A NOVEL PID-LIKE NEURAL NETWORK CONTROLLER

Shuang Cong Guodong Li Beichen Ji

Dept. of Automation University of Science & Technology of China
Anhui, Hefei, 230027, P. R. of China, E-mail address: scong@ustc.edu.cn

Abstract: In this paper, a novel PID-like neural network controller (PIDNNC) is created. It is composed of a neural network with no more than 3 neural nodes in hidden layer, and there are an activation feedback and (or) an output feedback in hidden layer, respectively. This special structure makes the network be able to be a P, PI, PD, or PID controller as needed. The proposed controller weights can be updated on-line according to errors caused by non-linear and uncertain factors of system, based on some adaptation mechanism. The resilient back-propagation algorithm with sign instead of the gradient is used to derive the rule of updating network weights. The basic ideas, techniques and analysis are presented. Finally, we give the simulation experiment of a double inverted-pendulum system and the comparison of the effects between the proposed control strategy and the conventional one.

Keywords: multivariable control system, neural networks, neural control system design, PID controllers, adaptive algorithms

1. INTRODUCTION

Proportional-integral-derivative (PID) control is a very popular control strategy in industry due to its simple architecture and easy tuning. Despite their wide spread use and considerable history, PID tuning is still an active area of research, both academic and industrial. During the past five decades, a comprehensive PID tuning literature has been developed. Roughly speaking, there are two different approaches to obtain PID and PID-like controller parameters. First, tune the parameters of the PID structure by following one of several available tuning techniques. Examples of these techniques include: Ziegler-Nichol (Z-N) method (Ziegler and Nichols, 1942), internal-model-control (IMC) based method (Morari and Zafiriou, 1989), optimization method (Zhuang and Atherton, 1993) and gain-phase margin method (Ho, *et al.*,1995). For single input/output (SISO) plants, satisfactory control can be achieved by using established tuning rules. These rules can be applied to multivariable plants with SISO characteristics as well. Many multivariable plants, however, show significant internal interaction. Since the tuning rules are developed for SISO system, their applications to such “true” multivariable plants are ineffective. Further, in the case of multivariable plants, the number of PID parameters becomes quite a lot. Trial and error techniques are thus inadequate to derive a good compromise between controller performance and robustness. The underlying theory for multi-loop plants is still immature. Secondly, assume that the controller has a PID structure, and

This paper supported by National Natural Science Foundation of China (No. 50375148)

find the PID parameters by using some well-known optimization methods, *e.g.*, H_∞ (Grimble, 1990), mixed H_2 / H_∞ (Chen, *et al*., 1995) and semi-definite programming approaches (Bao, *et al.*, 1999). These methods can be used to obtain the PID controller parameters such that the controllers have good time-domain performance and frequency-domain robustness. The main problem with this approach is that the resulting controllers are state-space controllers of high-order rather than low-order controllers with a fixed structure. Although one can reduce it or approximate it to one with a PID-like structure, it is not so far the reduced-order controller.

The main advantage of neural networks is that it is possible to train a neural network to perform a particular function by adjusting the values of connecting weights between elements. Due to the learning ability of the artificial neural networks, a number of studies for PID controller design have successively been proposed by using neural networks (Omatu, *et al*., 1990; Chen and Chang, 1996).

The usual idea using neural networks within PID controller is to tune the PID parameters. The structure of controller itself is still a PID controller's. Some other ideas of PID-like neural networks are also proposed (Hemerly and Nascimento, 1999), but the design and analysis procedures are complex and difficult. From these factors, the specifications required for real applications of control theories are that the control structures and algorithms should be simple enough to be implemented and understood. The control algorithm should have following features such as learning ability, flexibility, robustness and nonlinearity. The need to deal with systems that include uncertainties and unknown nonlinearities, operating in highly uncertain environments, has attracted a lot of research activities mainly through a neuro-control approach.

