Pergamon

Mechatronics Vol. 5, No. 8, pp. 949-962, 1995
Elsevier Science Ltd
Printed in Great Britain.

0957-4158 (95) 00063-1

A KNOWLEDGE-BASED APPROACH TO IMPROVE NEURAL NETWORK CONTROL DESIGN

MO-YUEN CHOW and JASON T. TEETER

Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, NC 27695-7911, U.S.A.

(*Received* 23 *December* 1994; *revised and accepted* 27 *June* 1995)

Abstract—Control engineers have been investigating and developing different on-line adaptation schemes to fine-tune performance of controllers after off-line design. Artificial neural network technology has shown satisfactory results for many control applications. However, certain types of nonlinear problems are difficult for the neural controller to learn by *conventional* on-line adaptation schemes, while the nonlinear system can be effectively controlled by incorporating heuristics knowledge. This paper presents an effective approach to incorporate heuristics control knowledge into a neural controller by off-line pre-training, then fine-tune the neural controller performance further by on-line adaptation. Experimental results on a servomotor system with significant nonlinear friction characteristics are used to demonstrate the effectiveness of the design approach.

1. INTRODUCTION

Artificial neural network (ANN) technology has become increasingly popular as a tool for performing tasks such as pattern recognition, time series prediction, system identification and automatic control [1–3]. The control of nonlinear time-varying systems is one application in which neurocontrollers have been shown to provide performance and flexibility superior to that of conventional linear controllers in many cases [3]. The choices of network classes and topologies often depend upon the characteristics of the system being controlled [1]. However, several general neurocontroller design approaches for nonlinear systems have been developed [3].

One of the most popular neural network control techniques involves using an ANN in the forward path of a feedback system as shown in Fig. 1.

The inputs to the network are usually present and past values of error, e, defined as the difference between the reference input r and system output y. The outputs of the network, u, are used as inputs to the plant. Different neural control design and

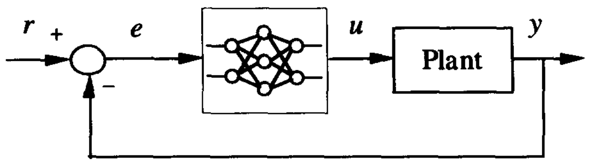

Fig. 1. A common neural control method.

adaptation approaches have been reported in different technical journals and transactions. One of the strong features of neural control is its capability to adapt on-line in order to minimize some prespecified cost index [4]. An inverse nonlinear control strategy using neural networks is proposed in [5]. The neurocontroller is trained to approximate the inverse mapping of the plant, and inputs to the neurocontroller are then conditioned in order to obtain desired error dynamics. A method in which a neural network is used in parallel with an existing neural controller is developed in [6]. The parallel network is adapted to correct system performance so that the mapping of the base controller is preserved and the convergence speed of the adaptation process is increased.

However, successful adaptation of on-line neural controllers in many cases requires careful and substantial design effort. One of the common practices is to pre-train a neural controller off-line based on some simplified design methods, such as a proportional integral (PI) controller based on *approximated* system models that can provide reasonable control performance, before putting the neural controller for on-line fine tuning [7]. This approach can speed up the neural controller on-line adaptation process and increase the closed-loop on-line adaptation stability because the initial weights of the neural controller are much closer to the optimal final weights (if they exist) after the pre-training process.

In many control problems, conventional control strategies may not provide satisfactory results, nor can the pre-trained neural controller fine-tune its performance even after on-line adaptation until a *significant* amount of time and effort is spent. However, in many cases, knowledge-based techniques can be incorporated in the control-loop to provide an effective solution. Different control schemes such as fuzzy control and gain scheduling have been investigated and developed in this avenue and have provided satisfactory results [8]. However, knowledge-based techniques are in general difficult to adapt on-line to provide optimal control. Recently, a significant amount of research effort has been invested in the adaptation of fuzzy logic. One of the popular approaches is to implement fuzzy logic in neural network structures for on-line adaptation.

This paper presents the design approach to incorporate heuristic knowledge into fuzzy logic controllers, which can provide reasonable performance for a servomotor with significant nonlinear friction characteristics, then pre-train the neural controller to learn the fuzzy control before placing the neural controller on-line to further optimize its performance.

The neural control implementation presented here is similar to that presented in [6] except that the development of the supplementary network is different. The supplementary network is first trained off-line to mimic a fuzzy system which has been designed using a knowledge base obtained through observation of the plant dynamics [8]. If a human operator were assigned the task of controlling the motor using a joystick, for example, he might experience difficulty compensating for the nonlinearities of the motor. As the operator obtains qualitative knowledge of the motor dynamics, however, he should be able to modify his control policies to compensate for the nonlinearities. Assuming that these modifications are describable by fuzzy rules of the form "If A and B then C," fuzzy logic can be used to incorporate the heuristic knowledge of the operator into the design of an automatic controller. The supplementry network which learns the fuzzy logic mapping is used to modify the control policies of the base neurocontroller in order to compensate for undesired performances.

