NEURAL ADAPTIVE TACKING CONTROL OF A LOW SPEED DC SERVO SYSTEM

Hu Hongjie Chen Jingquan Er Lianjie
College of Automation Science and Electrical Engineering
Beijing University of Aeronautics and Astronautics.
Beijing 100083, P.R.China
hongjiehu@263.net

ABSTRACT

Neural networks are well-suited for the modeling and control of complex physical systems because of their ability to handle complex input-output mapping without detailed analytical model of the systems. Representing a nonlinear dynamic mapping, diagonal recurrent neural networks (DRNN) are well-suited to deal with dynamic nonlinear problems. In this paper, a DRNN is used in a low speed DC servo system. First the DRNN is used as a identifier (DRNNI) to identify the inverse model of the system to be controlled though supervised, and then the DRNN is used as a feedforward controller (DRNNC) Combining with feedback controller, to generate control voltage to force the dc servo system to follow a pre-selected trajectories of position. Comparing with PID control method, experimental results are provided to illustrate the performance and effectiveness of the method proposed, even in the presence of much strong mechanical fiction and other nonlinear characteristics.

INTRODUCTION

For there are variation of parameters, such as load torque, inertia and mechanical friction, low speed dc servo system is a typical nonlinear system. It is difficult to get accurate system's mathematical model due to unstructured uncertainties for the unmodelled dynamics like nonlinear friction. It is impossible to get high accuracy response by means of traditional control method based on system's mathematical model, such as PID. The neural network area consist of a very promising direction to solve the problem relating to unknown nonlinear dynamic system. Hence, neural networks appear as a powerful tool for learning highly nonlinear dynamic systems. Their massive parallelism, very fast adaptation, and inherent approximation capability, have attracted extensively researchers in the field of system identification and control [1-4]. Many existing neural control laws for mechanical systems suffer from important shortcomings. Such as intensive computation effort and high storage capacity [5-6], so it is impossible to apply neural control algorithm to system such as sampling time is very short. This is a main cause for neural network controllers have not been widely used so far. This paper presented a neural-network based control scheme for a low speed dc servo system. To reduce calculating time, a DRNN is used to estimate the inverse model of plant to be controlled, parameters of neural network are updated on-line according to the dynamic backpropagation algorithm (DBP). To ensure system's initial robustness and close-loop stability, a fixed gain feedback controller is used.

$$\frac{di_a(t)}{dt}=\frac{R_a}{L_a}i_a(t)-\frac{K}{L_a}w(t)+\frac{1}{L_a}U_c(t)$$

DC SERVO SYSTEM

The low speed system's hardware setup is composed of a permanent dc motor, driving circuit, servo amplifier (PWM), a mechanical frame as an inertial load, interface circuit (A/D and D/A), an encoder for position sensing, and a personal computer (PETIUM I 133) is used as the programming environment, using Borlandc31 as programming language for the real-time control application. Sampling time is defined as 5ms. The block diagram of the hardware setup is shown in figure 1.

SYSTEM MODEL

The low speed dc servo system model is given by the follow equations

$$q(t)=\int w(t)dt \tag{1}$$

$$\frac{dw(t)}{dt}=\frac{K}{J_a}i_a(t)-\frac{D}{J_a}w(t)-\frac{1}{J_a}T_l(t) \tag{2}$$

$$O(k)=\sum_j w_j^o x_j(k), \qquad x_j(k)=f(s_j(k)), \tag{3}$$

where $w(t), U_c, i_a, R_a, L_a, J_a, K, D, T_l$ and $q(t)$ are the rotor speed ,terminal voltage, armature current, armature resistance ,armature inductance, rotor inertia, torque and back emf constant, damping constant, load torque, and the position angle, respectively. Solving

