18

The Artificial Neural Networks applied to servo control systems

Yuan Kang1 , Yi-Wei Chen2, Ming-Huei Chu2, Der-Ming Chry2
1*Department of Mechanical Engineering, Chung Yuan Christian University*
2*Department 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 dana 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 trough the emulator fails, causing poor or even unstable control performance.

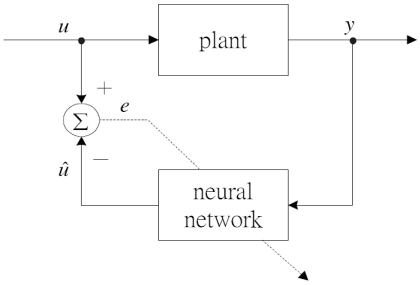


Fig. 1. The general learning architecture

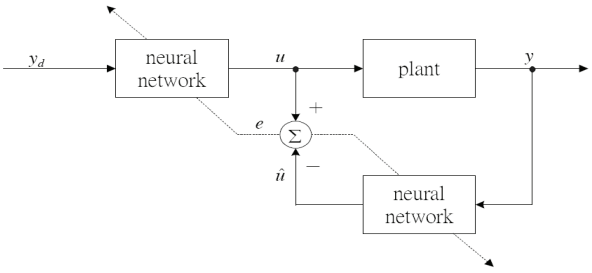


Fig. 2. The indirect control for general learning architecture

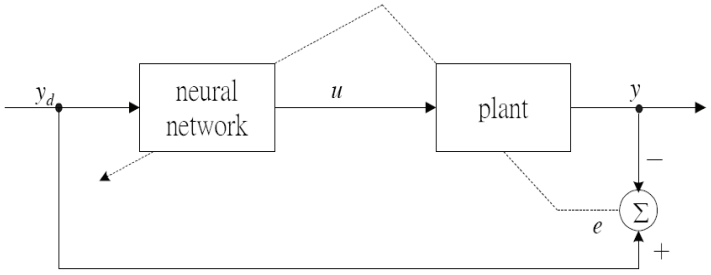


Fig. 3. The specialized learning architecture

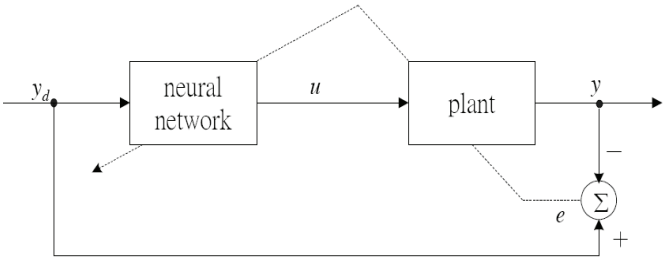


Fig. 4. The direct control for specialized learning architecture

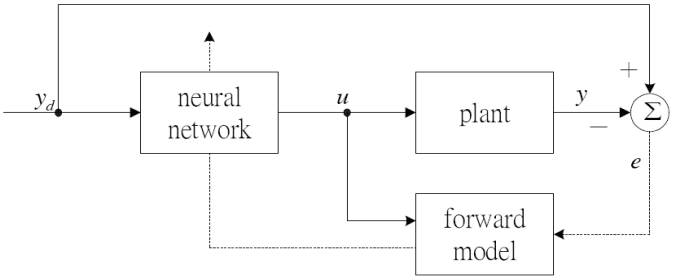


Fig. 5. The indirect control for specialized learning architecture

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 $\acute{e}$ 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 bedetermined based on the specified performance index of control system.

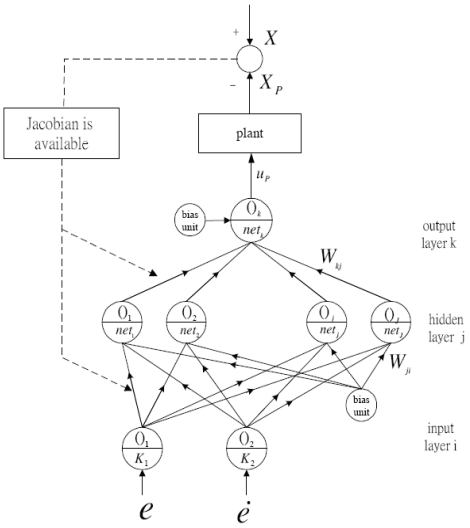


Fig. 6. A direct neural controller with three layers

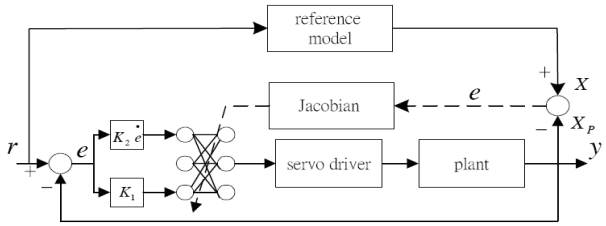


Fig. 7. The structure of a direct neural control system

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:

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

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

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

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. θ_j and θ_k denote the bias of the hidden and output layers, respectively.
For the Nth sampling time, the error function is defined as

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

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

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

where η is denoted as learning rate and α is the momentum parameter. The gradient of En with respect to the weights W_{kj} is determined by

$$\frac{\partial E_N}{\partial W_{kj}} = \frac{\partial E_N}{\partial net_k}\frac{\partial net_k}{\partial W_{kj}} = \delta_k O_j \tag{7}$$

and δ_k is defined as

$$\begin{aligned}\delta_k &= \frac{\partial E_N}{\partial net_k} = \sum_n \frac{\partial E_N}{\partial X_P}\frac{\partial X_P}{\partial u_P}\frac{\partial u_P}{\partial O_n}\frac{\partial O_n}{\partial net_k} = \sum_n \frac{\partial E_N}{\partial O_n}\frac{\partial O_n}{\partial net_k} \\ &= \sum_n \frac{\partial E_N}{\partial O_n}\beta(1 - O_k^2) \qquad n = 1,2,...K\end{aligned} \tag{8}$$

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 δ_k can besolved.
Similarly, the gradient of E_n with respect to the weights, W_{ji} is determined by

$$\frac{\partial E_N}{\partial W_{ji}} = \frac{\partial E_N}{\partial net_j}\frac{\partial net_j}{\partial W_{ji}} = \delta_j O_i \tag{9}$$

$$\delta_j = \frac{\partial E_N}{\partial net_j} = \sum_m \frac{\partial E_N}{\partial net_k} \frac{\partial net_k}{\partial O_m} \frac{\partial O_m}{\partial net_j}$$
$$= \sum_m \delta_k W_{km} \beta (1 - O_j^2) \qquad m = 1,2,\ldots J \tag{10}$$

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

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

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

where δ_j and δ_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 .

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

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

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

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

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

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

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

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

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 $X(N)$ $u(N)_{PP}$((<0 leads to

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

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

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

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

$$\frac{\partial E_N}{\partial O_k} = K_3 e + K_4 \frac{\mathrm{de}}{\mathrm{dt}} \tag{20}$$

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 $0.001 V\ rpm$, 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.

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

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 u_p 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.

