



# Neural network-based PID compensation for nonlinear systems: ball-on-plate example

Amin Mohammadi<sup>1</sup> · Ji-Chul Ryu<sup>1</sup>

Received: 26 July 2018 / Revised: 5 September 2018 / Accepted: 5 September 2018 / Published online: 11 September 2018  
© Springer-Verlag GmbH Germany, part of Springer Nature 2018

## Abstract

In this paper, a neural network (NN)-based feedback controller is proposed in order to compensate for the errors caused by using an approximated dynamic model in controller design. The controller consists of two subcontrollers working in parallel: base linear controller and NN-based PID compensator. The former can be any controller that is easily designed based on the system's linearized or simplified model. The latter is based on a PID controller with adjustable gains and a neural network is used to update the PID gains during control process, aiming to compensate for the nonlinear effects ignored in the base controller. The performance of the proposed controller is demonstrated with a ball-on-plate system built for this study. Approximate feedback linearization is used to design the base controller in this work and the NN-based PID compensator is used in parallel. Simulation and experimental results that achieve better stabilization and trajectory tracking performance are provided and discussed.

**Keywords** Neural network-based PID · Nonlinearity compensation · Feedback linearization · Ball-on-plate system

## 1 Introduction

Controller design of a nonlinear system is in general difficult. One way to avoid such complexity is using a linearly approximated model so that a linear control technique can be easily applied. However, a certain degree of control error is inevitable when using the simplified model.

The objective of this work is to reduce this error using a neural network (NN)-based PID compensator. The controller structure of the system consists of two parts: *base controller* and *NN-based PID*. The base controller can be any classic linear controller that can perform stabilization and/or trajectory tracking. The error of the base controller, which may be induced by model simplifications, is compensated by the NN-based PID.

In this work, the full-state feedback linearization method is used on an approximated model for the design of the base controller. The neural network in the PID compensator takes

state errors as input and generates PID gains as outputs. The PID compensator uses the updated gains and creates compensatory control input to the system. The neural network itself is also being updated online using the backpropagation learning algorithm. Finally, the two control inputs from the base linear controller and the NN-based PID compensator are added to generate the input to the system.

This control approach is implemented on the ball-on-plate system we built for this study and demonstrates improvement in stabilization and trajectory tracking control.

### 1.1 Literature review

The feedback linearization method which our proposed base controller is based on is a powerful tool in control design of nonlinear systems since a nonlinear system can be transformed to an equivalent linear system so that linear control techniques can be easily applied. Among numerous related studies, Ho et al. [1] used the aforementioned method to control a wheel that balances a ball on it. Ryu et al. [2] also used the feedback linearization method to balance a disk on a disk. In their work, the lower disk is attached to a DC motor and the upper disk which is free to roll is balanced on top of the lower disk.

✉ Ji-Chul Ryu  
jryu@niu.edu

Amin Mohammadi  
z1776184@students.niu.edu

<sup>1</sup> Department of Mechanical Engineering, Northern Illinois University, DeKalb, IL 60115, USA

However, this method is limited in that only a class of systems that satisfy necessary conditions is feedback linearizable. For those that do not satisfy the conditions, one possible solution would be simplification of the dynamic model so that feedback linearization becomes applicable with the approximate model. The ball-on-beam system is proven to be not feedback linearizable, so an approximated model approach is applied in [3], which we adopted in this study.

A typical approach to control the ball-on-plate system is to decouple the system into two independent ball-on-beam systems about each rotational axis of the plate through model simplification. A proven ball-on-beam controller can then be applied to each axis in order to finally control the ball-on-plate system.

Awtar et al. applied a double-loop control strategy where the outer loop calculates target angular positions of the plate while the inner loop uses PID control to achieve the target position [4]. Mochizuki and Ichihara proposed I–PD control, which is a variant of PID control, using the KYP (Kalman–Yakubovich–Popov) lemma [5]. Fuzzy logic has been also studied for the control of the ball-on-plate system in [6]. Ho et al. applied in [7] the approximate feedback linearization method which was first proposed in [3].

However, the simplification in controller design resulted from using an approximate model is often made at a cost of inferior performance of the control system. Therefore, we propose a compensation strategy to compensate for the effects of the ignored nonlinear terms as well as unmodeled dynamics. While a parallel control structure appears in fuzzy PID control [8,9], we chose a neural network system as a compensator due to its high adaptivity to different system conditions. In addition, the error-based control method of the PID can be used for neural network training as it has shown reliable performance. For these collective reasons, the design of a NN-based PID controller is proposed in this work.

Controller design in the structure of neural network has been increasingly employed in a wide range of applications from vehicle vibration control [10] to robotic manipulators [11].

Jung et al. designed a neural network-based pre-filter for a conventional PID, based on the reference compensation technique (RCT) that can stabilize a two-dimensional inverted pendulum [12] as well as a wheeled inverted pendulum [13]. The neural network in these studies basically represents the inverse dynamics of the control system. A variant of PID controller which is also based on neural networks is presented in [11,14] to improve the position control of a manipulator arm using pneumatic actuators. Similar to our proposed work, the backpropagation algorithm is used to update the neural network online. Cong and Liang [15] proposed a neural network controller that mimics a structure of PID using an activation and output feedback on three nodes in the hidden layer and implemented it on a single and a double inverted pendulum.

In this work, the control performance is dependent on initial conditions of the states. Kang et al. presented an adaptive neural network controller that uses a PID structure in [16] where the particle swarm optimization (PSO) algorithm is adopted to choose better initial weights that prevent trapping in local minima. Another adaptive controller is proposed in [17] to control a Segway-type inverted pendulum. The controller is based on radial basis function neural networks (RBFNNs) and the same structure of controller is applied each for balancing and yaw control.

Neural networks also have been used as an estimator [18]. In the reference, the neural network is designed to estimate the system's dynamic model and used with another neural network-based PI controller to control the speed of an induction motor under the presence of disturbances using the so-called projection algorithm. A high gain observer to estimate the state derivatives is also used in [19] with a neural network-based feedback controller that does not need offline training.

## 1.2 Paper outline

The rest of the paper is organized as follows. The dynamic model of the ball-on-plate system is derived in Sect. 2. Section 3 describes the design and structure of the proposed controller. The experimental setup we built for this study is explained in Sect. 4. Sections 5 and 6 present the simulation and experimental results, respectively, followed by concluding remarks in Sect. 7.

