PID-Like Neural Network Nonlinear Adaptive Control for Uncertain Multivariable Motion Control Systems

S. Cong and Y. Liang

***Abstract*—A mix locally recurrent neural network was used to create a proportional-integral-derivative (PID)-like neural network nonlinear adaptive controller for uncertain multivariable single-input/multi-output system. It is composed of a neural network with no more than three neural nodes in hidden layer, and there are included an activation feedback and an output feedback, respectively, in a hidden layer. Such a special structure makes the exterior feature of the neural network controller able to become a P, PI, PD, or PID controller as needed. The closed-loop error between directly measured output and expected value of the system is chosen to be the input of the controller. Only a group of initial weights values, which can run the controlled closed-loop system stably, are required to be determined. The proposed controller can update weights of the neural network online according to errors caused by uncertain factors of system such as modeling error and external disturbance, based on stable learning rate. The resilient back-propagation algorithm with sign instead of the gradient is used to update the network weights. The basic ideas, techniques, and system stability proof were presented in detail. Finally, actual experiments both of single and double inverted pendulums were implemented, and the comparison of effectiveness between the proposed controller and the linear optimal regulator were given.**

***Index Terms*—Neural network, nonlinear adaptive control, proportional-integral-derivative (PID), single-input/multi-output (SIMO), uncertain multivariable system.**

I. Introduction

In industry applications, proportional-integral-derivative (PID) control is a very popular control strategy due to its simple architecture and easy tuning. Despite their widespread 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: Ziegler–Nichol method [1], internal-model-control-based method [2], optimization method [3], and gain-phase margin method [4]. For single-input/single-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, particularly the single-input/multi-output (SIMO) plants, however, show significant internal interaction. Since the tuning rules are developed for SISO system, their applications to such "true" multivariable plants are ineffective. Furthermore, 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. Second, assume that the controller has a PID structure, and find the PID parameters by using some well-known optimization methods, e.g., H_∞ [5], mixed H_2/H_∞ [6], and semidefinite programming approaches [7]. 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 or approximate it with a PID-like structure, it is not so far the reduced-order controller.

With advantages like the abilities of adaptive, parallel, and fault tolerance, artificial neural networks plays an important role in many fields. From the end of the 1980s, introductions to neural network control are successively given. Feedforward controllers in which the neural network learns to mimic the inverse of the plant are intensively discussed by [8]–[11]. Currently, many different neural network control methods are proposed which include the adaptive neural networks to model the plant [12], the analytical redundancy method using neural network modeling of the induction motor in vibration spectra for machine fault detection and diagnosis [13], the method of in direct adaptive control using neural network model of the process and its inverse [14], and adaptive controller with neural network uncertain compensation [15].

In the real world, the system parameters are often time varying, so the conventional PID and PID-like control scheme becomes not flexible in dealing with the systems in which there exist uncertain factors such as modeling error and external disturbance. In addition, some modern control approaches are also proposed, 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

Manuscript received May 8, 2008; revised October 7, 2008, December 12, 2008, and January 29, 2009. First published April 7, 2009; current version published September 16, 2009. This work was supported by the National Science Foundation of China under Grant 60774098.

The authors are with the Department of Automation, University of Science and Technology of China, Hefei 230027, China (e-mail: scong@ustc.edu.cn; sunnylg@mail.ustc.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIE.2009.2018433

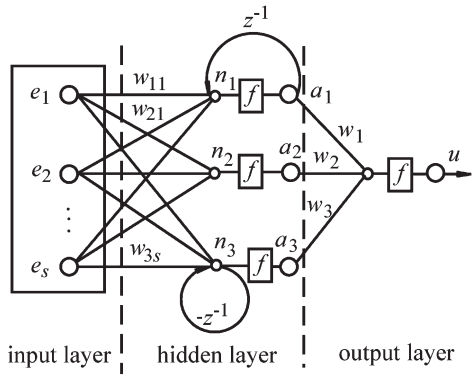


Fig. 1. Structure of the neural network controller.

implemented and understood. The control algorithms should have the following features: learning ability, flexibility, robustness, and nonlinearity. A neural network-based PID real-time tuning method was proposed and used in practical motor control [16], but the method is only suitable for SISO system. The treatment of the SIMO case is practically meaningful, since the plant to be controlled has only one actuator with two or more sensors, which commonly appear in real control applications to get the better control performance by putting extra sensors. Some results on performance limitations of a single-input two-output system, special form of the SIMO system, can be found in [17]. Early in 2005, Cong *et al.* [18], [19] proposed a novel multivariable PID-like neural network control scheme for SIMO system, and then discussed its design method in 2006, but the sufficient stability analysis of the close-loop system is still lacking.

The main contribution of this paper is that based on a mix locally recurrent neural network proposed in the [18], which was used to create a PID-like neural network adaptive controller (PIDNNC) for multivariable SIMO systems with uncertainty, the sufficient condition of close-loop stability is proved, and two typical actual motion control system experiments are implemented.

