Back-Propagation Neural Networks for Nonlinear Self-Tuning Adaptive Control

Fu-Chuang Chen

ABSTRACT: A back-propagation neural network is applied to a nonlinear self-tuning tracking problem. Traditional self-tuning adaptive control techniques can only deal with linear systems or some special nonlinear systems. The emerging back-propagation neural networks have the capability to learn arbitrary nonlinearity and show great potential for adaptive control applications. A scheme for combining back-propagation neural networks with self-tuning adaptive control techniques is proposed, and the control mechanism is analyzed. Simulation results show that the new self-tuning scheme can deal with a large unknown nonlinearity.

Introduction

Model uncertainty is a serious problem facing control engineers. Often, it is not possible to represent, adequately, system characteristics such as nonlinearity, time delay, saturation, time-varying parameters, and overall complexity. Recent developments in robust control and adaptive control have proven to be useful in some practical situations [1], [2]. Robust control design produces a constant gain controller, which stabilizes a class of linear systems over a range of system parameters, while adaptive control adjusts the controller characteristics to stabilize a system with unknown parameters.

Neural networks are of interest to the control community because they have the potential to treat many problems that cannot be handled by traditional analytic approaches. Back-propagation neural networks are the most prevalent neural network architectures for control applications because they have the capability to "learn" system characteristics through nonlinear mapping. Some investigations, in this respect, have been done, e.g., [3]–[6]. In [3] and [4], several schemes are proposed for the neural network to learn the inverse of the plant. In [5] and [6], neural nets are used directly as controllers, but this approach bears a less direct connection to traditional control design methods.

This article shows how neural networks can combine with self-tuning control algorithms to control a class of unknown discrete-time nonlinear systems. Back-propagation neural networks are reviewed briefly in the next section. The new neural network self-tuning control scheme is then described, followed by representative simulation results.

An early version of this article was presented at the Fourth IEEE International Symposium on Intelligent Control, Albany, New York, September 25-26, 1989. Fu-Chuang Chen is with the Department of Electrical Engineering, Michigan State University, East Lansing, MI 48824.

Back-Propagation Neural Networks

A back-propagation neural network is a layered network consisting of an input layer, an output layer, and at least one layer of nonlinear processing elements. The nonlinear processing elements, which sum incoming signals and generate output signals according to some predefined function, are called *neurons*. In this article, the function used by nonlinear neurons is called the *hyperbolic tangent function* h, which is similar to a smoothed step function

$$h(x) = (e^x - e^{-x})/(e^x + e^{-x})$$

The neurons are connected by terms with variable weights. The output of one neuron multiplied by a weight becomes the input of an adjacent neuron of the next layer.

In 1986, Rumelhart et al. [9] proposed a generalized delta rule known as *back-propagation* for training layered neural networks. For control engineers, it is appropriate to consider back-propagation neural networks as a tool to solve function approximation problems rather than pattern recognition problems. Consider a layered neural network $\hat{f}(x, W)$ with x as a vector representing inputs and W as a vector representing variable weights. It is desired to train $\hat{f}(x, W)$ to approximate a function $f: [a, b] \subset R \to R$, where $f([a, b])$ is a bounded set. In iteration k, let x_k which belongs to $[a, b]$ be selected randomly as the input of the neural net, let $\hat{f}[x_k, W(k)]$ be the output of the neural net, and let $f(x_k)$ which also belongs to $f([a, b])$ be the desired output. The task is to adjust all the variable weights of the neural net such that the error E_k can be reduced, where E_k is defined as

$$E_k = \tfrac{1}{2}\left\{\hat{f}[x_k, W(k)] - f(x_k)\right\}^2 \quad (1)$$

Let $w^i(k)$ be any element in $W(k)$. The effect of adjusting $w^i(k)$ to the error E_k is determined directly by the partial derivative $(\partial E_k)/[\partial w^i(k)]$. How to evaluate $(\partial E_k)/[\partial w^i(k)]$ is really the contribution of the back-propagation algorithm. Details can be found in [9]. Then $w^i(k)$ is adjusted in the direction to reduce E_k, where μ is a constant that specifies the update rate.