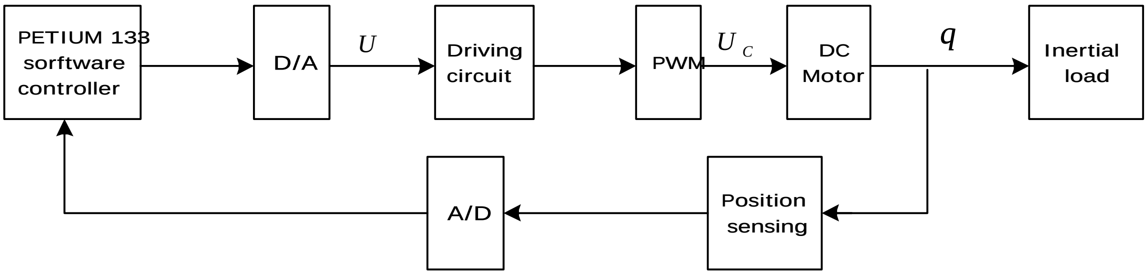


Fig.1 Block diagram of the hardware setup

equation (1), the motor transfer between the speed and the input voltage and the transfer between motor position angle and input voltage can be written as

$$\frac{w(s)}{U_c(s)} = \frac{K_m}{s^2 + a_1 s + a_2} \qquad (4)$$

$$\frac{q(s)}{U_c(s)} = \frac{K_m}{(s^2 + a_1 s + a_2)s} \qquad (5)$$

The armature voltage of the dc motor is supplied from the servo amplifier and is proportional to its control voltage $U(s)$. So the overall transfer function of the dc servo system can be written as bellow

$$\frac{w(s)}{U(s)} = \frac{K_s}{s^2 + a_1 s + a_2} \qquad (6)$$

$$\frac{q(s)}{U(s)} = \frac{K_s}{(s^2 + a_1 s + a_2)s} \qquad (7)$$

where constants a_1, a_2, K_m, K_s depend on the system parameters. Equation (6) and (7) can be written in the form of a differential equation as

$$U(t) = b_1\ddot{w}(t) + b_2\dot{w}(t) + b_3 w(t) \qquad (8)$$

$$U(t) = c_1\dddot{q}(t) + c_2\ddot{q}(t) + c_3\dot{q}(t) \qquad (9)$$

where $b_1, b_2, b_3, c_1, c_2, c_3$ are system parameters.

DRNN CONTROLLER

As shown in figure 2, The basic control scheme consists of a feedforward DRNN controller and a fixed gain feedback controller. The DRNN is first used as a identifier to emulate the inverse dynamics of the dc servo system, and this networks is called as DRNNI, it is trained off-line and on-line. When DRNNI is trained, it is used as a feedforward controller called as DRNNC. The system control voltage U is composed of the feedforward controller output voltage U_n and the feedback controller U_P. If the DRNNI has learned the inverse model of the system, the DRNNC alone provides all the necessary voltage for the system to track the desired trajectory and output of the feedback controller will tend to zero.

LEARNING OF DRNNI

Several studies have founded that a three-layered neural networks with one hidden layer can approximate any nonlinear function to any desired accuracy[1]. DRNN networks superior to multiplayer feedforward static neural networks to deal with dynamic problems[7]. The structure of three layer DRNN is shown in figure 3. It consists of an input layer, an output layer and one recursive hidden layer. Where $I_i(k), w_j, w_{ij}, S_j$ and $O(k)$ are the ith input to the DRNN, the connecting weight between jth recursive neuron and the output of networks, connecting weight between ith input to network and the jth hidden neuron, the output of jth hidden neuron and the output of the DRNN. The mathematical model of DRNN is shown below:

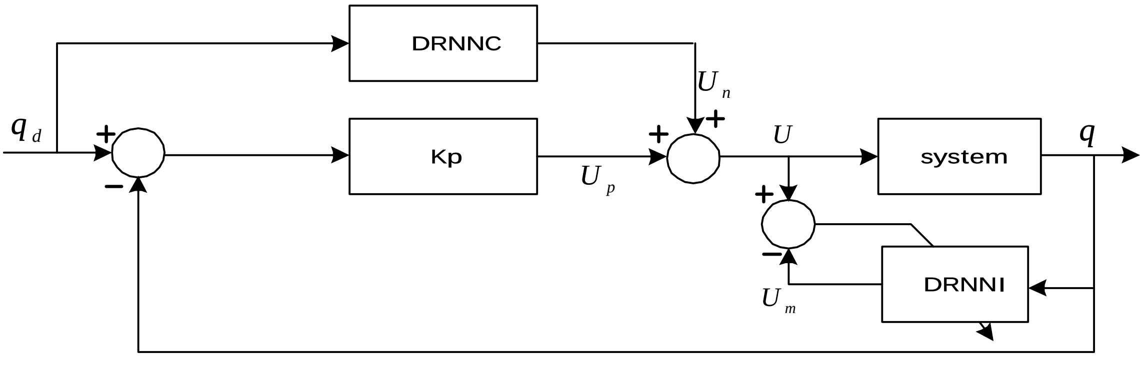


Fig.2 DRNN control for DC system

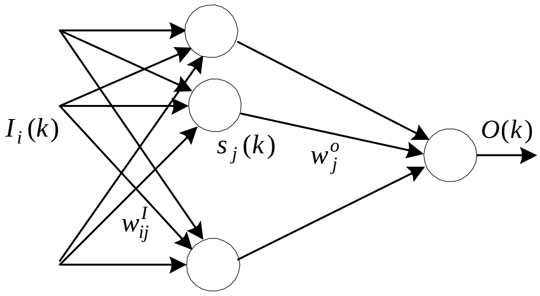


Fig.3 Three layers DRNN

Where $x_j(k)$ is the output of jth recursive neuron, w_j is the recursive weight of jth hidden neuron, $f(\bullet)$ is sigmoid function. When DRNN is used as DRNNI, output of networks $O(k) = U_m(k)$. When DRNN is used as DRNNC, $O(k) = U_n(k)$.

$$s_j(k) = w_j^D x_j(k-1) + \sum_i w_{ij}^I I_i(k) \qquad (10)$$

The cost function to train DRNNI is defined as

$$J = \frac{1}{2}(U - U_m)^2 = \frac{1}{2}e_m^2 \qquad (11)$$

The objective of the learning process is to adjust the network parameters (weights) so as to minimize the cost function J over the entire train set. The back propagation algorithm is given below[7].

$$\begin{aligned} \Delta w(k) &= -h \frac{\partial J}{\partial w} \\ &= h\, e_m(k) \frac{\partial U_m}{\partial w} \\ &= h\, e_m \frac{\partial O(k)}{\partial w} \qquad (12) \end{aligned}$$

Where $w(k)$ is any weight of DRNNI, h is the learning rate of this weight.

Define the output gradients with respect to output, recurrent, and input weight, respectively as below

$$\frac{\partial O(k)}{\partial w_j^o} = x_j(k) \qquad (13)$$

$$\frac{\partial O(k)}{\partial w_j^D} = w_j^o P_j(k) \qquad (14)$$

$$\frac{\partial O(k)}{\partial w_{ij}^I} = w_j^o Q_{ij}(k) \qquad (15)$$

$$P_j(k) = \frac{\partial x_j(k)}{\partial w_j^D} = f'(s_j) x_j(k-1) \qquad (16)$$

$$Q_{ij}(k) = \frac{\partial x_j(k)}{\partial w_{ij}^I(k)} = f'(s_j) I_i(k) \qquad (17)$$

From above equations, learning algorithm of weight w_{ij}, w_j and w_j can be got. The learning rate can be chosen properly[7].

IDENTIFICATION AND CONTROL

For the dc system position tracking, the DRNNI is used to identify the unknown system dynamics (dc motor, amplifier, and the mechanical friction) that mapping the control voltage U to the motor position. Because the DRNNI is used to identify the inverse model of the DC servo system, the inputs to feedforward controller DRNNC is a desired position trajectory and the output of DRNNC is control voltage for system to tack the desired trajectory. From function (9), the relation between control voltage and the motor position can be written as a difference equation below

$U(k) = d_1 q(k-3) + d_2 q(k-2) + d_3 q(k-1)$ (18)

If the aim is to track the desired speed, similarly can get the difference relationship between control voltage and the speed of dc motor as below

$U(k) = e_1 w(k) + e_2 w(k-1) + e_3 w(k-2)$ (19)

Where d_1, d_2, d_3 and e_1, e_2, e_3 are system parameters . Function (17) and (18) can be written in this form

$U(k) = h(q(k-1), q(k-2), q(k-3))$ (20)

$U(k) = g(w(k), w(k-1), w(k-2))$ (21)

The DRNNI is trained to emulate the unknown function $h(\bullet)$ or $g(\bullet)$. For position tracking, the inputs to the DRNNI are $q(k-1), q(k-2)$ and $q(k-3)$ .For speed tracking, the inputs to the DRNNI are $w(k), w(k-1)$ and $w(k-2)$.