This paper is organized as follows. A novel PID-like neural network controller scheme for multivariable SIMO systems is first described in Section II. Then, we derive in Section III updating weights rules of neural network controller according to the improved resilient back-propagation algorithm. The stability of closed-loop system is proven in Section IV, in which the Lyapunov stability theorem is applied to obtain the stable learning rate of controller. In Section V, we give numerical applications to both single and double inverted pendulum systems by using of the proposed controller. The comparison of control results without and with disturbance between PIDNNC and linear optimal controller are also given. Finally, conclusions are given in Section VI.

II. Novel PID-Like Neural Network Controller

The neural network controller we proposed is composed of an input layer, a 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 the hidden layer, there are no more than three neural nodes: we call them integral node a_1, proportional node a_2, and derivative node a_3, respectively. In the output layer, there is only one neural node, which outputs the control value. The activation functions in the hidden layer and output layer are all linear functions. Nodes in input layer and hidden layer are fully connected. The integral node a_1 in hidden layer has an 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 n_1. The derivative node a_3 in hidden layer has an activation feedback, which is realized by negatively feeding back the output of weighted sum node n_3 with a unit sampling period delay to the place of the input of the node n_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:

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

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

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

The final output of neural controller is

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

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 (1)–(4) and the output feedback, (1) of the first neural node in hidden layer can be written as

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

from which one can see that (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, (3) of the third neural node in hidden layer can be rewritten as

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

from which one can see that (6) has the relationship of differential, so we call $a_3(k)$ a differential node.

On the one hand, 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 (4)–(6), one 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 controller. The conventional PID controller is usually only suitable for the system with SISO system and without consideration of environmental and disturbing factors, while the PID-like neural network controller proposed in this paper is hoped to be suitable for SIMO system and can online regulate controller's parameters adaptively under the effect of uncertain and unmodeled factors.

On the other hand, conventional neural networks can have several layers. There are usually two main types of multilayer networks: feedforward and recurrent ones, in which, recurrent networks have at least one 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 neural network controller proposed has two layers, in which the hidden layer play does a multi-input PID function so it has the explicit signification of PID, while the output layer plays the rule of sum, so in the controller proposed the weights of output layer are set as

$$w_i = 1, \qquad (i = 1, 2, 3). \tag{7}$$

Although two layers of the network controller here use both the linear activation functions, the mix local recurrent network used in hidden layer makes the PIDNNC have the properties there do not exist in the general linear networks or conventional local recurrent networks.

Just because it is composed of a neural network with no more than three neural nodes in hidden layer, and in the same hidden layer there are an activation feedback and an output feedback, respectively, such a special structure of the proposed controller makes it a P, PI, PD, or PID controller as needed. In such a way, the proposed controller has both the property of simplicity of PID control and, at the same time, the neural networks' powerful capability of self-learning.

III. Updating Weights Rules of PIDNNC and Its Initial Weights Design

The structure of closed-loop control system with the neural controller proposed is shown in Fig. 2. Like general controllers, the goal of the control system with the PIDNNC is to minimize the error between the given command input and the output of closed-loop system. Because physical units used for each controlled variable in practical system are different (for example the meter for the distance and meter per second for the velocity), the cost function $J_j(k)$ to different controlled variables $y_j(k)$ $(j = 1, 2, \ldots, s)$ at k should be defined, respectively, and the total cost function $J(k)$ is the sum of all defined cost functions.

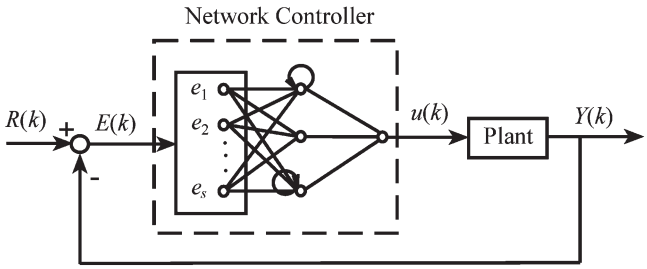


Fig. 2. Structure of the control system with network controller proposed.

The output error of closed-loop system is defined as

$$e_j(k) = r_j(k) - y_j(k), \qquad j = 1, 2, \ldots, s. \tag{8}$$

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

$$J_j(k) = \frac{1}{2}\left(r_j(k) - y_j(k)\right)^2 = \frac{1}{2}\left(e_j(k)\right)^2. \tag{9}$$

The total cost function $J(k)$ at k is

$$J(k) = \sum_{j=1}^{s} J_j(k). \tag{10}$$

Equation (10) is the final cost function used in every sampling period.