A servomotor with nonlinear mechanical characteristics, which will be explained in later sections, is used to illustrate the effectiveness of the proposed method. A neural network is trained off-line to learn a PI controller input/output mapping so that the pre-trained neural controller will have better initial weights for on-line adaptation. A cost index for on-line training to fine-tune the neural controller performance in order to compensate the undesirable closed-loop control performance due to the nonlinearity is described, and problems encountered with on-line training of the neurocontroller are discussed. The design methodology proposed in this paper is to effectively incorporate heuristic knowledge into a neural controller. Heuristic knowledge of the motor dynamics is used to design a fuzzy logic rule which can improve the performance of the closed-loop system. A supplementary network is trained off-line to learn the fuzzy mapping and is then trained on-line in order to further improve performance over the operating range of the system.

2. D.C. MOTOR SYSTEM DESCRIPTION

Servomotors are used extensively in industry for applications such as robot arm drives, machine tools, rolling mills and aircraft control [9, 10]. A block diagram of the popular unity-feedback motor control system is shown in Fig. 2, where r is the desired output and d represents a load disturbance.

In this paper, the speed control of a D.C. servomotor system with significant mechanical nonlinear characteristics is used to illustrate the effectiveness of the proposed control methodology. Linear time-invariant characteristics of D.C. motors have been emphasized in classical control text books, in which most conventional control techniques such as transfer functions, zero-pole placements, Bode plots, PI(D) (where D stands for derivative) algorithms can be applied directly. Unfortunately, in general, only expensive D.C. motors are manufactured to provide all the nice linear time-invariant characteristics over a reasonably wide operating range. In reality, many D.C. motor systems have different nonlinear characteristics. Many conventional control algorithms developed for D.C. servomotors require significant amounts of on-line fine-tuning to provide better performance. The PI(D) controller is a classical example. The gain settings on the design stage generally provide reasonable motor performance over a small operating range. Engineers need to adjust the two (three) knots, corresponding to the P, I (and D) gain settings, from the off-shelf PI(D) controller on-line for their systems, and repeat the tasks for different motor operating

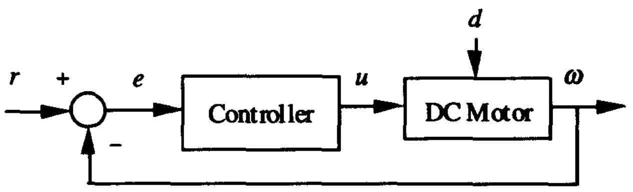

Fig. 2. Block diagram of the unity-feedback servomotor control system.

conditions. The motor used in this paper for illustration purposes has several nonlinear characteristics including backlash, dead zone and nonlinear friction [9]. The nonlinear friction has a significant effect on system dynamics, while the effects of the other nonlinearities are comparatively insignificant. Thus, we lump all the nonlinearity effects into the nonlinear friction effect in our future discussion for simplicity.

Several friction models have been proposed for the analysis of physical systems that involve some type of sliding motion [11]. A popular friction model known as the Stribeck curve [12] is shown in Fig. 3.

Purely velocity-dependent friction models such as the Stribeck curve do not include additional friction components, such as rising static friction and frictional memory, which may significantly affect performance [12]. Although it may be difficult to obtain an accurate analytical friction model, the *qualitative* effects of friction are well-known. At low velocities, motion may be intermittent and the resulting stick–slip phenomenon can lead to overshoot and large-amplitude limit cycles [13]. These effects are discussed in more detail in the following section, where they are demonstrated on an actual D.C. motor.

A schematic of the experimental system is shown in Fig. 4. The controllers are implemented on a 486 PC using the LabVIEW graphical programming package. The rotation of the motor shaft generates a tachometer voltage which is then scaled by interfacing electronic circuitry. A National Instruments data acquisition board receives the data via an Analog Devices isolating backplane. After a control value is computed, an output voltage is generated by the data acquisition board. The voltage is then scaled by the interfacing circuitry before being applied to the armature of the motor. Load disturbances are generated by subjecting a disc on the motor shaft to a

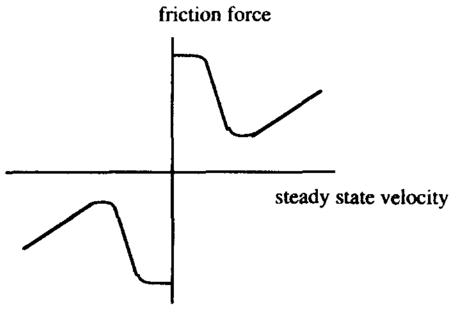

Fig. 3. Generalized Stribeck curve friction model.