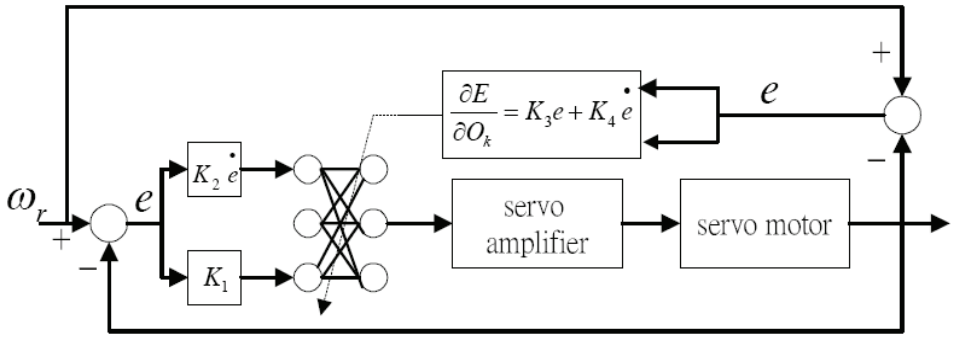


Fig. 8. The speed control system with direct neural controller

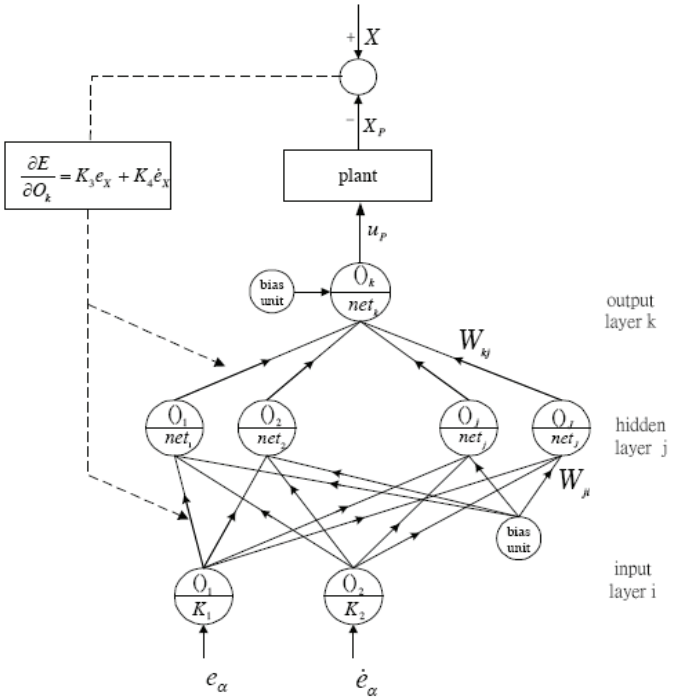


Fig. 9. The direct neural controller

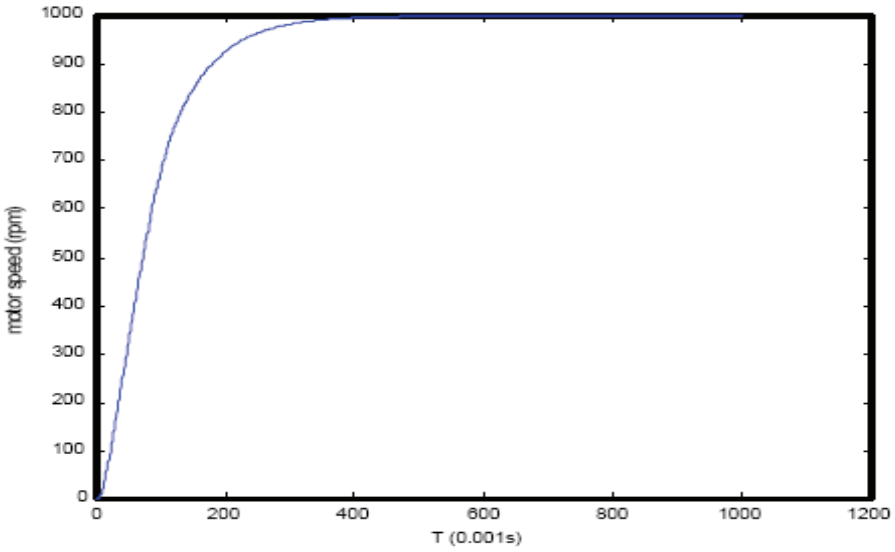


Fig. 10. Speed response for DC servo motor

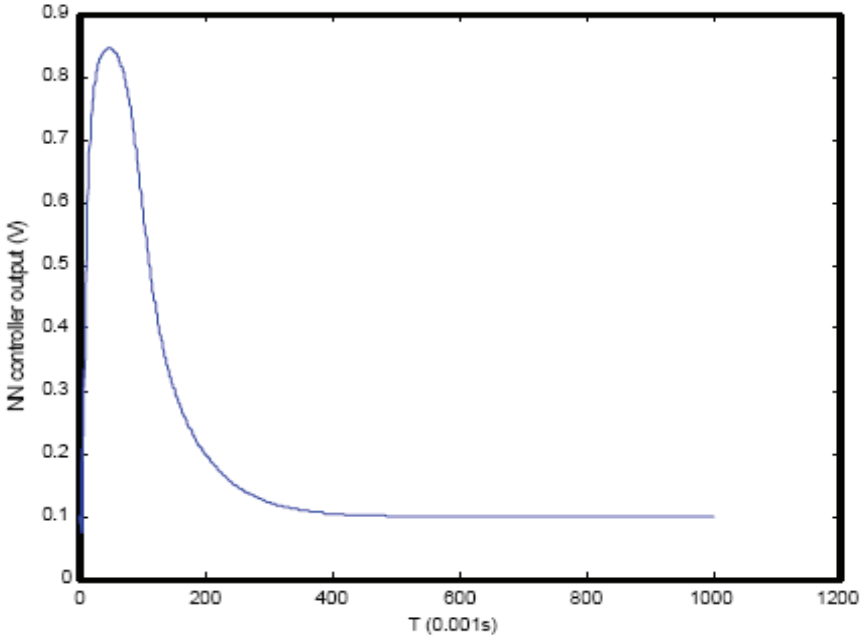


Fig. 11. The time response for control input

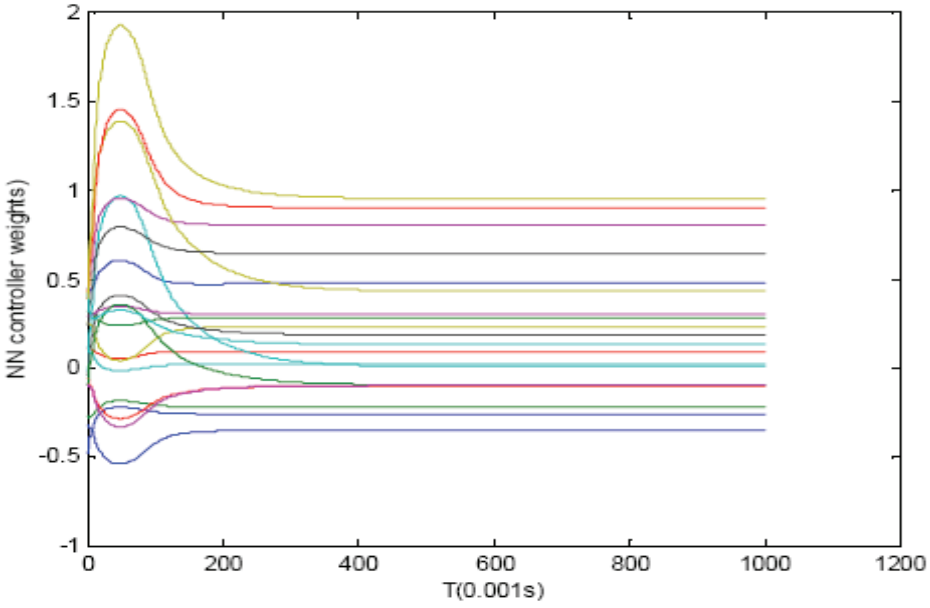


Fig. 12. All connective weights are convergent before 0.4s

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.

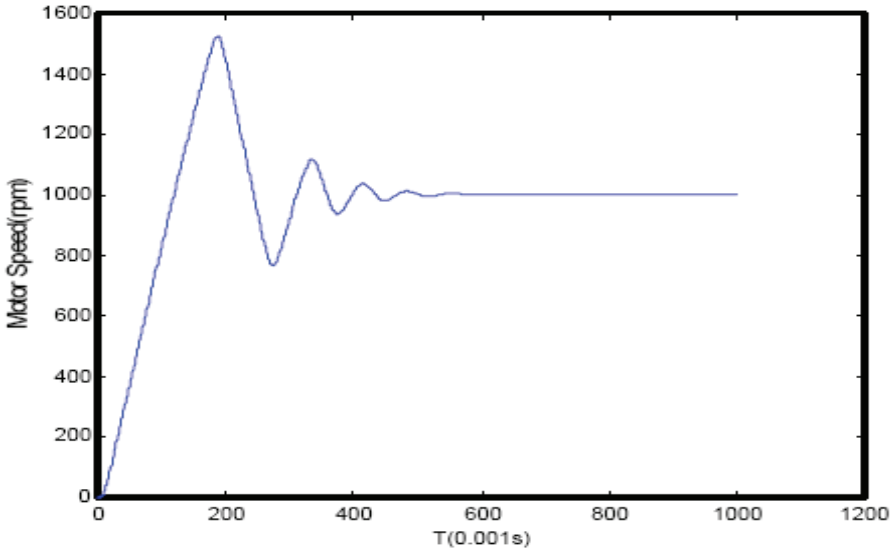


Fig. 13. Speed response for DC servo motor

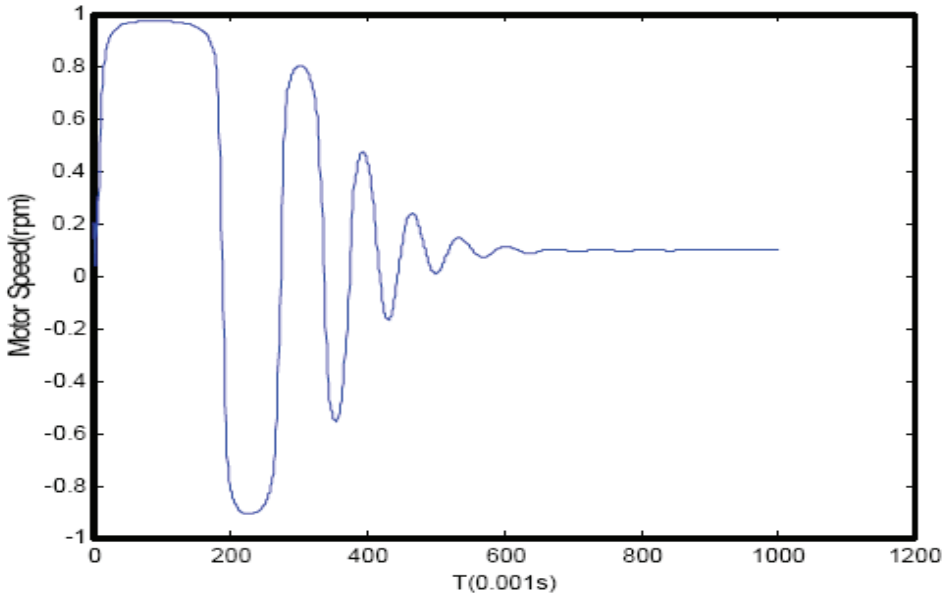


Fig. 14. The time response for control input

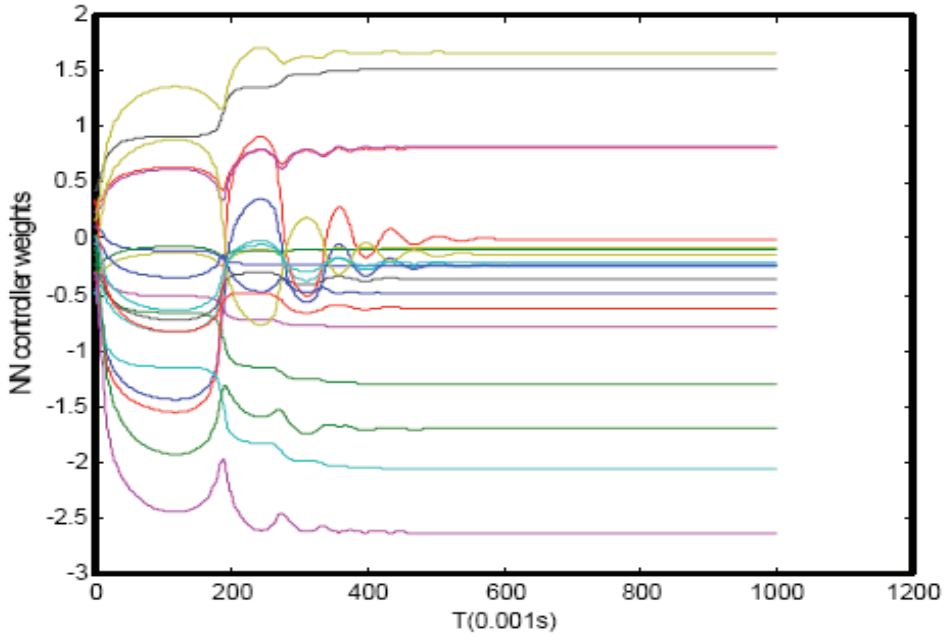


Fig. 15. All connective weights are convergent before 0.6s

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 $'_r$ is the speed command and $'$ 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.

