

# The Artificial Neural Networks applied to servo control systems

Yuan Kang<sup>1</sup>, Yi-Wei Chen<sup>2</sup>, Ming-Huei Chu<sup>2</sup>, Der-Ming Chry<sup>2</sup>

*1Department of Mechanical Engineering, Chung Yuan Christian University*

*2Department of Mechatronical Engineering, TungNan University*

**Abstract:** This chapter utilizes the direct neural control (DNC) based on back propagation neural networks (BPN) with specialized learning architecture applied to the speed control of DC servo motor. The proposed neural controller can be treated as a speed regulator to keep the motor in constant speed, and be applied to DC servo motor speed control. The proposed neural control applied to position control for hydraulic servo system is also studied for some modern robotic applications.

A tangent hyperbolic function is used as the activation function, and the back propagation error is approximated by a linear combination of error and error's differential. The simulation and experiment results reveal that the proposed neural controller is available to DC servo control system and hydraulic servo system with high convergent speed, and enhances the adaptability of the control system.

**Keywords:** Neural networks, DC servo motor, Speed regulator, Speed control, Hydraulic servo System

## 1. Introduction

The neural controls have been put into use in various fields owing to their capability of on line learning and adaptability. In recent years, many learning strategies for neural control have been proposed and applied to some specified nonlinear control systems to overcome the unknown model and parameters variation problems. In this chapter, a direct neural controller with specialized learning architecture is introduced and applied to the DC servo and hydraulic servo control systems.

The general learning architecture and the specialized learning architecture are proposed and studied in early development of neural control [1]. The general learning architecture shown in Fig. 1, uses neural network to learn the inverse dynamic of plant, and the well-trained network is applied to be a feed forward controller. In this case, the general procedure may not be efficient since the network may have to learn the responses of the plant over a larger operational range than is actually necessary. One possible solution to this problem is to combine the general method with the specialized procedure, so that an indirect control strategy for general learning was proposed, which is shown in Fig. 2. For a general learning architecture with some specialized procedures, the off line learning of the

inverse dynamic of plant still have to learn the responses of the plant over a larger operational range.

The specialized learning architecture shown in Fig. 3, trains the neural controller to operate properly in regions of specialization only. Training involves using the desired response as input to the network. The network is trained to find the plant input, which drives the system output to the desired command. This is accomplished by using the error between the desired and actual responses of the plant to adjust the weights of the network with a steepest descent procedure. The weights are adjusted to decrease the errors during iterations. This procedure requires knowledge of the Jacobian of the plant.

There are two strategies to facilitate the specialized learning, one is direct control shown in Fig. 4 and the other is indirect control shown in Fig. 5 [2]. In the former, the plant can be viewed as an additional but no modifiable layer of the neural network, and the dash line of Fig. 4 means the weights update need the knowledge of plant. The latter, which has been used in many applications [3-5], is a two-step process including identification of dynamics of plant and control.

In the indirect control strategy, a sub-network (called "emulator") is required to be trained before the control phase, and the quality of the trained emulator is crucial to the controlling performance. It is therefore very important that the data sets for training the emulator must cover a sufficiently large range of input and output data pairs, but if some of these values are outside the input range that was used during the emulator's training, the back propagation through the emulator fails, causing poor or even unstable control performance.

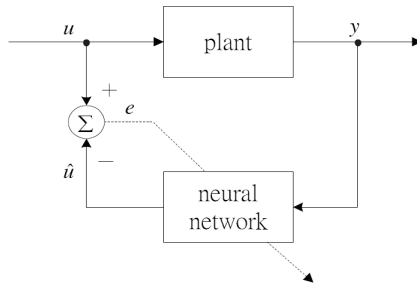


Fig. 1. The general learning architecture

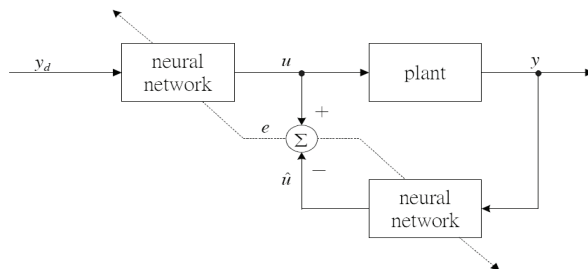


Fig. 2. The indirect control for general learning architecture

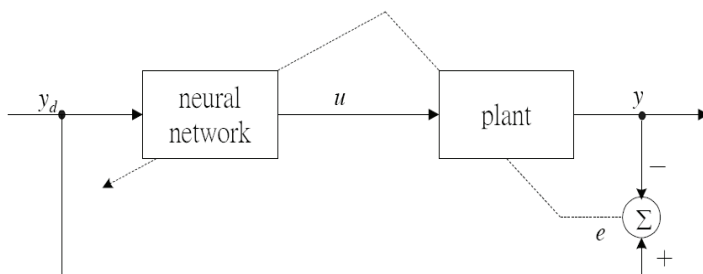


Fig. 3. The specialized learning architecture

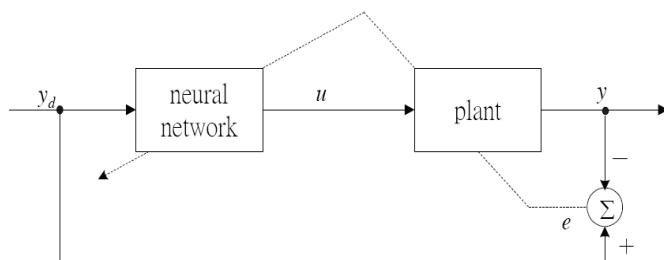


Fig. 4. The direct control for specialized learning architecture

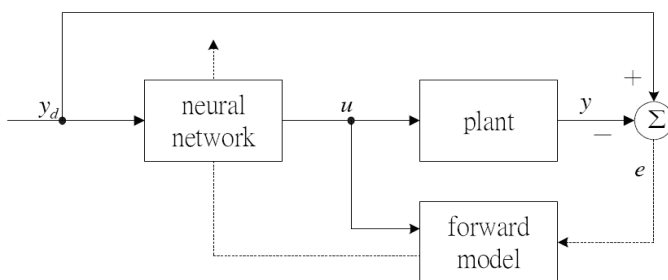


Fig. 5. The indirect control for specialized learning architecture

The direct control strategy can avoid this problem if a priori qualitative knowledge or Jacobian (the partial differential of plant output to input) of the plant is available. But it is usually difficult to approximate the Jacobian of an unknown plant. This chapter utilizes the direct neural control (DNC) based on back propagation neural networks (BPN) with specialized learning architecture applied to the speed controls of DC servo motor. The approximation methods of Jacobian are introduced for direct neural control scheme. The direct control strategies with the approximation methods have been successfully applied to DC servo and hydraulic servo control systems. The proposed neural controller also can be

treated as a speed regulator to keep the motor in constant speed. The corresponding performances are investigated and discussed.

