Design Issues of a Reinforcement-based Self-Learning Fuzzy Controller for Petrochemical Process Control

John Yen, Haojin Wang and Walter C. Daugherity

Center for Fuzzy Logic and Intelligent Systems Research
Department of Computer Science
Texas A&M University
College Station, TX 77843-3112

Abstract

Fuzzy logic controllers have some often-cited advantages over conventional techniques such as PID control, including easier implementation, accommodation to natural language, and the ability to cover a wider range of operating conditions. One major obstacle that hinders the broader application of fuzzy logic controllers is the lack of a systematic way to develop and modify their rules; as a result the creation and modification of fuzzy rules often depends on trial and error or pure experimentation. One of the proposed approaches to address this issue is a self-learning fuzzy logic controller (SFLC) that uses reinforcement learning techniques to learn the desirability of states and to adjust the consequent part of its fuzzy control rules accordingly. Due to the different dynamics of the controlled processes, the performance of a self-learning fuzzy controller is highly contingent on its design. The design issue has not received sufficient attention. The issues related to the design of a SFLC for application to a petrochemical process are discussed and its performance is compared with that of a PID and a self-tuning fuzzy logic controller.

1 Introduction

Conventional model-based control has the advantage of stability and proved optimality within the given range of operating conditions. For this reason, Proportional-Integral-Derivative (PID) control has been a major practical control technology for a long time. However, there are some serious limitations with this approach in dealing with ill-defined, non-linear and dynamic processes. One of the problems is its lack of adaptivity to the operating environment. When the operating conditions are out of the prescribed range, human intervention is needed to manually tune and adjust the operating parameters.

In the past few years, a great deal of interest has been generated in applying fuzzy logic and approximate reasoning to industrial process control and these efforts have resulted in various techniques of fuzzy control. The basic idea of fuzzy control is to transform human expert knowledge about controlling the process into fuzzy *if-then* rules and use approximate reasoning to deal with uncertainty and to derive the control actions. The advantage of fuzzy

control is that it can capture the imprecise and uncertain aspects of human reasoning and as a result the fuzzy controller can deal with those dynamic, ill-defined or non-linear systems more efficiently than conventional approaches. Since E. Mamdani [6] applied the basic concepts of fuzzy logic, coined by Lotfi Zadeh [7], to control in early 1970s, and especially during the past decade, there has been a great deal of research activity in this field, and many techniques and architectures have been proposed or developed.

Despite the advantages of the fuzzy logic controller mentioned above, there are some problems associated with the fuzzy controller that hinders its wider application. One of the issues is its lack of a systematic way to develop and modify its rules, and as a result the creation and modification of fuzzy rules often depend on trial and error or pure experimentation. Several approaches have been proposed to address this issue. One of the proposed approaches is to use a self-learning mechanism to learn the desirability of rules and modify the consequent part of the fuzzy rules accordingly.

The issue of self-learning is to let the system itself learn the proper control actions through a given number of trials. Several techniques have been developed or proposed to accomplish the goal of self-learning during recent years. Barto et al [2] proposed a learning mechanism composed of two neuron-like elements called the adaptive critic element (ACE) and the associative search element(ASE). Lee[5] integrated this idea into a fuzzy control system and applied it to the well-known pole-balancing problem. Chen[3] used a similar approach with slight modification, and applied it to three similar dynamic process. One important aspect of SFLC that has not been properly addressed is that the performance of a self-learning fuzzy controller is application dependent and the different dynamics of the controlled process requires different treatment in the design of a SFLC. In the following sections, the issues related to the design of a SFLC in general and for a particular petrochemical process are discussed.

2 Description of the Control Process and SFLC

The control process for this research is a simple gas-fired water heater, since it is widely used in the petrochemical industry and an accurate simulation model was available. The inlet water at a certain temperature and feed rate enter a stirred tank heated by a gas burner. At a certain point downstream the outlet water temperature is measured by a sensor. The resultant time delay is known as dead time. The controller calculates the temperature difference between the current value and the desired (or "setpoint") value, i.e., the error, and adjusts the valve controlling the gas supply accordingly. The initial temperature reading of the water tank is assumed to be at room temperature level. For a more detailed description of the control process, see [4]. The control task is first to heat the tank to the desired set point and then to keep the temperature at the desired level in the presence of sensor noise and changing operating conditions.