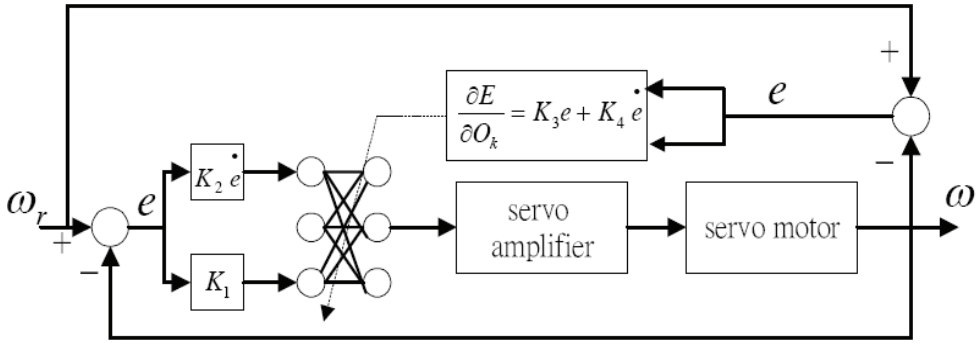


Fig. 14. The block diagram of speed control system

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

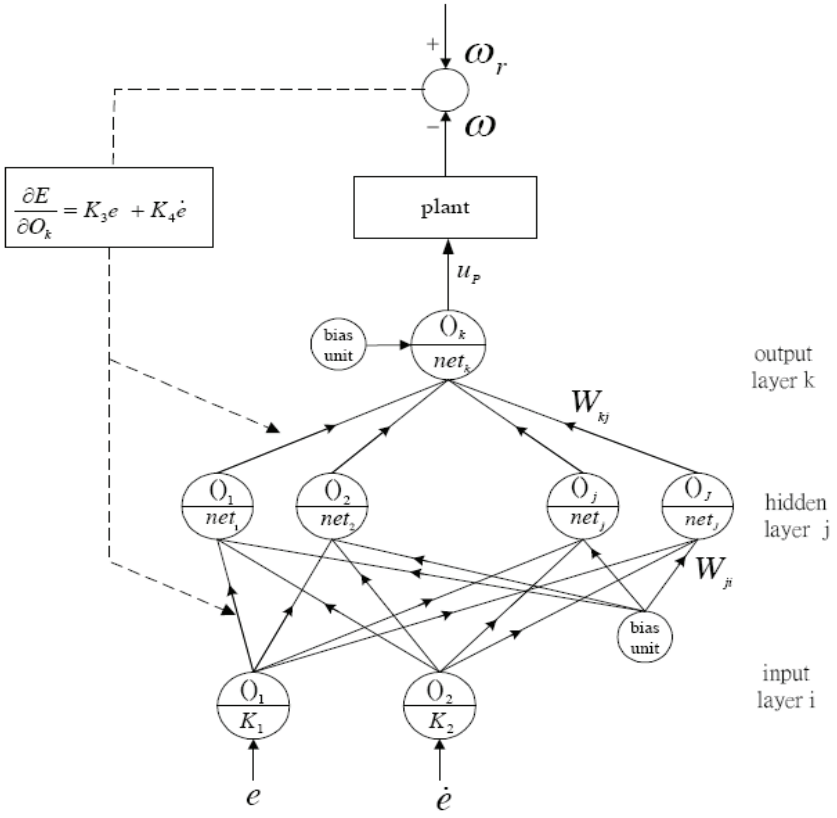


Fig. 15. The structure of proposed neural controller

The difference between command speed $'_r$ and the actual output speed $'$ is defined as error e. The error e and its differential e are normalized between A1 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 A 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 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 A5V to +5V and a servo amplifier with voltage gain of 2.3. The parameters of DC servo motor are listed in Table 1.

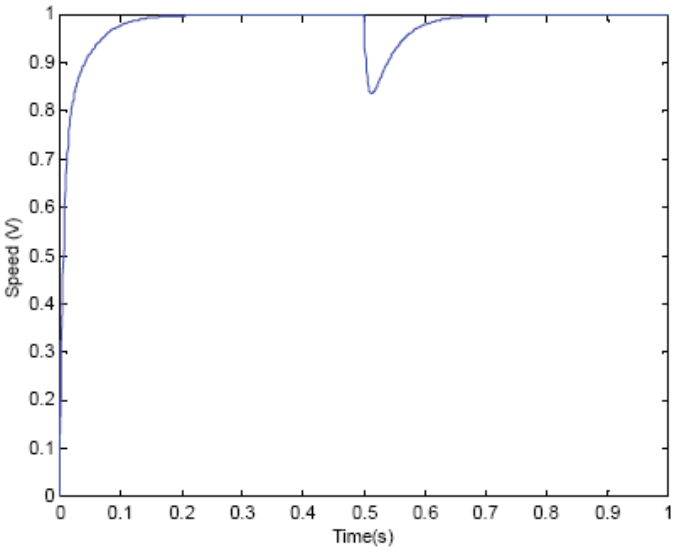
Motor resistance R_a	3.18Ω
Motor inductance L_a	$0.53mH$
Inertia of rotor J	$24.3\times10^{-7}kgm^2$
Torque constant K_T	$23mNm/A$
Back emf K_B	0.00241V/rpm

Table 1. The parameters of motor

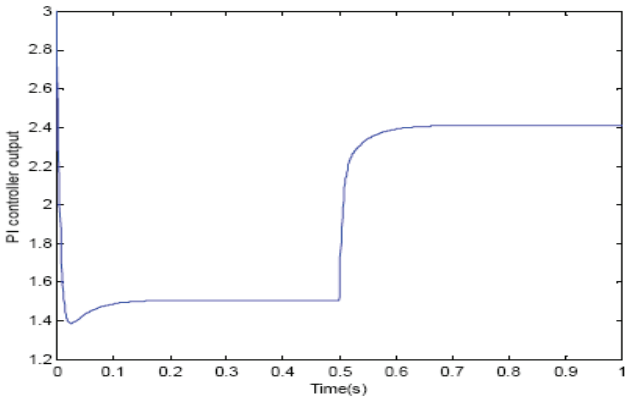
In the designed direct neural controller, the number of neurons is set to be 2, 5 and 1 for the input, hidden and output layers, respectively (see Fig.2). There is only one neuron in the output layer. The output signal of the direct neural controller will be between -1 and +1, which is converted into 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 Kmust 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 !"of the neural network is set to be 0.1 and the coefficient β=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_2 and K_4 will increase the damping effect of control system.

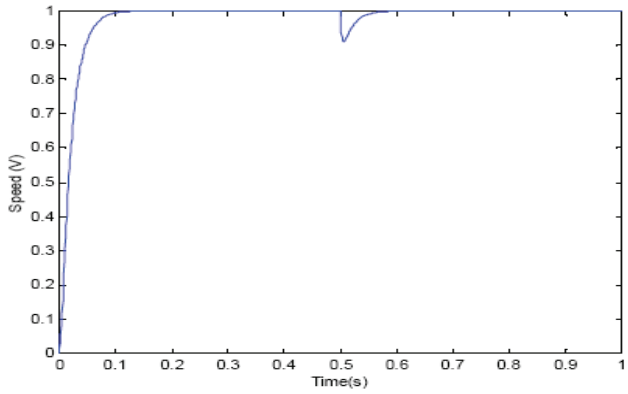


(a) Speed response

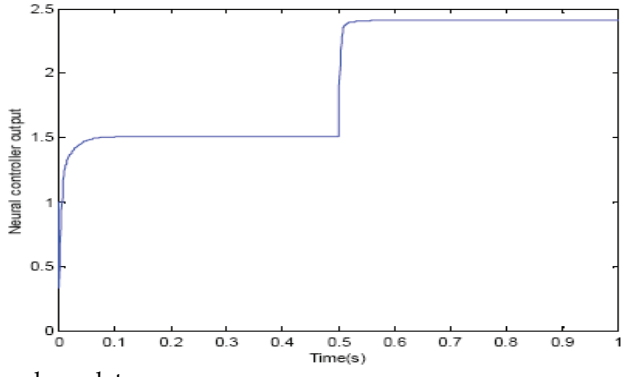


(b) Output of PI controller

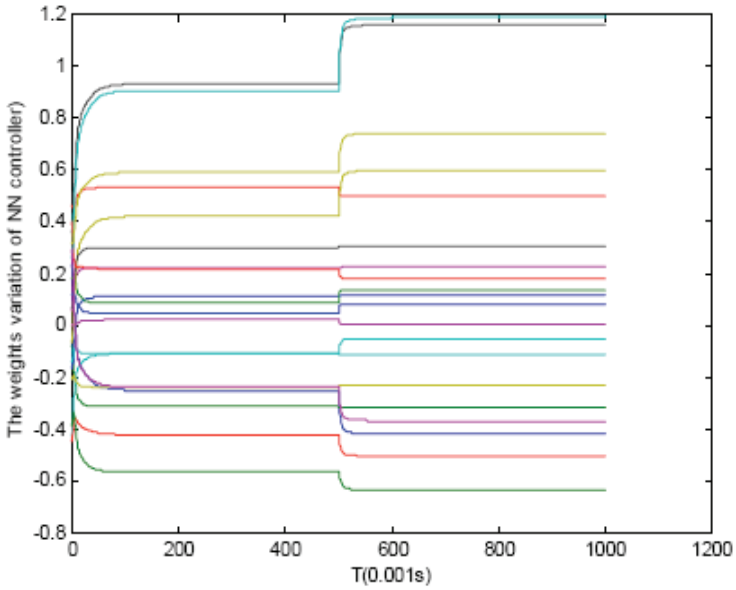
Fig. 16. Simulation results for speed regulation of DC servo motor with PI controller



