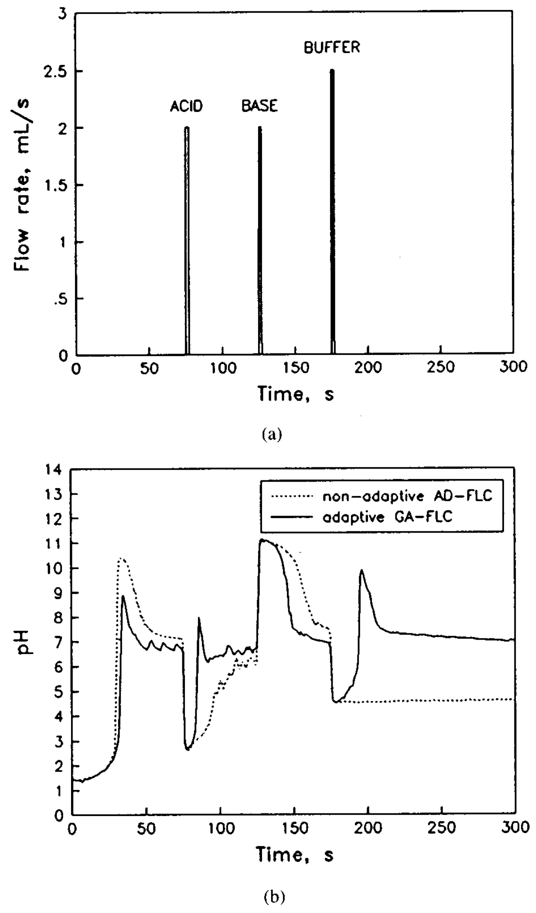

Fig. 5. (a) The external agent added acid, base, and buffer. (b) The adaptive GA FLC was able to drive the system pH to the set point of seven and to maintain the set point better than the nonadaptive FLC.

The results presented in this section demonstrate much of the power of adaptive FLC's. They are able to maintain a high degree of control over the laboratory pH system despite drastic changes in the system characteristics. Perhaps the most intriguing aspect of this application is the fact that the system parameters that were altered do not appear in the rule set the FLC employs. All changes in FLC performance are due to alterations in the membership functions. Based on the results presented for the three scenarios considered, adaptive GA FLC's allow the volatile characteristics of many industrial pH systems to be controlled via on-line changes to the membership functions.

VI. Summary

Scientists at the U.S. Bureau of Mines have developed a technique for producing adaptive FLC's in which GA's are used to alter fuzzy membership functions on-line. In this

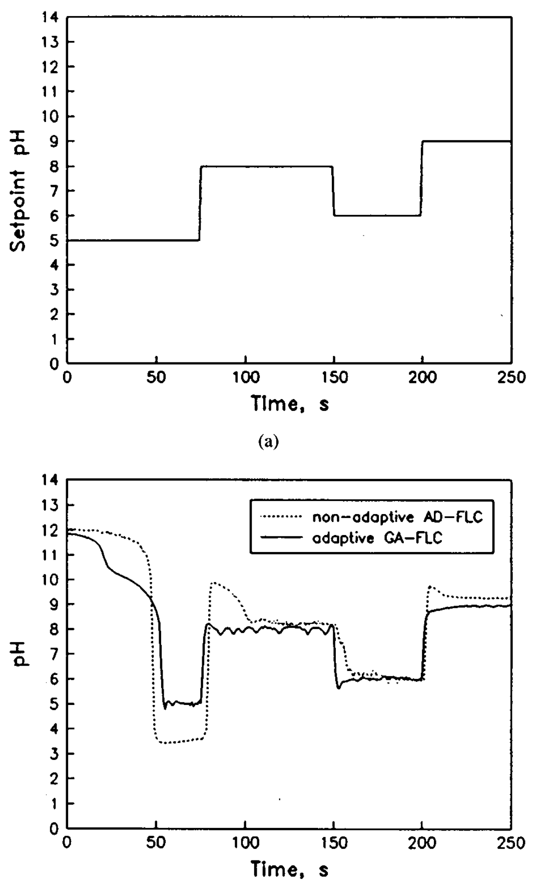

Fig. 6. (a) Changes in the system set point can create problems for a nonadaptive FLC. (b) The adaptive GA FLC is able to adjust to changes in the system set point more efficiently than a nonadaptive FLC.

