Dual Heuristic Programming for Fuzzy Control

George G. Lendaris, Thaddeus T. Shannon, Larry J. Schultz, Steven Hutsell and Alec Rogers
Northwest Computational Intelligence Laboratory and Systems Science Ph.D. Program,
P.O. Box 751, Portland State University, 97207
lendaris@sysc.pdx.edu, tads@sysc.pdx.edu

ABSTRACT

Overview material for the Special Session (Tuning Fuzzy Controllers Using Adaptive Critic Based Approximate Dynamic Programming) is provided. The Dual Heuristic Programming (DHP) method of Approximate Dynamic Programming is described and used to the design a fuzzy control system. DHP and related techniques have been developed in the *neuro*control context but can be equally productive when used with fuzzy controllers or neuro-fuzzy hybrids. This technique is demonstrated by designing a temperature controller for a simple water bath system. In this example, we take advantage of the TSK model framework to initialize the tunable parameters of our plant model with reasonable problem specific values.

I. INTRODUCTION

This is an introductory paper for the special session entitled Tuning Fuzzy Controllers Using Adaptive Critic Based Approximate Dynamic Programming. We include here a demonstration of using adaptive-critic-based approximate dynamic programming to design a fuzzy controller for a (simple) water bath problem.

A variety of Adaptive Critic Design techniques for training *neuro*-controllers have appeared in the literature recently, falling into model-based methods such as Dual Heuristic Programming (DHP), and non-model-based methods such as Action Dependent Heuristic Dynamic Programming (ADHDP) or Q-learning [1][4][5][6][9][10][11][14][15][18] [19].

The tuning of *fuzzy* controllers using Adaptive Critic based reinforcement learning has previously been accomplished (e.g., based on *temporal differencing* schemes) [3][4][7][8].

Adaptive Critic methods fall within the general category of Reinforcement Learning. The latter refers to learning based on *evaluative* type feedback (called *reinforcement*) from the environment. This is in distinction to Supervised Learning, which is based on rather specific (instructional type) feedback from an agent called a "teacher."

In the neural network context, model-based techniques, such as the DHP method emphasized in this paper, have been shown to be generally more effective than their non-model based counterparts – such as those described in [9][10][13].

There is, therefore, motivation to apply the model-based DHP method in the fuzzy context, and in [14], we demonstrated that the DHP techniques developed for neural networks carry over cleanly to the tuning of parameters in fuzzy control systems. In this paper we extend that demonstration by showing the "design from scratch" of a fuzzy control system via the DHP method.

In model based methods, the Jacobian of the coupled plant-controller system is used to train both the controller and critic networks. These derivatives could be found explicitly from an analytic model of the plant, when available. If not available, then an alternative method such as back-propagation through a *neural network* plant model could be used. Another alternative would be to use a fuzzy model of the plant, and in [16], we showed that such derivative information could be explicitly estimated in the form of a Takagi-Sugeno (TSK) fuzzy model of the plant [17][21]. In the present paper, we focus on the *controller*, and demonstrate that the approximate dynamic programming techniques can easily be adapted to the tuning of fuzzy controllers.

A key underlying feature in the fuzzy control context considered here, is the partitioning of (the continuous) state space in a fuzzy way. This allows "averaging" over adjacent partitions, which in turn allows the possibility of generalization, as states not previously known explicitly may be evaluated.

As has been observed in previous work with these techniques [14], there are significant advantages to starting the controller training (or, tuning) process with a set of computational structures (neural networks, fuzzy rule sets, etc.) that are well matched to

This work was supported by the National Science Foundation under grant ECS-9904378.

the task at hand. Structures that are too small will be unable to satisfactorily learn the desired task, while overly large structures tend to require longer training times and tend to exhibit poor generalization. Problem specific knowledge is required to match computational structures to specific tasks. Doing this "prestructuring" with the neural network architectures commonly used in this arena (feedforward or recurrent networks with sigmoidal activation functions) is a rather opaque task. Fuzzy systems offer a way around this difficulty in many application contexts, and we posit this to be true in the reinforcement learning context as well.