## 2. The direct neural controller

### 2.1 The structure of direct neural control

A direct neural controller with three layers was shown in Fig. 6. A three layers neural network with one hidden layer is sufficient to compute arbitrary decision boundaries for the outputs [6]. Although a network with two hidden layers may give better approximation for some specific problems, but the networks with two hidden layers are more prone to fall into local minima [7], and more CPU time is needed. In the following section, a back propagation network (BPN) with single hidden layer is considered.

Another consideration is the right number of units in a hidden layer. Lippmann [8] has provided comprehensive geometrical arguments and reasoning to justify why the maximum number of units in a single hidden layer should equal to  $M(N+1)$ , where  $M$  is the number of output units and  $N$  is the number of input units. Zhang et al. [2] have tested different numbers units of the single hidden layer for a ship tracking control system. It was found that a network with three to five hidden units is often enough to give good results.

The structure of direct neural control is shown in Fig. 7. The proposed neural network has three layers with two units in the input layer, one unit in the output layer and fine number of units in the hidden layer. The  $r = X$ ,  $X$  and  $y = X$  denote as the command input, output of the reference model and the output of the plant respectively. The two inputs of the network are the error  $e$  and its differential  $\dot{e}$  between  $X_R$  and  $X_P$ .

The reference model can be designed according to a second order dynamic model; the damping ratio and natural frequency can be determined based on the specified performance index of control system.

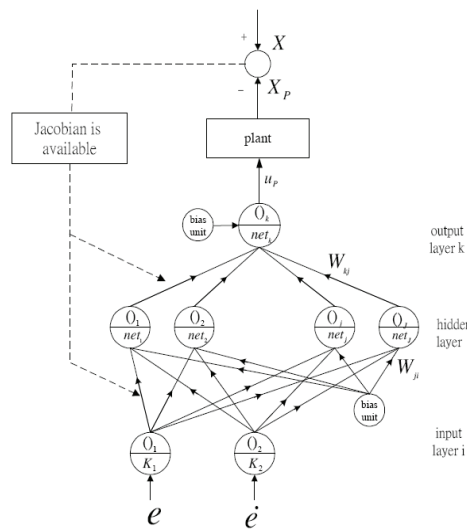


Fig. 6. A direct neural controller with three layers

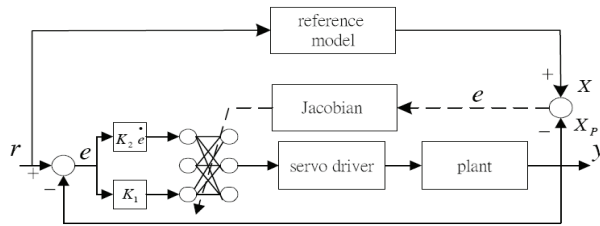


Fig. 7. The structure of a direct neural control system

**2.2 The algorithms for direct neural controller**

The proposed neural network shown in Fig. 6 has three layers with two units in the input layer, one unit in the output layer and right number of units in the hidden layer. The  $X_r$ ,  $X$  and  $X_p$  denote the required command input, output of the reference model and the output of the controlled plant respectively. The two inputs of the network are the error and its differential between  $X_r$  and  $X_p$ . The reference model can be designed according to a second order transfer function; the damping ratio and natural frequency can be defined based on the specified performance index. The algorithms and weights update equations of the proposed direct neural controller are described by the following equations. The proposed direct neural controller has the hidden layer (subscript "j"), output layer (subscript "k") and layer (subscript "i"). The input signal is multiplied by gains  $K_1$ ,  $K_2$  at the input layer, in order to be normalized between +1 and -1. A tangent hyperbolic function is used as the activation function of the nodes in the hidden and output layers, the number of units in hidden layer equals to  $J$ , the number of units in input layer equals to  $I$ , and the number of units in output layer equals to  $K$ , the net input to node j in the hidden layer is:

$$net_j = \sum (W_{ji} \cdot O_i) + \theta_j \quad i=1,2,\dots,I, j=1,2,\dots,J \tag{1}$$

the output of node j is

$$O_j = f(net_j) = \tanh(\beta \cdot net_j) \tag{2}$$

where  $\beta > 0$ , the net input to node k in the output layer is

$$net_k = \sum (W_{kj} \cdot O_j) + \theta_k \quad j=1,2,\dots,J, k=1,2,\dots,K \tag{3}$$

the output of node k is

$$O_k = f(net_k) = \tanh(\beta \cdot net_k) \tag{4}$$

The output  $O_k$  of node  $k$  in the output layer is treated as the control input  $u_p$  of the system for a single-input and single-output system. As expressed equations,  $W_{ji}$  represent the connective weights between the input and hidden layers and  $W_{kj}$  represent the connective weights between the hidden and output layers.  $\theta_j$  and  $\theta_k$  denote the bias of the hidden and output layers, respectively.