(a) Speed response



(b) Output of neural regulator

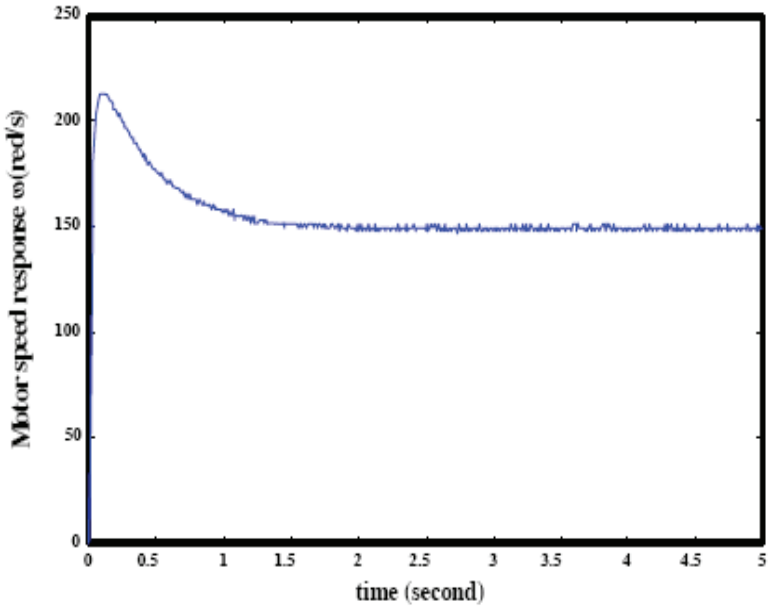


(c) The time responses for connective weights

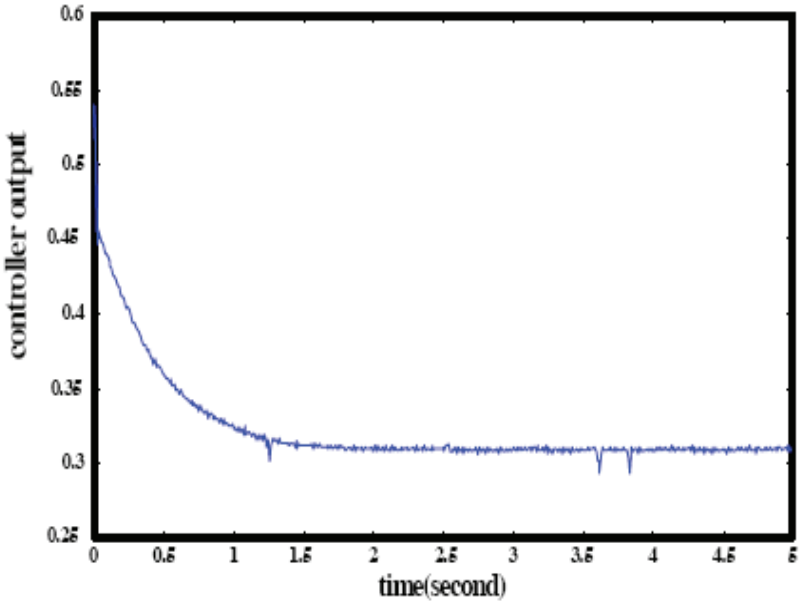
Fig. 17. Simulation results for speed regulator of DC servo motor with neural regulator

3.3 Experimental Results

The experimental apparatus consist of a 15W DC servo motor whose parameters shown in Table 1 are the same as that of simulation, an encoder with a unit of 0.01256 rad/pulse, an 12bits bipolar D/A converter with a voltage range of A5.13V to +5.13V and a servo amplifier with voltage gain of 2.3. 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. There is only one neuron in the output layer. The output signal of the direct neural controller will be between A1 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 +4.847V and -4.847V corresponding to the output signal between +1 and -1 of the neural controller. Then, the voltage signal is amplified by the servo-amplifier to provide high current for driving the DC servo motor. Furthermore, the parameters K_1 and K_2 must be adjusted in order to normalize the input signals of the neural regulator. In this experiment, the parameters K_3 and K_4 depend on K_1 and K_2, and $K_3 = K_1$ and $K_4 = K_2$ is assigned. A step signal of 120 pulses/0.01s (12000 pulses/s) corresponding to 150.8 rad/s is denoted as the speed command. The learning rate η=0.3, the coefficient β=0.5 and the sampling time of 0.01s are assigned. The parameter K_1=0.003 and K_2 =0.00003 are defined to normalize the input signals of the neural controller. The relations $K_1 = K_3$ and $K_2 = K_4$ are applied in our experiments. The results are shown in Fig. 18. It shows that the speed response of a DC motor is stable and accurate but with overshoot. The experiment result with $K_{24} = K = 0.00004$ and η=0.5 is shown in Fig. 19. It shows that parameters K_2 and K_4 exhibit a damping effect evidenced by an overshoot in the speed response of a DC servo motor as $K_2 = K_4 = 0.0003$, and the overshoot decreased as $K_2 = K_4 = 0.00004$. The settling time is decreased as $K_2 = K_4 = 0.0004$.

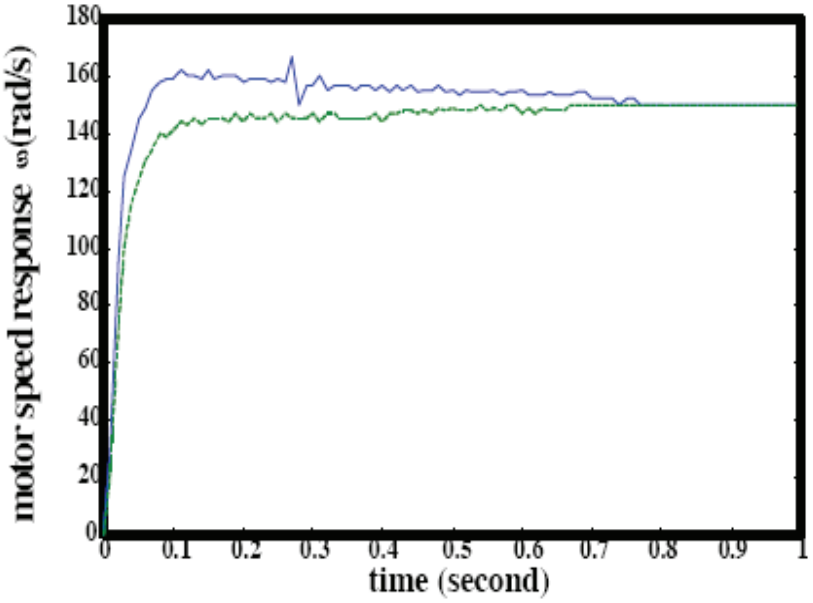


(a) Speed response of DC servo motor

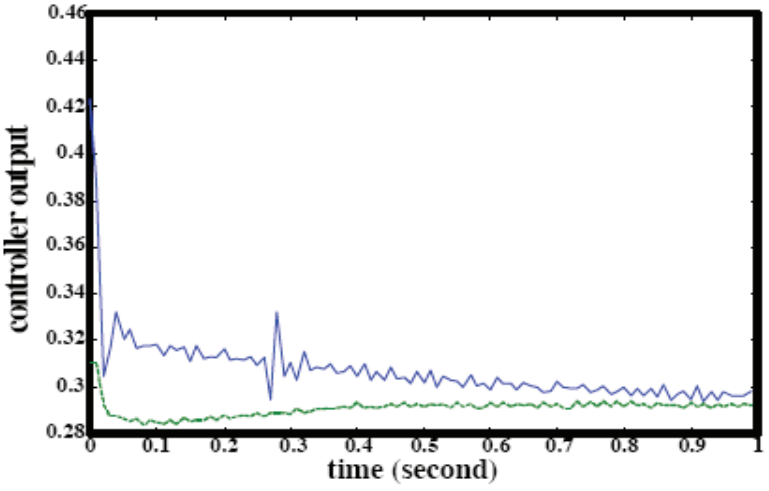


(b) The output of neural controller

Fig. 18. Experiment results (Sampling time=0.01s, η=0.3, $K_1 = K_3 = 0.003$, $K_2 = K_4 = 0.00003$)



