[4] K. Jensen, *Colored Petri Nets, Basic Concepts, Analysis Methods and Practical Use.* Berlin, Germany: Springer-Verlag, 1992.

[5] R. David and H. Alla, *Petri Nets and Grafcet: Tools for Modeling Discrete Events Systems.* London, U.K.: Prentice-Hall, 1992.

[6] M. Silva, J. Martinez, P. Ladet, and H. Alla, "Generalized inverses and the calculation of symbolic invariants for colored Petri nets," *Technique et Science Informatiques*, vol. 4, no. 1, pp. 113–126, 1985.

[7] K. R Baker, *Introduction to Sequencing and Scheduling.* New York: Wiley, 1974.

[8] A. Ichikawa and K. Hiraishi, "Analysis and control of discrete event systems represented by Petri nets," in *Discrete Event Systems: Models and Applications.* New York: Springer-Verlag, 1988.

[9] J. Martinez, P. Muro, and M. Silva, "Modeling, validation, and software implementation of production systems using high level Petri nets," in *IEEE Int. Conf. Robotics Automation,* Raleigh, NC, 1987, pp. 1180–1185.

[10] M. C. Zhou, K. McDermott, and P. A. Patel, "Petri net synthesis and analysis of a flexible manufacturing system cell," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, pp. 523–531, Mar./Apr. 1993.

[11] A. Finkel and G. Memmi, "An introduction to FIFO nets-monogeneous nets: A subclass of FIFO nets," *Theor. Comput. Sci.,* vol. 35, pp. 191–214, 1985.

[12] J. Hopcroft and J. Ullman, *Introduction to Automata Theory, Languages and Computation.* Reading, MA: Addison-Wesley, 1979.

[13] S. Wolfram, *Cellular Automata and Complexity.* Reading, MA: Addison-Wesley, 1994.

[14] M. Le Borgne, A. Benveniste, P. Le Guernic, "Polynomial dynamical systems over finite fields," *1st Eur. Conf. Algebraic Computing Control,* Paris, France, Mar. 1991.

[15] S.Lang, *Algebra.* Reading, MA: Addison-Wesley, 1993.

[16] A. Bourbaki, *Algèbre, Corps Commutatifs,* Masson Ed., 1981.

[17] H. Alla, "Réseaux de Petri colorés et réseaux de Petri continus," Ph.D. dissertation, Univ. Grenoble, France, 1987.

[18] B. H. Krogh and L. E. Holloway, "Synthesis of feedback control logic for discrete manufacturing systems," *Automatica,* vol. 27, no. 4, pp. 641–651, 1991.

[19] J.-M. Proth and X. Xie, *Les Réseaux de Petri pour la Conception et la Gestion des Systèmes de Production,* Masson Ed., Paris, 1994.

# A New Approach to Adaptive Fuzzy Control: The Controller Output Error Method

H. C. Andersen, A. Lotfi, and A. C. Tsoi

***Abstract*—The Controller Output Error Method (COEM) is introduced and applied to the design of adaptive fuzzy control systems. The method employs a gradient descent algorithm to minimize a cost function which is based on the error at the controller output. This contrasts with more conventional methods which use the error at the plant output. The cost function is minimized by adapting some or all of the parameters of the fuzzy controller. The proposed adaptive fuzzy controller is applied to the adaptive control of a nonlinear plant and is shown to be capable of providing good overall system performance.**

## I. Introduction

The Controller Output Error Method (COEM) which we will describe in this paper can be used for the on-line tuning or adaptation of the parameters of a fuzzy controller. This method can be used with any fuzzy controller design, the only requirement being that the controller is capable of stabilizing the plant before the commencement of tuning. Thus, any fuzzy rule-based model and any inference mechanism can be employed [1] to parameterize and initialize the controller of the system. COEM is applied subsequently for the purpose of achieving better performance.

Neither the initialization nor the subsequent COEM requires a plant model to be available. The initialization conforms to standard fuzzy control design techniques which usually do not rely on a plant model. COEM does not perform a system identification and does not require the plant output error to be propagated backward to the plant input through a reference model, as in indirect adaptive control [2], or directly through the plant as in [3].

The structure of this paper is as follows. In Section II some basic concepts of fuzzy control systems are presented, and in Section III we will describe methods for parameterizing and subsequently initializing a fuzzy controller. Section IV will then explain the concept of COEM and how it can be used in the design of an adaptive fuzzy control system. Section V presents the results of the application of the developed paradigm to a nonlinear plant, and Section VI offers relevant conclusions.