In this paper we develop a PID-like Neural Network Controller (PIDNNC). The use of a PID structure in the way eliminates networks design problems such as the choice of network topology, (i.e., the number of hidden units) and reduces the sensitivity of the network to the initial values of the weights. It is hoped that the combination will take the advantage of simplicity of PID control and the neural networks' powerful capability of self-learning and tackling non-linearity. The paper is organized as follows. In Section 2 the proposed PIDNNC structure is described. Then, we derive in Section 3 updating weights rules of neural network controller according to the improved resilient back-propagation algorithm. The stability analysis of closed-loop system is discussed in Section 4. In Section 5 we give a numerical application to a double inverted-pendulum system by using of the proposed controller. The comparison of control results without and with disturbance between PIDNNC and conventional controller are also given. Conclusions are given in Section 6.

2. STRUCTURE DESIGN OF THE PIDNNC

The PIDNNC is composed of input layer, hidden layer and output layer as shown in Fig. 1. In input layer there are s input neural nodes, where s is the number of measurable output variables of controlled system. In hidden layer there are no more than 3 neural nodes: we call them integral node a_1, proportional node a_2 and derivative node a_3, respectively. In output layer there is only one neural node which outputs the control value. The activation functions in hidden layer and output layer are all linear functions. Nodes in input layer and hidden layer are full connected. The integral node a_1 in hidden layer has characteristic of output feedback, which is realized by delaying the output of a_1 with a unit sampling period and then feeding it back to the weighted sum node of a_1. The derivative node a_3 in hidden layer has characteristic of activation feedback, which is realized by negatively feeding back the output of weighted sum node of a_3 with a unit sampling period delay to the place of the input of the node a_3. The proportional node a_2 in hidden layer is a general node without any feedback. According to the structure described above and as shown in Fig.1, the output of nodes in hidden layer $a_i(k)$, $(i = 1,2,3)$ at sampling times k may be calculated respectively as follows.

$$\begin{aligned} a_1(k) &= f\left(\sum_{j=1}^{s} w_{1j}(k)e_j(k) + a_1(k-1)\right) \\ &= \sum_{j=1}^{s} w_{1j}(k)e_j(k) + a_1(k-1) \end{aligned} \tag{1}$$

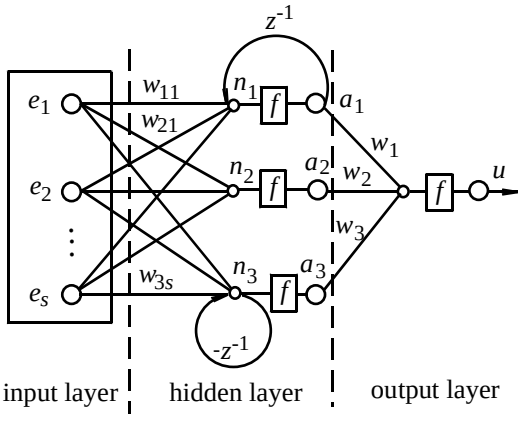

Fig. 1. The structure of the PIDNNC proposed.

$$a_2(k) = f\left(\sum_{j=1}^{s} w_{2j}(k)e_j(k)\right) = \sum_{j=1}^{s} w_{2j}(k)e_j(k) \quad (2)$$

$$a_3(k) = f\left(\sum_{j=1}^{s} w_{3j}(k)e_j(k) - \sum_{j=1}^{s} w_{3j}(k-1)e_j(k-1)\right) = \sum_{j=1}^{s} w_{3j}(k)e_j(k) - \sum_{j=1}^{s} w_{3j}(k-1)e_j(k-1) \quad (3)$$