Section II provides a brief overview of adaptive critic based approximate dynamic programming, and then delves into the details of the specific technique we demonstrate, Dual Heuristic Programming (DHP). In Section III, we introduce a water bath temperature control problem, and in Section IV, describe controller and critic architectures. We use a fuzzy TSK model to fill the role of *adaptive critic* function approximator, to show that DHP can be implemented without resort to neural networks. In practice, however, it may be desirable to mix neural network critics with fuzzy controllers. Section V introduces a TSK model of the plant for use in the tuning process, and describes the specifics of its estimation and its use in DHP. Section VI gives the details of the on-line tuning process, and finally, Section VII gives our results on the design problem.

II. APPROXIMATE DYNAMIC PROGRAMMING

Dynamic Programming is a general approach for sequential optimization applicable under very broad conditions. Fundamental to this approach is Bellman's Principle of Optimality [2]: that an optimal trajectory has the property that no matter how an intermediate point is reached, the rest of the trajectory must coincide with an optimal trajectory as calculated with the intermediate point as the starting point. This principle is applied by formulating a "primary" utility function *U(t)* that embodies a control objective for a particular context in one or more measurable variables. A *secondary* utility function is then formed

$$J(t) = \sum_{k=0}^{\infty} \gamma^k U(t+k),$$

which embodies the desired control objective through time. This is Bellman's equation, and the point of Dynamic Programming is to select the sequence of actions (controls) that maximize or minimize *J(t)*. A useful identity based on the above equation is the Bellman Recursion

$$J(t) = U(t) + \gamma J(t+1).$$

Unfortunately, this optimization is not computationally tractable for most real world problems, so we are forced to consider approximation methods that offer a greater chance of being computationally tractable.

A promising collection of such approximation techniques based on estimating the function *J(t)* using the Bellman Recursion with neural networks as function approximators was proposed by Werbos [18][19]. These networks are often called Adaptive Critics, though this term can be applied more generally to any network that provides learning reinforcement to another entity [20]. As a practical matter, any computational structure capable of acting as a universal function approximator can be used in this role (e.g., neural networks, fuzzy rule structures, etc.). The gradient of the estimated *J(t)* can then be used to train or tune a controller. Since the gradient is the important aspect for controller training, some techniques use critics that directly estimate the derivatives of *J(t)* instead of the function value itself.

The standard classification of these adaptive critic methods is based on the critic's inputs and outputs. In Heuristic Dynamic Programming (HDP) the critic's outputs are estimates of the value of *J(t)*. In Dual Heuristic Programming (DHP) the critic's outputs are estimates of the derivatives of *J(t)*. In the *action dependent* versions of HDP and DHP, the critic's inputs are augmented with the controller's output (action), hence ADHDP and ADDHP.

These approaches to approximate dynamic programming utilize at least two distinct training loops, a controller training loop and a critic training loop [5][6][15]. In the *neuro*control context, the controller training loop adapts a neural network to be an approximately optimal controller. Specifically, the controller is trained to optimize the secondary utility function *J(t)* for the problem context. Since the controller outputs control actions u(t), a gradient based learning algorithm requires estimates of the derivatives $\frac{\partial J(t)}{\partial u_i(t)}$ for controller training. The critic is trained based on the consistency of its estimates through time judged using the Bellman Recursion. The exact implicit relationship is depends on the type of critic used and the form of the primary utility function.

Our focus in this paper is on the DHP method, where the critic estimates the derivatives of J(t) with respect to the system states, i.e. $\lambda_i(t) \equiv \frac{\partial J(t)}{\partial R_i(t)}$.

From Bellman's Recursion we have

$$\frac{\partial}{\partial R_i(t)} J(t) = \frac{\partial}{\partial R_i(t)} \left(U(t) + \gamma J(t+1)\right),$$

so the identity used for this critic's training is (in tensor notation)

$$\lambda_i(t) \equiv \frac{\partial U(t)}{\partial R_i(t)} + \frac{\partial U(t)}{\partial u_j(t)} \frac{\partial u_j(t)}{\partial R_i(t)} + \gamma \lambda_k(t+1) \left[\frac{\partial R_k(t+1)}{\partial R_i(t)} + \frac{\partial R_k(t+1)}{\partial u_m(t)} \frac{\partial u_m(t)}{\partial R_i(t)} \right].$$

To evaluate the right hand side of this equation we need a *model of the system dynamics* that includes all the terms from the Jacobian matrix of the coupled plant-controller system, e.g. $\frac{\partial R_j(t+1)}{\partial R_i(t)}$ and $\frac{\partial R_j(t+1)}{\partial u_i(t)}$.