## II. Fuzzy Control Systems

The knowledge of an expert human controller can be expressed as a set of $N$ fuzzy linguistic rules. They are used to implement a fuzzy control system, as shown in Fig. 1. The fuzzy rule base $R$ contains a set of $N$ fuzzy rules as

$$R = \{\text{Rule}^1, \text{Rule}^2, \cdots, \text{Rule}^i, \cdots, \text{Rule}^N\}$$

where the $i$th fuzzy rule is

$$\text{Rule}^i\text{: If } z(k) \text{ is } \tilde{A}^i \text{ then } u(k) \text{ is } \beta^i$$

where

$$z(k) = [z_1(k), \cdots, z_\ell(k)]^T$$

is an $\ell$ vector containing *all* inputs to the fuzzy controller. The elements of $z(k)$ are linguistic variables in the universe of $Z = [Z_1, \cdots, Z_\ell]$, and the control signal $u(k)$ is a single crisp variable. The vector

$$\tilde{A}^i = [\tilde{A}^i_1, \cdots, \tilde{A}^i_j, \cdots, \tilde{A}^i_\ell]$$

is a linguistic vector referring to a vector of fuzzy variables $z(k)$. $\beta^i$ is a consequent parameter corresponding to the control signal $u(k)$. The superscript $\tilde{}$ represents the fuzzy values in contra-distinction to crisp values. The number of individual membership functions for a specific input value $z_j(k)$ is $K_j$. Note that $K_j \leq N$.

We further assume that the universe of antecedent, i.e., $Z_j, j = 1, 2, \cdots, \ell$, is limited to a specific domain interval

$$Z_j = [Z_j^-, Z_j^+] \qquad j = 1, 2, \cdots \ell. \tag{1}$$

In the experiment presented in this paper, the linguistic values are defined by Gaussian membership functions and are given as

$$\tilde{A}^i_j = e^{-(z_j(k) - \mu^i_j / \sigma^i_j)^2} \tag{2}$$

Manuscript received February 4, 1995; revised December 22, 1995 and May 19, 1996.

The authors are with the Department of Electrical and Computer Engineering, University of Queensland, Brisbane, Queensland 4072, Australia.

Publisher Item Identifier S 1083-4419(97)03883-1.

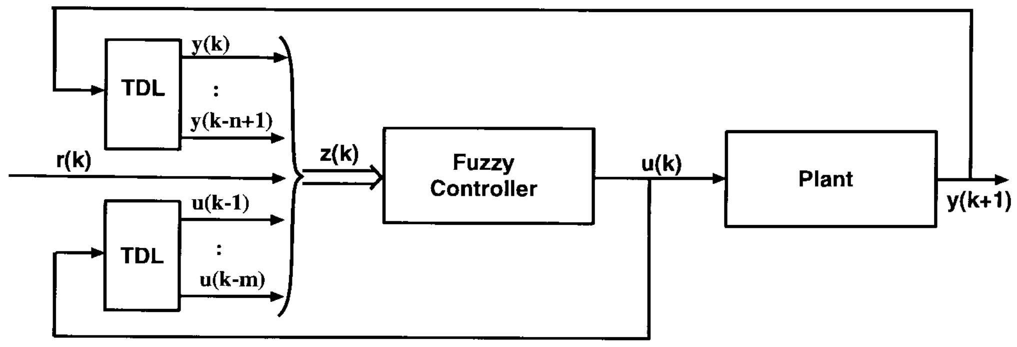


Fig. 1. Fuzzy control system.

where $\sigma_j^i$ and $\mu_j^i$ are unknown constant parameters. As will be shown subsequently, these parameters can be adjusted on-line using a gradient descent algorithm.

In the fuzzy controller proposed in this paper, we define the controller input $z(k)$ as

$$z(k) = [r(k), y(k), \cdots, y(k-n+1), u(k-1), \cdots, u(k-m)]^T \tag{3}$$

where $r(k)$ is a reference signal, $y(k)$ is the output of the plant, and $u(k)$ is the output of the controller. $k$ denotes the time instant, and $m$ and $n$ are constants. Note that $\ell = n + m + 1$.

The controller output is obtained as [4]

$$u(k) = \frac{\sum_{i=1}^{N} w^i \beta^i}{\sum_{i=1}^{N} w^i} \tag{4}$$

where $\beta^i$ are the consequent parameters and $w^i$ is the rule firing strength given by

$$w^i = \prod_{j=1}^{\ell} \tilde{A}_j^i(z_j(k)) \qquad i = 1, 2, \cdots, N. \tag{5}$$

The fuzzy controller is parameterized by

$$\Theta(k) = \{\mu_j^i(k), \sigma_j^i(k), \beta^i(k); i = 1, 2, \cdots, N; j = 1, 2, \cdots, \ell\}. \tag{6}$$

Note that we have chosen to use the Gaussian membership function, and the centroid method of fuzzy inference in this paper as a concrete fuzzy controller structure to convey the idea. The methods to be described in Section III are not dependent on the choice of the membership function, nor the choice of inference mechanism. Whether one chooses to use, for example, a triangular membership function or the Takagi–Sugeno inference mechanism, the method developed in Section III can be applied readily.