The final output of neural controller is

$$u(k) = \sum_{i=1}^{3} w_i(k)a_i(k) \quad (4)$$

Taking z^{-1} as an unit delay factor, we get $a(k-1) = a(k)\cdot z^{-1}$, $e(k-1) = z^{-1}\cdot e(k)$. From equations (1)-(4) of network, we may see that due to the output feedback, the equation (1) of the first neural node in hidden layer can be written as

$$a_1(k) = \frac{\sum_{j=1}^{s} w_{1j}(k)e_j(k)}{1-z^{-1}} \quad (5)$$

Equation (5) has the relationship of integral, so we call $a_1(k)$ is an integral node. On the other hand, due to the activation feedback, the equation (3) of the third neural node in hidden layer can be rewritten as

$$a_3(k) = \left[\sum_{j=1}^{s} w_{3j}(k)e_j(k)\right](1-z^{-1}) \quad (6)$$

Equation (6) has the relationship of differential, so we call $a_3(k)$ is a differential node.

The basic idea of PID is that the control action *u* should be proportional to the error, the integral of the error over time, and the temporal derivative of the error. From equations (4)-(6) we can see that when the neural controller adjusts input/output relation of the system, it shows the characteristic of proportional (the second node a_2) + differential (the third node a_3) + integral (the first node a_1), and it becomes actually a PID-like neural network controller. The conventional PID controller is usually only suitable for the system with single-input/single-output (SISO) system and without consideration of environmental and disturbing factors. The PIDNNC proposed here is hoped to be suitable for multi-input/multi-output (MIMO) system and can regulate system parameters adaptively and on-line under the effect of uncertain and un-modeled factors.

Conventional neural networks can have several layers. There are usually 2 main types of multi-layer networks – feed forward and recurrent, in which, recurrent networks have at least on feedback loop. This means an output of a layer feeds back to any proceeding layer. This gives the network partial memory. This makes recurrent networks powerful in approximating function depending on time. The PIDNNC proposed has two layers, in which the hidden layer plays a multi-input PID function so it has the explicit signification of PID, while the output layer plays the rule of sum. Although two layers of the network use the linear activation function, using of the non-normal mix local recurrent network, an activation feedback on the integral node and an output feedback on the derivative node in hidden layer, makes the PIDNNC have the features which are exist neither in the normal linear networks nor in conventional local recurrent networks and their PID-like controller. On the other hand, because of being a network, PIDNNC is also different from the PID controller by using neural networks tuning.

3. UPDATING WEIGHTS RULES OF THE PIDNNC

Like general controllers design procedure, the objective of the PIDNNC is to minimize the error between the closed-loop output of controlled system and the given command input. The structure of closed-loop control system using neural network controller is given in Fig. 2. The output error of closed-loop system is defined as

$$e_j(k) = r_j(k) - y_j(k)\ ,\ j = 1,\ 2,\ \ldots,s \quad (7)$$

It is assumed that $y_j(k)$ $(j = 1,2,\cdots s)$ are output variables of system that are measurable directly by sensor, where j is the number of measurable output variables, and $r_j(k)$, $(j = 1,2,\cdots s)$ are given command inputs. Then cost function to output $y_j(k)$ at k is defined as

$$J_j(k) = \frac{1}{2}\left[r_j(k) - y_j(k)\right]^2 \quad (8)$$

The total cost function at k is

$$J(k) = \sum_{j=1}^{s} J_j(k) \quad (9)$$

Equation (9) is the final cost function used in every sampling period.

Without loss of generality, we use $J(k)$ to derive the rules of updating weights of neural network controller. The rules of updating weights are obtained by means of minimizing cost function $J(k)$. Here we adopt the concept of gradient descent.

$$\frac{\partial J(k)}{\partial w_{ij}(k-1)} = \sum_{j=1}^{s}\left[\frac{\partial J_j(k)}{\partial y_j(k)}\cdot\frac{\partial y_j(k)}{\partial u(k-1)}\right]\cdot\frac{\partial u(k-1)}{\partial w_{ij}(k-1)}$$

$$\triangleq -\sum_{j=1}^{s}\left\{e_j(k)\cdot\operatorname{sgn}\left[\frac{y_j(k)-y_j(k-1)}{u(k-1)-u(k-2)}\right]\right\}\cdot e_j(k-1)$$