The self-learning fuzzy controller we developed is based on the approach proposed by Sutton, Barto and Lee and it is intended for application to industrial processes in general and to petrochemical processes in particular. The controller has two major components, namely, a fuzzy control component and a learning component. The fuzzy control component consists of a rule base which has a set of fuzzy rules and a fuzzy inference mechanism that uses the fuzzy rules and applies fuzzification and defuzzification operators to the input and output to obtain the actual control action. The learning component contains two neuron-like elements. They are the adaptive critic element (ACE) and the associative search element

(ASE) respectively. Initially, the consequent part of every control rules is initialized to an arbitrary fuzzy control value. When a rule fires with non-zero firing strength, the two neuron-like elements learn the desirability of the previous control action and adjust the weights of the fired rules. The consequent part is adjusted according to its weight. The control action is the result of applying a defuzzification operator to the control commands inferred by the fired rules. For more detailed information on the implementation of this type of controller, see Barto [2] and Lee [5].

The SFLC in this research has two input variables to the controller, namely, the error (difference between the current temperature reading and the set point) and the change of error (difference between the current temperature and the one at dead time steps back). The output variable is the amount of change to the valve that controls the gas supplied to the burner.

3 Design Issues

The performance of a SFLC is highly application dependent and one of the determining factors in the design of a SFLC is the dynamics of the control process. The design issues considered in this research are the choice of a training set, the choice of feedback, and the learning parameters.

3.1 The choice of Training Cases

A SFLC first needs to go through a learning session to learn the proper control action through a certain number of trials. After learning, the controller is put into actual operation where the learned rules are applied. The issue regarding training cases is the choice of cases presented to the controller during the learning session.

The dynamics of the control process directly influences the choice of training cases. For those highly dynamic processes, what the control system encounters during the learning session tends to cover a wide range of operating conditions and consequently this results in a better knowledge base for the control system. With a broader knowledge base, the control system can perform well under the various operating conditions. Therefore, the choice of training cases is not a real issue. However for those processes which are less dynamic, it is likely that the knowledge acquired during the learning session is not sufficient to cover a wide range of operating conditions if the training cases are generated in the same fashion. Then, the choice of test cases becomes very important, because what the SFLC learns will, to a large extent, determine how it performs in the operating environment.

The central idea of the choice of training cases is to design the training cases in such a way that all operating conditions that we anticipate the control system might encounter should be covered in the learning session. One approach to accomplish this is to map the state space of the control system into a two-dimensional space like the one in Figure 1 and then design the training cases in such a way that they are complementary to each other and that taken together, they can cover most of the state space. In this figure, each numbered square corresponds to a state the control system can be in. NB, NS, ZE, PS, and PB are fuzzy sets used for the SFLC and they are abbreviations of negative big, negative small, zero, positive small and positive big respectively. For more than two state variables, we can use a similar approach to map the rule base into a hyperspace. A state space like this can be used to design the training cases for a SFLC. In this figure, each curve is the trajectory

of the fired rules for one trial of a training instance. Together, they form a region that defines a desirable performance curve for the control system. If the control system is in a state that is within the desirable performance curve, it is expected to perform well because it has the knowledge regarding this particular state in its knowledge base. However, if the control system falls out of the desirable performance curve, the performance may be poor. A simple example will illustrate this point well. If the desirable performance curve is the one shown in Figure 1, and if the initial state for the control system happens to be the one in the lower right corner of the state space where the error and the change of error both are positive big, the knowledge acquired during the learning session will not be sufficient for the control system to handle this case effectively. The goal of designing training cases is to have the desirable performance curve cover as many states as possible in the state space.