Without loss of generality, the rules of updating weights are obtained by means of minimizing the cost function $J(k)$. Here, we adopt the method of gradient descent to obtain updating rules. The rule of updating weights in the hidden layer is

$$\begin{aligned}\frac{\partial J(k)}{\partial w_{ij}(k)} &= \left(\sum_{j=1}^{s}\left(\frac{\partial J_j(k)}{\partial y_j(k)}\frac{\partial y_j(k)}{\partial u(k)}\right)\right)\frac{\partial u(k)}{\partial w_{ij}(k)} \\ &= -\left(\sum_{j=1}^{s} e_j(k)\frac{\partial y_j(k)}{\partial u(k)}\right)e_j(k).\end{aligned} \tag{11}$$

Equation (11) is obtained as follows: From (4), one can see that u is independent of j, so $\partial u(k)/\partial w_{ij}(k)$ can be placed the out side of the $\sum_{j=1}^{s}(\bullet)$, and for a fixed i, then according to (1)–(3): One has $(\partial u(k)/\partial u_{ij}(k)) = (\partial a_i(k)/\partial w_{ij}(k)) - e_j(k)$.

Considering the complexity and uncertainty of functional relation between the output $y_j(k)$ and the input $u(k)$, the partial derivative $\partial y_j(k)/\partial u(k)$ is approximated by using the concept of difference, that is as follows: $(\partial y_j(k)/\partial u(k)) \approx (\Delta y_j(k)/\Delta u(k)) = ((y_j(k) - y_j(k-1))/(u(k) - u(k-1)))$. Moreover, in order not to consider the actual result of calculation the same concept using in the resilient backpropagation algorithm with sign instead of the gradient value is used to calculate the $\Delta y_j(k)/\Delta u(k)$. That is to say, the result of $\Delta y_j(k)/\Delta u(k)$ is set to be ± 1 according to the positive-negative characteristic of $y_j(k) - y_j(k-1)/u(k) - u(k-1)$. In such a way, we not only avoid the problem of stopping

updating weights owing to the gradient decent being too small, but also greatly simplify the calculation and satisfy the request of online control. Therefore, (11) can be rewritten as

$$\frac{\partial J(k)}{\partial w_{ij}(k)} \approx -\left(\sum_{j=1}^{s} e_j(k)\mathrm{sgn}\left(\frac{\Delta y_j(k)}{\Delta u(k)}\right)\right) e_j(k). \quad (12)$$

According to (12), the final rule of updating weights in hidden layer is

$$\begin{aligned}\Delta w_{ij}(k) =& -\eta_{ij}(k)\frac{\partial J(k)}{\partial w_{ij}(k)} \\ =& \eta_{ij}(k)\left(\sum_{j=1}^{s} e_j(k)\mathrm{sgn}\left(\frac{\Delta y_j(k)}{\Delta u(k)}\right)\right) e_j(k) \\ & i=1,2,3; \quad j=1,2,\ldots,s \end{aligned} \quad (13)$$

in which $\eta_{ij}(k)$ is the learning rate of the hidden layer.

The weights in the PIDNNC proposed in this paper need not be trained *a prior*. The only thing requested is that all of its initial weights are selected which can stabilize the controlled system. Just because the special structure of the controller proposed, there exist corresponding relations between the weights of PIDNNC and those of conventional controllers. In practice, some simple conventional controllers, such as PID or linear quadratic regulator (LQR) ones, can be first designed for the controlled system and make it to be stable, whose parameters can be then used as the initial weights in PIDNNC according to correspond relationships between them. Equation (13) is used to online adjust weights to adapt parameters change caused by the effects of nonlinearity and uncertainty to get better control performance of the system and stronger robustness to some external disturbance. In fact, in the case without (13) or $\Delta w_{ij}(k)=0$, the PIDNNC is just the conventional controller used to design the initial weights.

IV. Stability of Closed-Loop System

Theorem 1: If all the learning rate $\eta_{ij}(k)$ at the sampling time k have the same value $\eta(k)$, and $\eta(k)$ satisfies the following (14)–(16), then close-loop control system with the PIDNNC in Fig. 2 is stable if $(\sum_{j=1}^{s}\sum_{i=1}^{3}(e_j(k)/w_{ij}(k)))q(k)<0$ then

$$0<\eta(k)\le -2\frac{\sum_{j=1}^{s}\sum_{i=1}^{3}\frac{e_j(k)}{w_{ij}(k)}}{\left(\sum_{j=1}^{s}\left(\sum_{i=1}^{3}\frac{1}{w_{ij}(k)}\right)^2\right)q(k)} \quad (14)$$

if $(\sum_{j=1}^{s}\sum_{i=1}^{3}(e_j(k)/w_{ij}(k)))q(k)\ge 0$ then

$$-2\frac{\sum_{j=1}^{s}\sum_{i=1}^{3}\frac{e_j(k)}{w_{ij}(k)}}{\left(\sum_{j=1}^{s}\left(\sum_{i=1}^{3}\frac{1}{w_{ij}(k)}\right)^2\right)q(k)} \le \eta(k)\le 0 \quad (15)$$

where

$$q(k)=\left(\sum_{j=1}^{s} e_j(k)\mathrm{sgn}\left(\frac{\Delta y_j(k)}{\Delta u(k)}\right)\right)\left(\sum_{j=1}^{s} e_j^2(k)\right). \quad (16)$$