$$w^i(k+1) = w^i(k) - \mu(\partial E_k)/[\partial w^i(k)], \quad \forall w^i(k) \in W(k) \quad (2)$$

It can be shown that if μ is small enough, then

$$\left\{\hat{f}[x_k, W(k+1)] - f(x_k)\right\}^2 < \left\{\hat{f}[x_k, W(k)] - f(x_k)\right\}^2, \quad \text{if } E_k > 0 \quad (3)$$

The interpretation of the above equation is that the output of the updated neural network approaches the output of the function to be approximated for the same input, i.e., x_k. Although no theoretical convergence results are available about the training, it is expected that the average error

$$(1/N)\sum_{k=1}^{N} E_k$$

will decrease and converge as N gets large. This is based on past experience as reported in the literature.

Recently, Hecht-Nielsen [10] gave some analytic results about the capability of layered neural networks to approximate nonlinear functions. He showed in [10] that under certain conditions, given any $\epsilon > 0$, there exists a three-layered neural network (with an input layer, an output layer, and a nonlinear hidden layer) that can approximate a function f to within ϵ mean-squared error accuracy. This result shows that layered neural networks can be applied to many general situations encountered by control engineers. It is noted in [10], however, that the above analytic results do not deal with the ability of the neural network to acquire the correct

0272-1708/90/0400-0044 $01.00 © 1990 IEEE

weights by using existing learning laws, which is still an open question.

Neural Network for Self-Tuning Tracking

Many interesting systems can be described by a nonlinear model in which the control appears linearly. This article considers a class of *single-input/single-output* feedback linearizable systems, where y is the output, u is the input, and g is a nonzero function.

$$y_{k+1} = f(y_k, y_{k-1}, \ldots, y_{k-p}, u_{k-1}, u_{k-2}, \ldots, u_{k-p}) + g(y_k, y_{k-1}, \ldots, y_{k-p}, u_{k-1}, u_{k-2}, \ldots, u_{k-p})\, u_k \quad (4)$$

If we know both $f(\cdot)$ and $g(\cdot)$ of (4), we can use the following control, and the system output y_{k+1} will exactly track the desired output d_{k+1}.

$$u_k = -\frac{f(\cdot)}{g(\cdot)} + \frac{d_{k+1}}{g(\cdot)}$$

Since $f(\cdot)$ and $g(\cdot)$ are unknown, neural networks can be used to "learn" to approximate these functions and generate suitable controls. In order to simplify the problem and focus on the control mechanism, the discussion in the rest of the section treats the first-order system.

$$\textit{Plant} \quad y_{k+1} = f(y_k) + g(y_k)\, u_k \quad (5)$$

Although the scalar functions f and g are unknown, it will be assumed that the sign of $g(y_k)$ is known. Based on Hecht-Nielsen's theorem stated in the previous section, the system can be modeled by the neural network shown in Fig. 1, where $W = [w_0, w_1, \ldots, w_{2p}]$ and $V = [v_0, v_1, \ldots, v_{2q}]$.

$$\textit{Model} \quad \hat{y}_{k+1} = \hat{f}[y_k, W(k)] + \hat{g}[y_k, V(k)]\, u_k \quad (6)$$

Notice that $\hat{f}(0, W) = w_0$ and $\hat{g}(0, V) = v_0$. The neurons labeled "L" are linear ones, which are able to scale and shift incoming signals. The neurons labeled "H" are nonlinear ones using the hyperbolic tangent function. It is assumed that $\hat{f}(y_k, W)$ and $\hat{g}(y_k, V)$ are complex enough (i.e., contain enough neurons) to be able to approximate $f(y_k)$ and $g(y_k)$ to the desired accuracy. Although some preliminary results are available for predicting the optimal number of nonlinear hidden neurons [11], no mature techniques are available to determine a proper neural network size for this application. Therefore, trial and error must be used.