## 2 Dynamic model of the ball-on-plate system

In this section, we derive the equations of motion of the ball-on-plate system using the Euler–Lagrange equation. The derived model will later be simplified to two independent ball-on-beam systems such that the two rotational axis motions of the ball-on-plate system are controlled each by a ball-on-beam controller.

### 2.1 Dynamic model

Figure 1 shows the 3D CAD model of the ball-on-plate system we developed for this study. While the derivation of the equations of motion is also available in [7], we present important steps in a slightly different manner to improve understanding.

The Euler–Lagrange equation is given by

$$\frac{d}{dt} \left( \frac{\partial L}{\partial \dot{q}} \right) - \frac{\partial L}{\partial q} = Q \quad (1)$$

where Lagrangian  $L = T - V$ . Here,  $T$  and  $V$  are the total kinetic and potential energies of the system, respectively. The generalized coordinate vector  $q$  for the system is given by  $q = [x, y, \theta_x, \theta_y]^T$  in which  $(x, y)$  is the ball position measured from the center of the plate.  $\theta_x$  and  $\theta_y$  are the plate tilting angles about  $x$  and  $y$  axes, respectively. The definitions of these state variables are visualized in Figs. 1 and 2. The generalized force vector  $Q$  represents the input torques about each axis, given by  $Q = [0, 0, \tau_x, \tau_y]^T$ .

First, between the local coordinate frame  $oxyz$  attached to the plate, denoted by ‘1’, and the global coordinate frame  $OXYZ$ , denoted by ‘0’, the rotation matrix is given by

$$R_1^0 = \begin{bmatrix} c_y & 0 & s_y \\ s_x s_y & c_x & -s_x c_y \\ -c_x s_y & s_x & c_x c_y \end{bmatrix} \tag{2}$$

where  $s_x, c_x, s_y,$  and  $c_y$  denote  $\sin \theta_x, \cos \theta_x, \sin \theta_y,$  and  $\cos \theta_y,$  respectively.

The relationship of  $S(\omega_p^0) = \dot{R}_1^0 (R_1^0)^T$  from [20, chap. 4] gives the angular velocity of the plate in the form of a skew symmetric matrix.

$$\begin{bmatrix} 0 & -\omega_{pz} & \omega_{py} \\ \omega_{pz} & 0 & -\omega_{px} \\ -\omega_{py} & \omega_{px} & 0 \end{bmatrix} = \begin{bmatrix} 0 & -\dot{\theta}_y s_x & \dot{\theta}_y c_x \\ \dot{\theta}_y s_x & 0 & -\dot{\theta}_x \\ -\dot{\theta}_y c_x & \dot{\theta}_x & 0 \end{bmatrix} \tag{3}$$

From this relationship, the angular velocity of the plate in the global coordinate frame is obtained as

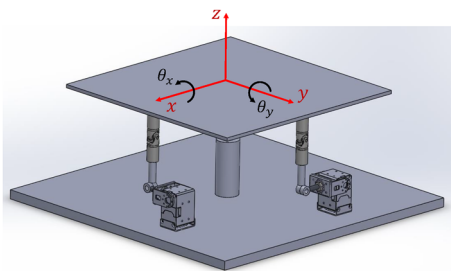


Fig. 1 Developed ball-on-plate system in CAD drawing

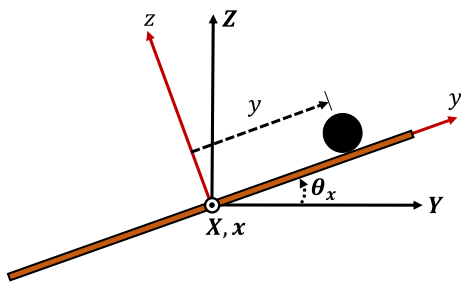


Fig. 2 Global  $OXYZ$  and local  $oxyz$  coordinate systems

$$\omega_p^0 = [\omega_{px}, \omega_{py}, \omega_{pz}]^T = [\dot{\theta}_x, \dot{\theta}_y c_x, \dot{\theta}_y s_x]^T \tag{4}$$

It is more convenient to calculate the rotational kinetic energy with the angular velocity expressed in the local coordinate frame where the moments of inertia remain constant. Thus, we have

$$\omega_p^1 = (R_1^0)^T \omega_p^0 \tag{5}$$

On the other hand, given the ball position in the local coordinate frame

$$p_b^1 = [x, y, r]^T \tag{6}$$

where  $r$  is the radius of the ball, the velocity of the ball can be obtained in the global coordinate frame such that

$$v_b^0 = \frac{dp_b^0}{dt} = \frac{d(R_1^0 p_b^1)}{dt} \tag{7}$$

Assuming no slip condition between the ball and the plate, the angular velocity of the ball relative to the plate is given by  $\left[-\frac{\dot{y}}{r}, \frac{\dot{x}}{r}, 0\right]^T$ . Furthermore, since the plate is also rotating, adding the angular velocity of the plate of Eq. (5) yields the angular velocity of the ball, expressed in the local coordinate frame, as

$$\omega_b^1 = \left[-\frac{\dot{y}}{r} + \dot{\theta}_x c_y, \frac{\dot{x}}{r} + \dot{\theta}_y, \dot{\theta}_x s_y\right]^T \tag{8}$$

Finally, the total kinetic and potential energies of the system are the sum of those of the plate and the ball, given by

$$T_{total} = T_{plate} + T_{ball} \tag{9}$$

$$V_{total} = V_{plate} + V_{ball} \tag{10}$$

where

$$T_{plate} = \frac{1}{2} (\omega_p^1)^T \mathcal{I}_p \omega_p^1 \tag{11}$$

$$V_{plate} = 0 \tag{12}$$

$$T_{ball} = \frac{1}{2} m_b (v_b^0)^T v_b^0 + \frac{1}{2} I_b (\omega_b^1)^T \omega_b^1 \tag{13}$$

$$V_{ball} = m_b g [0 \ 0 \ 1] p_b^0 \tag{14}$$

Note that  $\mathcal{I}_p$  is the constant inertia tensor expressed in the body-attached frame given by

$$\mathcal{I}_p = \begin{bmatrix} I_{px} & 0 & 0 \\ 0 & I_{py} & 0 \\ 0 & 0 & I_{pz} \end{bmatrix} \tag{15}$$

**Table 1** Description of the system parameters

$I_{px}$	Plate moment of inertia about $x$ axis
$I_{py}$	Plate moment of inertia about $y$ axis
$I_{pz}$	Plate moment of inertia about $z$ axis
$m_b$	Mass of the ball
$r$	Radius of the ball
$I_b$	Moment of inertia of the ball
$g$	Gravitational acceleration

where  $I_{px}$ ,  $I_{py}$ , and  $I_{pz}$  are the moment of inertia about  $x$ ,  $y$ , and  $z$  axes, respectively and  $I_b$  is the moment of inertia of the ball (Table 1). Now by substituting  $L = T_{total} - V_{total}$  into Eq. (1), one can obtain the full dynamic model of the ball-on-plate system.

$$M\ddot{q} + C\dot{q} + G = Q \tag{16}$$

Due to the length, the detailed expressions of the matrices of  $M$ ,  $C$ , and  $G$  are not presented here, but one could easily derive with help of mathematical software such as Maple or Matlab.