We now use the process for this research to illustrate the above idea. For the petrochemical process under consideration, the dynamics differs from the inverted pendulum that is often used to demonstrate the concept of reinforcement-based self-learning. The inverted pendulum is highly dynamic and the choice of training cases is easier because each training case tends to cover a large portion of the state space. Therefore, by randomly generating the training cases (i.e., the arbitrary initial angles and positions), the system is able to learn an appropriate response for most of the states in the state space. However, this is not the case for industrial processes in general and petrochemical processes in particular for the following two reasons:

- The slow-response nature of the process dynamics may result in smaller portion of the state space being covered during the learning session;
- With a proper choice of feedback, training a SFLC for an industrial process may not require a large number of training cases to reach the goal state.

Consequently, only a limited number of operating conditions are encountered in the learning session and the desired performance curve covers only a small portion of the state space. The choice of the training cases is thus the issue of ensuring that a large portion of the state space is covered. The suggested approach to this problem is to design multiple training cases which are complementary to each other so that taken together they can cover a large portion of the state space.

3.2 The choice of feedback

The basic idea of reinforcement learning is to use feedback from the environment to generate reinforcement signal that helps distinguish desirable states from undesirable states. The choice of feedback directly impacts the performance of a SFLC by influencing the quality of learning and the length of the learning cycle.

Reinforcement learning is implemented through the two neuron-like elements ACE and ASE. The ACE receives feedback from the environment and its main function is to provide a critique of the control action that took place at dead time steps back and in doing so, it generates an internal reinforcement signal to the ASE. The rationale is that when the process is moving from a less desirable state to a more desirable state, it should receive a positive reinforcement signal and when it moves in the opposite direction, it should receive a negative reinforcement signal. The choice of feedback directly impacts the quality and quantity of the internal reinforcement the ACE generates, and in turn it impacts the weight associated with each rule and eventually affects the control action the SFLC generates.

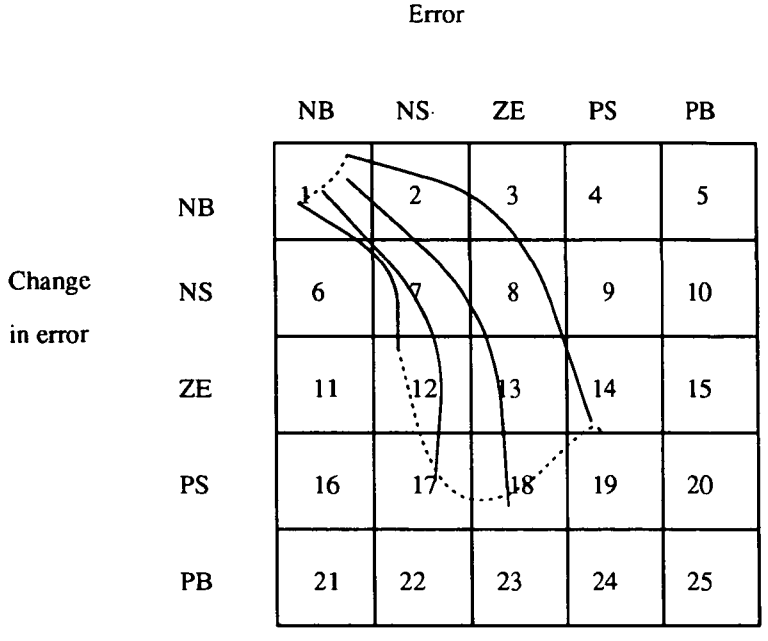

Figure 1: Trajectory of trained rule in state space

The choice of feedback is closely related to the dynamics of the control process. A difference in process dynamics may need a different choice of feedback to best suit the purpose of reinforcement. The method currently employed for the generation of feedback in the concept-demonstration control systems developed by Barto [2] and Lee [5] is to give a -1 as feedback once the control process falls into a "failure" state. All states that are outside a desirable region of the pole's angles are deemed failure states. This method does not suit petrochemical processes well for three reasons. First, there is a delay between a control action and the resultant response and the rules fired immediately before a failure state may not be the real "culprit." Secondly, for a less dynamic process, the feedback may be infrequent if only failure states cause feedback. Finally, the initial state of a training trial can be a state outside the desirable region (i.e., the initial temperature is not in $[T-\alpha, T+\alpha]$ where T is the set point and α is the threshold that specifies the desirable region) and if we give negative feedback to all the states outside the desirable region, then the process will never reach the the goal state. Taking into consideration the difference in the process dynamics, we discuss some general issues regarding the choice of feedback for reinforcement learning and then propose some design guidelines for addressing these issues.