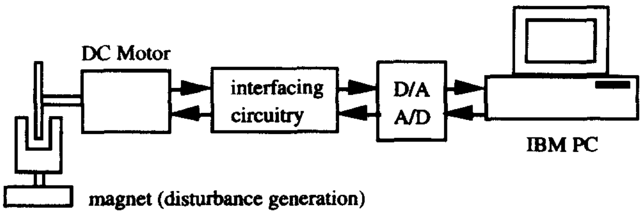

Fig. 4. Schematic of the experimental D.C. motor system.

magnetic field. The complete system setup is shown in Fig. 5, and a close-up view of the motor is given in Fig. 6.

3. PI CONTROLLER DESIGN AND PERFORMANCE

A PI controller is designed in order to provide training patterns that will be used to pre-train, thus initialize, a neurocontroller [14]. A second-order model of the form

$$\dot{\mathbf{x}} = A\mathbf{x} + bu \tag{1}$$

is used, where u is the armature voltage, $\mathbf{x} = [i_a \ \omega]^T$ and i_a and ω are the armature current and shaft angular velocity, respectively. A discrete version of (1) is obtained, and a digital PI controller is designed to provide critically damped step responses and rise time of about 1.5 sec.

The parameters of the motor are estimated via a standard system identification technique based on a linear D.C. motor model [15]. The parameters of the motor, based on the linear time-invariant model, are listed in Table 1. R_a and L_a are the resistance and the inductance of the motor armature circuit, respectively; J and f are the moment of inertia and the viscous-friction coefficient of the motor and load referred to the motor shaft, respectively; K is the constant relating the armature current to the motor torque, and K_b is the constant relating the motor speed to the D.C. motor's back-emf.

The amplifier for the motor saturates at 15 V, at which point the effectiveness of the feedback loop is lost. The output of the controller will remain saturated until the

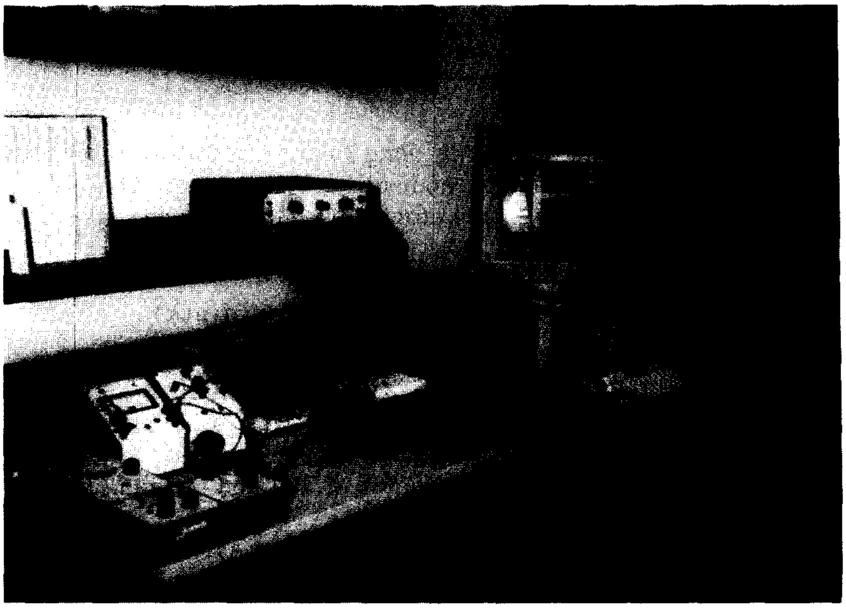

Fig. 5. Motor control system setup.

Fig. 6. Close-up view of the D.C. motor.

Table 1. D.C. motor parameters

R_a	4.67 Ω
L_a	170e−3 H
J	42.6e−6 kg-m^2
f	47.3e−6 N-m/rad/sec
K	14.7e−3 N-m/A
K_b	14.7e−3 V-sec/rad

error has been negative for a sufficiently long time to allow the magnitude of the integral term to become small. This phenomenon is called *integrator windup* [16]. To avoid potential problems caused by integrator windup, the following rules are used:

1. $e(-1) \equiv e(0)$.
2. If $|u_{\text{calc}}(k)| > 15$, set $u(k) = 15 \cdot \operatorname{sgn}(u(k))$,

where $u_{\text{calc}}(k)$ represents the control output calculated by the PI controller. The first rule prevents the control output from immediately taking on a large value due to the step input introduced at time $t = 0$. The second law is a simple back-calculation algorithm commonly used to avoid windup [16]. Typical responses of the linear model and the actual system for step reference inputs of 100 and 300 rad/sec are given in Figs 7 and 8.