As mentioned earlier, the base controller in our proposed control structure is supposed to be designed with a simplified dynamic model. Ignoring coupling terms results in two independent, simplified ball-on-beam equations.

$$\left(m_b + \frac{I_b}{r^2}\right)\ddot{x} - m_b x \dot{\theta}_y^2 - m_b g \sin \theta_y = 0 \tag{17a}$$

$$\begin{aligned} (m_b x^2 + I_b + I_p)\ddot{\theta}_y + 2m_b x \dot{x} \dot{\theta}_y \\ - m_b g x \cos \theta_y - m_b g r \sin \theta_y = \tau_y \end{aligned} \tag{17b}$$

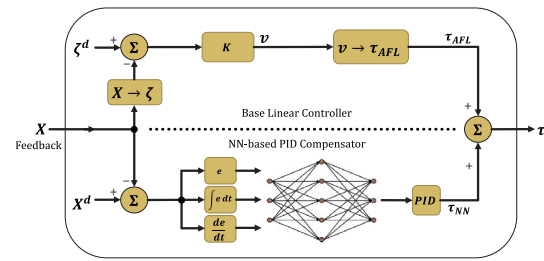
$$\left(m_b + \frac{I_b}{r^2}\right)\ddot{y} - m_b y \dot{\theta}_x^2 - m_b g \sin \theta_x = 0 \tag{18a}$$

$$\begin{aligned} (m_b y^2 + I_b + I_p)\ddot{\theta}_x + 2m_b y \dot{y} \dot{\theta}_x \\ - m_b g y \cos \theta_x - m_b g r \sin \theta_x = \tau_x \end{aligned} \tag{18b}$$

where  $I_p = I_{px} = I_{py}$  assuming the plate is square. Since these two sets of the equations in Eqs. (17) and (18) are basically identical for each axis of rotation, only Eq. (17) will be used in the controller design in the next section. Then, the same structure of the controller will be applied to the other rotational motion.

### 3 Controller design

The proposed controller consists of two subcontrollers: base linear controller and neural network (NN)-based PID com-



**Fig. 3** Structure of the proposed controller (for one axis)

pensator. When a simple controller design is chosen based on its approximated model, the NN-based PID compensator is designed such that it compensates for the errors that may have been caused by the simplification of the dynamic model with the base controller. This proposed controller works in a way that the control inputs from the two individual sub-controllers are added to finally generate torque input to the ball-on-plate system.

We use the approximate input-output feedback linearization method adopted from [3] to design the base controller. The method builds on the simplified model derived in Sect. 2.1 and further simplification is made during the controller design in Sect. 3.1. For the NN-based PID compensator, we design a neural network that takes state errors as input and updates the gains of the PID controller online to complement the base controller. The structure of proposed controller is shown in Fig. 3 and the detailed design of each subcontrollers are presented in this section.

#### 3.1 Base controller design using approximate feedback linearization

The simplified dynamic model in Eq. (17) can be rewritten in an input-affine form as

$$\dot{X} = f(X) + g(X)\tau_y \tag{19}$$

where  $X = [x, \dot{x}, \theta_y, \dot{\theta}_y]^T$ . This system is said to be feedback linearizable if an output  $h(X)$  exists satisfying

$$\nabla h \text{ ad}_f^i g = 0, \quad i = \{0, 1, 2\}, \tag{20a}$$

$$\nabla h \text{ ad}_f^3 g \neq 0, \tag{20b}$$

where the Lie bracket is defined as  $\text{ad}_f g = \nabla g f - \nabla f g$  [21]. Then, by solving Eq. (20a), the output is obtained as  $h(X) = x$ . Now to transform the nonlinear system to a linear one, we differentiate the output  $h$  with respect to time until the input appears.

$$\zeta_1 = h(X) = x \tag{21}$$

$$\zeta_2 = \frac{d\zeta_1}{dt} = \dot{x} \tag{22}$$

$$\zeta_3 = \frac{d\zeta_2}{dt} = \frac{1}{m_b + \frac{I_b}{r^2}} (m_b x \dot{\theta}_y^2 + m_b g s_y) \tag{23}$$

$$\zeta_4 = \frac{d\zeta_3}{dt} = \frac{1}{m_b + \frac{I_b}{r^2}} (m_b \dot{x} \dot{\theta}_y^2 + m_b g \dot{\theta}_y c_y) \tag{24}$$

in which the term  $2m_b x \dot{\theta}_y \ddot{\theta}_y / (m_b + I_b/r^2)$  is ignored in  $\zeta_4$ . This approximation is necessary in order to make the system feedback linearizable. Then, we finally obtain, with  $\zeta = [\zeta_1, \zeta_2, \zeta_3, \zeta_4]^T$ ,

$$\dot{\zeta} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \zeta + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} v \tag{25}$$

with the input transformation

$$v = \alpha(X) + \beta(X)\tau_y \tag{26}$$

where  $\alpha(X)$  and  $\beta(X)$  are a function of the state variables and their expressions are provided in ‘‘Appendix’’.

Since the transformed system in Eq. (25) is now linear, a linear control technique such as the pole placement method can be used to determine controller gains. The control input in Eq. (25) can be designed such that

$$v = (x^d)^{(4)} + K_4 \ddot{e} + K_3 \dot{e} + K_2 e + K_1 e \tag{27}$$

where  $x^d$  is the desired ball trajectory and the error  $e = x^d - x$ . The actual torque input  $\tau_y$  can then be obtained using the input transformation in Eq. (26).