## III. Fuzzy Controller Initialization

When designing a fuzzy controller, one attempts either to derive the linguistic rules directly from expert knowledge of the plant to be controlled, or one can use numerical techniques to generate the rules automatically. In this paper, we use a numerical technique which will be described below.

To design a fuzzy rule-based system, we need to set the number of rules, the membership functions, and the consequent parameters. The other information which must be available is the limit of universe for each input and the inference mechanism.

We develop the algorithm for initializing the controller using the following steps.

1) Generate $T$ numerical data samples, $(z^t; u^t)$, by applying signals (random or otherwise) to the input of the plant and sampling the corresponding output. $z^t = z(k^t)$ and $u^t = u(k^t)$ where $k^t; t = 1, 2, \cdots, T$ are the instances of sampling.
2) Specify respectively the minimum and maximum values of all inputs $[Z_j^-, Z_j^+]$ to the fuzzy controller. These can be obtained by sorting the minimum and the maximum of all the inputs $z^t$.
3) Define a number of individual membership functions $K_j$ for each input. The chosen number is just an initial estimate, and will be adjusted later if it is judged inadequate.
4) Determine the initial center, $\mu_j^i$, of each antecedent membership function, distributed uniformly over the universe of $Z_j$ using an interval $|Z_j^+ - Z_j^-|/K_j - 1$. The centers are chosen such that they are distributed evenly over the universe of $Z_j$.
5) Determine the *dispersions*, $\sigma_j^i$, of the antecedent membership functions as
$$\sigma_j^i = \alpha_j \frac{|Z_j^+ - Z_j^-|}{K_j} \qquad i = 1, 2, \cdots, N$$
where $\alpha_j$ is the *interaction coefficient* of membership functions.
6) Form the rule-based system with all possible combinations of the membership functions. The maximum possible number of rules is $\Pi_{j=1}^{\ell} K_j$.
7) A heuristic method [5] is used to set the parameters of the consequent part as
$$\beta^i = \frac{\sum_{t=1}^{T} \prod_{j=1}^{\ell} \tilde{A}_j^i(z_j^t) u^t}{\sum_{t=1}^{T} \prod_{j=1}^{\ell} \tilde{A}_j^i(z_j^t)} \qquad i = 1, 2, \cdots, N. \tag{7}$$
8) If the system behavior is not satisfactory, the above steps must be repeated with more individual membership functions and a larger interaction coefficient.

Once it has been established that the fuzzy controller is able to stabilize the system, tuning utilizing COEM can commence.

Note that step 6 in the above algorithm could result in a large number of rules, especially if $\ell$ is large. Hence, while the framework of the proposed method is suitable only for a small number of rules, $N$, and inputs, $\ell$, this is not usually a problem in control applications where these conditions are almost always satisfied.

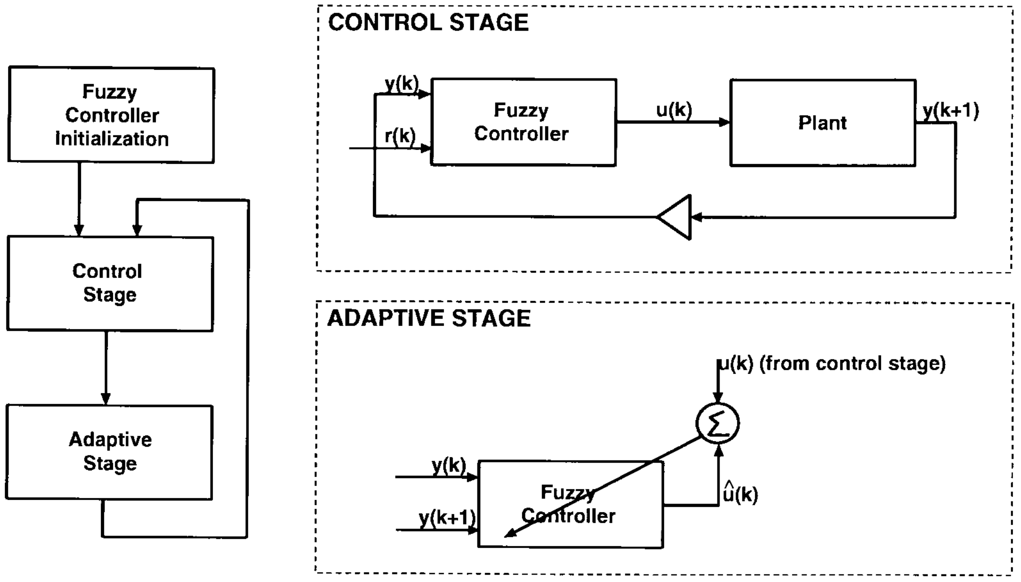


Fig. 2. A block diagram of a simplified version of the proposed adaptive fuzzy controller employing the controller output error method.