For the  $N$ th sampling time, the error function is defined as

$$E_N = \frac{1}{2}(X_N - X_{PN})^2 = \frac{1}{2}e_N^2 \quad (5)$$

where  $X_N$  and  $X_{PN}$  denote the outputs of the reference model and the outputs of the controlled plant at the  $N$ th sampling time, respectively. The weights matrix is then updated during the time interval from  $N$  to  $N+1$ .

$$\Delta W_N = W_{N+1} - W_N = -\eta \frac{\partial E_N}{\partial W_N} + \alpha \cdot \Delta W_{N-1} \quad (6)$$

where  $\eta$  is denoted as learning rate and  $\alpha$  is the momentum parameter. The gradient of  $E_N$  with respect to the weights  $W_{kj}$  is determined by

$$\frac{\partial E_N}{\partial W_{kj}} = \frac{\partial E_N}{\partial net_k} \frac{\partial net_k}{\partial W_{kj}} = \delta_k O_j \quad (7)$$

and  $\delta_k$  is defined as

$$\begin{aligned} \delta_k &= \frac{\partial E_N}{\partial net_k} = \sum_n \frac{\partial E_N}{\partial X_p} \frac{\partial X_p}{\partial u_p} \frac{\partial u_p}{\partial O_n} \frac{\partial O_n}{\partial net_k} = \sum_n \frac{\partial E_N}{\partial O_n} \frac{\partial O_n}{\partial net_k} \\ &= \sum_n \frac{\partial E_N}{\partial O_n} \beta(1 - O_k^2) \quad n = 1, 2, \dots, K \end{aligned} \quad (8)$$

where  $\partial X_p / \partial u_p$  is defined to be the Jacobian of plant. Assume the Jacobian of the plant can be evaluated. The  $\delta_k$  can be solved.

Similarly, the gradient of  $E_N$  with respect to the weights,  $W_{ji}$  is determined by

$$\frac{\partial E_N}{\partial W_{ji}} = \frac{\partial E_N}{\partial net_j} \frac{\partial net_j}{\partial W_{ji}} = \delta_j O_i \quad (9)$$

$$\begin{aligned} \delta_j &= \frac{\partial E_N}{\partial net_j} = \sum_m \frac{\partial E_N}{\partial net_k} \frac{\partial net_k}{\partial O_m} \frac{\partial O_m}{\partial net_j} \\ &= \sum_m \delta_k W_{km} \beta(1 - O_j^2) \quad m = 1, 2, \dots, J \end{aligned} \tag{10}$$

The weight-change equations on the output layer and the hidden layer are

$$\Delta W_{kj,N} = -\eta \frac{\partial E_N}{\partial W_{kj,N}} + \alpha \cdot \Delta W_{kj,N-1} = -\eta \delta_k O_j + \alpha \cdot \Delta W_{kj,N-1} \tag{11}$$

$$\Delta W_{j,N} = -\eta \frac{\partial E_N}{\partial W_{j,N}} + \alpha \cdot \Delta W_{j,N-1} = -\eta \delta_j O_i + \alpha \cdot \Delta W_{j,N-1} \tag{12}$$

where  $\delta_j$  and  $\delta_k$  can be evaluated from Eqs.(24) and (21). The connective weights in the neural network are updated during the time interval from N to N+1 .

$$W_{kj,N+1} = W_{kj,N} + \Delta W_{kj,N} \tag{13}$$

$$W_{j,N+1} = W_{j,N} + \Delta W_{j,N} \tag{14}$$

A tangent hyperbolic function is used as the activation function, so that the neural network controller output  $O_k = u_p$  evaluated from Eq. (4) is between A1 and +1, which is multiplied by the scaling factor  $K_o$  to be the input of plant. The weights and biases is initialized randomly in the interval between +0.5 and A0.5, and updated by Eqs. (13) and (14).

### 2.3 The on line trained adaptive neural controller

The Jacobian of plant needs to be available to Eq.(8) for back propagation algorithm. However, the exact  $\partial X_p / \partial u_p$  is difficult to determine because of the unknown plant dynamics. Two differential approximations are presented [1] by slightly changing each input to the plant at an operating point, and measuring the changes in the output. The jacobian is denoted by

$$\frac{\partial X_p}{\partial u_p} = \frac{X_p(u_p + \Delta u_p) - X_p(u_p)}{\Delta u_p} \tag{15}$$

Alternatively, by comparing the changes of the differential related variables with values in the previous iteration, the differential can be approximated using the relationship

$$\frac{\partial X_p(N)}{\partial u_p(N)} = \frac{X_p(N) - X_p(N-1)}{u_p(N) - u_p(N-1)} \quad (16)$$

It has been observed in earlier reported simulations [2] that the use of approximation (15) or (16) often causes ambiguity for network training when the controlled plant has large inertia or when disturbances are added. Ambiguity in training contrary to what would be expected from a clear understanding of the situation being investigated. A simple sign function proposed by Zhang et al. [2] is applied to approximate the Jacobian of plant, and called on-line trained adaptive neural controller for industrial tracking control application. Therefore, the differential  $\partial X_p(N)/\partial u_p$  is approximated by the ratio of the signs of the changes in  $X(N)_p$  and  $u(N)_p$ . The term  $\partial X_p(N)/\partial u_p(N)$  is replaced by its sign, so that Eq.8 takes the form

$$\delta_k = \frac{\partial E_N}{\partial net_k} = \sum_n \frac{\partial E_N}{\partial X_p} \text{sign}\left(\frac{\partial X_p}{\partial u_p}\right) \frac{\partial u_p}{\partial O_n} \frac{\partial O_n}{\partial net_k} \quad (17)$$

The clear knowledge of how the control signal  $u(N)_p$  influence the plant outputs  $X(N)_p$  will provide the required sign information. Therefore  $X(N)_p / u(N)_p < 0$  leads to

$$\text{sign}(\partial X_p(N)/\partial u_p(N)) = -1 \quad (18)$$

and  $\partial X_p(N)/\partial u_p(N) > 0$  leads to

$$\text{sign}(\partial X_p(N)/\partial u_p(N)) = 1 \quad (19)$$

Using Eq.(17) with the given differential signs provide in Eq.(18) and (19), the neural controller will effectively output control signals with the correct direction according to the plant output error  $e(N)$ .