### 3.2 NN-based PID controller

The combination of PID and neural network is used as a compensator in the form of

$$\tau_{NN} = K_p e + K_i \int_0^t e dt + K_d \dot{e} \tag{28}$$

where the position error  $e = x^d - x$ . The control gains  $K_p$ ,  $K_i$ , and  $K_d$  are updated online through the neural network while adapting to different situations.

The structure of the neural network used in the controller is shown in Fig. 4. The designed neural network has three inputs, five neurons in the hidden layer, and three in the output layer in which all neurons of one layers are connected to all neurons of the next. The three inputs to the neural network are

$$S_1 = e, \quad S_2 = \int_0^t e dt, \quad S_3 = \dot{e} \tag{29}$$

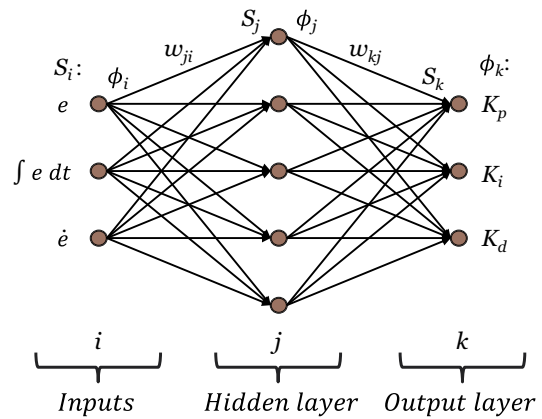


Fig. 4 Designed neural network (biases,  $b_j$  and  $b_k$ , are not shown)

The outputs are three PID control gains.

$$\phi_1 = K_p, \quad \phi_2 = K_i, \quad \phi_3 = K_d \tag{30}$$

A tangent sigmoid and linear functions are used as transfer functions for the hidden and the output layers, respectively. Those functions are given by

$$\phi_j(S_j) = \frac{2}{1 + e^{-2S_j}} - 1 \tag{31}$$

$$\phi_k(S_k) = S_k \tag{32}$$

The final torque control input to the system is the sum of the base linear control  $\tau_{AFL}$  and the neural network control  $\tau_{NN}$  as shown in Fig. 3, such that

$$\tau = \tau_{AFL} + \tau_{NN} \tag{33}$$

To update the weights and biases of the neural network, the gradient descent learning algorithm [22] is applied with the following error function.

$$E = \frac{1}{2}(x^d - x)^2 \tag{34}$$

Then, the update rules with an iterative index  $n$  are defined as

$$w_{kj}(n + 1) = w_{kj}(n) - \eta dw_{kj}(n) + \alpha dw_{kj}(n - 1) \tag{35}$$

$$b_k(n + 1) = b_k(n) - \eta db_k(n) + \alpha db_k(n - 1) \tag{36}$$

$$w_{ji}(n + 1) = w_{ji}(n) - \eta dw_{ji}(n) + \alpha dw_{ji}(n - 1) \tag{37}$$

$$b_j(n + 1) = b_j(n) - \eta db_j(n) + \alpha db_j(n - 1) \tag{38}$$

where

$$dw_{kj}(n) = \frac{\partial E}{\partial w_{kj}}(n), \quad db_k(n) = \frac{\partial E}{\partial b_k}(n)$$

$$dw_{ji}(n) = \frac{\partial E}{\partial w_{ji}}(n), \quad db_j(n) = \frac{\partial E}{\partial b_j}(n)$$

and the learning rate  $\eta$  and momentum rate  $\alpha$  can be determined experimentally. Furthermore, we have the following equations using the chain rule.

$$\frac{\partial E}{\partial w_{kj}} = \frac{\partial E}{\partial x} \frac{\partial x}{\partial \tau} \frac{\partial \tau}{\partial \tau_{NN}} \frac{\partial \tau_{NN}}{\partial \phi_k} \frac{\partial \phi_k}{\partial S_k} \frac{\partial S_k}{\partial w_{kj}} \tag{39}$$

$$\frac{\partial E}{\partial b_k} = \frac{\partial E}{\partial x} \frac{\partial x}{\partial \tau} \frac{\partial \tau}{\partial \tau_{NN}} \frac{\partial \tau_{NN}}{\partial \phi_k} \frac{\partial \phi_k}{\partial S_k} \frac{\partial S_k}{\partial b_k} \tag{40}$$

$$\begin{aligned} \frac{\partial E}{\partial w_{ji}} &= \frac{\partial E}{\partial x} \frac{\partial x}{\partial \tau} \frac{\partial \tau}{\partial \tau_{NN}} \left( \sum_{k=1}^3 \frac{\partial \tau_{NN}}{\partial \phi_k} \frac{\partial \phi_k}{\partial S_k} \frac{\partial S_k}{\partial \phi_j} \right) \\ &\quad \times \frac{\partial \phi_j}{\partial S_j} \frac{\partial S_j}{\partial w_{ji}} \end{aligned} \tag{41}$$

$$\begin{aligned} \frac{\partial E}{\partial b_j} &= \frac{\partial E}{\partial x} \frac{\partial x}{\partial \tau} \frac{\partial \tau}{\partial \tau_{NN}} \left( \sum_{k=1}^3 \frac{\partial \tau_{NN}}{\partial \phi_k} \frac{\partial \phi_k}{\partial S_k} \frac{\partial S_k}{\partial \phi_j} \right) \\ &\quad \times \frac{\partial \phi_j}{\partial S_j} \frac{\partial S_j}{\partial b_j} \end{aligned} \tag{42}$$

For the calculation of each term in Eqs. (39)–(42), we obtain from Eq. (34)

$$\frac{\partial E}{\partial x} = -(x^d - x) \tag{43}$$

Then, due to the nature of the gradient descent method, the direction of the term  $\frac{\partial x}{\partial \tau}$  is more important. Hence, using the finite difference method, one can simplify the term as follows.

$$\frac{\partial x}{\partial \tau}(n) = \text{sgn} \left( \frac{\Delta x}{\Delta \tau} \right) = \text{sgn} \left( \frac{x(n) - x(n-1)}{\tau(n) - \tau(n-1)} \right) \tag{44}$$

where  $\text{sgn}(\cdot)$  denotes the sign function.