Considering the complexity and uncertainty of functional relation between output $y_j(k)$ and input $u(k-1)$, we approximate the partial derivative $\frac{\partial y_j(k)}{\partial u(k-1)}$ by using concept of difference, then

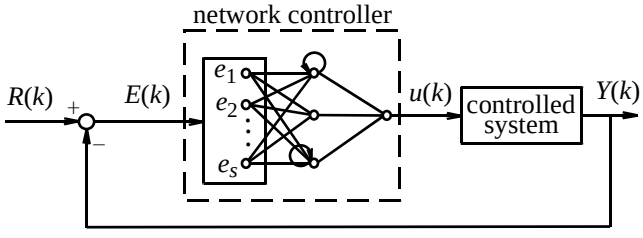

Fig. 2. The structure of close-loop control system.

$\frac{\partial y_j(k)}{\partial u(k-1)} \approx \frac{\Delta y_j(k)}{\Delta u(k-1)} = \frac{y_j(k)-y_j(k-1)}{u(k-1)-u(k-2)}$. We also adopt to use *sign* function rather than considering the actual result of calculation. That is to say, we replace the concept of resilient back-propagation algorithm result of $\frac{y_j(k)-y_j(k-1)}{u(k-1)-u(k-2)}$ with ± 1 according to its positive-negative characteristic. In such way we not only avoid the problem of stopping updating weights owing to gradient decent being too small, but also greatly simplify the calculation and satisfy the request of on-line control. Therefore the final rule of updating weights in hidden layer is

$$\Delta w_{ij}(k-1) = -\eta_{ij}\frac{\partial J(k)}{\partial w_{ij}(k-1)}$$

$$= \eta_{ij}\cdot\sum_{j=1}^{s}\left\{e_j(k)\cdot \operatorname{sgn}\frac{\Delta y_j(k)}{\Delta u(k-1)}\right\}\cdot\cdot e_j(k-1)$$

$$i = 1,2,3;\ j = 1,2,\cdots,s \qquad (10)$$

The weights of output layer are

$$w_i = 1, \quad i = 1,2,3 \qquad (11)$$

4. STABILITY ANALYSIS OF CLOSED-LOOP SYSTEM

In order to analyze the stability of closed-loop system, we define a Lyapunov function as

$$V(k) = \frac{1}{2}\sum_{j=1}^{s} e_j^2(k) \qquad (12)$$

Thus the change of Lyapunov function is

$$\Delta V(k) = V(k+1) - V(k) = \frac{1}{2}\sum_{j=1}^{s}\left[e_j^2(k+1) - e_j^2(k)\right] \qquad (13)$$

Note that (Ku and Lee, 1992)

$$e_j(k+1) = e_j(k) + \Delta e_j(k) = e_j(k) + \left[\frac{\partial e_j(k)}{\partial w_{ij}(k)}\right]^T \Delta w_{ij}(k) \qquad (14)$$

in which $\Delta w(k)$ represents the change of weights, then

$$\Delta V(k) = \frac{1}{2}\sum_{j=1}^{s}\left\{\left[e_j(k+1) - e_j(k)\right]\cdot\left[e_j(k+1) + e_j(k)\right]\right\}$$

$$= \frac{1}{2}\sum_{j=1}^{s}\left\{\Delta e_j(k)\left[2e_j(k) + \Delta e_j(k)\right]\right\}$$

$$= \sum_{j=1}^{s}\left\{\Delta e_j(k)\left[e_j(k) + \frac{1}{2}\Delta e_j(k)\right]\right\} \qquad (15)$$

On the one hand, we have

$$\Delta w_{ij}(k) = -\eta_{ij}\frac{\partial J(k)}{\partial w_{ij}(k)} = -\eta_{ij}\sum_{j=1}^{s}\left[e_j(k)\cdot\frac{\partial e_j(k)}{\partial w_{ij}(k)}\right] \qquad (16)$$