#### 2.4 The approximation of Jacobian

An accurate tracking response needs to increase the speed of convergence. However, for a single-input and single-output control system, the sensitivity of  $E_N$  with respect to the network output  $O_k$  can be approximated by a linear combination of the error and its differential according to the & adaptation law [8] shown as below

$$\frac{\partial E_N}{\partial O_k} = K_3 e + K_4 \frac{de}{dt} \quad (20)$$



where  $K_3$  and  $K_4$  are positive constants, so that Eq.8 takes the form

$$\delta_k = \frac{\partial E_N}{\partial net_k} = \sum_n \frac{\partial E_N}{\partial X_p} \frac{\partial X_p}{\partial u_p} \frac{\partial u_p}{\partial O_n} \frac{\partial O_n}{\partial net_k} = \sum_n (K_3 e + K_4 \frac{de}{dt}) \frac{\partial O_n}{\partial net_k} \tag{21}$$

**Example 1.**

A direct neural controller applied to DC servo speed control system is shown in Fig. 8. Assume the voltage gain of servo amplifier is unity. The gain of speed sensor is 0.001V rpm , the first order dynamic model of DC servo motor is

$$M(S) = \frac{10000}{S + 1}$$

According to Eq. 23, the direct neural controller using & adaptation law with three layers and five hidden neurons shown in Fig. 9. is used to control and regulate the motor speed.

$$\frac{\partial E_N}{\partial O_k} = K_3 e + K_4 \frac{de}{dt} \tag{23}$$

According to Fig. 8, the neural control system without reference model is a self-tuning type adaptive control, so that  $K_1=K_3$  and  $K_2=K_4$  conditions can be applied. The  $K_1=0.6$  and  $K_2=0.05$  can be determined for input signals normalization. The learning rate  $\eta = 0.1$  , sampling time=0.0001s,  $K_1=K_3=0.6$ ,  $K_2=K_4=0.05$  and the step command of 1V(1000rpm) assigned for simulation phase, and the simulation results are shown in Fig. 10, Fig. 11, and Fig. 12. The simulation results show that the connective weights will be convergent. The time response for  $v_u$  shows that the network will keep an appropriate output voltage signal to overcome the speed (motional) voltage, which is generated by the rotating armature. Similarly, the neural controller can provide output to overcome the torque load and friction. This is similar to a PI controller, but the neural controller can enhance the adaptability and improve performance of control system.