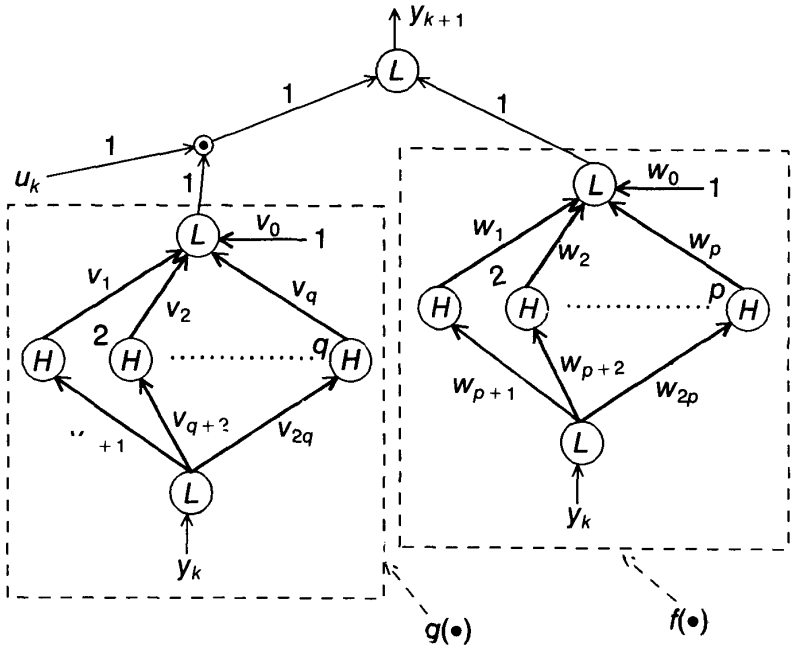

Fig. 1. The neural network model of the nonlinear system.

The control goal is for the plant output to track a command $\{d_k\}$. The overall control scheme is depicted in Fig. 2. At time-step k, the following control is applied to both the plant (5) and the model (6).

$$u_k = -\frac{\hat{f}[y_k, W(k)]}{\hat{g}[y_k, V(k)]} + \frac{d_{k+1}}{\hat{g}[y_k, V(k)]} \quad (7)$$

In contrast to the open-loop function approximation, feedback dramatically changes the role of the weights of the neural network. The output of the plant (8) depends on the weights of the neural net and serves as the desired output of the neural network.

$$y_{k+1} = f(y_k) + g(y_k)\left\{-\frac{\hat{f}[y_k, W(k)]}{\hat{g}[y_k, V(k)]} + \frac{d_{k+1}}{\hat{g}[y_k, V(k)]}\right\} \quad (8)$$

If (7) is substituted into (6), the output of the neural network is d_k, which is independent of $W(k)$ and $V(k)$. Let $e_{k+1} = d_{k+1} - y_{k+1}$, and define the output error to be

$$E_k = \tfrac{1}{2} e_{k+1}^2 = \tfrac{1}{2}(d_{k+1} - y_{k+1})^2 \quad (9)$$

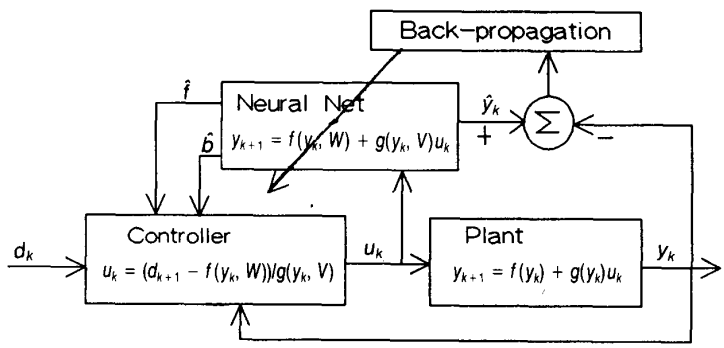

Fig. 2. Self-tuning control system using neural network.