(a) Speed response of DC servo motor

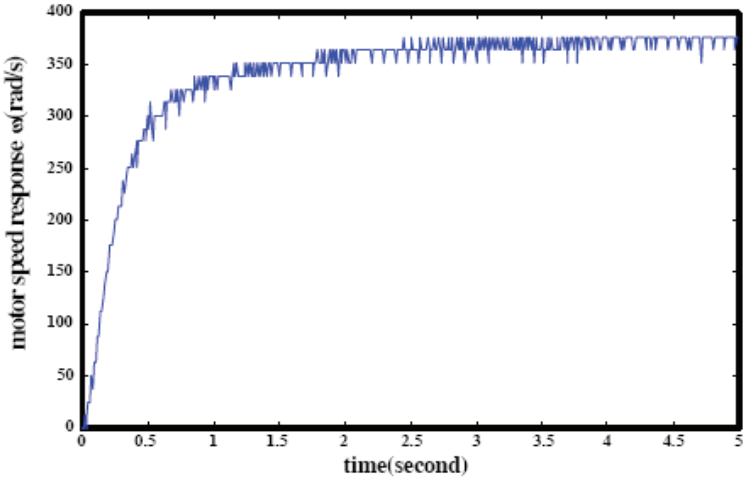


(b) The output of neural controller

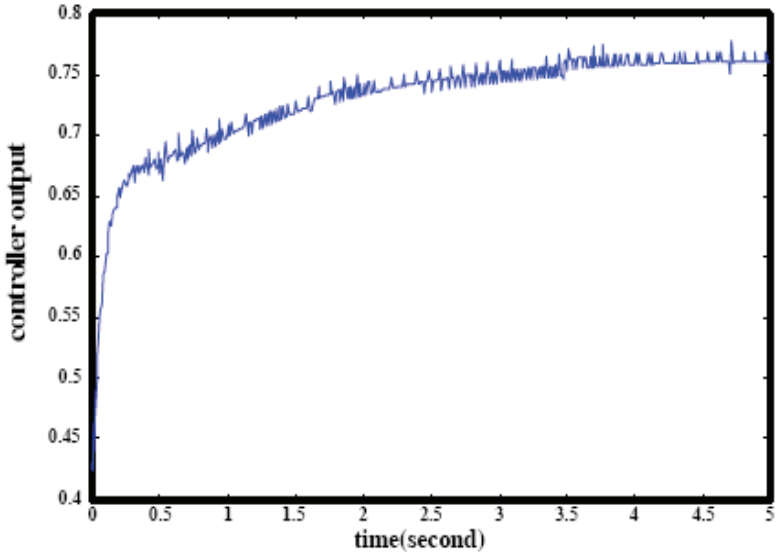
Fig. 19. Experiment results (Sampling time=0.01s, $K_1 = K_3 = 0.003$,$\eta = 0.5$,K_2 $K_4 = 0.00003$ **: ----,** $K_2 = K_4 = 0\ 00004$ **: ----**)

It is a better way for neural controller that the sampling time is as small as possible, then choosing a correspondent small learning rate. It means the connective weights can have more update times within the same time interval. The smaller sampling interval leads to

more weight updates per second. It is helpful for convergence of on line learning. So that, a smaller sampling interval of 0.001s and the speed command of 30 pulses/ms (30,000 pulses/s) corresponding to 377rad/s are applied to this experiment, it means the connective weights can be updated 1000 times per second. The parameters $K_1 = K_3 = 0.003$ and $K_2 \; K_4 = 0.00003$ are assigned for this experiment. Both of the learning rate of 0.3 and 0.5 are assigned, and the corresponding experiment results are shown in Fig. 20 and Fig. 21 respectively.

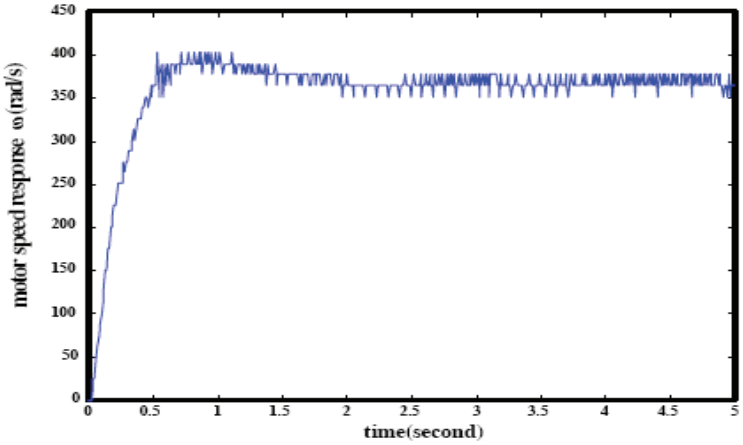


(a) Speed response of DC servo motor

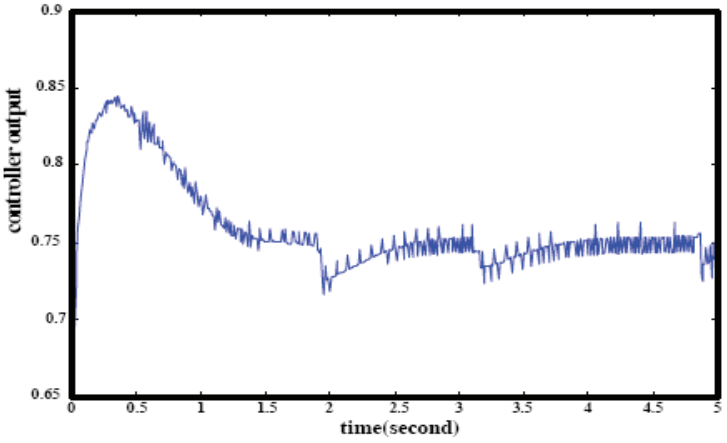


(b) The output of neural controller

Fig. 20. Experiment results (Sampling time=0.001s, η=0.3, $K_1 = K_3 = 0.03$, $K_2 = K_4 = 0.00003$)



(a) Speed response of DC servo motor



(b) The output of neural controller

Fig. 21. Experiment results (Sampling time=0.001s, η=0.5, $K_1 = K_3 = 0.03$, $K_2 = K_4 = 0.00003$)

Fig. 20 and Fig. 21 show the smaller sampling interval make the pulse number of one sampling interval become smaller, so that the speed error to speed command ratio will become larger. The speed error is between -1 and +1 pulse per sampling interval.

In Fig. 21, the speed response is still stable with $\eta = 0.5$, but more overshoot can be investigated; owing to the fact that more learning rate induces more neural controller output and get more overshoot. It can be investigated that the sampling time needs to be smaller, then choosing a correspondent small learning rate. It is proven that the speed response of a DC servo motor with the proposed direct neural controller is stable and accurate. The simulation and experimental results show the speed error comes from speed sensor characteristics, the measurement error is between -1 and +1 pulse per sampling interval. If the resolution of encoder is improved, the accuracy of the control system will be increased.

The speed error is in the interval of 1 pulses/0.01s as the sampling time of 0.01s, but it is in the interval of 1 pulses/0.001s as the sampling time of 0.001s. The step speed command is assigned as 120 pulses/0.01s (150.72rad/s) with the sampling interval of 0.01s, and the step speed command needs to be increased to 30pulses/ms (377rad/s) to keep the accuracy of the speed measurement. Furthermore, we have to notice the normalization of the input signals. From the experimental results, the input signals need to be normalized between +1 and A1. The learning rate should be determined properly depends on the sampling interval, the smaller sampling interval can match the smaller learn rate, and increase the stability of servo control system.

4. The Direct Neural Control Applied to Hydraulic Servo Control Systems

The electro-hydraulic servo systems are used in aircraft, industrial and robotic mechanisms. They are always used for servomechanism to transmit large specific powers with low control current and high precision. The electro-hydraulic servo system (EHSS) consists of hydraulic supply units, actuators and an electro-hydraulic servo valve (EHSV) with its servo driver. The EHSS is inherently nonlinear, time variant and usually operated with load disturbance. It is difficult to determine the parameters of dynamic model for an EHSS. Furthermore, the parameters are varied with temperature, external load and properties of oil etc. The modern precise hydraulic servo systems need to overcome the unknown nonlinear friction, parameters variations and load variations. It is reasonable for the EHSS to use a neural network based adaptive control to enhance the adaptability and achieve the specified performance.

4.1 Description of the electro-hydraulic servo control system

The EHSS is shown in Fig. 22 consists of hydraulic supply units, actuators and an electro-hydraulic servo valve (EHSV) with its servo driver. The EHSV is a two-stage electro hydraulic servo valve with force feedback. The actuators are hydraulic cylinders with double rods.

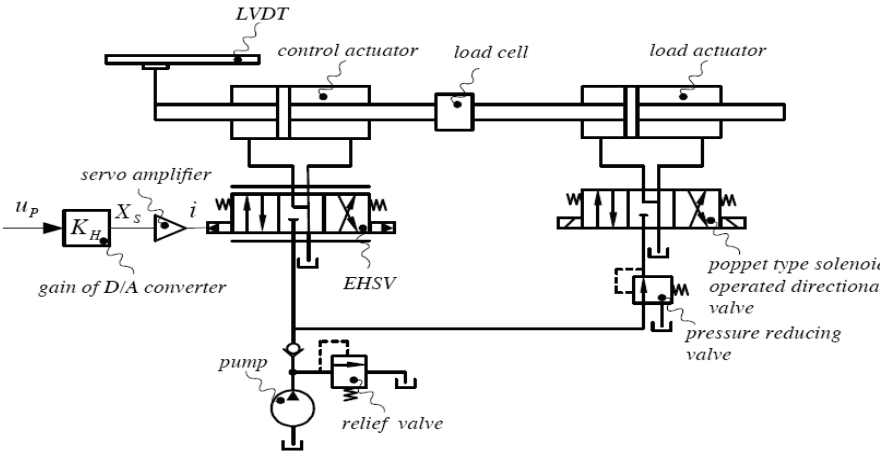


Fig. 22. The hydraulic circuit of EHSS

The application of the direct neural controller for EHSS is shown in Fig. 23, where y_r is the position command and y_p is the actual position response.