Controller training then utilizes the chain rule and the system model to translate critic outputs into estimates of $\frac{\partial J(t)}{\partial u_i(t)}$, i.e.,

$$\frac{\partial J(t)}{\partial u_k(t)} = \frac{\partial U(t)}{\partial u_k(t)} + \gamma \sum \lambda_i(t+1) \frac{\partial R_i(t+1)}{\partial u_k(t)}.$$

The entire process can be characterized as a simultaneous optimization problem; gradient based optimization of the critic function approximator together with gradient based optimization of controller parameters based on the *J(t)* estimates obtained from the critic. Different strategies have been utilized to get both of these optimizations to converge. Various authors propose a process which alternates doing a number of optimization cycles in one loop, and then doing a number of optimization steps in the other loop, repeating until the two loops converge [9][10][11][13]. In some past work (e.g., [5][6]), we demonstrated that performing optimization steps in *both* loops simultaneously does not appear to introduce significant instabilities into the dual convergence problem. Since the simultaneous stepping approach is about twice as fast as the alternating approach, we recommend its use.

As these techniques rely on gradient based optimization of *J(t)*, they inherently suffer from the problem of (unsatisfactory) local optima. Global optimization of *J(t)* in general is subject to the "No Free Lunch Theorem". What approximate dynamic programming techniques offer is a tractable method for local hill climbing on the *J(t)* landscape of controller parameter space. Initialized at a random point in parameter space, these methods may be trapped by a local optimum at an unsatisfactory control law. We can attempt to avoid this case by applying whatever problem specific knowledge is available a priori to the choice of initial controller parameters, in the hope of being near a satisfactorily high hill (or deep valley).

III. WATER BATH MODEL

A discrete time model for of the temperature of the water bath problem from [12] is

$$t(k+1) = e^{-aT} t(k) + \left[1 - e^{-aT}\right] t(0) + \frac{a\left[1 - e^{-aT}\right]}{b + be^{0.5t(k)-c}} u(k).$$

where $t(k)$ is the water temperature measured by sample k, and T is the sampling interval of the control system. The parameter values we use for our simulation are a = 1.00151x10^{-4}, b = 8.6797x10^{-3}, c = 40 and $t(0)$ = 25° C. The system is sampled at 30-second intervals. The objective we train and test for is a series of step increases in the target temperature:

$$\hat{t}(k) = \begin{cases} 35° & k \le 80, \\ 55° & 80 < k \le 160, \\ 35° & 160 < k. \end{cases}$$

IV. CONTROLLER AND CRITIC STRUCTURES

To keep this demonstration of using DHP to tune fuzzy controllers relatively simple, we use a simple Mamdami style controller with singleton consequents for the above plant. A more sophisticated implementation would doubtlessly yield finer control, but this simple controller is adequate for our demonstration. Important to the present development is the fact that the DHP method turns out to be quite capable of fine tuning even our naïve structure to produce acceptable control quality. The controller we crafted comprises 20 rules with consequents of the form

$$u(k) = \alpha_i,$$

where α_i is the singleton value for the rule, with Gaussian membership functions defined on t and e = $\hat{t}$ - t by

$$m_i(t, e) = \exp\left(-\frac{(\bar{t}_i - t))^2}{\sigma_1^2} - \frac{(\bar{e}_i - e)^2}{\sigma_2^2} \right),$$

where $\bar{t}_i$ and $\bar{e}_i$ specify the location of the center of rule i. These rules are placed uniformly over the temperature interval [25, 85], and the error interval [-10, 10] so as to completely cover the anticipated operating and training range of the controller (Figures 1 and 2). With this rule structure, we use center average defuzzification to conveniently calculate controls as

$$u(k) = \frac{\sum m_i(t(k), e(k))\alpha_i}{\sum m_j(t(k), e(k))}.$$

While we could apply the DHP method to tuning all controller parameters, for clarity we limit the tuning process to the rule consequent parameters α_i. Thus we have a fixed grid of rules imposed upon the plant's state space.