Proof: In order to stabilize the closed-loop control system with the PIDNNC, the Lyapunov stability theorem is used here. The Lyapunov function is defined as

$$V(k)=\frac{1}{2}\sum_{j=1}^{s} e_j^2(k). \quad (17)$$

Thus, the change of the Lyapunov function is

$$\Delta V(k)=V(k+1)-V(k)=\frac{1}{2}\sum_{j=1}^{s}\left(e_j^2(k+1)-e_j^2(k)\right). \quad (18)$$

According to the structure of PIDNNC in Figs. 1 and 2, one has

$$\Delta e_j(k)=\sum_{l=1}^{s}\left(\sum_{i=1}^{3}\left(\frac{\partial e_j(k)}{\partial w_{il}(k)}\Delta w_{il}(k)\right)\right). \quad (19)$$

From (1)–(3), one can get

$$\frac{\partial e_j(k)}{\partial w_{il}(k)}=\frac{\frac{\partial a_i(k)}{\partial w_{il}(k)}}{\frac{\partial a_i(k)}{\partial e_j(k)}}=\frac{e_l(k)}{w_{ij}(k)}. \quad (20)$$

According to (13) and (20), (19) can be rewritten as

$$\begin{aligned}\Delta e_j(k)=\sum_{l=1}^{s}\Bigg(\sum_{i=1}^{3}\Bigg(&\frac{e_l(k)}{w_{ij}(k)}\eta_{il}(k) \\ &\times\left(\sum_{j=1}^{s} e_j(k)\mathrm{sgn}\left(\frac{\Delta y_j(k)}{\Delta u(k)}\right)\right)e_l(k)\Bigg)\Bigg). \end{aligned} \quad (21)$$

Let

$$p(k)=\sum_{j=1}^{s}\left(e_j(k)\mathrm{sgn}\left(\frac{\Delta y_j(k)}{\Delta u(k)}\right)\right). \quad (22)$$

Then, (21) can be rewritten as

$$\Delta e_j(k)=\sum_{l=1}^{s}\left(\sum_{i=1}^{3}\left(\frac{p(k)e_l^2(k)}{w_{ij}(k)}\eta_{il}(k)\right)\right). \quad (23)$$

Let all the learning rate $\eta_{ij}(k)$ in the sampling time k have the same value $\eta(k)$, according to (23), one has

$$\begin{aligned}\Delta e_j(k)=&\sum_{i=1}^{3}\left(\frac{p(k)}{w_{ij}(k)}\sum_{l=1}^{s}\left(e_l^2(k)\right)\right)\eta(k) \\ =&\left(\sum_{i=1}^{3}\frac{1}{w_{ij}(k)}\right)\left(p(k)\sum_{j=1}^{s}\left(e_j^2(k)\right)\right)\eta(k). \end{aligned} \quad (24)$$

Let

$$q(k) = \left(p(k) \sum_{j=1}^{s} \left(e_j^2(k) \right) \right). \tag{25}$$

Then, (24) can be represented as

$$\Delta e_j(k) = \left(\sum_{i=1}^{3} \frac{1}{w_{ij}(k)} \right) q(k)\eta(k). \tag{26}$$

Since

$$e_j(k+1) = e_j(k) + \Delta e_j(k). \tag{27}$$

According to (26) and (27), (18) can be rewritten as

$$\begin{aligned}
\Delta V(k) &= \frac{1}{2} \sum_{j=1}^{s} \left(\left(e_j(k+1) + e_j(k) \right) \left(e_j(k+1) - e_j(k) \right) \right) \\
&= \frac{1}{2} \sum_{j=1}^{s} \left(\left(2e_j(k) + \Delta e_j(k) \right) \Delta e_j(k) \right) \\
&= \sum_{j=1}^{s} \left(e_j(k) \Delta e_j(k) + \frac{1}{2} \left(\Delta e_j(k) \right)^2 \right) \\
&= \sum_{j=1}^{s} \left(e_j(k) \left(\sum_{i=1}^{3} \frac{1}{w_{ij}(k)} \right) q(k)\eta(k) \right. \\
&\qquad \left. + \frac{1}{2} \left(\sum_{i=1}^{3} \frac{1}{w_{ij}(k)} \right)^2 q^2(k)\eta^2(k) \right) \\
&= \left(\sum_{j=1}^{s} \left(\sum_{i=1}^{3} \frac{e_j(k)}{w_{ij}(k)} \right) \right) q(k)\eta(k) \\
&\quad + \frac{1}{2} \left(\sum_{j=1}^{s} \left(\sum_{i=1}^{3} \frac{1}{w_{ij}(k)} \right)^2 \right) q^2(k)\eta^2(k).
\end{aligned} \tag{28}$$

According to Lyapunov stability theorem, only if $\Delta V(k) \leq 0$ in any sampling time k, the stability of close-loop control system with PIDNNC will be guaranteed. From (28), one can concludes the sufficient condition for $\Delta V(k) \leq 0$ is that $\eta(k)$ satisfies (14)–(16).