Then, $W(k)$ and $V(k)$ are to be adjusted such that E_k can be reduced. It can be verified that

$$\frac{\partial E_k}{\partial w^i(k)} = \frac{g(y_k)}{\hat{g}[y_k, V(k)]} \cdot \left\{ \frac{\partial \hat{f}[y_k, W(k)]}{\partial w^i(k)} \right\} e_{k+1} \tag{10}$$

and

$$\frac{\partial E_k}{\partial v^j(k)} = \frac{g(y_k)}{\hat{g}[y_k, V(k)]} \cdot \left\{ \frac{\partial \hat{g}[y_k, V(k)]}{\partial v^j(k)} \right\} u_k e_{k+1} \tag{11}$$

The quantities $\{\partial \hat{f}[y_k, W(k)]\}/[\partial w^i(k)]$ and $\{\partial \hat{g}[y_k, V(k)]\}/[\partial v^j(k)]$ can be calculated by using the techniques provided in [9], although some minor modification is needed. The function $g(y_k)$ in (10) and (11) is unknown, but the sign is assumed known. So $g(y_k)$ is replaced by $\text{sgn}[g(y_k)]$ before (10) and (11) are used in the following updating rules:

$$w^i(k+1) = w^i(k) - \eta_k \left\{ \text{sgn}[g(y_k)] \div \hat{g}[y_k, V(k)] \cdot \left\{ \frac{\partial \hat{f}[y_k, W(k)]}{\partial w^i(k)} \right\} e_{k+1} \right. \tag{12}$$

$$v^j(k+1) = v^j(k) - \mu_k \left\{ \text{sgn}[g(y_k)] \div \hat{g}[y_k, V(k)] \right\} \cdot \left\{ \frac{\partial \hat{g}[y_k, V(k)]}{\partial v^j(k)} \right\} u_k e_{k+1} \tag{13}$$

The positive scalars η_k and μ_k specify the learning rates at time-step k. With y_{k+1}, $W(k+1)$, and $V(k+1)$ available [see (8), (12), and (13)], the functions $\hat{f}[y_{k+1}, W(k+1)]$ and $\hat{g}[y_{k+1}, V(k+1)]$ are calculated by the neural network. Then $\hat{f}[y_{k+1}, W(k+1)]$, $\hat{g}[y_{k+1}, V(k+1)]$, and d_{k+2} can be used to generate u_{k+1} by (7) for the next time-step.

It is important to consider how the quadratic error in function approximation (1) is different from that in self-tuning control (9). Equation (1) defines the error between the neural network and the function to be approximated by the neural network. Equation (9) defines the error between the plant output and the reference command. The purpose of using the neural network in self-tuning control is to learn two functions, i.e., $f(y_k)$ and $g(y_k)$, *so that* the correct control can be calculated using (7). Reducing $E_k = \frac{1}{2}(d_{k+1} - y_{k+1})^2$ alone does not imply that the neural network is learning the plant dynamics. In fact, the neural network may not be learning. An extreme example is to set η_k of (12) to 0 (for all k), then the neural network can never come close to the plant dynamics and the plant output will not track the command. On the other hand, how does one know that the neural network is learning if η_k is set to be equal to μ_k? Fortunately, it can be shown that under the gradient-descent updating rules of (12) and (13), if the scalar η_k is chosen correctly, then the error will decrease. Let

$$\eta_k = \mu_k \left(\sum_j \left\{ \partial \hat{g}[y_k, V(k)]/[\partial v^j(k)] \right\}^2 \right) \div \left(\sum_i \left\{ \partial \hat{f}[y_k, W(k)]/[\partial w^i(k)] \right\}^2 \right)$$