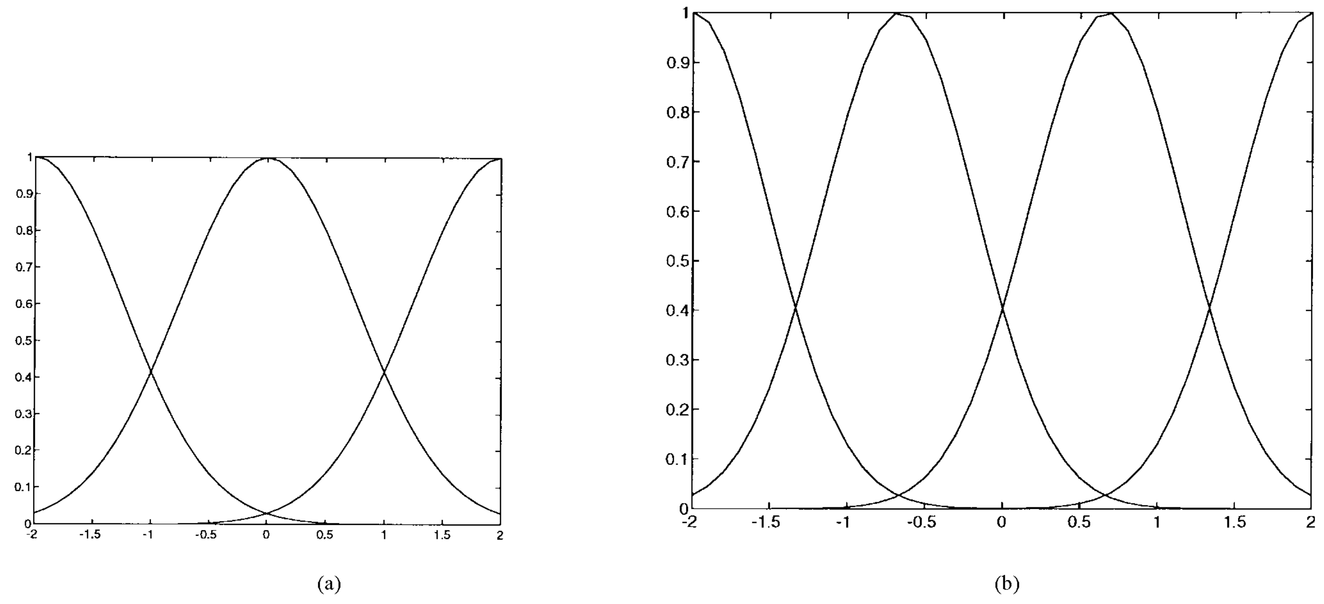


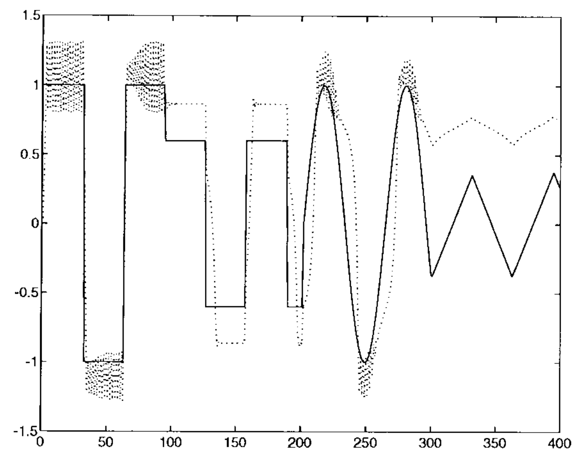
Fig. 3. Membership functions for the fuzzy controller with (a) nine rules and (b) 16 rules.

Note also that the method described in this section is not meant to be optimal. It is only meant to obtain a starting point from which the COEM algorithm to be described in Section IV can commence. Hence, the estimates involved in most of the steps could be quite conservative in nature. For example, the system will perform reasonably even if the bounds on the input domain (step 2) are conservative.