Remark: From conditions (14) and (15) on the one hand, one can see that the reciprocal of weights $w_{ij}(k)$ is needed to calculate. More deep analysis can deduce that no matter how weights are, too great or too small, $\eta(k)$ can be a reasonable value. In practice, usually $\eta(k)$ can be fixed as an enough small value, and it does not calculated at each sampling time in order to save the time. On the other hand, one can see also from (14) and (15) that $\eta(k)$ even can be selected the negative value. We will do the experiments in both situations.

V. Actual Applications on Inverted Pendulum Systems

In this section, real-time experiments of the control system with PIDNNC are implemented. They are applied to the stabilizing controls of single and double inverted pendulum systems.

A. Real-Time Implementation of PIDNNC Strategy

Under the description and design above, the proposed control system can be realized as the following procedure.

1) Determine the number of input layer nodes in PIDNNC according to the number of output variables of the controlled SIMO system.
2) Choose a suitable controller structure (P, PI, PD, or PID) through the selection of hidden layer nodes according to the practical control necessity.
3) Initialize the weights $w_{ij}(0)$ in hidden layer of PIDNNC by means of conventional control method such as LQR or PID, or neural network training based on the input and output data of the close-loop SIMO system controlled by any other stabilizing controller. The output layer weights $w_i = 1$, $(i = 1, 2, 3)$. Let $u(-1) = 0$, $a_1(-1) = 0$, $e_j(-1) = 0$, $w_{3j}(-1) = 0$.
4) Cascade the controlled SIMO system and PIDNNC to form the close-loop SIMO system showed in Fig. 2.
5) In every sample step $k(k \geq 0)$ when system is running, do the following.
   a) Read the system output $y_j(k)$ through A/D sensor.
   b) Calculate the error $e_j(k)$ as inputs of PIDNNC according to (8).
   c) Calculate the output of the hidden layer $a_1(k)$, $a_2(k)$, and $a_3(k)$ according to (1)–(3).
   d) Calculate control signal $u(k)$ according to (4).
   e) Determine the learning rate $\eta_{ij}(k)$ according to Theorem 1.
   f) Calculate $\Delta w_{ij}(k)$ according to (13), and get $w_{ij}(k+1)$ through $w_{ij}(k+1) = w_{ij}(k) + \Delta w_{ij}(k)$.
   g) Output control signal $u(k)$.
   h) $k = k + 1$.
   i) Return to step (1) in 4) and repeat the online operating.

B. Stabilizing Inverted Pendulum Control System With PIDNNC Controller

The Inverted Pendulum is one of the most important classical problems of Control Engineering. Broom Balancing (Inverted Pendulum on a cart) is a well-known example of nonlinear unstable control problem. The problem of balancing an inverted pendulum is of the delicate control subset of problems in the greater field of robotics: It is almost purely mechanical, precluding the need for higher level planning or behavioral programming that a more complex robot might require. Inputs over (limited) time in the form of pendulum position readings are mapped algorithmically to output in the form of motor control. The aim of this paper is to stabilize the Inverted Pendulum such that the position of the carriage on the track

is controlled quickly and accurately so that the pendulum is always erected in its inverted position during such movements.

Good performance of system comes from the ability of rejecting external disturbance and uncertainty of system. In order to verify effectiveness and advantages of PIDNNC, actual experiments of single and double inverted pendulum systems are both implemented.

The system devices used in the experiments, which are based on the inverted pendulum produced by Googol Technology (Shenzhen) Ltd., are composed of three parts. The first part is the equipment including a cart, a drive, and links. The second one is an electrical control box with motor drives and interface circuit. The third one is a general four-axis movement control board fixed in a PC.

1) Comparison of Control Performance Between LQR Controller and PIDNNC in Single Inverted Pendulum System: In this case, variables in a single inverted pendulum system are two: position of cart x and offset angle of link θ. The inverted pendulum has two inputs and two outputs. In order to have full state feedback control, two PID controllers have to be used. Neural networks have a big advantage here due to their parallel nature. In such a case, only one PIDNNC can be used instead of two PID controllers. The initial weight values of PIDNNC are obtained by LQR whose control law is

$$u(k) = -KX = -(k_1 x + k_2 \dot{x} + k_3 \theta + k_4 \dot{\theta}).$$

The objective of such state regulation may be attained by selecting the control input $u(k)$ to minimize the following performance index of the type:

$$J_{\mathrm{LQR}} = \sum_{k=1}^{\infty} \left(X(k)^{\mathrm{T}} Q X(k) + u(k)^{\mathrm{T}} R u(k) \right)$$

where $Q \geq 0$ and $R > 0$ are symmetric weighing matrices. LQR has been a standard and typical control design which can be solved easily under MATLAB. Regarding how to design the feedback matrix K, please refer to the [20]. The feedback matrix K is designed as

$$K = [k_x \quad k_{\dot{x}} \quad k_\theta \quad k_{\dot{\theta}}] = [-21.49 \quad 14.79 \quad 54.74 \quad 9.00].$$