First, we should ensure that no strong negative feedback is given while the control process is on its way to the goal state even though the intermediate states are failure states. This is one of the major differences between petrochemical processes and the processes used to demonstrate the concept of SFLC, like the inverted pendulum. On the other hand, a negative feedback should be generated once the control process falls into a failure state that is not on the desired performance curve that leads the process from the initial state to the goal state. This requires that we define the desired performance curve and distinguish those

failure states that are on the performance curve from those that aren't.

Next, the amount of domain-specific information used should be limited to a minimal level. The fundamental assumption for the self-learning fuzzy logic controller is that it should learn its own control action in the absence of knowledge about input and output relations.

Another factor is regarding how early or frequently feedback is generated. In order to shorten the learning cycle, ideally the feedback should be generated as frequently and as early as possible. However, there is a trade-off between a short learning cycle and the amount of domain specific information required. It is usually the case that to increase the frequency of feedback often requires more domain-specific information. A balance between the two can be struck depending on the availability of domain-specific information and the requirement for the length of the learning session. For instance, if the learning system is given the desired performance curve (which is highly specific to the particular process being considered), a feedback can be generated every cycle based on the distance between the current state and the corresponding state on the desired performance curve. However, such an approach is not feasible if the desired performance curve is not readily available. Under such a circumstance, a mechanism that generates feedback less frequently but relies on less domain-specific information should be used.

Having discussed these issues, we now outline some design guidelines for addressing them. First, the feedback can be expressed as a function of the factors it depends on. It can be expressed as $Feedback(S)$, where S stands for state. We generalize the notion of state-based feedback to the notion of performance-based feedback. A state-based feedback, as demonstrated by Barto, Sutton and Lee using the inverted pendulum problem, generates a feedback signal entirely based on the current state of the system. Therefore, a *performance-based feedback* incorporates the initial and goal states into the function for generating the feedback, in addition to the current state. It can be expressed as $Feedback(S, I, G)$ where I stands for initial state and G for the goal state. The advantage of this approach is its flexibility. A state can be given different feedback depending on whether it is on the desired performance curve, which is determined by the initial operating conditions and the goal state. For instance, in Figure 2, the state s is the same state for cases a and b. Because the initial states are different for the two cases, the state s is on the desired performance curve in a but it is not in case b. By incorporating the initial state into the feedback function, we are able to give different feedback for the same state s under different circumstances. A variation of this method is to use global performance history instead of a single failure state to generate the feedback. It can be expressed as $Feedback(I, G, \tilde{S})$ where I and G are same as above and $\tilde{S}$ is the global performance history, e.g., the average of all errors. A method similar to this is employed in in Y. Y. Chen's[3] system.

The second design guideline for generating feedback is to incorporate the performance objectives such as reaching time or overshoot to generate feedback. The performance objectives serve as constraints to the control process. Once the control process fails any of the performance objectives, feedback is generated. Thus, the feedback can be expressed as $Feedback(S, O_1, ..., O_n)$ where O_i represents ith performance objective. The method we employed for this research is a combination of using the global performance history and incorporating a performance objective into feedback function.

The third design guideline is to use general knowledge about the control process such as the dynamics of the process to generate feedback. This is control process dependent and detailed implementation hinges on the actual control process in question.