## IV. The Controller Output Error Method

COEM is a strategy for adapting the parameters of a controller without the use of an implicit or explicit plant model. The underlying idea of COEM is as follows. *Each time the response of a plant to a set-point signal is observed, we learn how to repeat that response, should it be required in future.*

Traditionally, adaptive control strategies have been categorized into two groups: direct and indirect [2], [6]. These approaches rely implicitly or explicitly on a plant model in order to derive the appropriate change in each controller parameter from the *plant output error*, $e_y(k) = r(k) - y(k)$. In COEM, the plant output error is not utilized and therefore no plant model is required. The lack of dependence on the plant output error, however, introduces a restriction on the application of COEM. Since the cost function is not based on the plant output error, this error is not directly minimized. This implies that if the controller does not already stabilize the plant, COEM is not likely to cause it to do so. In other words, the controller *must* be able to stabilize the plant before COEM is utilized. Hence, the method should be viewed as a tuning device as opposed to an automated controller design methodology. In the context of fuzzy control, this implies that the control engineer is required to take the usual steps of fuzzy controller design and verification (see Section III) before employing COEM to tune the parameters of the fuzzy controller.


(a)

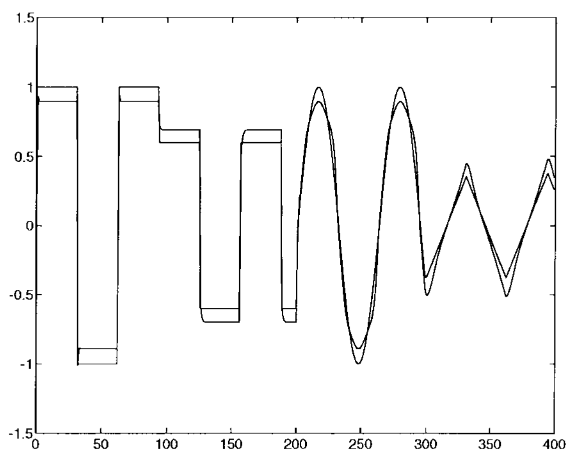
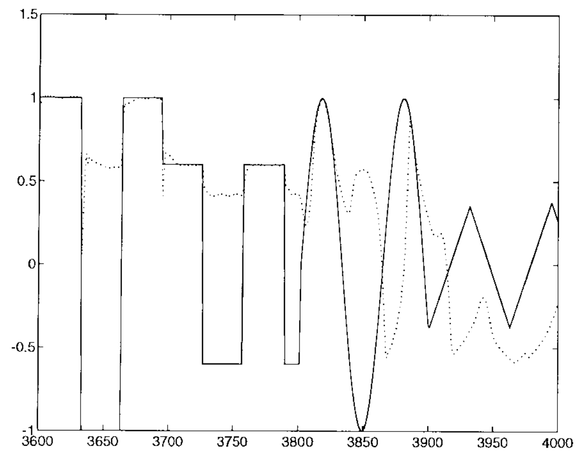

(b)

Fig. 4. Performance of controllers *without COEM* for (a) nine rules and (b) 16 rules.


(a)

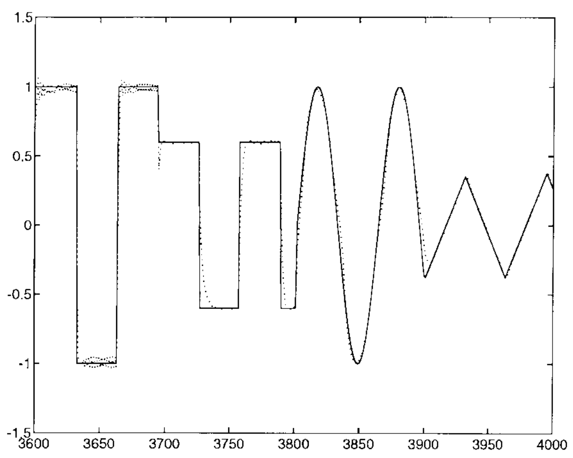

(b)

Fig. 5. Performance of controller *with COEM* for nine rules ($\eta = 0.01$). Partial updates are used in (a) and full updates are used in (b).

A concept similar to COEM has been described previously by Psaltis, Sideris, and Yamamura [3] and Andersen, Teng, and Tsoi [7] who applied it to neural network controllers. In this context the name, "indirect learning" was used. In [8] Johnson also described a similar idea for linear systems. He used the phrase "one-step-ahead input matching" to describe the method[1]. We have chosen the name, Controller Output Error Method, because i) in the control field the term "indirect" is more commonly used in connection with the concept of indirect adaptive control [2], [6] and use of this term is thus thought to be misleading in this context, and ii) we consider it to be important to emphasize the controller output rather than the plant input since this is where the control signal originates.