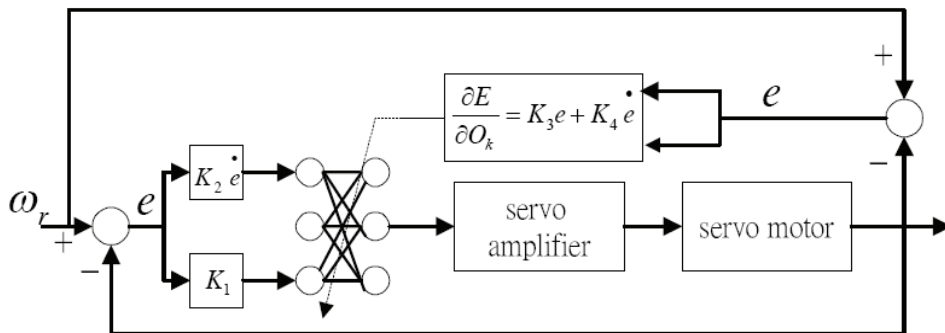


Fig. 8. The speed control system with direct neural controller

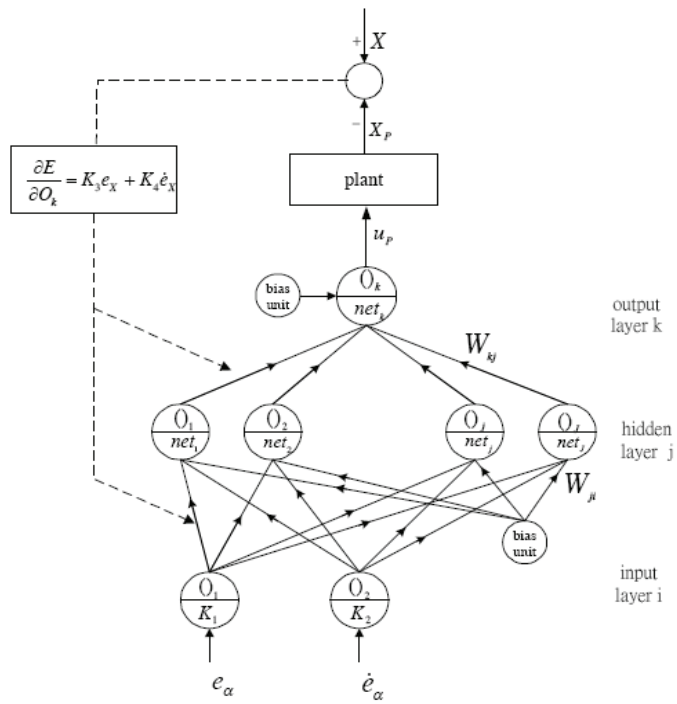


Fig. 9. The direct neural controller

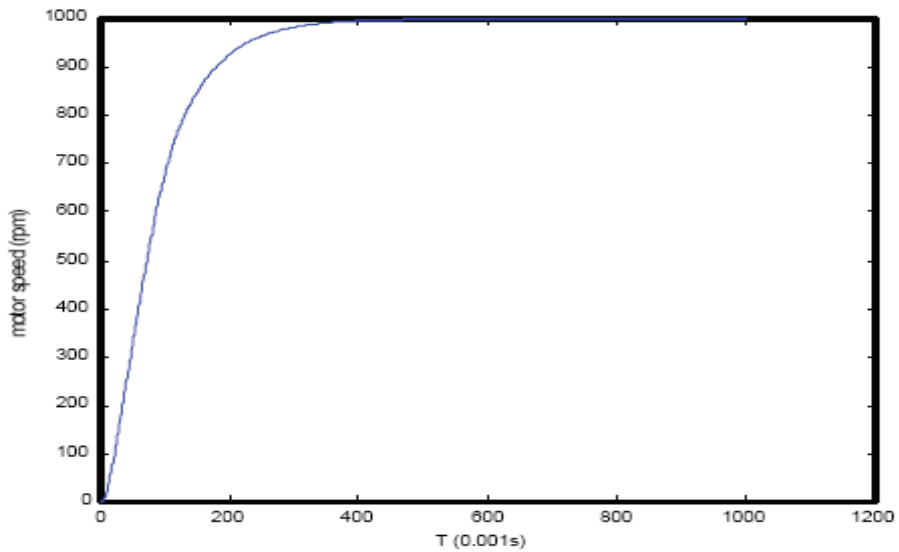


Fig. 10. Speed response for DC servo motor

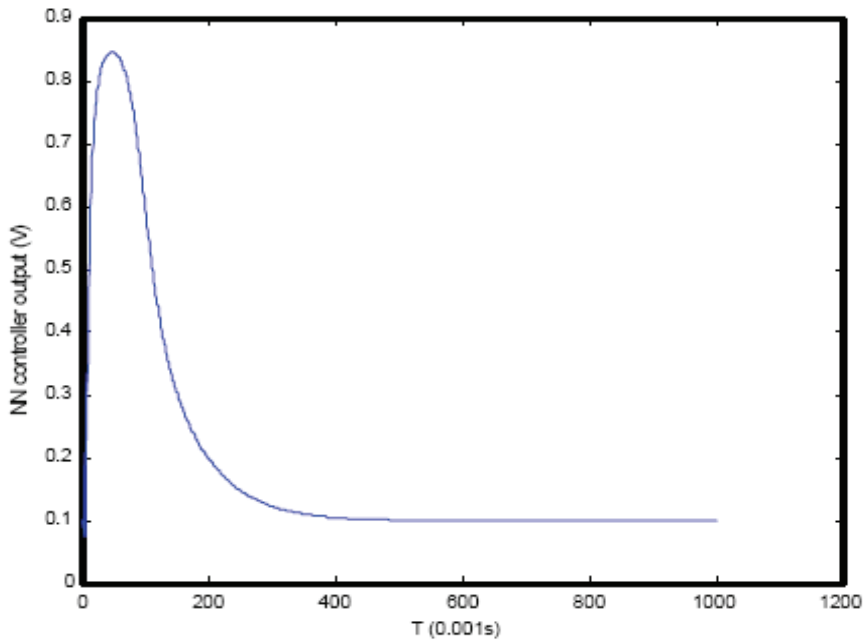


Fig. 11. The time response for control input

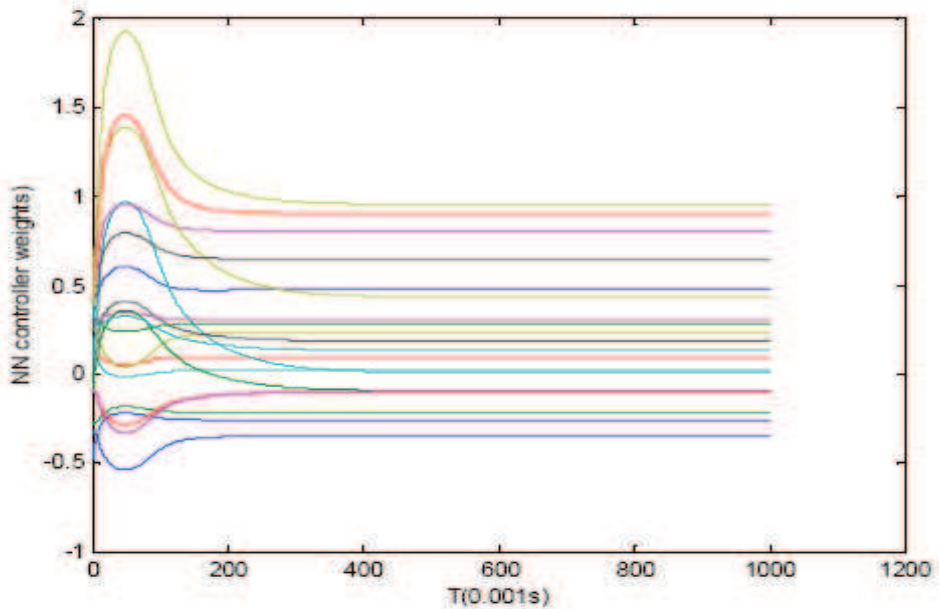


Fig. 12. All connective weights are convergent before 0.4s

The on-line trained neural controller using sign function approximation of Jacobian is also applied to this speed control system. The simulation results shown in Fig. 13, Fig. 14, and Fig. 15, which reveal that the on-line trained method takes more time for convergence.