It is clear that the friction nonlinearity has an adverse effect on system response, causing overshoot for small reference inputs and oscillatory transient responses for large reference inputs. It is reasonable to expect that load disturbances could induce limit cycles due to the nonlinear friction characteristics of the motor. To verify this,

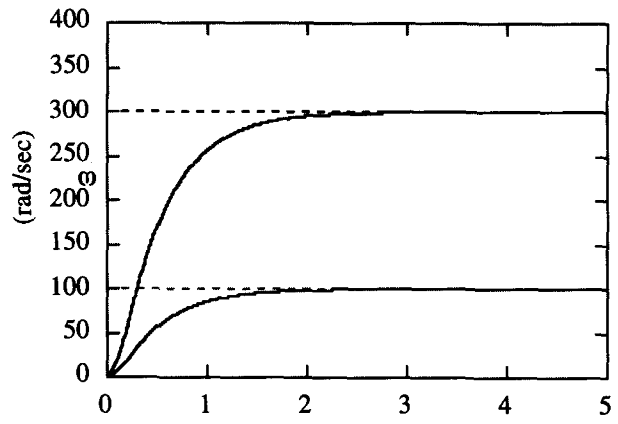

Fig. 7. Typical step responses of the linear D.C. motor model under PI control.

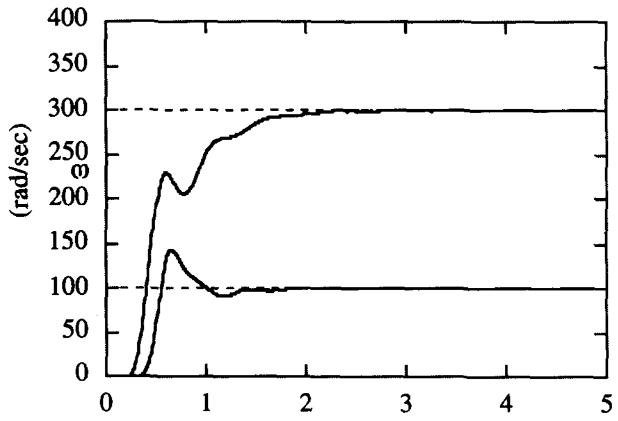

Fig. 8. Typical step responses of the actual D.C. motor under PI control.

step disturbances are introduced at steady-state conditions for the same reference inputs used in Figs 7 and 8. Figure 9 shows typical responses of the PI control system under load disturbance.

The load disturbances cause large-amplitude limit cycles, as expected, and this behavior is not limited to small reference inputs. Adjusting PI controller constants in order to provide smooth step responses and stable disturbance responses results in unacceptably large response times. Instead, a neural network is initialized using the PI controller and the resulting neurocontroller is trained on-line in an effort to improve response characteristics.

4. NEURAL CONTROL

A neurocontroller is trained on-line in an attempt to compensate for the friction nonlinearity that the PI controller cannot achieve. The symbol N_{n_i,n_h,n_u} is used to

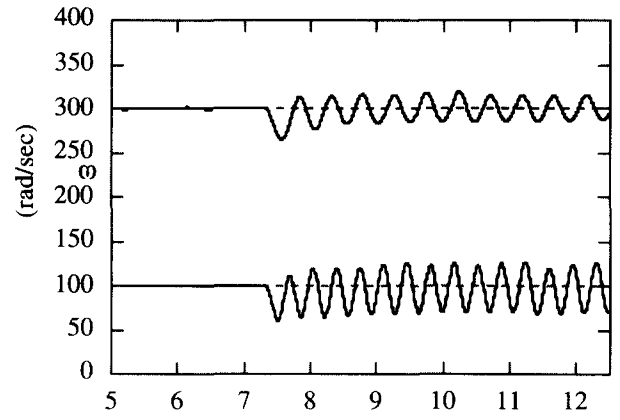

Fig. 9. Typical responses of the PI control system under load disturbance.

designate a class of three-layer feedforward ANN with n_i inputs, n_h hidden-layer neurons and n_u outputs. The PI controller uses values of $e(k)$ and $e(k-1)$ to compute $\Delta u(k)$, so inputs to the PI controller can be considered to be error (e) and change in error (ce), where change in error is defined to be the backward difference $e(k)-e(k-1)$.