Also, from Eq. (33),

$$\frac{\partial \tau}{\partial \tau_{NN}} = 1 \tag{45}$$

Since the neural network outputs are the gains of the PID controller, we have

$$\frac{\partial \tau_{NN}}{\partial \phi_1} = x^d - x \tag{46}$$

$$\frac{\partial \tau_{NN}}{\partial \phi_2} = \int_0^t (x^d - x) dt \tag{47}$$

$$\frac{\partial \tau_{NN}}{\partial \phi_3} = \dot{x}^d - \dot{x} \tag{48}$$

The neural network structure, as shown in Fig. 4, yields

$$\frac{\partial S_k}{\partial w_{kj}} = \phi_j, \quad \frac{\partial S_k}{\partial b_k} = 1, \quad \frac{\partial S_k}{\partial \phi_j} = w_{kj}, \tag{49}$$

$$\frac{\partial S_j}{\partial w_{ji}} = \phi_i, \quad \frac{\partial S_j}{\partial b_j} = 1 \tag{50}$$

Furthermore, from the transfer functions we selected in Eqs. (31) and (32),

$$\frac{\partial \phi_j}{\partial S_j} = 1 - \phi_j^2 \tag{51}$$

$$\frac{\partial \phi_k}{\partial S_k} = 1 \tag{52}$$

Finally, following the rules in Eqs. (35)–(38) will update the weights and biases of the neural network.

As it is often the case, the designed neural network requires offline training beforehand. This training is done by setting an acceptable set of PID gains, which can be first obtained experimentally, as desired outputs of the neural network.

## 4 Experimental setup

The experimental setup built for this study is shown in Fig. 5. It consists of three main components: mechanical system, vision system, and controller system.

### 4.1 Mechanical system

The experimental setup is designed such that a steel ball is free to roll on an acrylic plate that is placed on top of a center rod through a universal joint. The plate is controlled by two motors through a linkage mechanism. Each linkage is attached to the plate through a universal joint so that it can rotate around  $x$  and  $y$  axes at the same time. Dynamixel motors (XM430-W210-T) mounted on the bottom plate are used as actuators and the motor shaft is attached to the other end of each linkage.

Using geometry and small angle approximations, we have the following relationship between the plate angle  $\theta_y$  and the motor shaft angle  $\theta_m$ .

$$\theta_y = \frac{d}{l} \theta_m$$

and between the torque applied to the plate  $\tau$  and the motor output torque  $\tau_m$ ,

$$\tau = \frac{l}{d} \tau_m \cos \theta_m \cos \theta_x$$



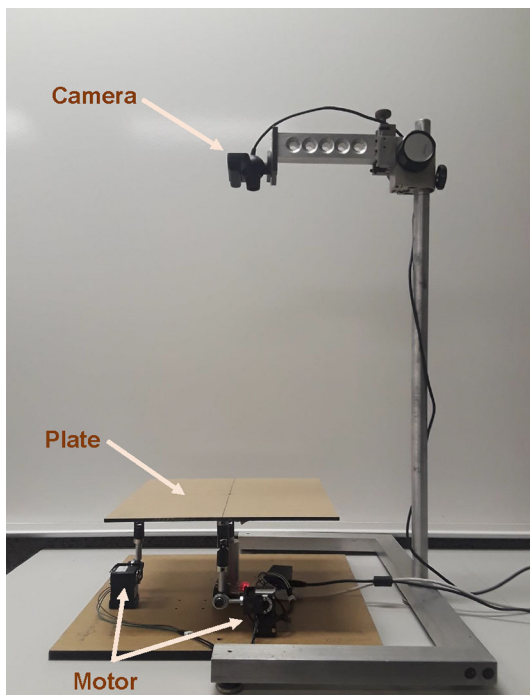


Fig. 5 Experimental setup

Table 2 Values of the system parameters

Parameters	Values
$I_{px}$	$4.315 \times 10^{-6}$ (kg m <sup>2</sup> )
$I_{py}$	$4.315 \times 10^{-6}$ (kg m <sup>2</sup> )
$I_{pz}$	$8.630 \times 10^{-6}$ (kg m <sup>2</sup> )
$m_b$	0.067 (kg)
$r$	0.013 (m)
$I_b$	$4.077 \times 10^{-3}$ (kg m <sup>2</sup> )
$g$	9.81 (m/s <sup>2</sup> )
$l$	0.140 (m)
$d$	0.056 (m)

where  $d$  denotes the length of the lower link attached to the motor shaft and  $l$  denotes the distance from the center of the plate to the point where the upper (universal joint) link is attached.

Table 2 shows the system parameters and their values of the actual system.

## 4.2 Vision system

A low-cost USB camera (Sony SLEH-00448) is used to track the ball on the plate. It runs at a frame rate of 120 Hz providing  $320 \times 240$  pixel resolution. The OpenCV library is used for image processing. The camera is mounted at a height of 48 cm from the plate. Since the camera measures the ball's position with respect to the global coordinate frame, the coordinates

are transformed to the local frame using the rotation matrix in Eq. (2).

## 4.3 Control system

A 2.1 GHz laptop with Ubuntu Linux 16.04 LTS is used to control the entire system. The control code is written in C++ utilizing the OpenCV library for vision tracking and the Dynamixel library for motor control. While the Dynamixel motors can run at up to 330 Hz using low latency timer in Linux and the vision system at up to 120 Hz, the entire control loop is set to run at 100 Hz due to necessary computations such as neural network online update.

From the experimental torque–current relation data provided by the manufacturer under current mode, we adopted the following linear equation to control torque input of the actuators by sending corresponding current commands to the motors.

$$\tau = 1.3i - 0.32 \quad (53)$$

where  $i$  is in A and  $\tau$  is in N m.

## 5 Simulation results

This section presents simulation results of stabilization and trajectory tracking control by our controller designed in Sect. 3. The results are compared with those of the linear controller alone.

The control gain  $K_i$  of the base linear controller in Eq. (27) are chosen as  $[K_1, K_2, K_3, K_4] = [2401, 1372, 294, 28]$ , which is the case when all the roots to the characteristic equation lie at  $-7$ . The same gain matrix is used for control of the other axis as well as for experiments. As mentioned in Sect. 3.2, we first need a conventional PID controller working along with the base controller for offline training. For this training purpose, the PID control gains are chosen experimentally as  $(K_p, K_i, K_d) = (1.0, 1.0, 0.1)$  which yields an acceptable control performance. For simulation study, the original nonlinear model in Eq. (16) is used to represent the physical system.