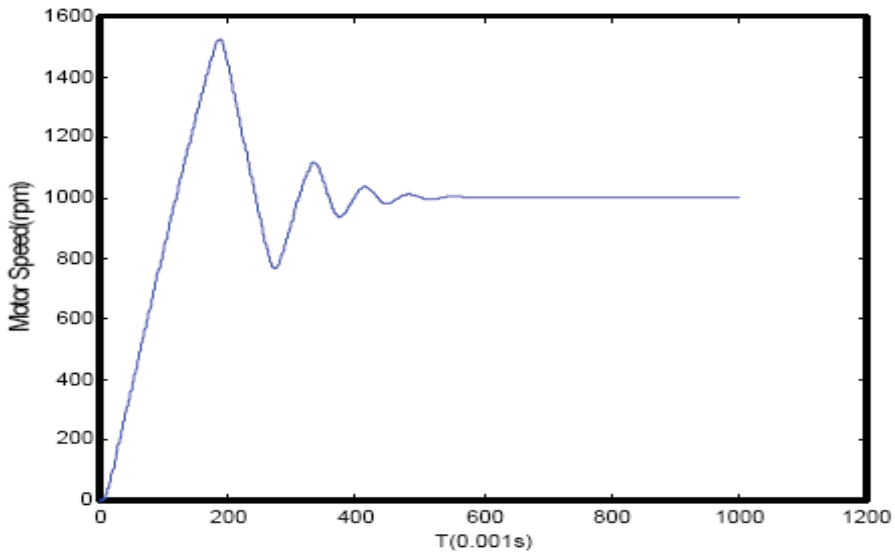


Fig. 13. Speed response for DC servo motor

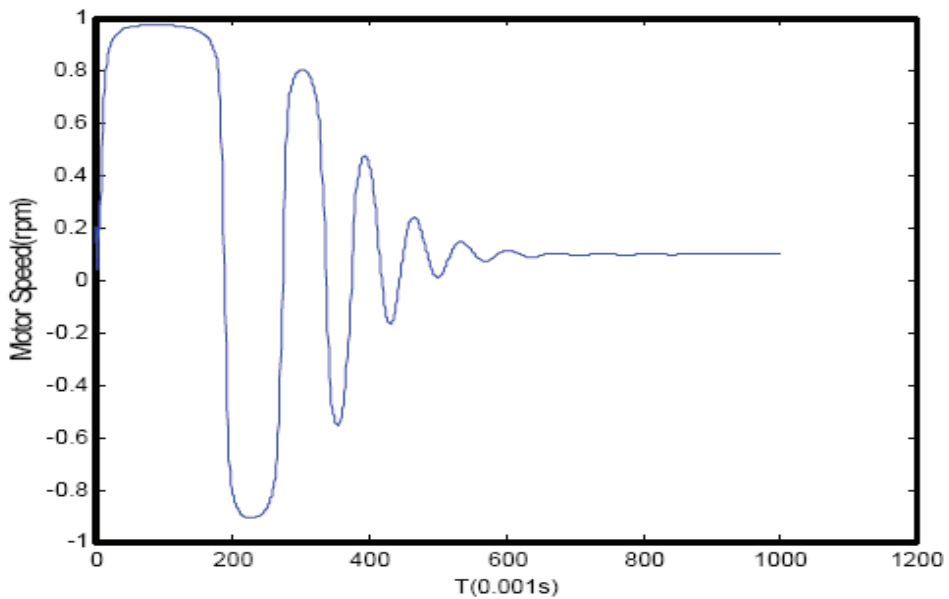


Fig. 14. The time response for control input

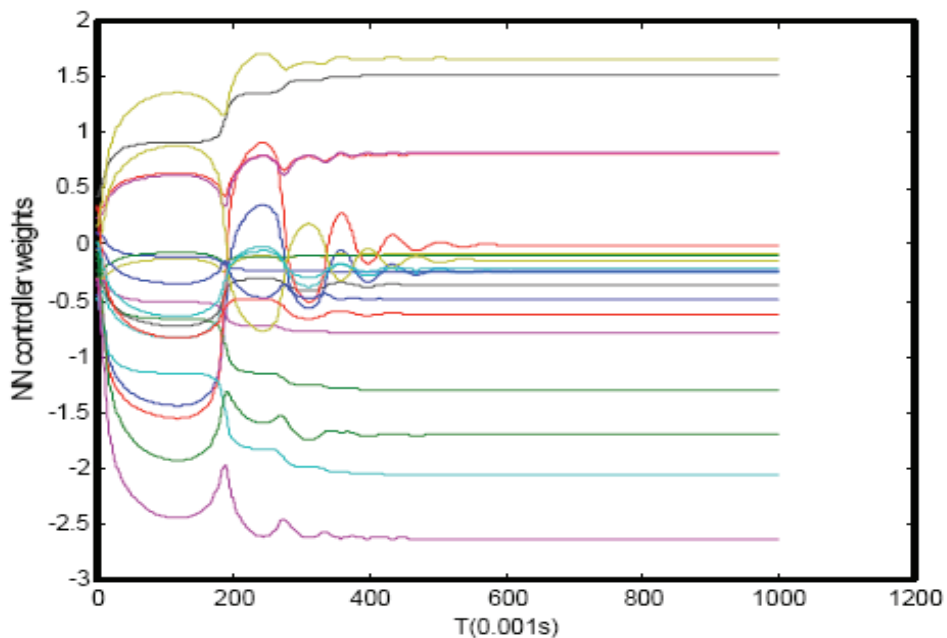


Fig. 15. All connective weights are convergent before 0.6s

The direct neural controllers using & adaptation law can provide better performance than using on-line trained method. The & adaptation law uses the error differential to increase the damping of the error convergent process, and improve the stability and convergent speed for weight update algorithm.

### 3. THE direct neural control applied to speed regulation for DC servo motor

The modern precise DC servo systems need to overcome the unknown nonlinear friction, parameters variations and torque load variations. It is reasonable to apply adaptive control to the DC servo system for speed control. But the conventional adaptive control techniques are usually based on the system model parameters. The unavailability of the accurate model parameters leads to a cumbersome design approach. The real-time implementation is difficult and sometimes not feasible because of using a large number of system parameters in these adaptive schemes. The proposed direct neural controllers can precisely regulate the speed for a DC servo motor, but don't have to the knowledge of system model parameters.

#### 3.1 Description of the speed regulation system

The application of the direct neural controller for DC servo motor speed regulation is shown in Fig. 14, where  $v_r$  is the speed command and  $v$  is the actual output speed. The proposed neural network is treated as a speed regulator, which can keep the motor in constant speed against the disturbance.

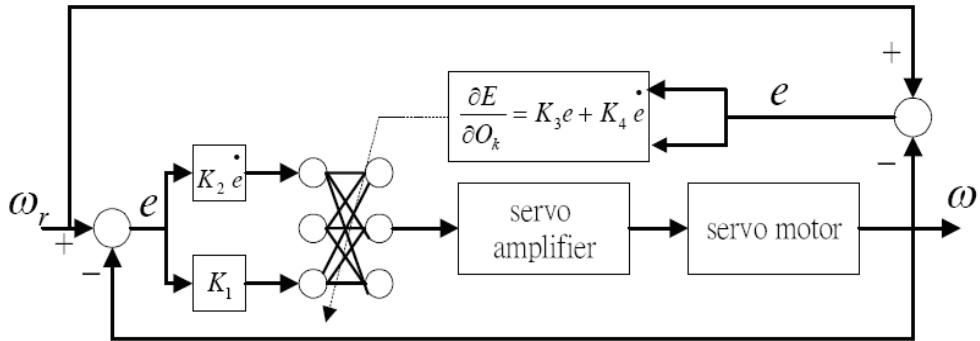


Fig. 14. The block diagram of speed control system

There are 5 hidden neurons in the proposed neural regulator. The proposed DNC is shown in Fig. 15 with a three layers neural network.