An ANN of the class $N_{3,12,1}$, which has sufficient number of weights to learn the proper control knowledge discussed in this paper [17], is trained to learn the PI controller mapping $\{e(k), ce(k)\} \to \Delta u(k)$ described in the previous section by using the concept of a *functional link* [2, 18] so that the ANN inputs are defined by the mapping

$$H\colon \{x_i\} \to \{x_i, \underset{j>i}{x_i x_j}\}. \tag{2}$$

Thus, the inputs to the ANN are $\{e(k), ce(k), e(k)\cdot ce(k)\}$ and the output is the change in armature voltage. The initialized ANN is then trained on-line using a popular cost function of the form

$$J = \tfrac{1}{2}[c_1 e^2 + c_2 ce^2], \tag{3}$$

where c_1 and c_2 are weighting factors that represent the penalties placed on the error and change in error of the output.

The popular backpropagation method [14, 15] is used to train the neurocontroller on-line. ANN weights are adjusted at each time step using a small learning stepsize. An estimate of the plant input/output sensitivity $\partial y/\partial u$ needed for on-line training is obtained from the linear model of the plant. The weighting constants in Eqn (3) are chosen to be $\{c_1, c_2\} = \{0.1, 5.0\}$ in order to balance the speed and smoothness of response. Sinusoidal reference inputs of different frequencies and amplitudes are used to train the neurocontroller until the observed performance of the closed-loop system over the operating range of interest has converged. The performance of the trained neurocontroller is shown in Fig. 10, where the reference inputs used are the same as those shown in Fig. 8.

Figures 8 and 10 reveal that the step responses of the PI and neurocontrol systems

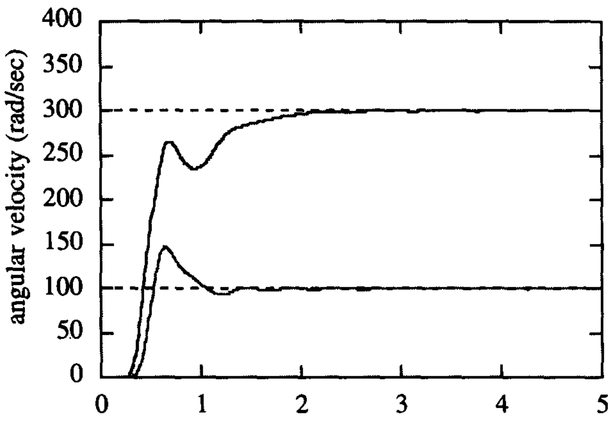

Fig. 10. Typical step responses of the trained neurocontrol system.

are very similar even after the on-line adaptation. The neurocontrol system exhibits slightly better convergence, but on-line training has *not* significantly compensated for the effects of the nonlinear motor friction. To further illustrate this point, step load disturbances are introduced during steady-state conditions for reference values of 100 and 300 rad/sec. The disturbance responses are shown in Fig. 11.

Basically, the results imply that the conventional on-line neural controller adaptation could not compensate the nonlinear friction effects. There are several options to be considered to improve the on-line training in order for the network to compensate the nonlinear friction effect on the servomotor system. One option for improving performance is to use a different cost index for on-line training. Due to the complexity of the friction nonlinearity, it will require a significant amount of effort and development time to find an appropriate cost index. Using a more elaborate cost index may also significantly increase the complexity of the adaptation process.

As mentioned previously, in general it is difficult to develop an accurate nonlinear

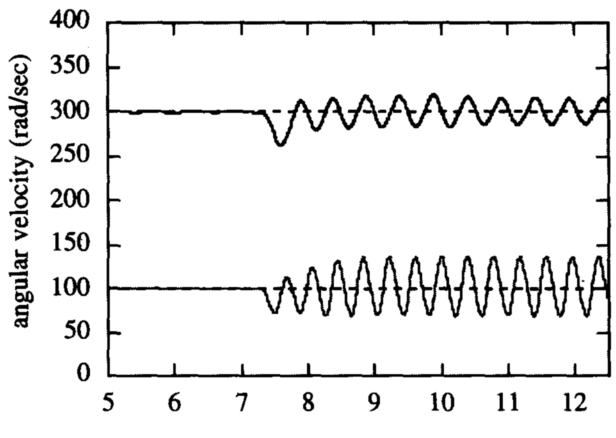

Fig. 11. Typical disturbance responses of the trained neural control system.

friction model for the servomotor system [9]. Although approximate models have been shown to be sufficient in some cases [16, 18, 19], an effective model-free approach may be more appealing if it results in a simpler design. This paper proposes another option to use fuzzy logic to incorporate heuristic knowledge of the effects of the nonlinear friction in order to improve performance, from which we incorporate the knowledge to train the neural controller.