To approximate the DHP *critic* function, we construct a TSK model based on this grid. The DHP critic mapping in this case takes $t(k)$ and $e(k)$ as inputs and produces $\frac{\partial J(k)}{\partial t(k)}$. Our critic system is therefore composed of 20 rules with consequents

$$\frac{\partial J(k)}{\partial t(k)} = \lambda(k) = \gamma_i t(k) + \psi_i e(k),$$

and membership functions identical to those used for the controller. Once again, we choose to only tune the consequent parameters of these rules.

For the primary utility function, we use

$$U(k) = -\frac{1}{2} e(k)^2,$$

which we seek to minimize over time. The choice of an appropriate discount factor for forming *J(k)* is much less critical in DHP, where *J(k)* is not explicitly estimated, as it is in HDP, where explicit estimates of *J(k)* are typically more important. For this example, we choose a discount factor of 0.9. The final requirement for implementing DHP is to have a differentiable model of the plant from which to obtain partial derivatives. In the present example, we have an analytic expression for the plant and hence could evaluate the derivatives explicitly. To reflect more realistic situations, we proceed to develop a plant model estimate in the next section.

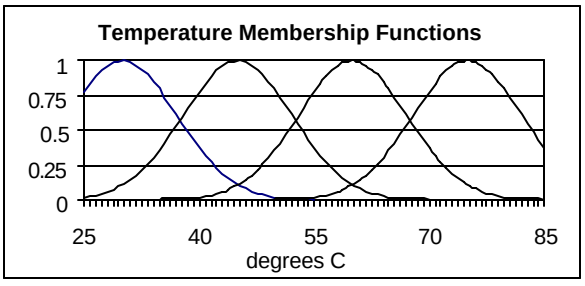

Figure 1 Temperature Membership Functions.

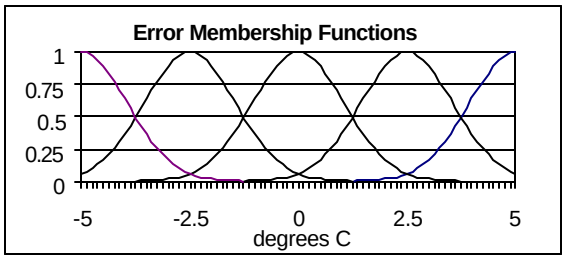

Figure 2 Error Membership Functions.

V. CONSTRUCTING A MODEL

It is relatively straight forward to prestructure an appropriate model for this DHP training context. We use a TSK model containing 4 rules with consequents of the form

$$t(k+1) = \zeta_i t(k) + \chi_i u(k),$$

and membership functions identical to those of the controller and critic rules. Training is based on minimizing squared one-step prediction error using a backpropagation of error approach.

A useful benefit of using a TSK model is that the coefficients in the output model of each rule provide local approximations of the partial derivatives we need during DHP tuning. For example, we can obtain

$$\left.\frac{\partial t(k+1)}{\partial u(k)}\right|_{(t(k),\,u(k))} = \frac{\sum_i m_i(t(k), u(k))\chi_{1,i}}{\sum_i m_i(t(k), u(k))}.$$

Our previous work with such models showed that as long as the approximate derivative values obtained in the above manner had the correct sign (positive or negative) *most* of the time, the model was adequate for use in DHP [15] [16].

VI. THE TUNING SEQUENCE

Prior to tuning, all the model consequent parameters were prestructured using local models based on direct linearization of the system equations. All controller and critic consequent parameters were initially set to zero. The plant simulation was then initialized and run for 2-hour periods with the controller providing *u(t)*, and both controller and critic updates taking place each sampling period. Training commenced in this fashion with the plant reinitialized every 120 minutes. The specific simulation/update sequence used was:

1) calculate control (k);
2) simulate one sampling interval (k);
3) evaluate critic (at new state = k+1);

4) calculate target critic value (for old state = k);
5) calculate controller error signal (k);
6) update controller (k);
7) evaluate critic (at old state = k);
8) update critic using actual minus target (k);
9) calculate the model error (k+1 actual vs. predicted)
10) update the model (k)

The controller update equations are

$$\alpha_i(k+1) = \alpha_i(k) + lc * m(t,e) * \left(\frac{\partial J(k)}{\partial u(k)} \right),$$

where lc is the learning coefficient,

$$m(t,e) = \frac{m_i(t(k), e(k))}{\sum_k m_k(t(k), e(k))},$$

and

$$\frac{\partial J(k)}{\partial u(k)} = 0.9\, \lambda_1(k+1) \frac{\partial t(k+1)}{\partial u(k)}.$$