Then

$$\Delta e_j(k) = -\eta_{ij}\sum_{j=1}^{s}\left[e_j(k)\cdot\frac{\partial e_j(k)}{\partial w_{ij}(k)}\right]\cdot\left(\frac{\partial e_j(k)}{\partial w_{ij}(k)}\right) \qquad (17)$$

Let $\frac{\partial e_j(k)}{\partial w_{ij}(k)} = e_{jw}$. Put e_{jw}, equations (16) and (17) into equation (15), we can obtain

$$\Delta V(k) = \sum_{j=1}^{s}\left\{\eta\sum_{j=1}^{s}\left[e_j(k)\cdot e_{jw}\right]\cdot e_{jw}\left[-e_j(k) + \frac{1}{2}\eta_{ij}\sum_{j=1}^{s}\left[e_j(k)\cdot e_{jw}\right]e_{jw}\right]\right\}$$

$$= \sum_{j=1}^{s}\left\{\eta\left(\sum_{j=1}^{s}\left[e_j(k)\cdot e_{jw}\right]\right)^2\left(\frac{-e_j(k)e_{jw}}{\sum_{j=1}^{s}\left[e_j(k)\cdot e_{jw}\right]} + \frac{1}{2}\eta_{ij}e_{jw}^2\right)\right\} \qquad (18)$$

from (18) we can see that because $\left(\sum_{j=1}^{s}\left[e_j(k)\cdot e_{jw}\right]\right)^2 > 0$, under the action of control $u(k)$, the sufficient condition of closed-loop system $\Delta V(k) < 0$ becomes $\eta_{ij}\cdot(-\lambda_j + \frac{1}{2}\eta_{ij}e_{jw}^2) < 0$. Because λ_j can be either positive or negative value , when $\lambda_j > 0$, system stable condition is $0 < \eta_{ij} < \frac{2\lambda}{e_{jw}^2}$, while $\lambda_j < 0$, system stable condition becomes $-\frac{2|\lambda|}{e_{jw}^2} < \eta_{ij} < 0$. On the other hand

$$\frac{\partial e_j(k)}{\partial w_{ij}(k)} = e_{jw} = -\frac{\partial y_j(k)}{\partial u(k)}\cdot\frac{\partial u(k)}{\partial w_{ij}(k)}$$

$$= -\frac{1}{\frac{\partial u(k)}{\partial y_j(k)}}\cdot e_j(k) = \frac{1}{\frac{\partial u(k)}{\partial e_j(k)}}\cdot e_j(k) = \frac{e_j(k)}{\sum_{i=1}^{3} w_{ij}(k)}$$

Let

$$e_{\max}(k) = \max\left(\sum_{j=1}^{s}\frac{e_j^2(k)}{\sum_{i=1}^{3} w_{ij}}\right) \quad , \quad w_{\min}(k) = \min\left(\left|w_{ij}(k)\right|\right)$$

we can have

$$\frac{2|\lambda|}{e_{jw}^2} = \frac{\left|\sum_{i=1}^{3} w_{ij}(k)\right|}{\sum_{j=1}^{s}\frac{e_j^2(k)}{\sum_{i=1}^{3} w_{ij}}} < \frac{w_{\min}(k)}{e_{\max}(k)}$$

The value range of leaning rate that can guarantee the control system stability is

$$-\frac{w_{\min}(k)}{e_{\max}(k)} < \eta < \frac{w_{\min}(k)}{e_{\max}(k)} \qquad (19)$$

5. NUMERICAL APPLICATION TO DOUBLE INVERTED-PENDULUM

The controller proposed is applied into a typical self-unstable inverted pendulum system in order to verify its effectiveness and advantage. As we know, the inverted pendulum system is typically used to benchmark new control techniques, as it's a high non-linear unstable system. The system model used in the simulation is based on the double inverted-pendulum GIP-200-L, which is produced by Googol Technology (Shenzhen) Ltd, is composed of three parts. The first part is an equipment including a cart, a drive and links. The second one is an electrical control box with motor drive and interface circuit.