5. FUZZY LOGIC FRICTION COMPENSATION

A fuzzy logic friction compensation methodology has been described in [8] and is briefly described in this section for completeness. The input to the motor can be written as

$$u(k) = u(k-1) + N(e(k), ce(k), \mathbf{w}), \tag{4}$$

where $N(\cdot)$ denotes the neural network mapping and $\mathbf{w}$ denotes the adjustable connection weights of the network. The following fuzzy logic rule is used to modify the controller output:

$$\begin{array}{l} \text{IF } r(k) \text{ IS SMALL} \\ \text{AND } u(k-1) \text{ IS LARGE} \\ \text{AND } \omega(k) \text{ IS SMALL} \\ \text{THEN } \mathit{DECREASE\ K}, \end{array} \tag{5}$$

where r is the reference, u is the motor input, ω is the motor speed and K is an attenuation factor incorporated into Eqn (4) which yields

$$u(k) = u(k-1) + K(r(k), u(k-1), \omega(k))N(e(k), ce(k), \mathbf{w}). \tag{6}$$

The membership functions $\{\mu(\cdot)\}$ for the fuzzy rule are shown in Fig. 12. Correlation-minimum encoding [20], which yields the minimum fuzzy degree of truth of the rule antecedents, is used to compute the rule output so that

$$K = 1 - 0.9 \min\{\mu_{\text{SMALL}}(r), \mu_{\text{LARGE}}(u), \mu_{\text{SMALL}}(\omega)\}. \tag{7}$$

The fuzzy rule compensates for the nonlinear friction by *reducing* the gain of the integral term when the system is most sensitive to the nonlinearity. The 0.9 factor is included in Eqn (7) to preserve the integrating operation of the controller so that steady-state tracking error is eliminated, provided the system is asymptotically stable.

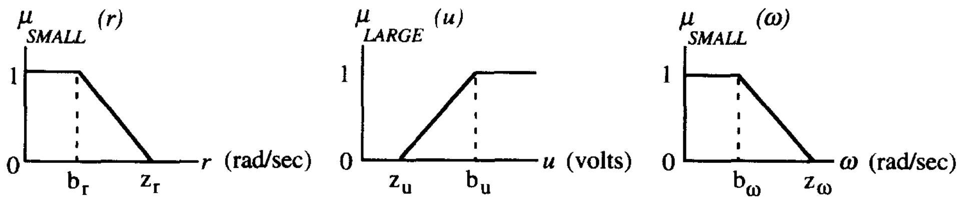

Fig. 12. Membership functions for fuzzy rule.

The parameters of the membership function are chosen to be

$$\{b_r, b_u, b_\omega\} = \{200, 6, 100\}$$
$$\{z_r, z_u, z_\omega\} = \{600, 2, 600\}. \tag{8}$$

Note that $z_r = z_\omega = 600$ even though the maximum attainable speed of our motor is about 500 rad/sec. The parameters in Eqn (8) describe the *shapes* of the membership functions over the range of attainable values for each variable; it is not required that the parameters themselves lie in these ranges.

6. RESULTS

A neural network of the class $N_{3,4,1}$ is trained to learn the mapping given by the fuzzy logic rule. This network is used as a supplementary network which modifies the gain of the network used in the initial control scheme. The overall structure is depicted in Fig. 13.

On-line training is performed again using the cost function given by Eqn (3), but only the weights of the supplementary network are adjusted. Typical step responses of the actual motor control system with the trained supplementary network are shown in Fig. 14.

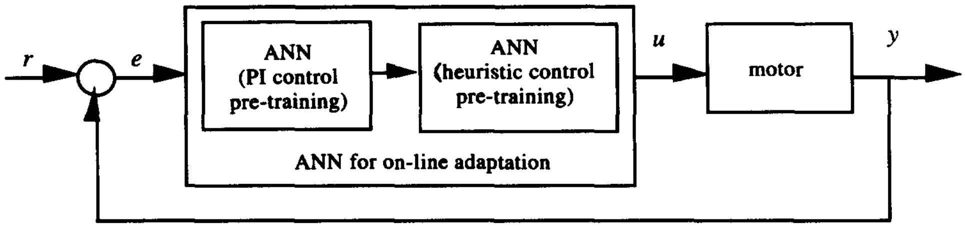

Fig. 13. Schematic diagram of the servomotor system neural control scheme.