Because the controller of stabilizing inverted pendulum is a regulator, there are $r_i = 0$. The inputs of PIDNNC are $>$

$$e_1 = r_1 - x = -x \quad e_2 = r_2 - \theta = -\theta.$$

In order to exam the robustness and recover ability of eliminating the external disturbance of the system, some external pulse force with small limited amplitude is added during the experiment. The control performances are obtained from actual experiment of single inverted pendulum under LQR and PIDNNC with or without external disturbance. The sampling time of the experiment is 0.005 s. For the experiment implementation simplicity, we selected the fixed stable small learning rate value as $\eta = 0.001$. The experimental results are shown in Fig. 3, in which Fig. 3(a) shows the cart displacement under the control of PIDNNC and LQR controller, respectively, without external disturbance, from which one can see that when there is no external disturbance applied to the system, the cart stabilizes at the equilibrium position of 0.014 m under the control of LQR controller, while the equilibrium position of PIDNNC is 0.006 m in the same condition. It is evident that the PIDNNC is superior to the LQR. This is because the inverted pendulum is actually a nonlinear system. Although LQR is an optimal controller, it is effective only for the linear system. The experiment verifies that the PIDNNC is a nonlinear controller. When there is the external disturbance applied to the system, as shown in Fig. 3(b), it demonstrates that the single inverted pendulum under the control of PIDNNC has much smaller displacement and much shorter time of coming back to the equilibrium position compared to the LQR. Fig. 3(c) shows the displacement responses with the positive and negative learning rates, respectively, from which one can see that system can work even better than it works with the positive one, which satisfies the conditions of (14) and (15) in Theorem 1.

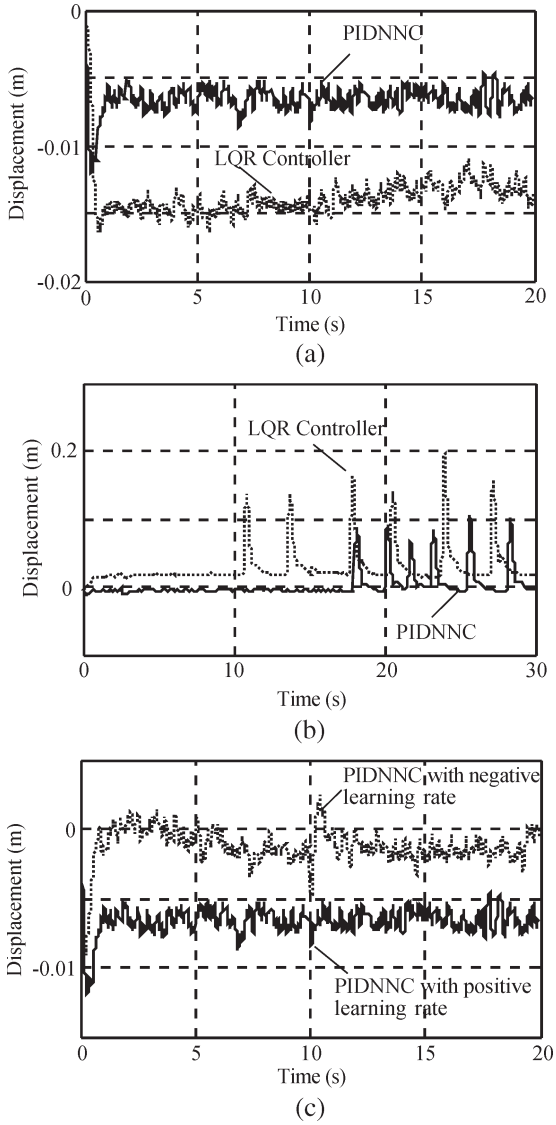


Fig. 3. (a) Displacement under LQR and PIDNNC without disturbance. (b) Displacement under LQR and PIDNNC with disturbance. (c) Displacement under PIDNNC with positive and negative learning rates.

2) Comparison of Control Performance Between LQR Controller and PIDNNC in Double Inverted Pendulum System: In this case, variables in double inverted pendulum system become three: x, position of cart; θ_1, offset angle of link 1 (short link);

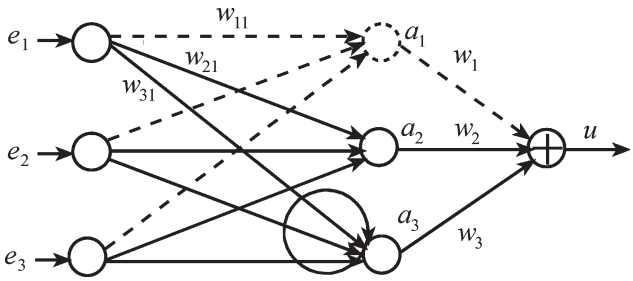


Fig. 4. Structure of PIDNNC used in a double inverted pendulum system.

and θ_2, offset angle of link 2 (long link). The inverted pendulum has three inputs and three outputs. The LQR control law is

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

in which feedback matrix K is designed as

$$\begin{aligned} K &= [k_x \quad k_{\dot{x}} \quad k_{\theta_1} \quad k_{\dot{\theta}_1} \quad k_{\theta_2} \quad k_{\dot{\theta}_2}] \\ &= [32.1522 \quad 28.1289 \quad 129.3426 \\ &\qquad 1.9087 \quad -234.6127 \quad -33.4221]. \end{aligned}$$

The values of such a K are used as the initial weight values of the neural network controller. The structure of neural network controller is shown in Fig. 4 in which there are three nodes in input layer, two nodes in hidden layer, and one node in output layer. Notice that there are only two nodes in hidden layer, which are proportional node and derivative node. The integral node a_1 (as be shown with dashed line in the Fig. 4) needs not be used in the given application, which is taught by the LQR controller.