### 5.1 Stabilization

Figure 6 shows the simulation results for stabilizing the ball at  $(x^d, y^d) = (0.1, -0.1)$  m. To increase clarity, only the variables related to the  $y$ -axis rotational motion are shown since the other axis motion is analogous. Figure 6a shows the result of only using the base linear controller (AFL hereafter) for comparison purpose while Fig. 6b shows our proposed controller (AFLNN hereafter) in which the neural network-based PID compensator is used along with AFL. The initial

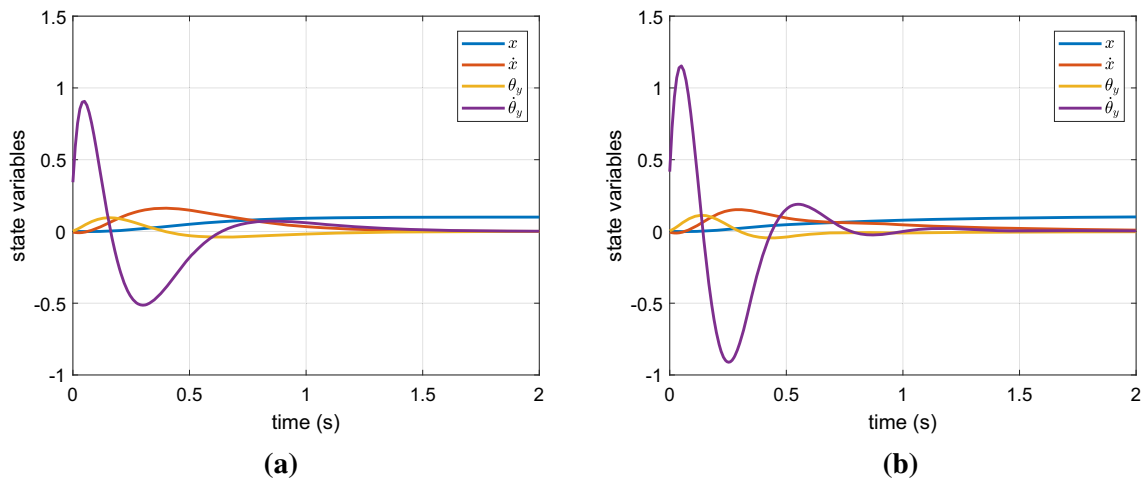


Fig. 6 Simulation results for stabilizing the ball at (0.1, -0.1). For clarity, only the y-axis rotational motion is plotted: **a** AFL alone and **b** AFLNN

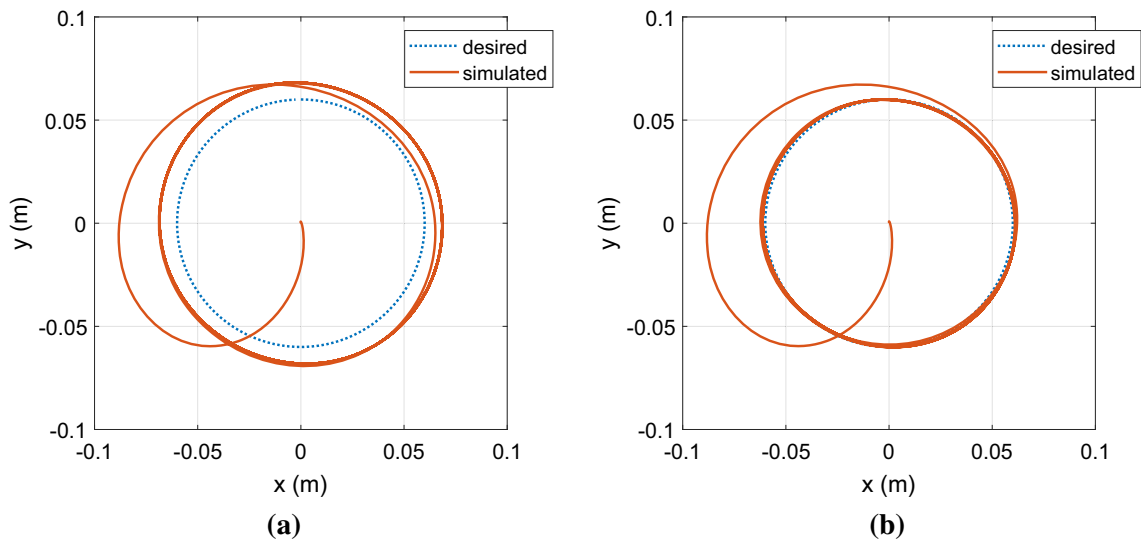


Fig. 7 Simulation results for circular trajectory tracking ( $R = 0.06$  m and  $\omega = 6$  rad/s): **a** AFL alone and **b** AFLNN

conditions for all states were given zero in the simulations. Although a target point off the plate center was selected to see the performance difference between AFL and AFLNN, except for initial larger overshoots of AFLNN, both controllers resulted in the error of 0.001 m in both  $x$  and  $y$  axes. However, as it will be shown later in the experiment section, in practice AFL shows a fair amount of steady state error whereas AFLNN is capable of significantly reducing the errors which might be resulted from the model simplification.

### 5.2 Trajectory tracking

To further demonstrate the performance of the proposed controller, a circular trajectory tracking control is conducted. The desired trajectory is chosen as

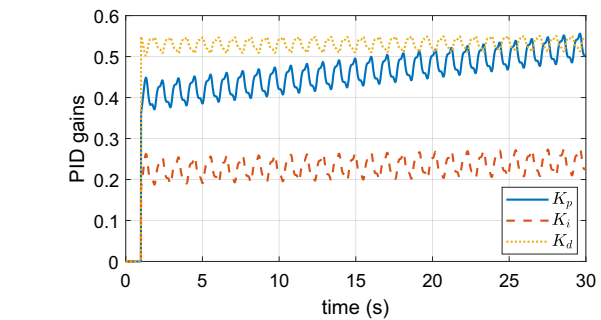
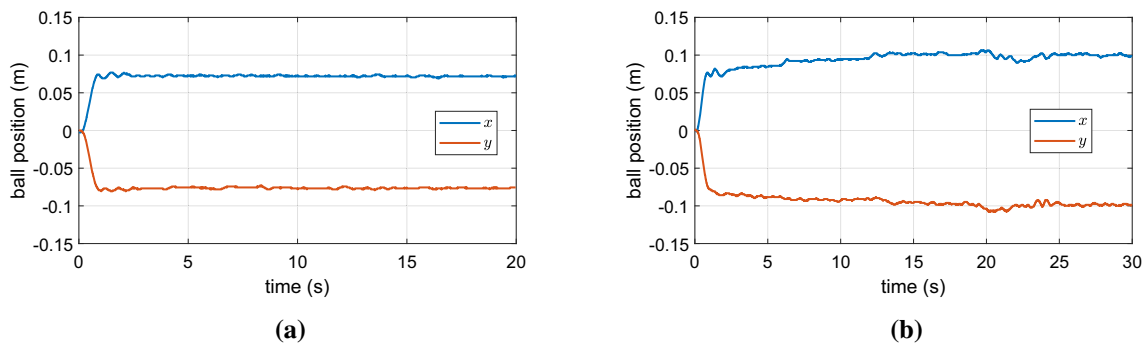