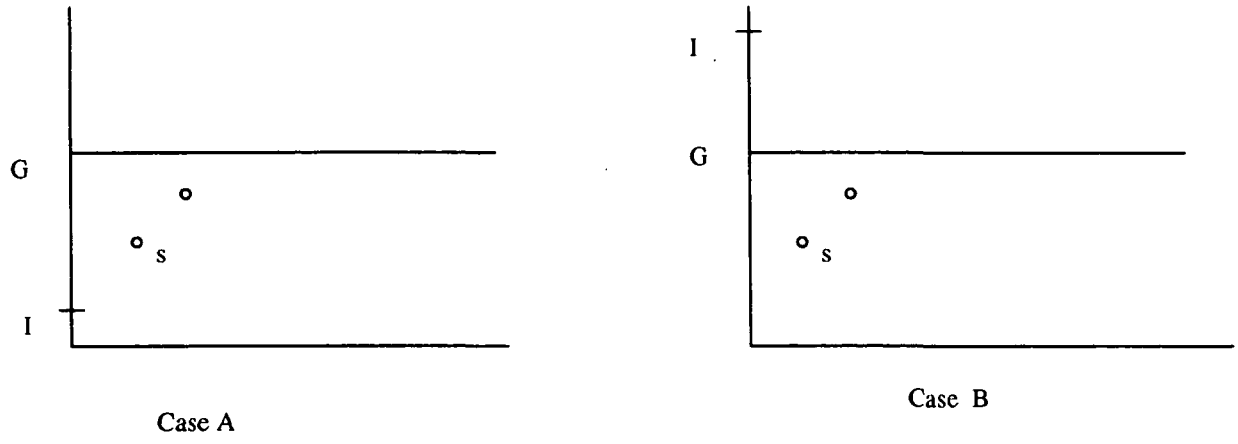

Figure 2: The two different training cases

3.3 Learning Parameters

There are many parameters for the learning rules. The values of those parameters are usually determined in a more or less trial and error fashion. It is a research issue as how to determine the parameter values systematically. Following are some observations about the relations between the parameter values and process dynamics.

Rule trace decay parameter

This determines how fast rule traces decay. Rule trace is the history of a rule's firing strength and frequency. This parameter is highly dependent on the dynamics of the control process. We observed that the more dynamic a process, the faster the decay and the less dynamic a process, the slower the decay. Intuitively, a large amount of information is likely to be required to compensate for the higher rate of decay for very dynamic processes.

Sigmoid gain parameter

This determines to what extent the weight of a rule is transferred into the consequent part of a fuzzy rule. This parameter is also highly related to the dynamics of the control process. It appears that the less dynamic the process is, the greater the sigmoid gain should be.

4 Empirical Experiments and Discussion

4.1 Description of Learning Rules

For the ACE, the learning rules are as follows:
Internal reinforcement is defined as

$$\hat{r} = r(t) + \gamma p(t) - p(t-1)$$

where $r(t)$ is feedback and $p(t)$ is total desirability of all states at time t:

$$p(t) = \sum_{i=1}^{n} v_i(t) x_i(t)$$

where v_i is the desirability of the ith state and x_i is the firing strength of the ith rule. In turn, v_i is defined by

$$v_i(t) = v_i(t-1) + \beta r(t)\bar{x}_i(t),$$

where $\bar{x}_i$ is the local memory trace defined by

$$\bar{x}_i(t) = \lambda \bar{x}_i(t-1) + (1-\lambda)x_i(t).$$

For the ASE, the learning rules are as follows:
The weight of each fuzzy rule is determined by

$$w_i(t) = w_i(t-1) + \bar{a}(t-1)\bar{r}(t)e_i(t-d),$$

where e_i is the rule trace, d is the dead time delay and $\bar{a}(t)$ is the dynamic positive learning rate. The rule trace ϵ_i is given by

$$\epsilon_i(t) = \delta \epsilon_i(t-1) + (1-\delta)y(t)x_i(t),$$

where δ is rule trace decay parameter, and

$$\bar{a}(t) = \frac{\alpha k}{k+t},$$

where α is initial value and k is a weight freeze parameter. The consequent of each fuzzy rule is determined by a sigmoid function:

$$y_i = f(w_i(t), noise(t)),$$

where the dynamic sigmoid function f is defined by
$f(x,t) = \frac{x}{T(x)+x}$ for $x > 0$
$f(x,t) = \epsilon$ for $x = 0$
$f(x,t) = \frac{x}{T(x)-x}$ for $x < 0$
where $T(x)$ is the tuning parameter defined by

$$T(x) = \kappa\ max(w_i(t))$$

where κ is the sigmoid gain parameter.