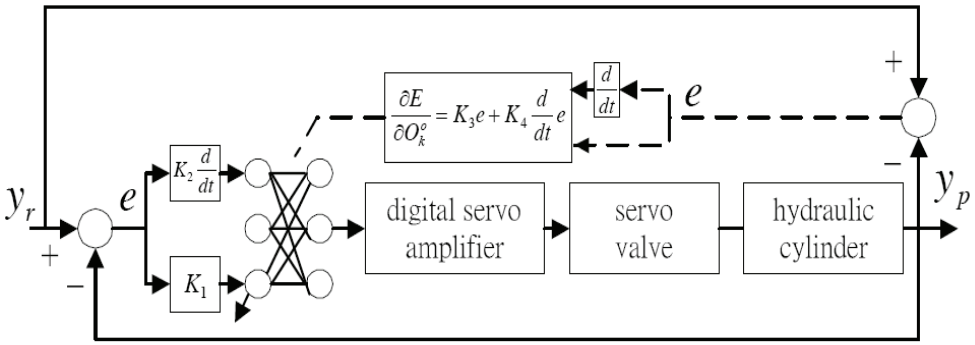


Fig. 23. The block diagram of EHSS control system

A. The simplified servo valve model

The EHSV is a two-stage electro hydraulic servo valve with force feedback. The dynamic of EHSV consists of inductance dynamic, torque motor dynamic and spool dynamic. The inductance and torque motor dynamics are much faster than spool dynamic, it means the major dynamic of EHSV determined by spool dynamic, so that the dynamic model of servo valve can be expressed as:

$$\frac{\Delta x_v}{\Delta e} = \frac{K_t}{1 + S/K_{vf}} \tag{24}$$

Δx_v : The displacement of spool

Δe : The input voltage

B. The dynamic model of hydraulic cylinder

The EHSV is 4 ports with critical center, and used to drive the double rods hydraulic cylinder. The leakages of oil seals are omitted and the valve control cylinder dynamic model can be expressed as [8]:

$$X_P = \frac{\frac{kq}{A_P} x_v - \frac{k_c}{A_P^2}(1 + \frac{V_t}{4\beta_e k_c} s) F_L}{s(\frac{V_t M_t}{4\beta_e A_P^2} s^2 + (\frac{k_c M_t}{A_P^2} + \frac{B_P V_t}{4\beta_e A_P^2}) s + (1 + \frac{B_P k_c}{A_P^2}))} \tag{25}$$

x_v: The displacement of spool

F_L: The load force

X_P:The piston displacement

C. Direct Neural Control System

There are 5 hidden neurons in the proposed neural controller. The proposed DNC is shown in Fig. 24 with a three layers neural network.

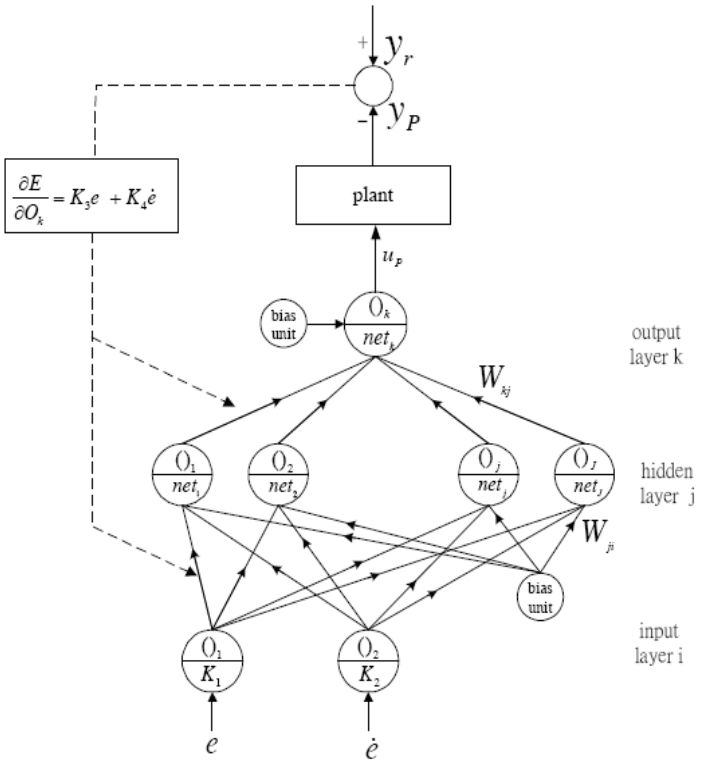


Fig. 24. The structure of proposed neural controller

The difference between command y_r and the actual output position response y_p is defined as error e. The error e and its differential $\dot{e}$ are normalized between A1 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 A 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 EHSV. A square command is assigned as the reference command in order to simulate the position response of the EHSS. The proposed three layers neural network, including the hidden layer (j), output layer (k) and input layer (i) as illustrated in Fig. 24. The input signals e and $\dot{e}$ are normalized between A 1 and +1, and defined as signals O_i feed to hidden neurons. A tangent hyperbolic function is used as the activation function of the nodes in the hidden and output layers. The net input to node j in the hidden layer is

$$net_j = \sum (W_{ji} \cdot O_i) + \theta_j \qquad i = 1,2,...I, \quad j = 1,2,...J \tag{26}$$

the output of node j is

$$O_j = f(net_j) = \tanh(\beta \cdot net_j) \tag{27}$$

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

$$net_k = \sum (W_{kj} \cdot O_j) + \theta_k \qquad j = 1,2,...J, \qquad k = 1,2,...K \tag{28}$$

the output of node k is

$$O_k = f(net_k) = \tanh(\beta \cdot net_k) \tag{29}$$

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 θ_j and θ_k denote the bias of the hidden and output layers, respectively. The error energy function at the Nth sampling time is defined as

$$E_N = \frac{1}{2}(y_{r\,N} - y_{PN})^2 = \frac{1}{2}e_N^2 \tag{30}$$

where $y_{r\,N}$, y_{PN} and e_N denote the the reference command, the output of the plant and the error term at the Nth sampling time, respectively. The weights matrix is then updated during the time interval from N to N+1.

$$\Delta W_N = W_{N+1} - W_N = -\eta \frac{\partial E_N}{\partial W_N} + \alpha \cdot \Delta W_{N-1} \tag{31}$$

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

$$\frac{\partial E_N}{\partial W_{kj}} = \frac{\partial E_N}{\partial net_k} \frac{\partial net_k}{\partial W_{kj}} = \delta_k O_j \tag{32}$$

and δ_k is defined as

$$\delta_k = \frac{\partial E_N}{\partial net_k} = \sum_n \frac{\partial E_N}{\partial X_P}\frac{\partial X_P}{\partial u_P}\frac{\partial u_P}{\partial O_n}\frac{\partial O_n}{\partial net_k} = \sum_n \frac{\partial E_N}{\partial O_n}\frac{\partial O_n}{\partial net_k}$$
$$= \sum_n \frac{\partial E_N}{\partial O_N}\beta(1-O_k^2) \qquad n=1,2,\cdots,K \tag{33}$$

where $\partial X_P / \partial u_P$ is difficult to be evaluated. The EHSS is a single-input and single-output control system (i.e., n =1), in this study, the sensitivity of E_N with respect to the network output O_k is approximated by a linear combination of the error and its differential shown as:

$$\frac{\partial E_N}{\partial O_k} = K_3 e + K_4 \frac{de}{dt} \tag{34}$$

where K_3 and K_4 are positive constants. Similarly, the gradient of E_N with respect to the weights, W_{ji} is determined by

$$\frac{\partial E_N}{\partial W_{ji}} = \frac{\partial E_N}{\partial net_j}\frac{\partial net_j}{\partial W_{ji}} = \delta_j O_i \tag{35}$$

where

$$\delta_j = \frac{\partial E_N}{\partial net_j} = \sum_m \frac{\partial E_N}{\partial net_k}\frac{\partial net_k}{\partial O_m}\frac{\partial O_m}{\partial net_j}$$
$$= \sum \delta_k W_{km}\beta\left(1-O_j^2\right) \qquad m=1,2,\cdots,J \tag{36}$$

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

$$\Delta W_{kj,N} = -\eta\frac{\partial E_N}{\partial W_{kj,N}} + \alpha\cdot\Delta W_{kj,N-1}$$
$$= -\eta\delta_k O_j + \alpha\cdot\Delta W_{kj,N-1} \tag{37}$$

$$\Delta W_{ji,N} = -\eta\frac{\partial E_N}{\partial W_{ji,N}} + \alpha\cdot\Delta W_{ji,N-1}$$
$$= -\eta\delta_j O_i + \alpha\cdot\Delta W_{ji,N-1} \tag{38}$$

where η is denoted as learning rate and α is the momentum parameter δ_j and δ_k can be evaluated from Eq.(34) and (31), The weights matrix are updated during the time interval from N to N+1 :

$$W_{kj,N+1} = W_{kj,N} + \Delta W_{kj,N} \tag{39}$$

$$W_{ji,N+1} = W_{ji,N} + \Delta W_{ji,N} \tag{40}$$

4.2 Numerical Simulation

An EHSS shown as Fig.1 with a hydraulic double rod cylinder controlled by an EHSV is simulated. A LVDT of 1 V/m measured the position response of EHSS. The numerical simulations assume the supplied pressure $Ps = 70 Kg_f/cm^2$, the servo amplifier voltage gain of 5, the maximum output voltage of 5V, servo valve coil resistance of 250 ohms, the current to voltage gain of servo valve coil of 4 $mA\ V$ (250 ohms load resistance), servo valve settling time ≈ 20ms, the serve valve provides maximum output flow rate = 19.25 l /min under coil current of 20mA and ΔP of $70 Kg_f/cm^2$ condition. The spool displacement can be expressed by percentage (%), and then the model of servo valve can be built as

$$\frac{x_v(100\%)}{i\ (mA)} = \frac{0.05}{S/200+1} \tag{41}$$

or

$$\frac{x_v(100\%)}{v\ (V)} = \frac{0.2}{S/200+1} \tag{42}$$

The cylinder diameter =40mm, rod diameter=20mm, stroke=200mm, and the parameters of the EHSS listed as following:

$$A_p = 9.4248 cm^2 = 0.00094248 m^2$$

$$V_t = 188.5 cm^3 = 0.0001885 m^3$$

$$B_p = 40\ N \cdot s/m$$

$$k_c = 3.727(10^{-5})\ m^3/MPa \cdot s$$

$$M_t = 1 kgm$$

$$k = 0\ N/m$$

$$\beta_e = 1000 MPa$$

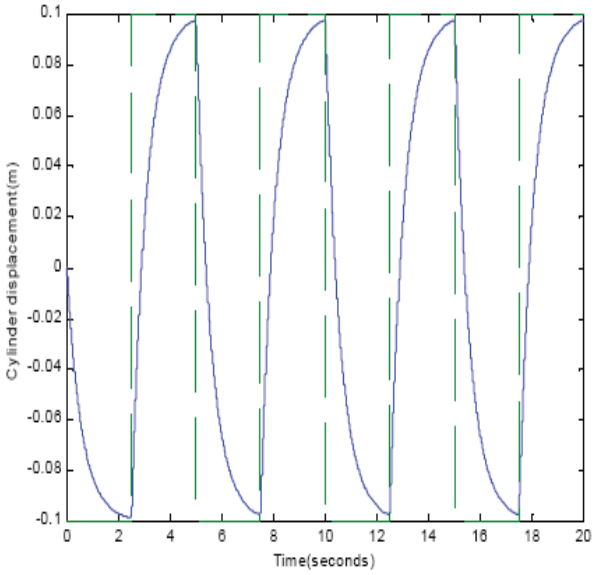
$$kq = 19.25\ \ell/\min \quad (\text{at}\quad \Delta P = 70.3\ Kg_f/cm^2)$$

$$= 320.833 m^3/s = 0.000320833\ m^3/s$$

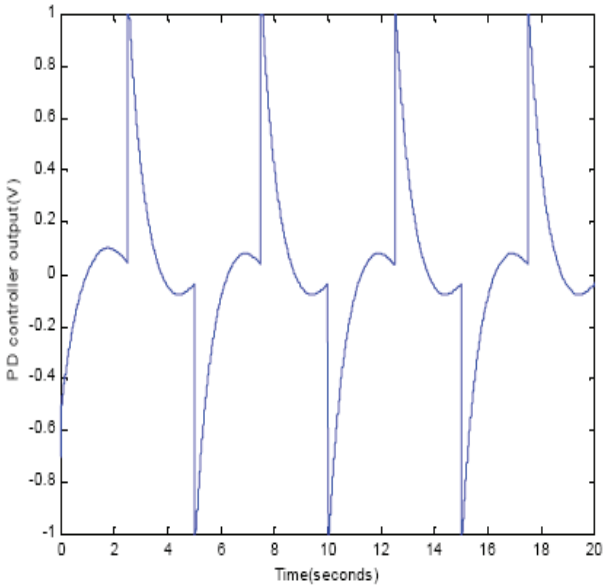
According to Eq(25), the no load transfer function is shown as

$$\frac{y_P}{x_v} = \frac{340414}{s(0.053s^2 + 44.122s + 1001678)} \tag{43}$$

The direct neural controller is applied to control the EHSS shown as Fig. 24, and the time responses for piston position are simulated. A tangent hyperbolic function is used as the activation function, so that the neural network controller output is between -1 . This is converted to be analog voltage between -) *Volt* by a D/A converter and amplified in current by a servo amplifier to drive the EHSV. The constants K_3 and K_4 are defined to be the parameters for the linear combination of error and its differential, which is used to approximate the BPE for weights update. A conventional PD controller with well-tuned parameters is also applied to the simulation stage as a comparison performance. The square signal with a period of 5 *sec* and amplitude of 0.1m is used as the command input. The simulation results for PD control is shown in Fig. 25 and for DNC is shown in Fig. 26. Fig. 26 reveals that the EHSS with DNC track the square command with sufficient convergent speed, and the tracking performance will become better and better by on-line trained. Fig. 27 shows the time response of piston displacement with 1200N force disturbance. Fig. 27 (a) shows the EHSS with PD controller is induced obvious overshoot by the external force disturbance, and Fig. 27 (b) shows the EHSS with the DNC can against the force disturbance with few overshoot. From the simulation results, we can conclude that the proposed DNC is available for position control of EHSS, and has favorable tracking characteristics by on-line trained with sufficient convergent speed.

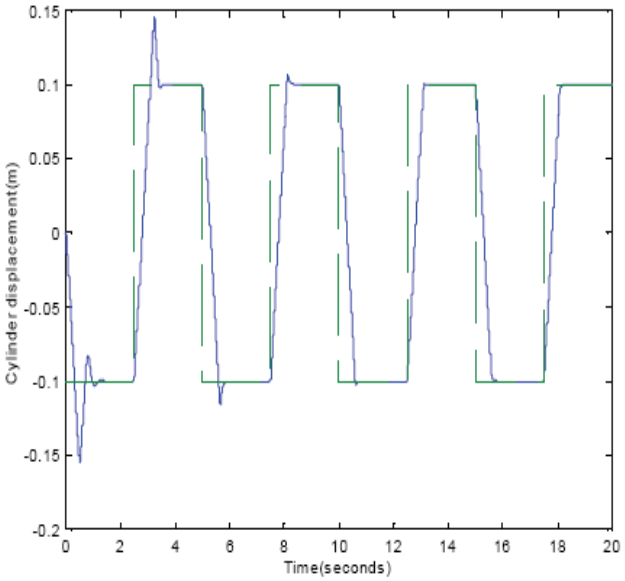


(a) Time response for piston displacement

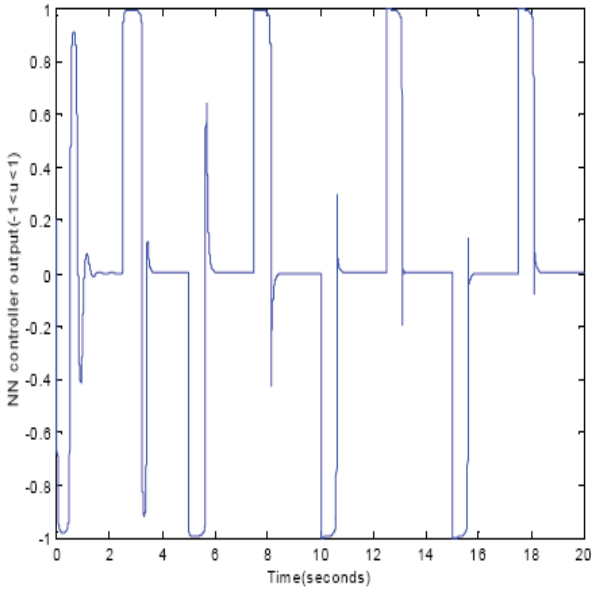


(b) Controller output

Fig. 25. The simulation results for EHSS with PD controller (Kp=7, Kd=1, Amplitude=0.1m and period=5 sec)

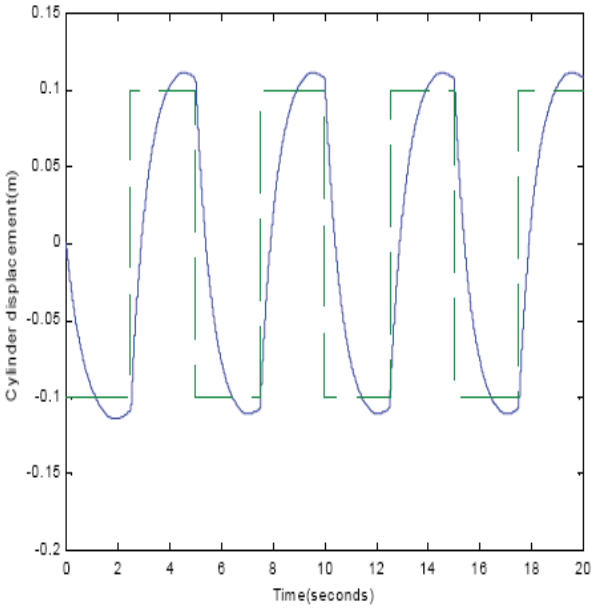


(a) Time response for piston displacement

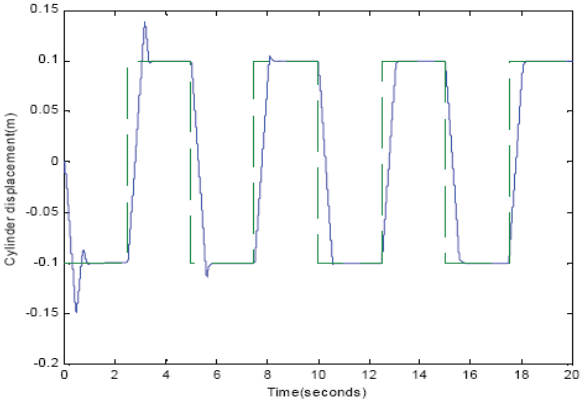


(b) Controller output

Fig. 26. The simulation results for EHSS with DNC (Amplitude=0.1m and period=5 *sec*)