Fig. 8 PID control gains being updated through the neural network during trajectory tracking control

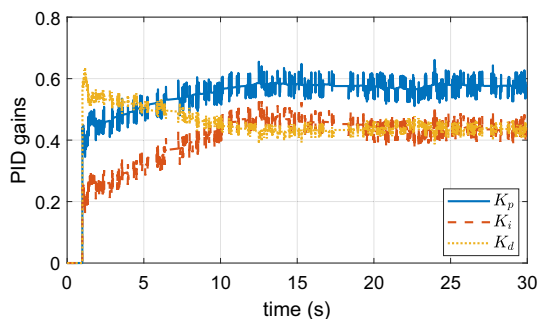
$$x^d(t) = R \sin(\omega t) \tag{54}$$

$$y^d(t) = R \cos(\omega t) \tag{55}$$





**Fig. 9** Experimental results for stabilizing the ball at  $(0.1, -0.1)$ : **a** AFL alone and **b** AFLNN



**Fig. 10** PID control gains being updated during the stabilization control experiment

where  $R$  denotes the radius of the circle and  $\omega$  the angular velocity of the circular motion. Considering the size of the plate, it is selected that  $R = 0.06$  m and  $\omega = 6$  rad/s. Figure 7a shows the simulation results of AFL alone while Fig. 7b shows the result of the proposed controller. As it can be seen from the figure, the RMS deviation of the circular motions from the desired trajectory decreased from 8.4 mm to 0.6 mm in the steady state.

Figure 8 shows the adjustment of the PID gains by the neural network over time. It is seen that in this case the proportional gain,  $K_p$ , plays an important role in reducing the steady state error as it rapidly increases in the beginning while the other gains are reasonably stable within a range.

## 6 Experimental results

It is worth mentioning that in the implementation of AFL, we need to measure higher time derivatives,  $\dot{x}$  and  $\dot{y}$ , of the ball position apart from  $x$  in the feedback control as seen in Eq. (27). These derivatives can be obtained from making use of the coordinate transformation in Eqs. (23) and (24), which are  $\dot{x} = \zeta_3$  and  $\dot{y} = \zeta_4$ , to avoid impractical numerical differentiations for such high derivatives.

It also should be mentioned that in order to minimize unwanted instability of AFLNN in practice, the NN-based

PID compensator is set to be activated after one second from start when the response is relatively settled with the base controller.

The learning rate of  $\eta = 0.3$  and momentum rate of  $\alpha = 0.6$  are used for the neural network in AFLNN for the experiments.

### 6.1 Stabilization

Figure 9 shows the experimental results of stabilization of the ball at  $(0.1, -0.1)$  m. The ball was initially placed near the center of the plate when the control starts. As it can be seen in Fig. 9a, there is a steady state error in both  $x$  and  $y$  coordinates with AFL alone as the ball is stabilized at the averaged coordinates of  $(0.073, -0.076)$  m. This error is caused by the simplifications and approximations that have been made on the system model in the design of the AFL base controller as explained in Sect. 3.1. As shown in Fig. 9b, the steady state error becomes almost eliminated when AFLNN is used as the ball is now stabilized at  $(0.093, -0.093)$ .

Figure 10 shows the changes in PID gains. Since the amount of additional torque input needed for the stabilization correction is relatively small, the PID gain updates occur within a small range. The error that AFL alone is not capable of eliminating is being gradually reduced as the PID gains are updated and the gains are also eventually stabilized within a small range as the ball is stabilized at the desired location.

### 6.2 Trajectory tracking

Figure 11 shows the experimental results of the ball following a circular trajectory with  $R = 0.08$  m and  $\omega = 2$  rad/s. Figure 11a is with only AFL and Fig. 11b shows the improved result with AFLNN. It shows that in the steady state the deviation error decreases from 0.0115 to 0.0077 m in terms of the radius of the circular trajectory.

Figure 12 shows that the PID gains by AFLNN are being adjusted online successfully. It shows that the neural network tries to increase the system response by increasing  $K_p$

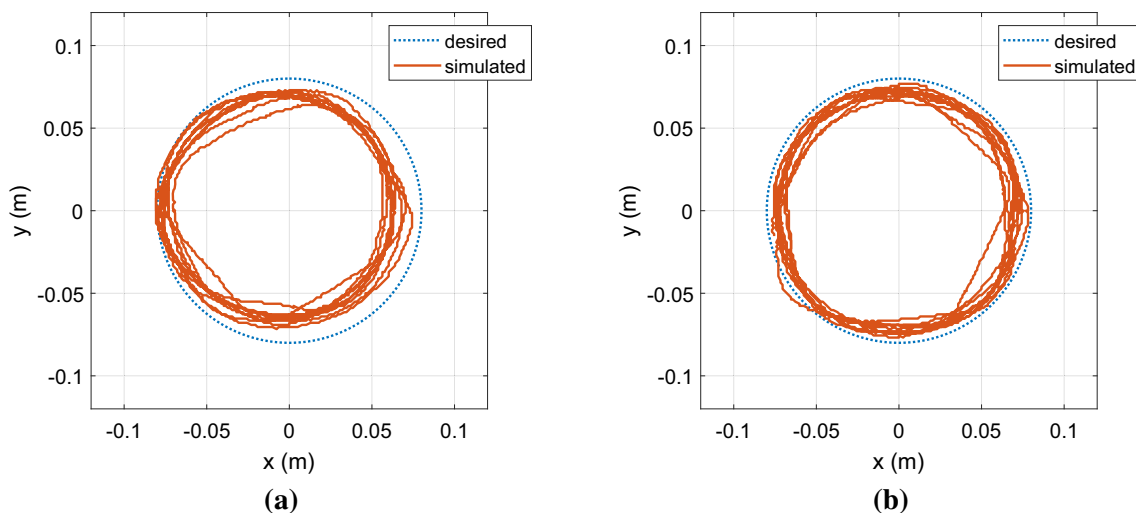


Fig. 11 Experimental results for circular trajectory tracking ( $R = 0.08$  m and  $\omega = 2$  rad/s): **a** AFL alone and **b** AFLNN

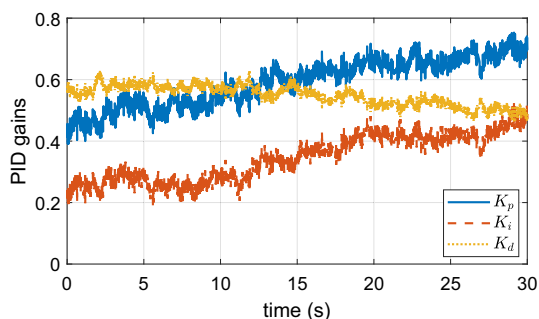


Fig. 12 PID control gains being updated during the trajectory tracking control experiment

while the trajectory tracking accuracy becomes improved by increasing  $K_i$ .