We incorporate both the performance objective, the initial state and performance history into the function to generate feedback:

$r = 0$ if the system neither fails a performance objective nor falls into a failure state;

$r = -\frac{1}{|a-b|}(\frac{1}{N}\sum_{k=1}^{N}|E(T)|)$ where a is the initial temperature which is the initial state for the process, b is the performance objective overshoot requirement and $E(T)$ is the average learning period;

$r = -1$ if $c \notin [min\{i, s-o\}, s+o]$ for $i < s$ or $c \notin [max\{i, s+o\}, s-o]$ for $i > s$. c, i, s, o represent the current state, initial state, set point and overshoot limit respectively.

Y. Y. Chen [3] used a method similar to this in form but the interpretation of a and b is different.

In the present research, the following parameter values were used: $\alpha = 0.05, \beta = 0.5, \delta = 0.93, \epsilon = 0.01, \gamma = 0.95, \kappa = 0.25. \lambda = 0.9,$

For more detailed information on the derivation of these learning rules, see Barto[2], Barto[1] and Lee[5].

4.2 Simulation Results

The work presented in this paper is the continuation of a previous project for developing a self-tuning fuzzy controller [4]. We simulated our system on a IBM PS/2 and compared its performance with that of three other control strategies: PID, the previously developed self-tuning fuzzy controller and a bare-bones fuzzy controller (without any learning or tuning). The SFLC is trained for 200 time steps, a set point of 200, and feed rate of 10 gallons per minute.

The general performance of the four regimes is shown in Figure 3. This is the performance of the controllers without any variation in operating parameters. We can see the self-learning scheme shows a performance very similar to that of the PID and the self-tuning scheme has a faster reaching time but slightly more overshoot.

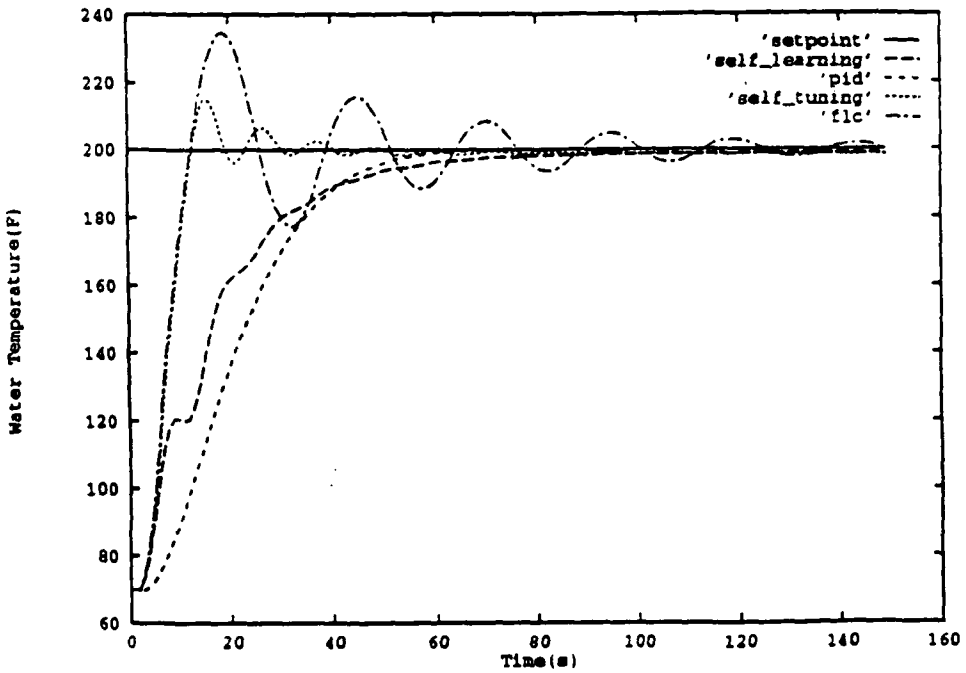

Figure 3: Four different control schemes

The advantages of the self-learning controller over the other schemes are demonstrated in three aspects. First is the short learning cycle. Four learning trials were sufficient for the system to reach the required performance level. Second, the stability of performance. Figure 4 shows the effects of changing the feed rate on the overshoot for PID and self-tuned systems. The SFLC has very little overshoot while varying the feed rate from 2 to 20 gallons per minute.