The inputs to neural network are

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

Initial weight values are

$$\begin{aligned} w_{21}(0) &= k_x = 32.1522 & w_{31}(0) &= k_{\dot{x}}/T = 5625.78 \\ w_{22}(0) &= k_{\theta_1} = 129.3426 & w_{32}(0) &= k_{\dot{\theta}_1}/T = 381.74 \\ w_{23}(0) &= k_{\theta_2} = -234.6127 & w_{33}(0) &= k_{\dot{\theta}_2}/T = -6684.42 \end{aligned}$$

in which T is the sampling period and $T = 0.005$ s. Control law is

$$\begin{aligned} u(k) = w_2(k) \cdot \left[\sum_{j=1}^{s} w_{2j}(k) e_j(k) \right] + w_3(k) \\ \cdot \left[\sum_{j=1}^{s} w_{3j}(k) e_j(k) - \sum_{j=1}^{s} w_{3j}(k-1) e_j(k-1) \right]. \end{aligned}$$

Fig. 5 is the actual experiment device of the double inverted pendulum control system.

The experimental results of double inverted pendulum system are shown in Fig. 6, in which Fig. 6(a) shows the cart displacement versus time under the control of LQR and PIDNNC without external disturbance. It is evident that under the control of PIDNNC the cart stabilizes in the position of 0.05 m, and

Fig. 5. Control system of double linear inverted pendulum device.

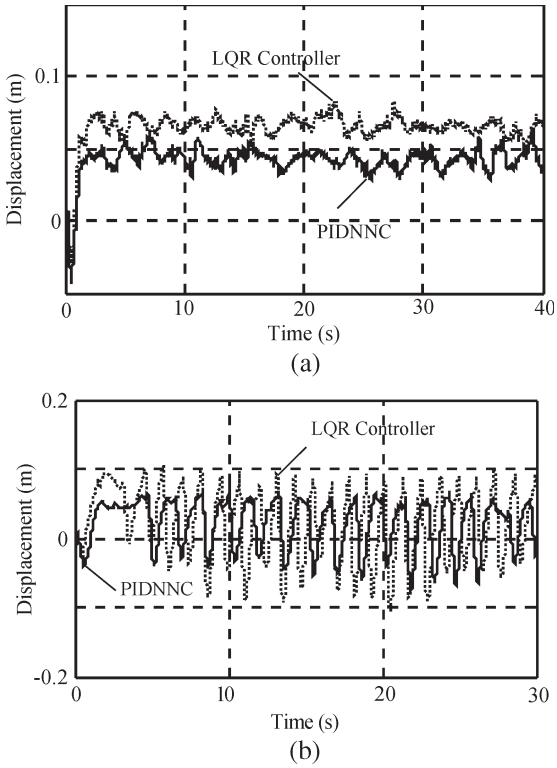


Fig. 6. (a) Displacement under LQR and PIDNNC without disturbance. (b) Displacement under LQR and PIDNNC with disturbance.

the maximum displacement is 0.06 m. While under the control of LQR controller, the equilibrium position is about 0.07 m, and the maximum displacement is 0.08 m. Fig. 6(b) shows the cart displacement with external disturbance, from which it can verified that under the control of PIDNNC, the displacement of the cart depart from center (or starting position) is smaller and the time of coming back to the equilibrium position is shorter.

Owing to the online learning capability, the proposed PIDNNC can adjust its network weight parameters according to the changes of system structure parameters and external environment. Actual experiments indicate that PIDNNC has stronger antidisturbance ability than LQR does, which will give system better adaptive capacity toward uncertainty of system and external disturbance in actual experiment. That is to say, controlling inverted pendulum system by PIDNNC needs a shorter horizontal rail. Fig. 6 also shows under the stability

condition, two controllers have almost the same control results of angles. This is reasonable because angles in stable control region usually do not change greatly.

VI. Conclusion

An adaptive PID-like neural network controller suitable for a large class of SIMO 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 experiments on single and double inverted pendulum are implemented, and the results are compared with that of the LQR, which demonstrates the advantages of controller proposed in simple structure, easy-design, and higher accuracy response performance for multivariable SIMO systems with disturbance and uncertainty.

References

[1] J. G. Ziegler and N. B. Nichols, "Optimum settings for automatic controllers," *Trans. ASME*, vol. 64, pp. 759–768, 1942.