$$\text{sgn}\left\{ \hat{g}[y_k, V(k)] \right\} = \text{sgn}[g(y_k)]$$

and μ_k is small enough, then

$$\phi_{k+1}^2(y_k) + \psi_{k+1}^2(y_k) < \phi_k^2(y_k) + \psi_k^2(y_k), \text{ if } E_k > 0 \tag{14}$$

where

$$\phi_{k+1}(y_k) = \hat{g}[y_k, V(k+1)] - g(y_k)$$

$$\phi_k(y_k) = \hat{g}[y_k, V(k)] - g(y_k)$$

$$\psi_{k+1}(y_k) = \hat{f}[y_k, W(k+1)] - f(y_k)$$

and

$$\psi_k(y_k) = \hat{f}[y_k, W(k)] - f(y_k)$$

The partial derivatives $\{\partial \hat{f}[y_k, W(k)]\}/[\partial w^i(k)]$ and $\{\partial \hat{g}[y_k, V(k)]\}/[\partial v^j(k)]$ are assumed available because they are needed by the updating rules [i.e., (12) and (13)] and can be calculated by back-propagation techniques [9] as mentioned before. These statements suggest that, in practice, if $\hat{g}[y_k, V(k)]$ is of the same sign as $g(y_k)$ and if one chooses μ_k and η_k properly, the quadratic error $\phi_k^2(y_k) + \psi_k^2(y_k)$ can be reduced at each time-step in the same way as in the open-loop function approximation [see (3)]. This gives more confidence in applying neural networks to self-tuning control, although no closed-loop stability or convergence result is available.

If the neural network has learned to approximate the plant dynamics well, the back-propagation procedure may be turned off. Then the control u_k can be generated from the neural network quite rapidly compared with the time constant of the plant. In the case of applying computed-torque-style methods for controlling robotics, neural networks coupled with the self-tuning algorithm have at least two advantages compared with traditional approaches:

- No detailed model of the robotics is needed, and the neural network may have the capability to catch those features traditionally omitted in mathematical models.
- Because of the massive parallel processing capability of neural networks, the bottleneck introduced by serial computers may be eliminated.

Back-propagation learning is a time-consuming process, however. Searching for faster learning methods is a current research issue [10].

Simulation

The simulation is carried out using Neuralworks Professional II, a neural network simulation package, running on a 20M Hz 80386 PC AT. The self-tuning control system is shown in Fig. 2. The layered neural network part of the control system can be constructed fairly easily by the package, and the back-propagation algorithm is available in the package. The rest of the control system is written into an interrupt service routine (coded in C), which interacts with the neural network. Whenever the neural network needs to receive or send out data, an interrupt is generated. The interrupt service routine not only provides and receives data, it also generates reference commands, calculates the control, and implements the plant. The plant used in this simulation is

$$y_{k+1} = 0.8 \sin(2y_k) + 1.2u_k$$

The plant is modeled by a neural network shown in Fig. 3. The control u_k is applied to neuron #2; the plant output y_k is applied to neuron #3. The unknown positive coefficient of u_k is represented by the weight $\hat{g}$ between neuron #2 and neuron #25 in Fig. 3. The unknown nonlinear function $0.8 \sin(2y_k)$ is modeled by the neural network between neuron #3 and neuron #24 with two hidden layers. Each hidden layer contains 10 nonlinear neurons. While one nonlinear hidden layer is shown to be enough for approximating nonlinear mapping, using more than one hidden layer may have the advantage of significantly reducing the number of hidden