"Specialized learning architecture" is another noteworthy adaptive control strategy which was also proposed in [3]. This method also does not require a plant model. Instead, it relies on finding an approximation of the derivative of each controller parameter with respect to the plant output error. This approximation is obtained by experimentally measuring the response of the system to small perturbations in the controller parameters.

[1]The term "input matching" was used since the plant input error was considered.

## A. Definition

In this section, we will define COEM mathematically and in Section IV-C a step-by-step algorithm procedure will be given.

We consider the nonlinear plant

$$y(k+1) = \mathcal{F}(y(k), \cdots, y(k-q+1), u(k), \cdots, u(k-p+1))$$

where $y(k)$ is the plant output at instance $k, \mathcal{F}(\cdot)$ is a nonlinear function, and $u(k)$ is the control signal. The constants $p$ and $q$ define the order of the plant. The order may be unknown, so we *cannot* assume that $p = m$ and $q = n$ [see (3)].

The aim of many practical control problems is to produce a controller which will drive the plant output toward a given reference signal, $r(k)$. A fuzzy controller can be defined as a functional relation between a set of inputs and an output. This relationship is described by

$$u(k) = \mathcal{G}(z(k), \Theta(k))$$

where $\mathcal{G}(\cdot)$ is the function defining the controller characteristics and $\Theta(k)$ is the set of controller parameters. The controller input vector, $z(k)$, is as defined in (3).

In most existing adaptive control strategies the plant output error, $e_y(k) = r(k) - y(k)$, is calculated and is either directly or indirectly

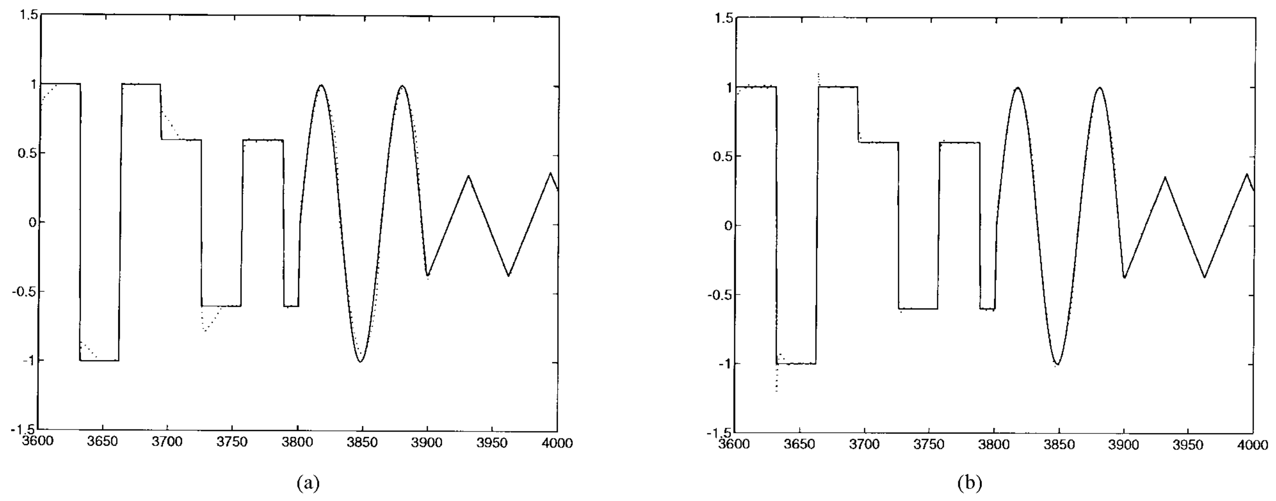


Fig. 6. Performance of controller *with COEM* for 16 rules ($\eta = 0.01$). Partial updates are used in (a) and full updates are used in (b).

used for the adaptation of the controller parameters, $\Theta(k)$, (see [2] and [6]). However, COEM does not use the plant output error to adapt the controller parameters. Instead the *controller output error*, $e_u(k)$, is used. The derivation of the controller output error can be explained as follows.

At instance $k$, the state of the plant may be defined by $S = [y(k), \cdots, y(k-p+1)]^T$ (assuming that the plant is observable). The fuzzy controller produces a control signal, $u(k)$, which drives the output of the plant to $y(k+1)$. Regardless of whether or not this was the intended response, we now know that, if the transition from a state $S$ to an output $y(k+1)$ is ever required again, the appropriate control signal is $u(k)$.

The fuzzy controller is now tested to see if it does indeed output a signal equal to $u(k)$ when required to drive the plant through this same transition. Instead of producing a control signal $u(k)$, however, the controller outputs the signal $\hat{u}(k)$. Thus, the controller output is in error by $e_u(k) = u(k) - \hat{u}(k)$.