The third one is a general 4-axis movement control board fixed in a PC.

As it is well known, variables in a double inverted-pendulum system are: x, position of cart; θ_1, offset angle of link 1(short link); θ_2, offset angle of link 2(long link). The inverted pendulum has 3 outputs, in order to have full state feedback control 3 PID controllers would have to be used. Neural networks have a big advantage here due to their parallel nature. One PIDNNC could be used instead of 3 PID's. The initial weight values of neural network are obtained by Linear Quadratic Regulation (LQR) algorithm whose control law is

$$U = -KX = -(k_1 x + k_2 \dot{x} + k_3 \theta_1 + k_4 \dot{\theta}_1 + k_5 \theta_2 + k_6 \dot{\theta}_2)$$

which is derived by LQR, and

$$K = \begin{bmatrix} k_x & k_{\dot{x}} & k_{\theta_1} & k_{\dot{\theta}_1} & k_{\theta_2} & k_{\dot{\theta}_2} \end{bmatrix}$$

$$= \begin{bmatrix} 16.03 & 18.13 & 130.75 & 3.55 & -218.26 & -36.12 \end{bmatrix}$$

Here we use such K as the initial weight values of the neural network controller. The structure of neural network controller is shown in Fig.3 in which there are 3 nodes in input layer, 2 nodes in hidden layer and 1 node in output layer. Notice that there are only 2 nodes in hidden layer, which are proportional node and derivative node, and integral node a_1 (as be shown with dashed line in the Fig. 3 needs not be used in the given application. The inputs to neural network are

$$e_1 = -x \quad , \quad e_2 = -\theta_1 \, , \quad e_3 = -\theta_2 \, .$$

According to the variables physical meanings of LQR, initial corresponding weight values of PIDNN are

$$w_{21}(0) = k_x = 16.03 \, , \quad w_{31}(0) = k_{\dot{x}} / T = 3626$$

$$w_{22}(0) = k_{\theta_1} = 130.75 \, , \quad w_{32}(0) = k_{\dot{\theta}_1} / T = 710$$

$$w_{23}(0) = k_{\theta_2} = -218.26 \, , \quad w_{33}(0) = k_{\dot{\theta}_2} / T = -7224$$

where T is the sampling period and $T = 0.005s$.

Control law is

$$u(k) = w_2(k) \cdot \left[\sum_{j=1}^{s} w_{2j}(k) e_j(k) \right]$$

$$+ w_3(k) \cdot \left[\sum_{j=1}^{s} w_{3j}(k) e_j(k) - \sum_{j=1}^{s} w_{3j}(k-1) e_j(k-1) \right]$$

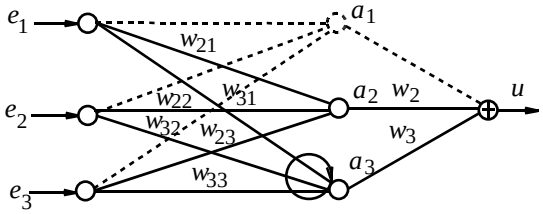

Fig. 3. Structure of PIDNNC used in a double inverted-pendulum.

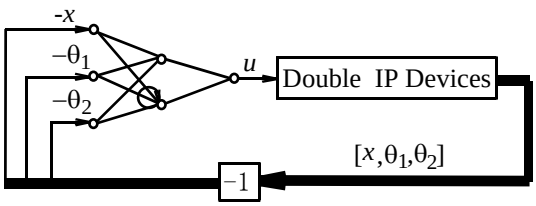

Fig. 4. Control system of double linear inverted-pendulum devices.