When the DRNNI is trained, it is used as a feedforward controller DRNNC. For position tracking, the inputs to DRNNC are desired trajectory $q_d(k-1), q_d(k-2)$ and $q_d(k-3)$. For speed tracking, the inputs to DRNNC are desired speed $w_d(k), w_d(k-1)$ and $w_d(k-2)$. Control voltage U, is the sum of the DRNNC , U_n, and the feedback controller, U_p.

$U = U_n + U_p$ (22)

EXPERIMENTAL RESULTS

The capability of this control method was tested in laboratory by applying different kinds of position trajectories. The control scheme was implemented in PETIUM I 133 using BorlandC3.1 language, and the sample time is defined as 5ms. The results of control are shown in figure 4, figure 5, figure 6 and figure 7. To illustrate the correctness of this method, results of PID control with the same condition are given in figure 8 and figure 9, also. From figure 4, it can be seen that the control voltage U_n of DRNN plays an important role in the whole control voltage U. From figure 5, figure 6 and figure 7, it can be seen that when desired position is very small (low system speed), and with strong mechanical friction, neural networks controlled system can track desired position very closely, whereas traditional PID controlled system tracking results are much bad, shown in figure 8 and figure 9.

CONCLUSION

This paper presents a real-time control of a low speed dc servo system. It is shown that, DRNN is efficient for system identification and control, and this system through this proposed method, can tack any selected trajectories with high performance under strong mechanical friction and other nonlinear factors. This control scheme is implemented with PETIUM I programming environment using BorlandC3.1 language. Also, it can be seen that with the higher system frequency of programming environment, the better control results will be got with this method proposed in this paper.

REEFERENCES

[1] Hornik,K.,Stinchcombe, M.,and White,H. Multilayer feedforward networks are universal approximators. Neural networks, 1989. (2):359-366.

[2] Psaltis, D., Siderris, A., and Yamamura, A.A. A multiplayered neural network controller. IEEE Control Syst. Mag., April 1988. 17-21.

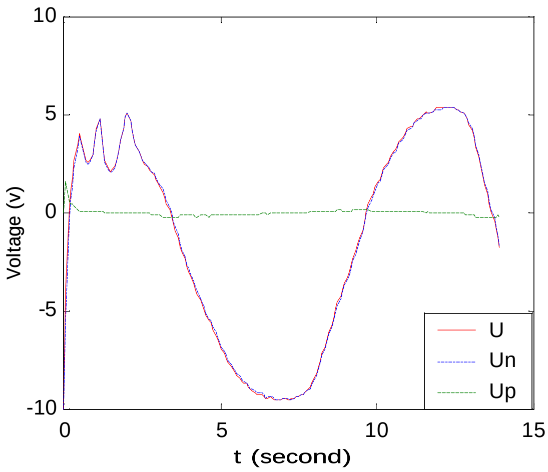
[3] K.S.Narenda and K.Parthasaathy. Identification and control of dynamical systems using neural networks. IEEE Trans. on neural networks, 1990. 1(1):4-27.

[4] Siri Weerasooriya and M.A.EI-Sharkawi, Identification and control of a DC motor using backpropgation neural networks. Proceedings of IEEE/PES winter meeting 1991.1-7.

[5] Jui-Hong Horng. Neural adaptive tracking control of a DC motor. Information sciences. 1999. (118):1-13.

[6] L.Jin, P.N.Nikiforruk, and M.M.Gupta. Direct adaptive output tracking control using multilayered neural networks, IEE Proceedings-D, 1993. 140(6): 393-398.