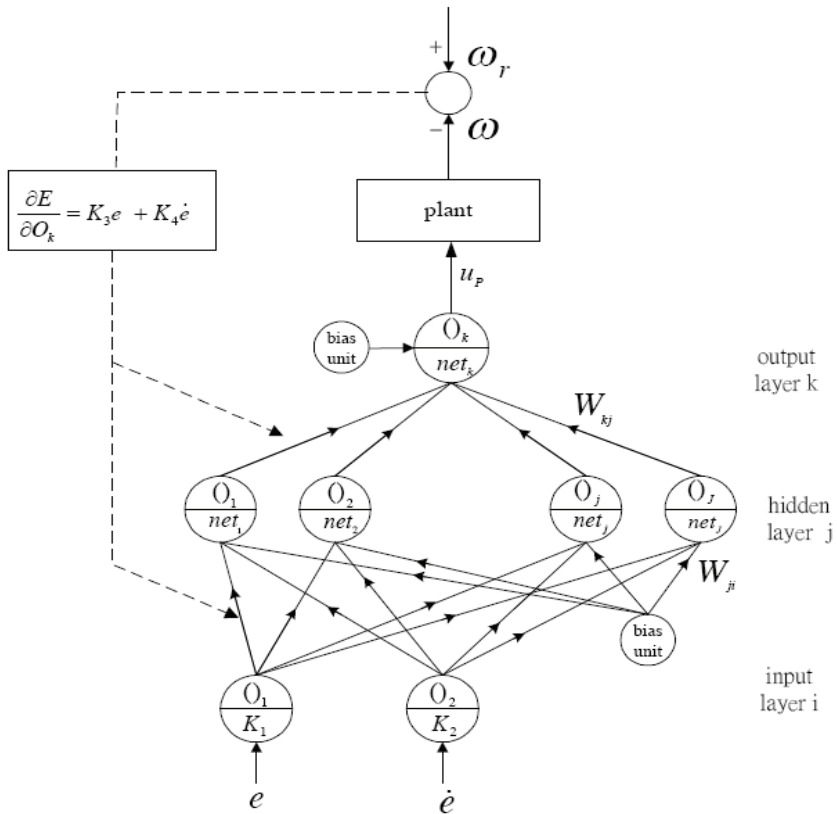


Fig. 15. The structure of proposed neural controller

The difference between command speed  $v_r$  and the actual output speed  $v$  is defined as error  $e$ . The error  $e$  and its differential  $\dot{e}$  are normalized between -1 and +1 in the input neurons before feeding to the hidden layer. In this study, the back propagation error term is approximated by the linear combination of error and error's differential. A tangent hyperbolic function is designed as the activation function of the nodes in the output and hidden layers. So that the net output in the output layer is bounded between -1 and +1, and converted into a bipolar analogous voltage signal through a D/A converter, then amplified by a servo-amplifier for enough current to drive the DC motor. A step speed command is assigned to be the reference command input in order to simulate the step speed response of a DC servo motor. The proposed three layers neural network, including the hidden layer (  $j$  ), output layer (  $k$  ) and input layer (  $i$  ) as illustrated in Fig. 15. The input signals  $e$  and its differential  $\dot{e}$  are multiplied by the coefficients  $K_1$  and  $K_2$ , respectively, as the normalized signals  $O_i$  to hidden neuron. A tangent hyperbolic function is used as the activation function of the nodes in the hidden and output layers. The algorithms for weight update are described in previous section.

### 3.2 Dynamic Simulations

The block diagram of the DC servo motor speed control system with the proposed neural regulator is shown in Fig. 14, which consists of a 15W DC servo motor, an tachometer with a unit of 1/150.8 V/rad/s, an 12 bits bipolar D/A converter with an output voltage range of -5V to +5V and a servo amplifier with voltage gain of 2.3. The parameters of DC servo motor are listed in Table 1.

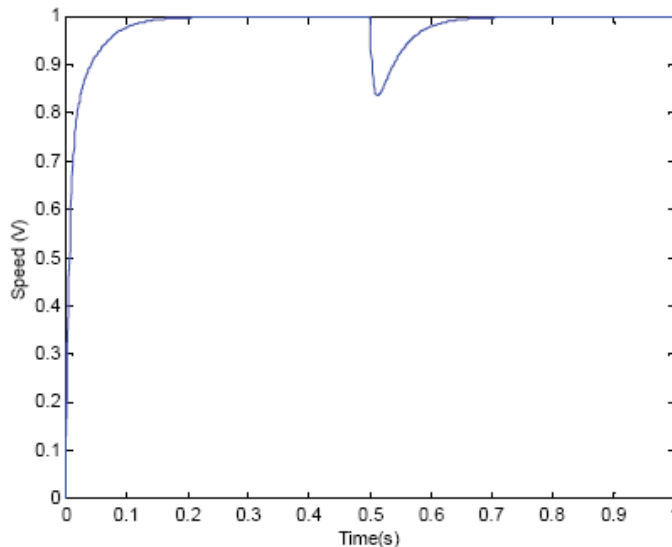
Motor resistance $R_a$	3.18 $\Omega$
Motor inductance $L_a$	0.53mH
Inertia of rotor $J$	24.3 $\times 10^{-7}$ kgm <sup>2</sup>
Torque constant $K_T$	23mNm/A
Back emf $K_B$	0.00241 V/rpm

Table 1. The parameters of motor

In the designed direct neural controller, the number of neurons is set to be 2, 5 and 1 for the input, hidden and output layers, respectively (see Fig.2). There is only one neuron in the output layer. The output signal of the direct neural controller will be between -1 and +1, which is converted into a bipolar analogous voltage signal by the D/A converter. The output of the D/A converter is between +5V and -5V corresponding to the output signal between +1 and -1 of the neural controller. It means the output of neural controller multiplied by a conversion gain of 5V. Then, the voltage signal is amplified by the servo amplifier to provide high current for driving the DC servo motor. The parameters  $K_1$  and  $K_2$  must be adjusted in order to normalize the input signals for the neural controller.