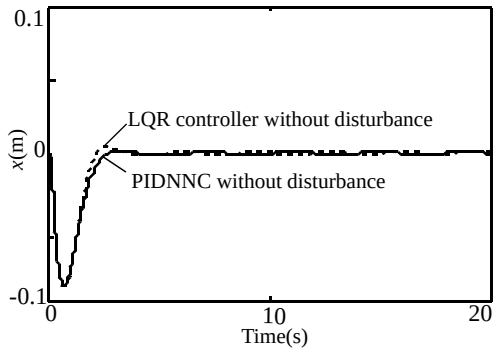

Fig. 5(a) Comparison of cart displacement x.

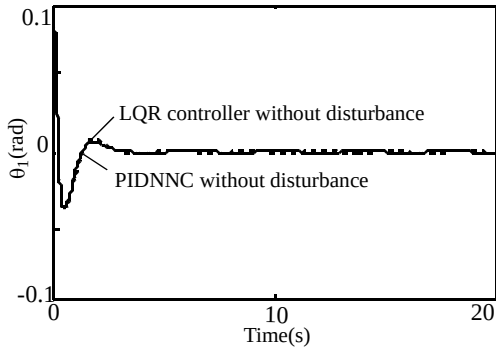

Fig. 5(b) Comparison of angle θ_1 response.

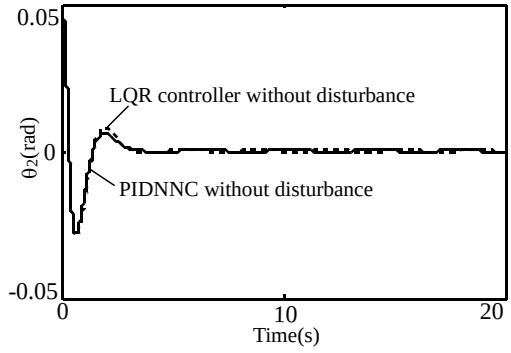

Fig. 5(c) Comparison of angle θ_2 response.

The whole control system is shown in Fig.4.

Two kinds of experiments were done: 1) Comparable experiment of LQR and PIDNNC under no external disturbance. 2) Comparable experiment of anti-disturbance ability of two kinds of controllers under external disturbance. Disturbance given in system experiment is a pulse signal with amplitude of 0.0001 in every four sampling periods. Fig. 5 shows comparable results of displacement x, angle θ_1 and angle θ_2 respectively controlled by LQR and PIDNNC under no external disturbance.

θ_1 In fact the main difference of two kinds of controllers is that PIDNNC has an extra ability of adjusting controller parameters (network weights) on-line according to measured system errors in each sample period. It can be seen from Fig.5 that under the situation without disturbance, the PIDNNC results are slightly better than that of LQR controller but the difference between them is small. All experiments must be done under the condition of the

system stability, which can be satisfied by properly adjusting learning rate.

Comparable results of displacement x, angle θ_1 and angle θ_2 respectively controlled separately by LQR and PIDNNC under disturbance are given in Fig. 6, from which it can be seen obviously that when giving external disturbance to system, two controllers' results demonstrate distinct different, especially on the displacement variable. The displacement response controlled by LQR changes a lot. This means that links move greatly in horizontal direction compared with the situation without disturbance. While the same variable controlled by

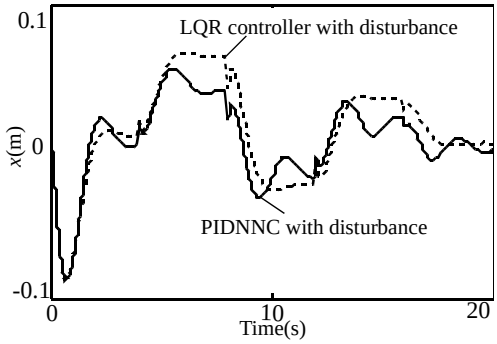

Fig. 6(a) Comparison of cart displacement x.