(a) EHSS with PD controller

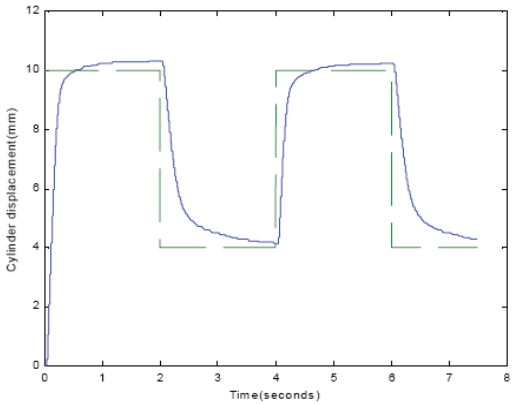


(b) EHSS with DNC

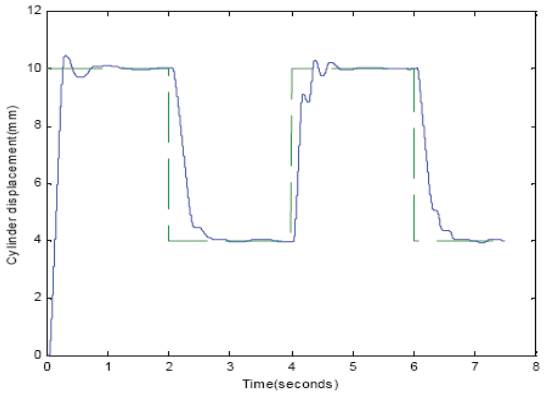
Fig. 27. Simulation results of position response with 1200N force disturbance

4.3. Experiment

The EHSS shown in Fig. 22 is established for our experiment. A hydraulic cylinder with 200mm stroke, 20mm rod diameter and a 40mm cylinder diameter is used as the system actuator. The Atchley JET-PIPE-206 servo valve is applied to control the piston position of hydraulic cylinder. The output range of the neural controller is between -1 , and converted to be the analog voltage between -5 *Volt* by a 12 bits bipolar DA /AD servo control interface, It is amplified in current by a servo amplifier to drive the EHSV. A crystal oscillation interrupt control interface provides an accurate 0.001 sec sample rate for real time control. A square signal with amplitude of 10mm and period of 4 sec is used as reference input. Fig. 28 shows the EHSS disturbed by external load force, which is induced by load actuator with operation pressure of 9 kg/cm^2. Fig. 28 (a) shows the EHSS with PD controller is induced obvious overshoot by the external force disturbance, and Fig. 28 (b) shows the EHSS with the DNC can against the force disturbance with few overshoot. The experiment results show the proposed DNC is available for position control of EHSS.



(a) EHSS with PD controller



(b) EHSS with DNC

Fig. 28. Experiment results of position response with the load actuator pressure of 9 kg/cm^2

The proposed DNC is applied to control the piston position of a hydraulic cylinder in an EHSS., and the comparison of time responses for the PD control system is analyzed by simulation and experiment. The results show that the proposed DNC has favorable characteristic, even under external force load condition.

5. Conclusion

The conventional direct neural controller with simple structure can be implemented easily and save more CPU time. But the Jacobian of plant is always not easily available. The DNC using sign function for approximation of Jacobian is not sufficient to apply to servo control system. The & adaptation law can increase the convergent speed effetely, but the appropriate parameters always depend on try and error. It is not easy to evaluated the appropriate parameters. The proposed self tuning type adaptive control can easily determined the appropriate parameters. The DNC with the well-trained parameters will enhance adaptability and improve the performance of the nonlinear control system.

6 References

D. Psaltis, A Sideris, and A. A. Yamamura (1988). A Multilayered Neural Network Controller, *IEEE Control System Magazine*, v.8, pp. 17-21.

Y. Zhang, P. Sen, and G. E. Hearn (1995). An On-line Trained Adaptive Neural Controller, *IEEE Control System Magazine*, v.15, pp. 67-75.

S. Weerasooriya and M. A. EI-Sharkawi Hearn (1991). Identification and Control of a DC Motor Using Back-propagation Neural Networks, *IEEE Transactions on Energy Conversion*, v.6, pp. 663-669.

A. Rubai and R. Kotaru (2000). Online Identification and Control of a DC Motor Using Learning Adaptation of Neural Networks, *IEEE Transactions on Industry Applications*, v.36, n.3.

S. Weerasooriya and M. A. EI-Sharkawi (1993). Laboratory Implementation of A Neural Network Trajectory Controller for A DC Motor, *IEEE Transactions on Energy Conversion*, v.8, pp. 107-113.

G. Cybenko (1989). Approximation by Superpositions of A Sigmoidal Function, *Mathematics of Controls*, Signals and Systems, v.2, n.4, pp. 303-314.

J. de Villiers and E. Barnard (1993). Backpropagation Neural Nets with One and Two Hidden layers, *IEEE Transactions on Neural Networks*, v.4, n.1, pp. 136-141.

R. P. Lippmann (1987). An Introduction to Computing with Neural Nets, *IEEE Acoustics, Speech, and Signal Processing Magazine*, pp. 4-22.

F. J. Lin and R. J. Wai (1998). Hybrid Controller Using Neural Network for PM Synchronous Servo Motor Drive, *Instrument Electric Engine Process Electric Power Application*, v.145, n.3, pp. 223-230.

Omatu, S. and Yoshioka, M. (1997). Self-tuning neuro-PID control and applications, *IEEE, International Conference on Systems*, Man, and Cybernetics, Computational Cybernetics and Simulation, Vol. 3.

Appendix: The simulation program

The simulation program for Example X.1 is listed as following=

1. Simulink block diagram

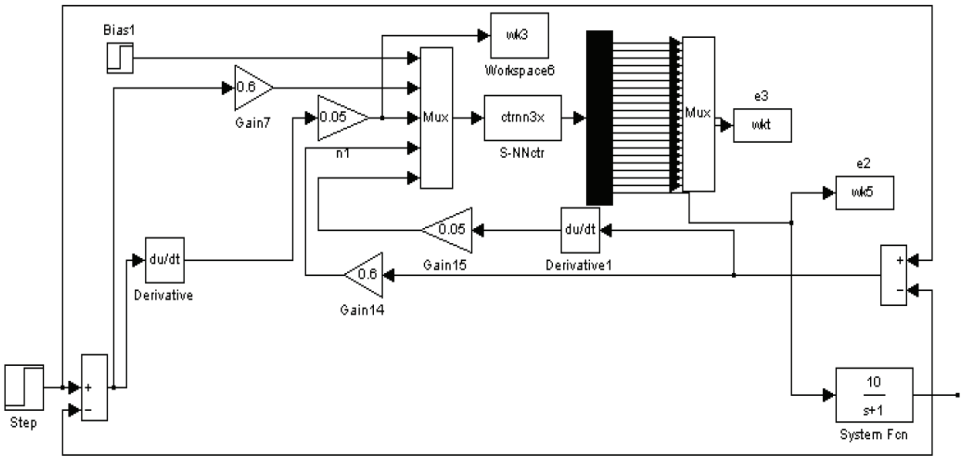


Fig. 1. The simulink program with S-function ctrnn3x

2. The content of S-function *ctrnn3x(t, x, u, flag)*

```
function [sys,x0,str,ts] = ctrnn3x(t,x,u,flag)
switch flag,
case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
case 2,
    sys=mdlUpdate(t,x,u);
case 3,
```

```
    sys=mdlOutputs(t,x,u);
case {1,4,9}
    sys=[];
otherwise
    error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates = 0;
sizes.NumDiscStates = 21;
sizes.NumOutputs = 21;
sizes.NumInputs = 5;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = [rand(1)-0.5;rand(1)-0.5;rand(1)-0.5;rand(1)-0.5;rand(1)-0.5;
    rand(1)-0.5;rand(1)-0.5;rand(1)-0.5;rand(1)-0.5;rand(1)-0.5;
    rand(1)-0.5;rand(1)-0.5;rand(1)-0.5;rand(1)-0.5;rand(1)-0.5;
    rand(1)-0.5;rand(1)-0.5;rand(1)-0.5;rand(1)-0.5;rand(1)-0.5;0.2];
%%%set the initial values for weights and states
%%%the initial values of weights randomly between -0.5 and +0.5
%%%the initial values of NN output assigned to be 0.2
str = [];
ts = [0 0];
function sys=mdlUpdate(t,x,u);
nv=0;
for j=1:5
  for i=1:3
    nv=nv+1;
    w1(j,i)=x(nv);
end
end
k=1;
  for j=1:5
nv=nv+1;
w2(k,j)=x(nv);
  end
  for j=1:5
jv(j)=w1(j,:)*[u(1);u(2);u(3)]; %u(1)=K1*e ,u(2)=K2*de/dt
%u(3)=1 is bias unity
ipj(j)=tanh(0.5*jv(j)); %outputs of hidden layer
  end
kv(1)=w2(1,:)*ipj';
opk(1)=tanh(0.5*kv(1)); %output of output layer
  for j=1:5
dk=(u(4)+u(5))*0.5*(1-opk(1)*opk(1));
%%%delta adaptation law, dk means delta K,u(4)=K3*e ,u(5)=K4*de/dt
dw2(1,j)=0.1*dk*ipj(j); %dw2 is weight update quantity for W2
  end
  for j=1:5
sm=0;
sm=sm+dk*w2(1,j);
sm=sm*0.5*(1-ipj(j)*ipj(j));
dj(j)=sm; %back propogation, dj means delta J
  end
  for j=1:5
    for i=1:3
```

```
dw1(j,i)=0.1*dj(j)*u(i); %dw1 is weight update quantity for W1
  end
end
  for j=1:5
w2(1,j)=w2(1,j)+dw2(1,j); %weight update
    for i=1:3
w1(j,i)=w1(j,i)+dw1(j,i); %weight update
    end
  end
nv=0;
for j=1:5
  for i=1:3
nv=nv+1;
x(nv)=w1(j,i); %assign w1(1)-w1(15) to x(1)-x(15)
  end
end
k=1;
  for j=1:5
nv=nv+1;
x(nv)=w2(k,j); %assign w2(1)-w2(5) to x(16)-x(20)
  end
x(21)=opk(1); %assign output of neural network to x(21)
sys=x; %Assign state variable x to sys
function sys=mdlOutputs(t,x,u)
for i=1:21
sys(i)=x(i);
end
```