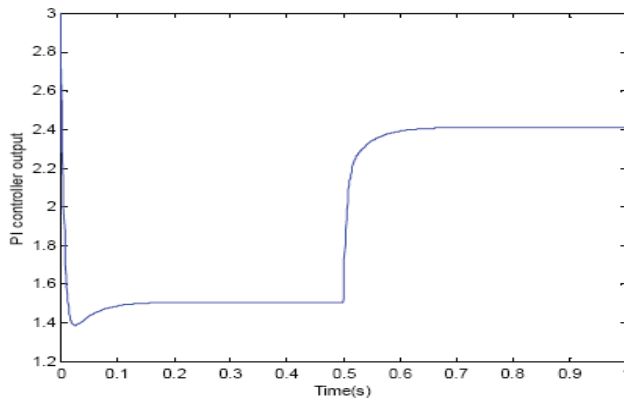
In this simulation, the parameters  $K_3$  and  $K_4$  can be determined, and  $K_3 = K_1$  and  $K_4 = K_2$  are assigned. In this simulations, a step signal of 1V corresponding to 150.8 rad/s is denoted as

the speed command, the sampling time is set to be 0.0001s, the learning rate  $\eta$  of the neural network is set to be 0.1 and the coefficient  $\beta=0.5$  is assigned. Since the maximum error-voltage signal is 1V, the parameters  $K_1$  and  $K_2$  are assigned to be 0.6 and 0.01, respectively, in order to obtain an appropriate normalized input signals to the neural network. The parameters  $K_3 = K_1 = 0.6$  and  $K_4 = K_2 = 0.01$  are assigned for better convergent speed of the neural network. And a conventional PI controller with well tuning parameters applied to this speed regulate system is also simulated. Assumes a disturbance torque load of 0.015 Nm applies to this control system at  $t=0.5s$ . The simulation results are shown in Fig. 16 and Fig. 17 4, where Fig. 16 (a) represents the speed response of the DC motor with PI controller, Fig. 16 (b) represents the output signal of the PI controller; Fig. 17 (a) represents the speed response of the DC motor with neural controller. Fig. 17 (b) represents the output signal of the neural controller. It exhibits a steady state error in the speed response is eliminated by the proposed neural regulator, which keeps appropriate voltage output as the inputs near 0. Fig. 17 (c) shows the convergent time of the connective weights is smaller than 100ms, and the speed response of the DC motor is stable. Consequently, the proposed neural speed regulator enhances the adaptability in speed control system. In addition, an extra attention should be taken on the disturbing torque load. The conventional PI controller does not have fast performance of speed regulation as the proposed neural speed regulator. The output of PI controller will saturate, if its performances are increased to near the neural regulator. Fig.3 (b) shows the maximum output of PI controller is close to 3V. Fig.4 (b) shows the maximum output of neural regulator is only 2.41V, and the speed regulation performance of neural regulator is better than that of the PI controller. The simulation results exhibit the neural regulator is available for the high-precision speed control systems. If the speed command is increased and over 1.66V, the parameters  $K_1$  and  $K_2$  need to be adjusted, the parameters  $K_3$  and  $K_4$  also need to be adjusted to appropriate values. Basically increasing the values of  $K_3$  and  $K_4$  will increase the damping effect of control system.



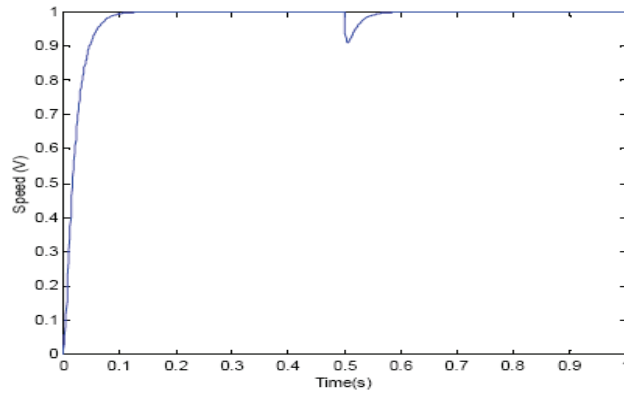
(a) Speed response



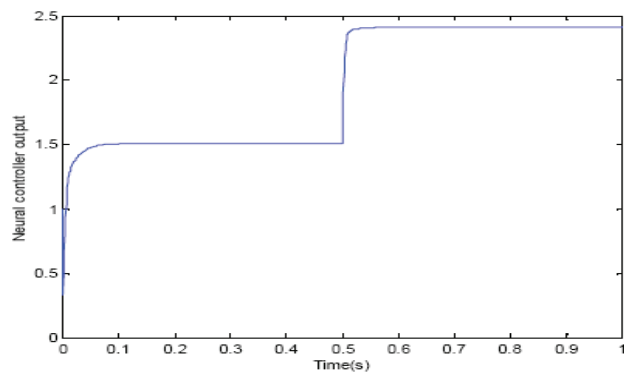


(b) Output of PI controller

Fig. 16. Simulation results for speed regulation of DC servo motor with PI controller



(a) Speed response



(b) Output of neural regulator

## Thank You for previewing this eBook

You can read the full version of this eBook in different formats:

- HTML (Free /Available to everyone)
- PDF / TXT (Available to V.I.P. members. Free Standard members can access up to 5 PDF/TXT eBooks per month each month)
- Epub & Mobipocket (Exclusive to V.I.P. members)

To download this full book, simply select the format you desire below