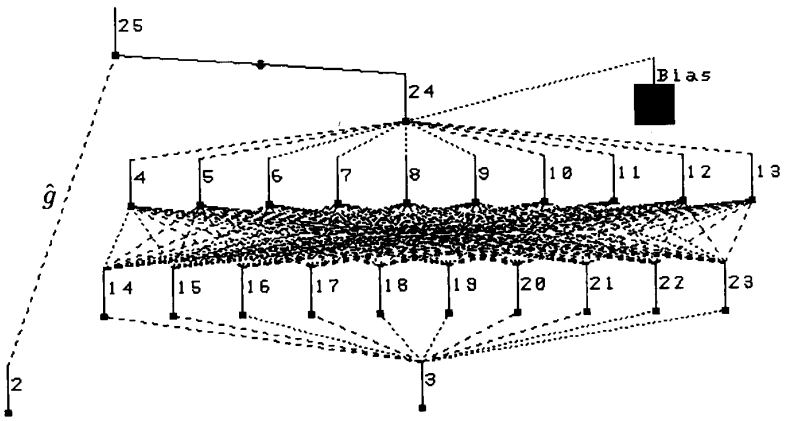

Fig. 3. The neural network model of the plant.

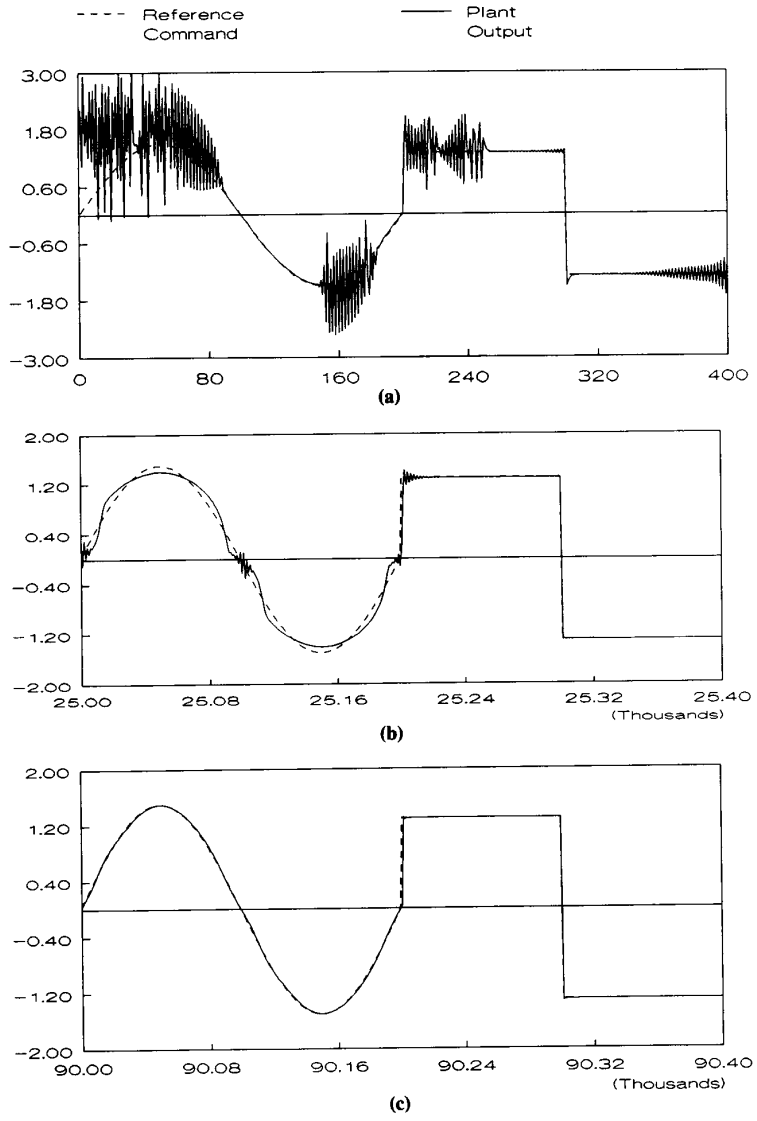

Fig. 4. Representative simulation results.

neurons needed for the same task. The initial weights, except $\hat{g}(0)$, of the neural network are selected randomly between -0.1 and 0.1. The specific weight $\hat{g}(0)$ is chosen between 0.01 and 0.1 such that it is of the same sign as the unknown positive number it is expected to approximate. Since the inverse of $\hat{g}(k)$ has a direct effect on the magnitude of u_k, $\hat{g}(k)$ is set to be 0.01 if $\hat{g}(k) < 0.01$. It is desirable to select small initial weights in order to avoid the saturation of nonlinear neurons.