The critic update equations are

$$\theta_i(k+1) = \theta_i(k) + lc * m(t,e) * \left(\hat{\lambda}(k) - \lambda(k) \right) * t(k),$$

$$\psi_i(k+1) = \psi_i(k) + lc * m(t,e) * \left(\hat{\lambda}(k) - \lambda(k) \right) * e(k),$$

where lc is the learning coefficient, and the $\hat{\lambda}(k)$ is given by

$$\hat{\lambda}(k) = \frac{\partial U(k)}{\partial t(k)} + 0.9\, \lambda(k+1) \left(\frac{\partial t(k+1)}{\partial t(k)} + \frac{\partial t(k+1)}{\partial u(k)} \frac{\partial u(k)}{\partial t(k)} \right)$$

The model update equations are

$$\zeta_i(k+1) = \zeta_i(k) + lc * m(t,e) * \left(\hat{t}(k+1) - t(k+1) \right) * t(k),$$

$$\xi_i(k+1) = \xi_i(k) + lc * m(t,e) * \left(\hat{t}(k+1) - t(k+1) \right) * u(k),$$

where $\hat{t}(k+1)$ is the predict value for $t(k+1)$.

VII. RESULTS

The average RMS error of the controller during the tuning process is shown in Figure 3. The final step response tracking error of the trained controller is shown in Figure 4.

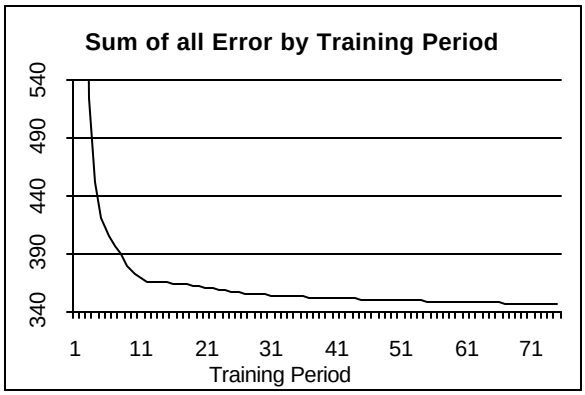

Figure 3. Sum of all error by training period.

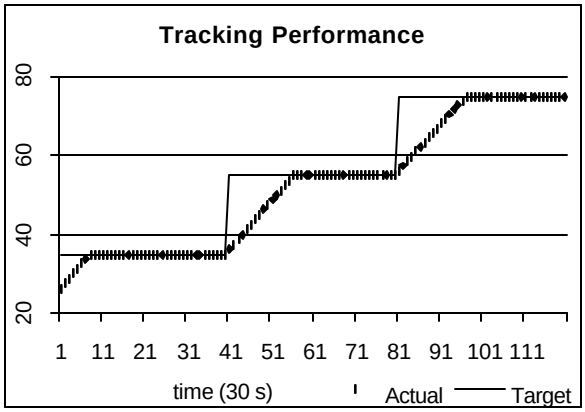

Figure 4. Tracking performance of the final controller – 30 second sampling intervals.

VIII. CONCLUSION

We have demonstrated how the DHP methodology (a model-based adaptive critic method) can successfully design a fuzzy *controller* using a fuzzy plant model. While this approximate dynamic programming technique has previously been used for training/designing *neuro*controllers, embedding DHP in a fuzzy framework more easily allows *a priori* knowledge to be used. In particular, use of first order TSK models offers a direct approach to representing the characteristics of the plant relevant to the needs of model-based approximate dynamic programming. The TSK models used also permit a simple approach to prestructuring the controller, based on known first principles models and approximate parameter values. Our belief is that DHP and related techniques are ideally suited for neuro-fuzzy hybrid implementations. Fuzzy controllers and models more easily allow the incorporation of a priori knowledge, while neural networks may be more natural as function approximators in the critic role. While the above demonstration was limited to tuning consequent pa-

rameters, DHP can also be used for training membership parameters if the structure of the fuzzy rule base is suitable. Thus DHP and related techniques should be helpful for state space segmentation and partitioning.

It is also important to notice the applicability of these techniques to adaptive control problems. For non-stationary plants, the controller, critic and model can all be continuously updated on-line to track changes in the plant's dynamics. Stability considerations would, of course, have to be taken into account.