## 7 Conclusion

In order to avoid the difficulties of controller design of nonlinear systems, we proposed a controller consisting of a base linear controller and a neural network-based PID compensator. The controller is structured in a way that the outputs of the two subcontrollers are added together to generate a resultant control input to the system. We implemented the strategy on the ball-plate system and demonstrated the effectiveness of the proposed controller in stabilization and trajectory tracking control through simulations and experiments.

The suggested future research directions include the use of different training algorithms such as Levenberg–Marquardt for better convergence for the neural network’s online training and construction of a portable ball-on-plate system using a small single-board computer such as Raspberry PI.

## Appendix

The expressions for  $\alpha(X)$  and  $\beta(X)$  in Eq. (26) are written as

$$\begin{aligned} \alpha(X) &= A_x^2 m_b \dot{\theta}_y^2 (m_b x \dot{\theta}_y^2 + m_b g \sin \theta_y) \\ &\quad - A_x m_b g \dot{\theta}_y^2 \sin \theta_y + A_x B_x (2m_b \dot{x} \dot{\theta}_y + m_b g \cos \theta_y) \\ &\quad \times (-2m_b x \dot{\theta}_y + m_b g x \cos \theta_y + m_b g r \sin \theta_y) \\ \beta(X) &= A_x B_x (2m_b \dot{x} \dot{\theta}_y + m_b g \cos \theta_y) \end{aligned}$$

where

$$A_x = \frac{1}{m_b + I_b/r^2}, \quad B_x = \frac{1}{m_b x^2 + I_b + I_p}$$

## References

1. Ho MT, Tu YW, Lin HS (2009) Controlling a ball and wheel system using full-state-feedback linearization [focus on education]. *IEEE Control Syst* 29(5):93–101
2. Ryu JC, Ruggiero F, Lynch KM (2013) Control of nonprehensile rolling manipulation: balancing a disk on a disk. *IEEE Trans Robot* 29(5):1152–1161
3. Hauser J, Sastry S, Kokotovic P (1992) Nonlinear control via approximate input–output linearization: the ball and beam example. *IEEE Trans Autom Control* 37(3):392–398
4. Awtar S, Bernard C, Boklund N, Master A, Ueda D, Craig K (2002) Mechatronic design of a ball-on-plate balancing system. *Mechatronics* 12(2):217–228
5. Mochizuki S, Ichihara H (2013) I–PD controller design based on generalized KYP lemma for ball and plate system. In: 2013 European control conference (ECC), pp 2855–2860. IEEE
6. Fan X, Zhang N, Teng S (2004) Trajectory planning and tracking of ball and plate system using hierarchical fuzzy control scheme. *Fuzzy Sets Syst* 144(2):297–312

7. Ho MT, Rizal Y, Chu LM (2013) Visual servoing tracking control of a ball and plate system: design, implementation and experimental validation. *Int J Adv Robot Syst* 10(7):287
8. Xu JX, Hang CC, Liu C (2000) Parallel structure and tuning of a fuzzy PID controller. *Automatica* 36(5):673–684
9. Rubaai A, Castro-Sitiriche MJ, Ofoli AR (2008) Design and implementation of parallel fuzzy PID controller for high-performance brushless motor drives: an integrated environment for rapid control prototyping. *IEEE Trans Ind Appl* 44(4):1090–1098
10. Eski I, Yildirim Ş (2009) Vibration control of vehicle active suspension system using a new robust neural network control system. *Simul Model Pract Theory* 17(5):778–793
11. Thanh TDC, Ahn KK (2006) Nonlinear PID control to improve the control performance of 2 axes pneumatic artificial muscle manipulator using neural network. *Mechatronics* 16(9):577–587
12. Jung S, Cho HT, Hsia TC (2007) Neural network control for position tracking of a two-axis inverted pendulum system: experimental studies. *IEEE Trans Neural Netw* 18(4):1042–1048
13. Jung S, Kim SS (2008) Control experiment of a wheel-driven mobile inverted pendulum using neural network. *IEEE Trans Control Syst Technol* 16(2):297–303
14. Ahn KK, Thanh TDC (2005) Nonlinear PID control to improve the control performance of the pneumatic artificial muscle manipulator using neural network. *J Mech Sci Technol* 19(1):106–115
15. Cong S, Liang Y (2009) PID-like neural network nonlinear adaptive control for uncertain multivariable motion control systems. *IEEE Trans Ind Electron* 56(10):3872–3879
16. Kang J, Meng W, Abraham A, Liu H (2014) An adaptive PID neural network for complex nonlinear system control. *Neurocomputing* 135:79–85
17. Tsai CC, Huang HC, Lin SC (2010) Adaptive neural network control of a self-balancing two-wheeled scooter. *IEEE Trans Ind Electron* 57(4):1420–1428
18. Chen TC, Sheu TT (2002) Model reference neural network controller for induction motor speed control. *IEEE Trans Energy Convers* 17(2):157–163
19. Ge SS, Hang CC, Zhang T (1999) Adaptive neural network control of nonlinear systems by state and output feedback. *IEEE Trans Syst Man Cybern Part B (Cybern)* 29(6):818–828
20. Spong MW, Hutchinson S, Vidyasagar M (2006) *Robot modeling and control*. Wiley, New York
21. Khalil HK (2001) *Nonlinear systems*, 3rd edn. Prentice-Hall, Englewood Cliffs
22. Baldi P (1995) Gradient descent learning algorithm overview: a general dynamical systems perspective. *IEEE Trans Neural Netw* 6(1):182–195