The third advantage is the wide range of operating conditions. When we varied the feed rate from 2 to 20 gallons per minute, the SLFC can perform as well as under normal conditions with very little fluctuation in performance and more importantly, without any re-training. Figure 5 shows the number of retraining steps needed for various feed rates.

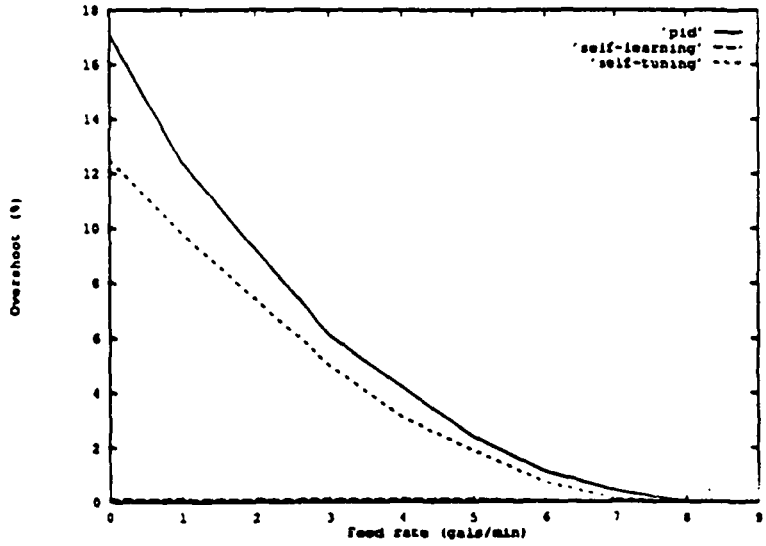

Figure 4: Effect of change of feed rate on overshoot

When the feed rate changes to 25 gallons per minute, the number of training steps needed increased only slightly.

5 Summary

In summary, we have discussed some of the design issues for a reinforcement-based self-learning fuzzy controller for application to a petrochemical process based on the approaches proposed by Barto and Lee. The main issues were the choice of training cases and the choice of feedback. Simulation results show that it has some advantages as discussed above over other schemes and that the choice of training cases and feedback has direct impact on the performance of a SFLC. Some issues such as finding a systematic way to determine the parameter values will be considered in future research.

6 Acknowledgement

Balaji Rathakrishnan was involved in the early stage of this project. His contribution is greatly appreciated and acknowledged here. Our research also benefited from discussion with C. C. Lee (Sony). Texaco provided the simulator of the water heating system used in this research.

feed rate	self-tuning FLC	self-learning FLC
2.0	2	0
5.0	1	0
8.0	0	0
15.0	1	0
20.0	1	0
25.0	1	2
30.0	1	2

Figure 5: Tuning/learning steps needed for new feed rates

References

[1] A. G. Barto and R. S. Sutton. Associative search network: a reinfrocement learning associative memory. *Biol. Cybern.*, 40:213–246, 1981.

[2] A. G. Barto, R. S. Sutton, and C. W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transaction on Systems, Man and Cybernetics*, 13:834–846, 1983.

[3] Y. Y. Chen, K. Z. Lin, and S. T. Hsu. A self-learning fuzzy controller. In *IEEE International Conference on Fuzzy Systems*, pages 189–196, March 1992.

[4] Walter Daugherity, Balaji Rathakrishnan, and John Yen. Performance evaluation of a self-tuning fuzzy controller. In *Proceedings of IEEE Internaltional Conference on Fuzzy Systems (FUZZY-IEEE)*, San Diego, California, March 1992.

[5] C. C. Lee. A self-learning rule-based controller employing approximate reasoning and neural net concepts. *International Journal of Intelligent Systems*, 6:71–93, 1991.

[6] E. H. Mamdani. Advances in the linguistic synthesis of fuzzy controllers. *Int. J. Man Mach. Studies*, 8(6):669–678, March 1976.

[7] Lotfi A. Zadeh. Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-3(1):28–44, January 1973.