It is important to note that, although $\hat{u}(k)$ is produced by the controller, it is not applied to the plant. Its only purpose is to calculate $e_u(k)$. $\hat{u}(k)$ is calculated by producing a new controller input vector, $\hat{z}(k)$, which is passed through the fuzzy controller as

$$\begin{aligned}\hat{u}(k) &= \mathcal{G}(\hat{z}(k), \Theta(k))\\ \hat{z}(k) &= [y(k+1), y(k),\\ &\qquad \cdots, y(k-n+1), u(k-1), \cdots, u(k-m)]^T.\end{aligned}$$

The input vector $\hat{z}(k)$ only differs from $z(k)$ in the first element, where $y(k+1)$ replaces $r(k)$. Note also that, for each time instance, two control commands, $u(k)$ and $\hat{u}(k)$, are produced[2], although only one of these, $u(k)$, is passed to the plant.

### B. Parameter Adaptation

The controller output error, $e_u(k)$, is now used in a cost function such as

$$J(k) = \tfrac{1}{2} e_u(k)^2. \tag{8}$$

[2]It is noted that, since $\hat{u}(k)$ depends on $y(k+1)$, it cannot be calculated until instance $k+1$.

The aim of the adaptation of the controller parameters, $\Theta(k)$, is to minimize this cost function. One method is to use gradient descent to adjust the parameters of the fuzzy controller such that $J(k)$ is minimized. For this technique to be effective, it is desirable that the membership functions employed have a continuous first derivative.

The parameters are updated by

$$\Theta(k+1) = \Theta(k) - \eta \frac{\partial J(k)}{\partial \Theta(k)}.$$

As defined in (6), $\Theta(k)$ is the set of parameters of the fuzzy controller. The partial derivative of the cost function, $J(k)$, with respect to each parameter is

$$\frac{\partial J(k)}{\partial \beta^i} = -\frac{e_u(k) w^i}{\sum_{i=1}^{N} w^i}$$

$$\frac{\partial J(k)}{\partial \mu_j^i} = -\frac{2 e_u(k)(\hat{z}_j(k) - \mu_j^i) \sum_{l=1}^{N} [c_{ij}^l \, w^i \, (\beta^i - \hat{u}(k))]}{(\sigma_j^i)^2 \sum_{i=1}^{N} w^i}$$

$$\frac{\partial J(k)}{\partial \sigma_j^i} = -\frac{2 e_u(k)(\hat{z}_j(k) - \mu_j^i)^2 \sum_{l=1}^{N} [c_{ij}^l \, w^i \, (\beta^i - \hat{u}(k))]}{(\sigma_j^i)^3 \sum_{i=1}^{N} w^i}$$

where $\eta$ is the rate of descent which may be chosen arbitrarily and $c_{ij}^l$ is 1 if the $\ell$th rule is dependent on the $i$th membership function of the $j$th input or 0 if it is not dependent.

The updating of the parameters in the fuzzy controller can be implemented in two ways: *partial updating* and *full updating*. In partial updating only the parameters of the consequent part of the rules, $\beta^i$, are adapted. The parameters in the antecedent, $\mu_j^i$ and $\sigma_j^i$, are left unchanged. This means that the meaning which may have been assigned to the linguistic variables is preserved. Full updating gives the adaptive algorithm freedom to change any parameters of the fuzzy controller, i.e., $\mu_j^i, \sigma_j^i$, and $\beta^i$.

Having defined the parameter update mechanism for the fuzzy controller, the system can be put on line.

At the present moment there are no general proofs of stability for fuzzy control systems. We do not have a proof of convergence for the controller design described in this paper, however, we have applied this method quite extensively, and found that the method appears to work quite satisfactorily. In Section V, we will describe a simulation example which is intended to demonstrate that the method appears to yield reasonable performance.

### C. Method

In summary, the procedure for designing and implementing a COEM adaptive fuzzy controller can be described in three steps.

1) **Initialization:** Design a fuzzy controller which stabilizes the given plant. A method such as that described in Section III may be used.
2) **Control:** Sample the plant output, $y(k)$, and, given a reference signal, $r(k)$, produce a control command, $u(k)$. This control command is applied to the plant. No controller parameters are changed in this step.
3) **Adaptation:** Using the next plant output, $y(k+1)$, in place of the reference signal, $r(k)$, produce a control command, $\hat{u}k$. This control command is *not* applied to the plant, but is instead used to calculate a controller output error, $e_u(k) = u(k) - \hat{u}k$, which is utilized in the cost function, $J(k) = \frac{1}{2}e_u(k)^2$. The parameters of the fuzzy controller are adapted by gradient-descent according to $J(k)$.