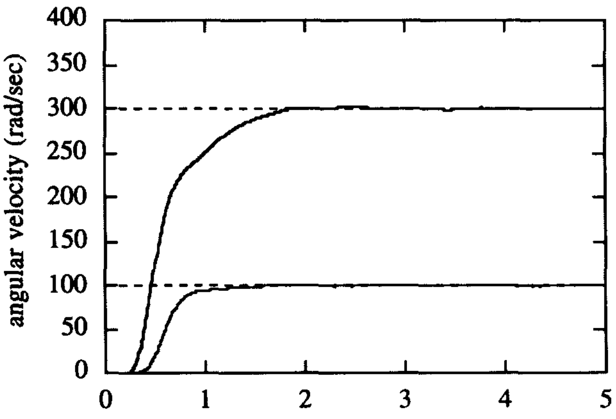

Fig. 14. Typical responses of the modified neural control system.

The step responses are smooth and exhibit negligible overshoot. Settling times are approximately the same as those of the original controller. Typical responses of the system with supplementary network under load disturbance are shown in Fig. 15. The step disturbances are the same as those used for the responses shown in Fig. 9.

The convergence of the response corresponding to the smaller reference is slower since the friction nonlinearity is more significant at lower speeds and the supplementary network has more of an effect on the change in input. A set of 200 reference/disturbance pairs has been used to test the stability of the modified system under load disturbance, and the system has exhibited asymptotically stable responses in every case.

Trajectories of the input and output of the motor with and without the supplementary network are shown in Fig. 16 in order to further illustrate the effects of the

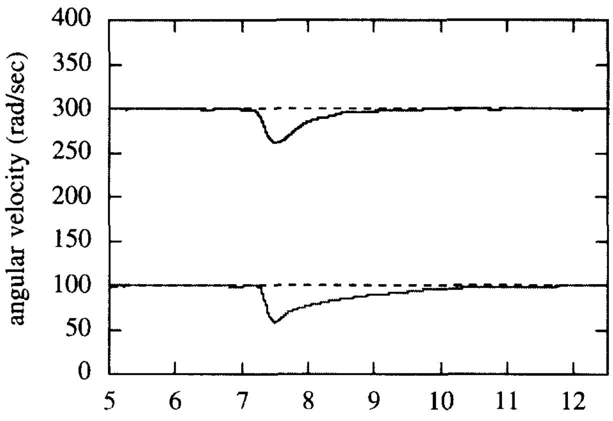

Fig. 15. Typical responses of the modified neural control system under load disturbance.

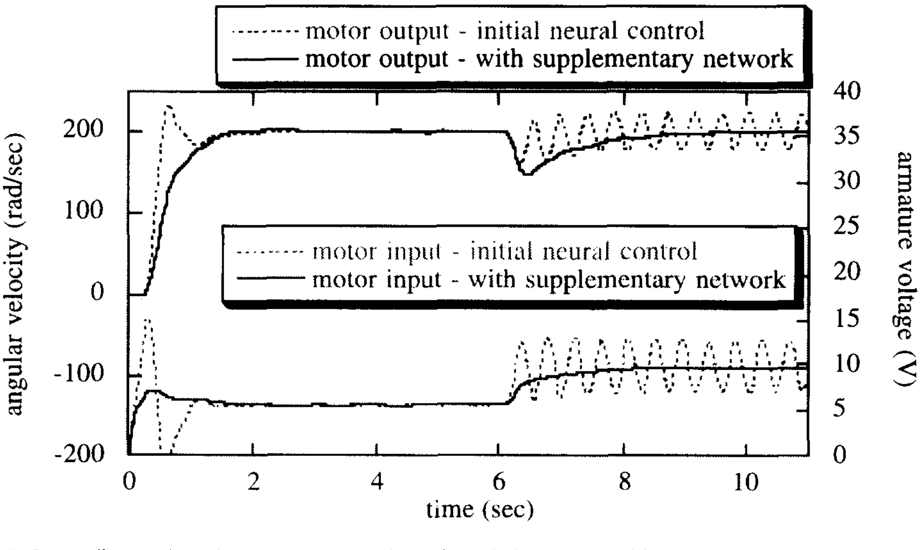

Fig. 16. Input (bottom) and output (top) trajectories of the motor with and without the supplementary network.

supplementary network. A step reference input of 200 rad/sec is used, and a step disturbance is introduced after the step responses have settled.

As indicated in the control structure shown in Fig. 13, there are several ways to view the controller structure. The PI control pre-trained neural network and the fuzzy control (knowledge-based) pre-trained neural network can be viewed separately as a two-step control. This modular viewpoint can provide the designer with more insight as to how the neural network controllers process information. On the other hand, the two neural controllers can be synthesized and viewed as a single controller. This approach views the controller providing correct input–output control mapping to compensate the system nonlinearity as well as performance control as an integrated controller.