The learning rates μ and η are chosen to be 0.016 and 0.95 initially and decreased gradually later for better convergence. The choice of μ and η are made after observing the behavior of the neural network for a long period of time and enough experience is accumulated. When μ is too large, it is observed that $\hat{g}(k)$ jumps around and $\hat{f}[\, y(k), W(k)]$ hardly learns to approximate $f(\, y_k)$ well. So, it is important to select μ and η such that learning takes place at about the same pace in the two subparts of the neural network, i.e., $\hat{g}$ and $\hat{f}(\, y_k, W)$.

Figure 4 shows illustrative simulation results. The 20M Hz 80386 CPU can run about seven time-steps per second. The processing speed depends on the complexity of the neural network and on the efficiency of the interrupt service routine. In the first 400 time-steps [Fig. 4(a)], wild oscillation is observed at the plant output. Although the plant output finally satisfactorily converges to the reference command [Fig. 4(c)], it takes a long time for the system to achieve this convergence.

Conclusion

Self-tuning adaptive control is traditionally limited to unknown linear systems. By introducing back-propagation neural networks into the self-tuning control scheme, it is demonstrated that the new control method has the potential to deal with unknown feedback linearizable nonlinear systems. Behind the control scheme proposed in this article, many practical and theoretical questions can be raised. Currently, some stability and minimal phase issues are under investigation [12].

Acknowledgment

I thank Dr. Hassan Khalil of Michigan State University for his guidance and tremendous help and support.

References

[1] A. Yue and I. Postlethwaite, "H_∞-Optimal Design for Helicopter Control," *Proc. 1988 Amer. Contr. Conf.*, pp. 1679–1684.

[2] J. V. Amerongen, "Adaptive Steering of Ships—A Model Reference Approach," *Automatica*, vol. 20, no. 1, pp. 3-14, 1984.
[3] D. Psaltis, A. Sideris, and A. A. Yamamura, "A Multilayered Neural Network Controller," *IEEE Contr. Syst. Mag.*, Apr. 1988.
[4] W. Li and J. J. E. Slotine, "Neural Network Control of Unknown Nonlinear Systems," *Proc. 1989 Amer. Contr. Conf.*, pp. 1136-1141.
[5] A. Guez and J. Selinsky, "A Neuromorphic Controller with a Human Teacher," *Proc. IEEE 1988 Intl. Conf. Neural Networks*, pp. II-595-602.
[6] V. V. Tolat and B. Widrow, "An Adaptive Broom Balancer with Visual Inputs," *Proc. IEEE 1988 Intl. Conf. Neural Networks*, pp. II-641-647.
[7] P. P. Kumar and P. Varaiya, *Stochastic Systems: Estimation, Identification and Control*, Prentice-Hall Inc., 1986.
[8] M. L. Minsky and S. S. Papert, *Perceptron*, MIT Press, 1969.
[9] D. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation," *Parallel Distributed Processing*, D. Rumelhart and J. McClelland (Eds.), vol. 1, MIT Press, 1986.
[10] R. Hecht-Nielsen, "Theory of the Back-Propagation Neural Network," *Proc. IEEE 1989 Intl. Conf. Neural Networks*, pp. I-593-605.
[11] S. Y. Kung and J. N. Hwang, "An Algebraic Projection Analysis for Optimal Hidden Units Size and Learning Rates in Back-Propagation Learning," *Proc. IEEE 1988 Intl. Conf. Neural Networks*, pp. I-363-370.
[12] F. C. Chen, "A Local Convergence Result on Self-Tuning Regulation Using Neural Networks," submitted to *1990 Amer. Contr. Conf.*

Fu-Chuang Chen received the B.S. degree in power mechanical engineering in 1983 from National Tsing Hua University, Taiwan, Republic of China, and the M.S. degree in systems science in 1986 from Michigan State University. Currently, he is completing his Ph.D. work in electrical engineering at Michigan State University. From 1983 to 1985, he was in the national service in the Chinese Army. His research interests are in control theory, neural networks, and artificial intelligence.