Step 1 is only performed once. When the system is put on line, the algorithm cycles between steps 2 and 3. Fig. 2 illustrates the method for a simple system where $m = 1$ and $n = 0$, i.e., the fuzzy controller input vector is $z(k) = [r(k), y(k)]^T$.

## V. Simulation Results

The following nonlinear plant, reported in [7], is used for the simulation study.

$$y(k+1) = 0.8\,\sin(2y(k)) + 1.2u(k). \tag{9}$$

Two adaptive fuzzy controllers utilizing COEM are implemented. Both are of the form described in Section IV where $m = 0$ and $n = 1$. That is, the input to the controller is $z(k) = [r(k), y(k)]^T$. As shown in Fig. 3, the first implementation uses nine rules, with three individual membership functions for $y(k)$ and $r(k)$, respectively, i.e., $K_1 = 3$ and $K_2 = 3$. The interaction coefficients are $\alpha_1 = \alpha_2 = 0.8$. The second implementation uses 16 rules, with four individual membership functions for each of $y(k)$ and $r(k)$, i.e., $K_1 = 4$ and $K_2 = 4$. The interaction coefficients are $\alpha_1 = \alpha_2 = 0.7$. For both cases, $\eta = 0.01$ was used.

Fig. 4(a) and (b) shows the performance of the 9 and 16 rule systems, respectively, when no on-line training is used. Figs. 5 and 6 contain plots of the plant output (solid lines) and the reference (dotted lines). In each figure, (a) shows the results when partial updating is used and (b) shows the results with full updating.

The contrast between the plot in Fig. 4(a) and those in Fig. 5, as well as the contrast between the plot in Fig. 4(b) and those in Fig. 6, shows that COEM improves the performance of the system significantly.

Comparison of Fig. 5(a) and (b) shows the performance of the method with full updating is far superior to the performance with partial updating only. The reason for this is that full updating allows the system more freedom in adapting to the circumstances. It should however be noted that allowing the membership functions to change can violate the spirit of fuzzy logic in that the meaning initially assigned to the linguistic values can be lost. There is a much smaller difference between Fig. 6(a) and 6(b). This is because the use of four membership functions, instead of three, gives the fuzzy controller more degrees of freedom in matching the control function. Because it can match the function more accurately, there is a lesser need for adjusting the parameters of the controller, and hence the changes made by the on-line training are smaller in magnitude.

## VI. Conclusions

A novel algorithm for adaptively updating the parameters of a fuzzy controller is proposed. This algorithm does not rely on any plant model. The controller output error is used in the update of the parameters associated with the fuzzy parameterization of the controller. It is observed that once the controller is initialized, using any fuzzy control design methodology, the tuning method takes over, and drives the system such that the tracking error approaches zero. The proposed method is shown to work well in the control of a nonlinear plant.

## References

[1] A. Lotfi and A. C. Tsoi, "Importance of membership functions: A comparative study on different learning methods for fuzzy inference systems," in *Proc. 3rd IEEE Int. Conf. Fuzzy Systems*, Orlando, FL, June 1994, pp. 1791–1796.

[2] R. R. Bitmead, M. Gevers, and V. Wertz, *Adaptive Optimal Control.* Englewood Cliffs, NJ: Prentice-Hall, 1990.

[3] D. Psaltis, A. Sideris, and A. A. Yamamura. "A multilayered neural network controller," *IEEE Contr. Syst. Mag.*, vol. 8, no. 2, pp. 17–21, 1988.

[4] L. X. Wang and J. M. Mendel, "Fuzzy basis functions, universal approximation, and orthogonal least-squares learning." *IEEE Trans. Neural Networks*, vol. 3, pp. 807–814, Sept. 1992.

[5] H. Ishibuchi, K. Nozaki, and H. Tanaka. "Empirical study on learning in fuzzy systems by rice taste analysis," *Fuzzy Sets Syst.*, vol. 64, pp. 129–144, 1994.

[6] K. J. Åström and B. Wittenmark, *Computer-Controlled Systems: Theory and Design.* Englewood Cliffs, NJ: Prentice-Hall, 1990.

[7] H. C. Andersen, F. C. Teng, and A. C. Tsoi. "Single net indirect learning architecture," *IEEE Trans. Neural Networks*, vol. 5, pp. 1003–1005, Nov. 1994.

[8] C. R. Johnson and E. Tse, "Adaptive implementation of one-step-ahead optimal control via input matching," *IEEE Trans. Automat. Contr.*, vol. AC-23, pp. 865–872, Oct. 1978.

[9] A. Lotfi and A. C. Tsoi, "Learning fuzzy inference systems using an adaptive membership function scheme," *IEEE Trans. Syst., Man, Cybern.*, vol. 26, pp. 326–331, Apr. 1996.