In situations where we feel confident with either the PI and/or fuzzy controllers' optimal design, we do not need to *train* neural networks to learn their performance for further on-line adaptation. Unfortunately, this is hardly the case in most real world situations. If we are confident about the optimality of PI design while less confident on the fuzzy control design, we can just train a neural network to learn the fuzzy control, and cascade it with the PI controller for further on-line adaptation. Similar arguments apply to the less confident PI design and confident fuzzy control design. Of course, this hybrid control system complicates the neural control structure and requires modifications for the on-line adaptation schemes [1]. Furthermore, implementing the PI and fuzzy control in neural network structure can significantly improve the controller speed due to the inherent nature of parallel processing that neural network possesses, in addition to its easy adaptation features.

7. CONCLUSIONS

This paper has discussed a methodology to train a neural controller by incorporating heuristic knowledge to the control effort. The methodology includes a pre-training session and an on-line adaptation session. The pre-training session trains the network to learn reasonably well control surfaces from different knowledge sources, thus providing better initial weights for the neural controller for future successful on-line adaptation. Then the on-line adaptation session fine-tunes the controller performance, based on actual system performance. A servomotor system with significant nonlinear effect is used in this paper to illustrate the proposed methodology.

Acknowledgments—The authors would like to acknowledge the support of the Electric Power Research Institute, Research Contract RP8004-24: Intelligent Energy Control.

REFERENCES

1. Narendra K. S. and Parthasarathy K., Identification and control of dynamical systems using neural networks. *IEEE Trans Neural Networks* **1**(1), 4–27 (1990).
2. Pao Y., *Adaptive Pattern Recognition and Neural Networks*. Addison-Wesley, Reading, MA (1990).
3. Hunt K. J., Sbarbaro D., Zbikowski R. and Gawthrop P. J., Neural networks for control systems—a survey. *Automatica* **28**, 1083–1112 (1992).

4. Werbos P., Neural networks for control and system identification. In *Proceedings of the IEEE International Conference on Decision and Control*. IEEE Press, New York (1989).
5. Low T. S., Lee T. H. and Lim H. K., A methodology for neural network training for control of drives with nonlinearities. *IEEE Trans. Ind. Elec.* **39**, 243–249 (1993).
6. Lee T.-H., Tan W. K. and Ang M. H. Jr, A neural network control system with parallel adaptive enhancements applicable to nonlinear servomechanisms. *IEEE Trans. Ind. Elec.* **41**, 269–277 (1994).
7. Canudas C., Astrom K. J. and Braun K., Adaptive friction compensation in DC-motor drives. *IEEE J. Robotics Automat.* **RA-3**, 681–685 (1987).
8. Teeter J., Chow M. and Brickley J. J. Jr, A novel fuzzy friction compensation approach to improve the performance of a DC motor control system. Accepted for publication in *IEEE Trans. Ind. Elec.* (1995)
9. Kuo B. C. and Tal J., *DC Motors and Control Systems.* SRL Publishing, Champaign, IL (1978).
10. Say M. G. and Taylor E. O., *Direct Current Machines.* John Wiley, New York (1980).
11. Johnson C. T. and Lorenz R. D., Experimental identification of friction and its compensation in precise, position controlled mechanisms. *IEEE Trans. Ind. Applic.* **28**, 1392–1398 (1992).
12. Armstrong-Helouvry B., Stick slip and control in low-speed motion. *IEEE Trans. Automat. Control* **38**, 1483–1496 (1993).
13. DuPont P. E., Avoiding stick slip through PD control. *IEEE Trans. Automat. Control* **39**, 1094–1096 (1994).
14. Chow M. and Menozzi A., On the comparison of emerging and conventional techniques for DC motor control. *Proceedings of IECON* (1993).
15. Dorf R. C., *Modern Control Systems*, Sixth Edn. Addison-Wesley, Reading, MA (1992).
16. Astrom K. J. and Rundqwist L., Integrator windup and how to avoid it. In *Proceedings of the American Control Conference*, Volume 2, pp. 1693–1698 (1989).
17. Chow Mo-Yuen, Bilbro G. and Yee, Sui Oi, Application of learning theory to a single phase induction motor incipient fault detection artificial neural network. *Int. J. Neural Syst.* **2(1&2)**, 91–100 (1991).
18. Yip P. P. C. and Pao Y., A recurrent neural net approach to one-step ahead control problems. *IEEE Trans. Syst. Man, Cybernet.* **24**, 678–683 (1994).
19. Rumelhart D. E., Hinton G. E. and Williams R. J., Learning internal representations by error propagation. In *Parallel Distributed Processing* (Edited by Rumelhart D. E. and McClelland J. L.). MIT Press, Cambridge, MA (1986).
20. Werbos P. J., Backpropagation through time: what it does and how to do it. *Proc. IEEE* **78**, 1550–1560 (1990).