[7] Chao-Chee Ku and Kwang Y. Lee, Diagonal recurrent neural networks for dynamic system control, IEEE Trans. on Neural networks. 1995. 6(1): 144-156



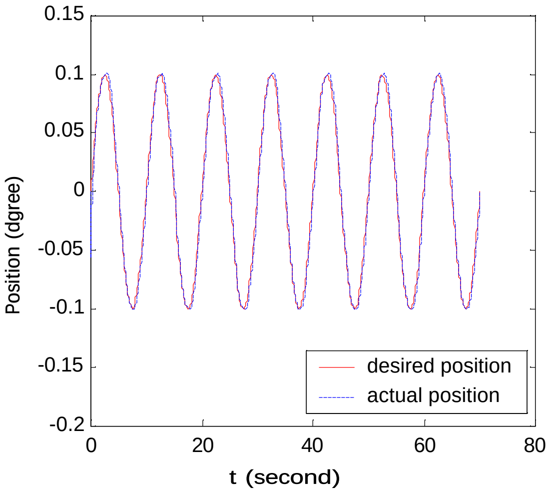


Fig.5 System position

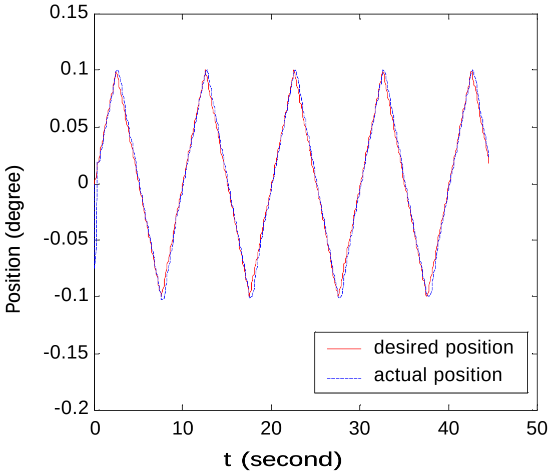


Fig.6 Sysytem position

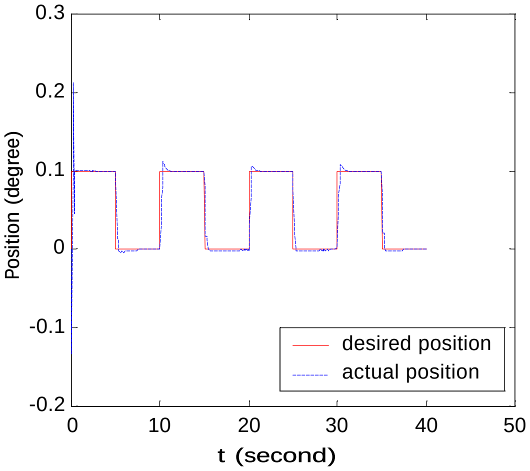


Fig.7 System position

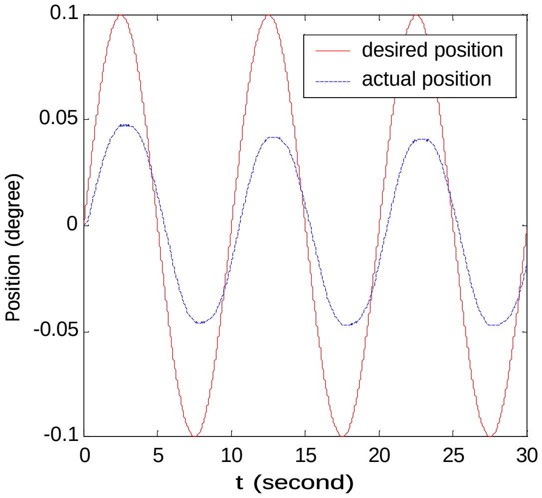


Fig.8 PID control

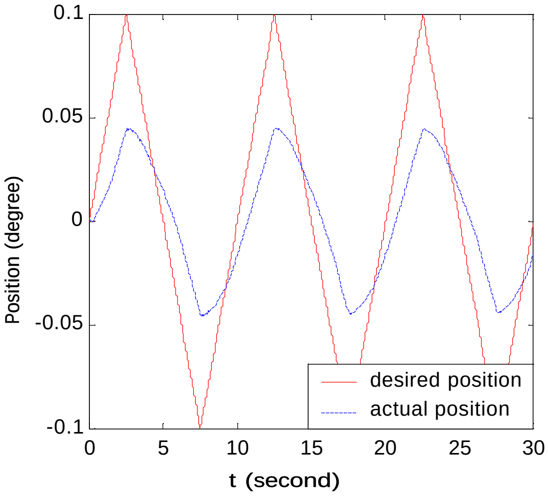


Fig.9 PID control