Control Society Award Nominations

Nominations are soon expected for four awards given out by the IEEE Control Systems Society. Further information and help on nominations are available from the Society Vice President for Member Activities, Prof. Jason L. Speyer, UCLA, Mechanical, Aerospace and Nuclear Engineering Department, 5713 Boelter Hall, Los Angeles, CA 90024; phone: (213) 206-4451.

George S. Axelby Outstanding Paper Award

Every year, the Control Systems Society awards up to three outstanding paper awards to authors of papers published in the *IEEE Transactions on Automatic Control* during the two preceding calendar years. The award is named after George S. Axelby, founding editor of the *Transactions*. We are now soliciting nominations for this year's award, from papers published in the *IEEE Transactions on Automatic Control* from January 1988 through December 1989 (volumes 33 and 34). This is an open invitation to nominate papers for consideration. The deadline for nominations is May 15, 1990. Nominations should be sent to: Prof. Tamer Basar, Axelby Award Chairman, Coordinated Science Laboratory, University of Illinois, 1101 W. Springfield Avenue, Urbana, IL 61801/USA; phone: (217) 333-3607 (E-mail: thasar@uicsl.csl.uiuc.edu).

1989 Control Systems Technology Award

Nominations are open for the the new Control Systems Technology Award, which will be awarded for the second time this year. This award is to be given for outstanding contributions to control systems technology, either in design and implementation or in project management. It may be conferred on either an individual or a team. The prize is $1000 and a certificate. Nominations are open to all. The prize is to be awarded annually at the IEEE Conference on Decision and Control. Deadline for nominations is April 30, 1990. Please send your nominations, together with supporting documents, to the Chairman of the Technology Award Committee: Prof. William R. Perkins, Coordinated Science Laboratory, 1101 W. Springfield Ave., University of Illinois, Urbana, IL 61801; phone: (217—333-0283 (E-mail: Perkins at uicsl.csl.uiuc.edu).

1990 Bode Lecture Prize

Nominations are open for the new Control Systems Society Bode Lecture Prize. Selection of the recipient for 1991 will be announced at the 1990 Conference on Decision and Control in December.

This award is to be given to an individual who has made distinguished contributions to control and systems science or engineering. The recipient will give a plenary lecture at the CDC, which will survey and evaluate a significant contribution to control science or to engineering and will be asked to write a similar article for the *Transactions*. The award consists of $1000 and a certificate plus reasonable travel expenses to the CDC to receive the award and present the lecture. This award is open to all. The deadline for submission is April 30, 1990.

The Bode Award Prize Committee consists of the Society President and the members of the Executive Committee. Nominations should be submitted to the President, William S. Levine, Dept. of Electrical Engineering, University of Maryland, College Park, MD 20742; phone: (301) 454-6841.

Control Systems Society Distinguished Member Award

The Distinguished Member Award is given to individuals who have made substantial technical contributions to the field of interest of the Society and to the work of the Society. Attainment of IEEE Fellow grade constitutes sufficient evidence of substantial technical contributions. Normally, a nominee shall be expected to have served on the Board of Governors or on one of the committees of the Society. However, the expected level of contributions to the Society goes beyond mere membership on the board or on committees. Service as Society President or as *Transactions* or *Magazine* Editor shall constitute sufficient, but not necessary, evidence of the required level of service to the Society. Nominations are due by October 1, 1990, and should be submitted to the President, William S. Levine.