[2] M. Morari and E. Zafiriou, *Robust Process Control*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[3] M. Zhuang and D. P. Atherton, "Automatic tuning of optimum PID controllers," *Proc. Inst. Elect. Eng.—Control Theory Appl.*, vol. 140, no. 3, pp. 216–224, May 1993.

[4] W. K. Ho, C. C. Hang, and L. S. Cao, "Tuning of PID controllers based on gain and phase margin specifications," *Automatica*, vol. 31, no. 3, pp. 497–502, Mar. 1995.

[5] M. T. Ho, "Synthesis of H-infinity PID controllers: A parametric approach," *Automatica*, vol. 39, no. 6, pp. 1069–1075, 2003.

[6] H. J. Uang and C. C. Lien, "Mixed H_2/H_∞ PID tracking control design for uncertain spacecraft systems using a cerebellar model articulation controller," *Proc. Inst. Elect. Eng.—Control Theory Appl.*, vol. 153, no. 1, pp. 1–13, Jan. 2006.

[7] G. Cheng and K. Peng, "Robust composite nonlinear feedback control with application to a servo positioning system," *IEEE Trans. Ind. Electron.*, vol. 54, no. 2, pp. 1132–1140, Apr. 2007.

[8] D. Psaltis, A. Sideris, and A. A. Yamamura, "A multilayered neural network controller," *IEEE Control Syst. Mag.*, vol. 8, no. 2, pp. 17–21, Apr. 1988.

[9] T. Pajchrowski and K. Zawirski, "Application of artificial neural network to robust speed control of servodrive," *IEEE Trans. Ind. Electron.*, vol. 54, no. 1, pp. 200–207, Feb. 2007.

[10] S. S. Ge and F. Hong, "Adaptive neural network control of nonlinear systems with unknown time delays," *IEEE Trans. Autom. Control*, vol. 48, no. 11, pp. 2004–2010, Nov. 2003.

[11] C.-H. Lu and C.-C. Tsai, "Adaptive predictive control with recurrent neural network for industrial processes: An application to temperature control of a variable-frequency oil-cooling machine," *IEEE Trans. Ind. Electron.*, vol. 55, no. 3, pp. 1366–1375, Mar. 2008.

[12] D. Wang and J. Huang, "Neural network-based adaptive dynamic surface control for a class of uncertain nonlinear systems in strict-feedback form," *IEEE Trans. Neural Netw.*, vol. 16, no. 1, pp. 195–202, Jan. 2005.

[13] H. Su and H. T. Chong, "Induction machine condition monitoring using neural network modeling," *IEEE Trans. Ind. Electron.*, vol. 54, no. 1, pp. 241–249, Feb. 2007.

[14] S. Madhusudan and S. Smriti, "Identification and control of a nonlinear system using neural networks by extracting the system dynamics," *IETE J. Res.*, vol. 53, no. 1, pp. 43–50, 2007.

[15] C. J. Lin and C. H. Chen, "A compensation-based recurrent fuzzy neural network for dynamic system identification," *Eur. J. Oper. Res.*, vol. 172, no. 2, pp. 696–715, 2006.

[16] T. J. Ren and T. C. Chen, "Motion control for a two-wheeled vehicle using a self-tuning PID controller," *Control Eng. Pract.*, vol. 16, no. 3, pp. 365–375, Mar. 2008.

[17] T. Bakhtiara and S. Hara, "Regulation performance limitations for SIMO linear time-invariant feedback control systems," *Automatica*, vol. 44, no. 3, pp. 659–670, Mar. 2008.

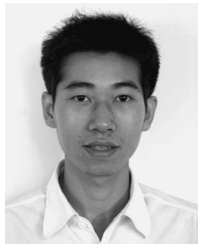
[18] S. Cong, G. Li, and B. Ji, "A novel PID-like neural network controller," in *Proc. 16th IFAC World Congr.*, Prague, Czech Republic, 2005, pp. 1–6.

[19] S. Cong, Y. Liang, and G. Li, "The design of multivariable adaptive PID-like neural network controller and its design method," *Inf. Control*, vol. 35, no. 5, pp. 568–573, 2006.

[20] Y. Liang, "Nonlinear adaptive control of time-varying uncertain electromechanical motion system," Ph.D. dissertation, Univ. Sci. Technol. China, Hefei, China, 2008.

S. Cong received the B.S. degree in automatic control from Beijing University of Aeronautics and Astronautics, Beijing, China, in 1982 and the Ph.D. degree in system engineering from the University of Rome "La Sapienza," Rome, Italy, in 1995.

She is currently a Professor with the Department of Automation, University of Science and Technology of China, Hefei, China. Her research interests include advanced control strategies for motion control, fuzzy logic control, neural networks design and applications, robotic coordination control, and quantum system control.



Y. Liang received the B.S. degree in automation from Southwest University of Science and Technology, Mianyang City, China, in 2002 and the Ph.D. degree in control theory and control engineering from the University of Science and Technology of China, Hefei, China, in 2008.

He is currently with the Department of Automation, University of Science and Technology of China. His research interests include the neural network controller design and implementation, motion control, and robust control for nonlinear system.