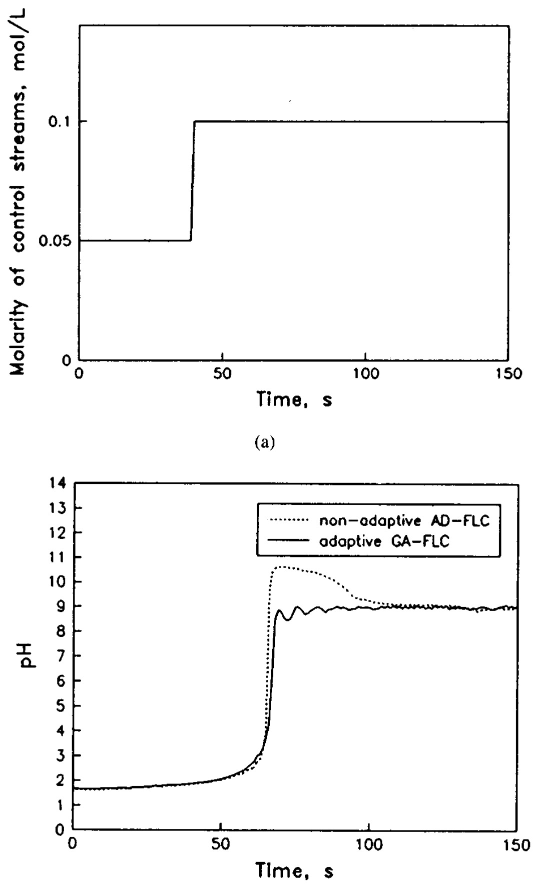

Fig. 7. (a) Changes in the concentrations of the control input streams dramatically alter the system dynamics. (b) The adaptive GA FLC is able to reach the system set point faster than a nonadaptive FLC.

paper, the technique has been applied to a laboratory acid–base system in which the process dynamics change in several different ways. Initially, an overview of GA's is provided, including a discussion of how they differ from conventional search techniques and details of their basic mechanics. Next, a GA is used to produce an adaptive FLC that is able to accommodate the introduction of a buffer, changes in the desired system set point, and alterations in the concentrations of acid and base used to drive the system pH. In all instances, the adaptive GA FLC was able to successfully drive the system pH to the desired set point in a reasonable time.

The approach that is presented in this paper differs from most of the work that has been done in the development of adaptive FLC's. For the most part, researchers have focused on altering the rule set associated with a FLC instead of taking the approach presented here, which is to modify the membership functions that drive the controller. The results presented demonstrate the effectiveness of altering the membership functions for a number of situations that are difficult to control. The adaptive pH GA FLC is an effective system that deserves more consideration.

FLC's have become increasingly popular as solutions to process control problems. However, if they are going to continue to grow in popularity, and if they are going to provide practical solutions to such complex process control problems as those found in the mineral and chemical industries, techniques must be developed that will provide FLC's with adaptive capabilities. This paper has presented results obtained from actual laboratory tests of a physical acid–base system. Based on these results, it is concluded that GA's represent such a technique and can provide a valuable resource for the design of FLC's.

References

[1] G. W. Evans, W. Karwowski, and M. R. Wilhelm, Eds., *Applications of Fuzzy Set Methodologies in Industrial Engineering.* Amsterdam: Elsevier, 1989.
[2] I. B. Turksen, Ed., *Proc. NAFIPS'90—A Quarter Century of Fuzziness.* Toronto: North American Fuzzy Information Society, 1990.
[3] E. G. Kelly, and D. J. Spottiswood, *Introduction to Mineral Processing.* New York: John Wiley, 1982.
[4] W. W. Gottinger, *Economic Models and Applications of Solid Waste Management.* New York: Gordon and Breach Science Publishers, 1991.

[5] H. S. Fogler, *Elements of Chemical Reaction Engineering.* Englewood Cliffs, NJ: Prentice-Hall, 1986.
[6] M. Galluzzo, V. Cappellani, and U. Garofalo, "Fuzzy control of pH using NAL," *Int. J. Approx. Reasoning*, vol. 5, no. 6, pp. 505–519, 1991.
[7] C. L. Karr, and E. J. Gentry, "Real-time pH control using fuzzy logic and genetic algorithms," in *Proc. Ann. Meeting of the SME*, to be published.
[8] T. J. Procyk, and E. H. Mamdani, "A linguistic self-organizing process controller," *Automatica*, vol. 15, pp. 15–30, 1978.
[9] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning.* Reading, MA: Addison-Wesley, 1989.
[10] C. L. Karr, "Genetic algorithms for fuzzy logic controllers," *AI Expert*, vol. 6, no. 2, pp. 26–33, 1991.
[11] T. J. McAvoy, E. Hsu, and S. Lowenthal, "Dynamics of pH in controlled stirred tank reactor," *Industrial Engineering and Chemical Process Design Developments*, vol. 11, no. 1, pp. 68–70, 1972.
[12] C. W. Hand, and G. L. Blewitt, *Acid-Base Chemistry.* New York: Macmillan, 1986.
[13] C. L. Karr, "Analysis and optimization of an air-injected hydrocyclone," Ph.D. thesis, University of Alabama, Tuscaloosa, AL, 1989.
[14] J. H. Holland, *Adaptation in Natural and Artificial Systems.* Ann Arbor, MI: University of Michigan Press, 1975.

Charles L. Karr received the Ph.D. in engineering mechanics from the University of Alabama in 1989.

He is a mechanical engineer with the Tuscaloosa Research Center, U.S. Bureau of Mines, Tuscaloosa, AL, where he is involved in research on intelligent process control. His interests include fuzzy logic, genetic algorithms, machine learning, and computational fluid dynamics.

Edward J. Gentry received his B.S. degree in computer science from the University of Alabama in May 1992.

He is currently a solfware developer for SEER Technologies, Cary, NC. His interests include fuzzy logic controller design and application- and event-driven object-oriented systems.