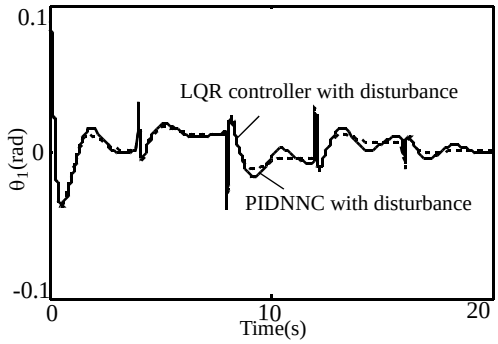

Fig. 6(b) Comparison of angle θ_1 response.

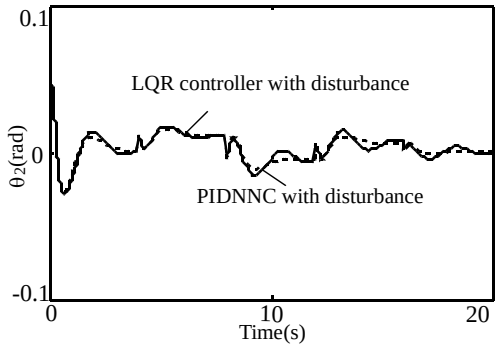

Fig. 6(c) Comparison of angle θ_2 response.

PIDNNC which can adjust weights on-line, the response has stronger capacity of adapting to change than it has under control of LQR.

6. CONCLUSIONS

An adaptive PID-like neural network controller suitable for multivariable nonlinear system is proposed in this paper. Rules of updating weights for the controller are derived. The effect of learning rates to stability of closed-loop system is analyzed. The actual numerical experimental implementation on a double linear inverted-pendulum based on the control strategy proposed and compared with the effect of the LQR control strategy demonstrates that the controller proposed has the advantages of simple structure, easy-design and higher response performance.

REFERENCES

Bao, J., J. F. Forbes, and P. J. McLellan (1999). Robust multiloop PID controller design: a successive semidefinite programming approach, *Ind. Eng. Chem. Res.*, **vol. 38**, pp. 3407-3413.

Chen C. L., F. Y. Chang(1996), Design and anakysis of neural/fuzzy variable structural PID control systems, *IEE Proceedings-Control Theory Application*, **vol.143, no.2**, pp.200-208

Chen, B. S., Y. M. Chiang, and C. H. Lee (1995). A genetic approach to mixed H_2/H_∞ optimal PID control, *IEEE Control System Magazine*, **vol. 15**, pp. 51-56.

Grimble, M. J. (1990). H_∞ controllers with a PID structure, *Trans. ASME J. Dynamic System Measurement Control*, **vol. 112**, pp. 325-330.

Hemerly E. M. and C. L. Nascimento Jr(1999), An NN-based approach for tuning servocontrollers, *Neural Networks*, **vol. 12**, pp.513-518

Ho, W. K., C. C. Hang, and L. S. Cao (1995). Tuning of PID controllers based on gain and phase margin specifications, *Automatica*, **vol. 31, no. 3**, pp. 497-502.

Ku C. C., and K. Y. Lee (1992). Diagonal Recurrent Neural Networks for Nonlinear System Control, *Proc. IJCNN International Joint Conference on Neural Networks*, **vol. I**, pp. 315-320, Baltimore, MD.

Morari, M. and E. Zafiriou (1989). *Robust Process Control.* Englewood Cliffs NJ.: Prentice-Hall.

Omatu S., M. Khalid and R. Yusof(1996). *Neuro-control and its application*, Springer, London

Zhuang, M. and D. P. Atherton (1993). Automatic tuning of optimum PID controllers, *IEE Proc.on Control and Applications*, **vol. 140**, pp. 216-224.

Ziegler, J. G. and N. B. Nichols (1942). Optimum settings for automatic controllers, *Trans. ASME*, **vol. 62,** pp. 759-768.