REFERENCES

[1] Barto, A.G., R.S. Sutton & C.W. Anderson, "Neuron-like Adaptive Elements That Can Solve Difficult Learning Control Problems", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 13, pp. 834-846, 1983.

[2] Bellman, R.E., *Dynamic Programming*, Princeton University Press, 1957.

[3] Berenji, H.R. & P. Khedkar, "Learning and Tuning Fuzzy Logic Controllers Through Reinforcement", *IEEE Transactions on Neural Networks*, 1992.

[4] Lee, C.C., "A Self-Learning Rule-Based Controller Employing Approximate Reasoning and Neural Net Concepts", *International Journal of Intelligent Systems*, Vol. 6, pp.71-93, 1991.

[5] Lendaris, G., C. Paintz and T. Shannon, "More on Training Strategies for Critic and Action Neural Networks in Dual Heuristic Programming Method," *Proceedings of IEEE-SMC'97 International Conference*, IEEE, 1997

[6] Lendaris, G.G., T.T. Shannon & A. Rustan, "A Comparison of Training Algorithms for DHP Adaptive Critic Neuro-Control", in *Proceedings IJCNN'99*, Washington D.C., IEEE, 1999.

[7] Lin, C.T. & C.S.G. Lee, *Neural Fuzzy Systems*, Prentice Hall, Upper Saddle River, NJ, 1996.

[8] Lin, C.T., & Y.C. Lu, "A Neural Fuzzy System with Linguistic Teaching Signals", *IEEE Transactions on Fuzzy Systems*, Vol. 3, # 2, 1995.

[9] Prokhorov, D., *Adaptive Critic Designs and their Application*, Ph.D. Dissertation, Department of Electrical Engineering, Texas Tech University, 1997.

[10] Prokhorov, D. & D. Wunsch, "Adaptive Critic Designs", *IEEE Transactions on Neural Networks*, vol. 8 (5), 1997, pp. 997-1007.

[11] Prokhorov, D., R. Santiago & D. Wunsch, "Adaptive Critic Designs: A Case Study for Neurocontrol", *Neural Networks*, vol. 8 (9), pp. 1367 - 1372, 1995.

[12] Tanomaru, J. and S. Omatu, "Process Control by On-line Trained Neural Controllers", IEEE Trans. on Ind. Elec., vol. 39, December 1992, pp. 511-521.

[13] Santiago, R. & P.J. Werbos, "New Progress Towards Truly Brain-Like Control", *Proceedings of WCNN'94*, San Diego, CA, 1994, pp. 27-33.

[14] Shannon, T.T & G. G. Lendaris, "Adaptive Critic Based Approximate Dynamic Programming for Tuning Fuzzy Controllers", in *Proc. of IEEE-FUZZ 2000*, San Antonio TX, IEEE Press, May 2000.

[15] Shannon T.T., "Partial, Noisy and Qualitative Models for Adaptive Critic Based Neurocontrol", in *Proceedings of IJCNN'99*, Washington D.C., IEEE, 1999.

[16] Shannon T.T., G.G. Lendaris "Qualitative Models for Adaptive Critic Neurocontrol", in *Proceedings of SMC'99*, IEEE, 1999.

[17] Takagi, T, & M. Sugeno, "Fuzzy Identification of Systems and its Application to Modeling and Control", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 15, # 1, 1985.

[18] Werbos, P.J., "A Menu of Designs for Reinforcement Learning Over Time", in Miller, W.T., R.S. Sutton, & P.J. Werbos eds., *Neural Networks for Control*, MIT Press, Cambridge, MA, 1990, pp. 67- 95.

[19] Werbos, P.J., "Approximate Dynamic Programming for Real-Time Control and Neural Modeling", in D.A. White & D.A. Sofge eds., *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches*, Van Nostrand Reinhold, New York, 1992, pp. 493 - 525.

[20] Widrow, B., N Gupta & S. Maitra, "Punish/Reward: Learning with a Critic in Adaptive Threshold Systems", *IEEE Transactions on Systems, Man & Cybernetics*, vol. 3 (5), pp. 455-465, 1973.

[21] Yen, J. & R. Langari, *Fuzzy Logic*, Prentice Hall, Upper Saddle River, NJ, 1999.