ICIC Express Letters
Part B: Applications
Volume **9**, Number **11**, November **2018**

ICIC International ©2018 ISSN 2185-2766
pp. 1125–1132

INDIRECT ADAPTIVE NEURAL CONTROL USING A HYBRID LEARNING ALGORITHM

Fadwa Damak, Mounir Ben Nasr and Mohamed Chtourou
Control and Energy Management Laboratory (CEMLab)
Department of Electrical Engineering
National School of Engineers of Sfax
BP 1173, Sfax, Tunisia
fadwa.damak@gmail.com; mounir.bennacer@enetcom.usf.tn; Mohamed.Chtourou@enis.rnu.tn

Received June 2018; accepted August 2018

Abstract. *In this paper, we have investigated the application of a hybrid algorithm to indirect adaptive control of nonlinear system using a multilayer perceptron. The proposed hybrid algorithm is employed to reduce the learning speed of neural controller. The overall control scheme includes two neural networks. The first is used to identify the nonlinear system model while the second is the controller. Both of these are adjusted on-line by different errors based on the hybrid algorithm. This algorithm uses Kohonen algorithm to train the parameters of the hidden layer and the gradient descent method to train the parameters of the output layer. Simulation results obtained with both classical backpropagation and hybrid algorithms for two different examples clearly show the advantage of the proposed approach.*

Keywords: Indirect adaptive neural control, Nonlinear system, Gradient descent method, Kohonen algorithm, Hybrid learning

1. Introduction.

Recently, adaptive neural networks have been widely used for various control systems. In process control application neural networks can be employed in the controller in either direct or indirect control design [1-14]. In [1], the authors demonstrated the use of dynamic backpropagation based neural networks for identification and control of nonlinear dynamical systems. In [2], the authors investigated the application of neural networks for tracking based on inverse control. In [3], the authors proposed a fast learning algorithm based on a new cost function and a linearized error signal to control a nonlinear dynamic system using feed forward neural networks. In [4], the authors proposed a variable neural network for adaptive control of nonlinear system. The parameters of the variable neural networks are adjusted by adaptation laws developed using Lyapunov synthesis technique. In [5], the authors proposed an approach for enhancing the estimation of the plant Jacobian which is on-line used in direct adaptive neural inverse control of nonlinear plants with noise. The works in [6] concerns with the adaptive control of continuous-time nonlinear dynamical systems using variable structure neural networks. In this approach the number of the radial basis function (RBF) networks can be either increased or decreased with time to achieve desired control performance. In [7], the authors proposed a neural adaptive control scheme for a class of nonlinear systems with unknown dead zone using backstepping design techniques. In [8], adaptive neural network based backstepping control was proposed for a class of uncertain multi-input multi-output (MIMO) nonlinear systems with input nonlinearities. An adaptive neural network decentralized backstepping out-feedback control for nonlinear large scale system with time delay was presented in [9]. Neural network based adaptive output feedback control for a class of MIMO non-affine system was presented in [10]. In [11], a direct adaptive neural control

DOI: 10.24507/icicelb.09.11.1125

of nonlinear systems was presented. The proposed control scheme incorporates a neural controller and a sliding mode controller. The parameters of neural controller are adjusted with extreme learning machine. An adaptive neural control of stochastic nonlinear system with multiple time varying delays and input saturation was presented in [12]. In [13], the authors investigated an adaptive based neural network for non-linear systems with full state constraints. In [14], the authors proposed an adaptive neural control of uncertain MIMO non-linear systems with state and input constraints. In most cases, the neural network used for control problems is the multilayer perceptron trained with the well-known backpropagation algorithm (BP). Despite its popularity, the main drawback of the basic backpropagation algorithm is its slow convergence rate. Therefore, some works have been proposed in the literature in order to overcome the disadvantage of the traditional backpropagation algorithm. Recently, a hybrid algorithm based on a combination of Kohonen algorithm and gradient descend method is also proposed for performances improvement [15]. In [15], the authors proposed a hybrid method to control a nonlinear dynamic system using feedforward neural network. The control scheme contains two neural networks. One is the neural model and the other is the neural controller. The neural networks parameters are adapted separately and off-line. This algorithm could conceptually be split-up into two steps. The first step is to train the weights connecting the input and the hidden layers by Kohonen algorithm [16], and the second step is to use the gradient descend method [17] to adjust the weights connecting the hidden and output layers. The main characteristic of this hybrid method is to define an appropriate number of neuron in the hidden layer of feedforward neural network, based on clustering method. The number of active hidden neurons directly affects the generalization and training time, which are two important factors in neural controller. This method trains fast and has good generalization performances compared to backpropagation algorithm. This paper extends the work of [15] by using a new indirect adaptive control of nonlinear system. In this design approach the plant parameters are trained on-line and used to calculate the controller parameters. The advantage of this configuration is to reduce the effect of parameter variations in order to raise robustness. The overall control scheme includes two neural networks. The first is used to identify the nonlinear system model while the second is the controller. Both of these are adjusted simultaneously and on-line at each sampling instant by considering two different error functions based on the hybrid algorithm. The paper is organized as follows. In the next section an adaptive control scheme is presented. The simulation results are presented in Section 3. Conclusions are given in the last section.

2. **Analysis and Design of Indirect Adaptive Neural Controller.**

The structure of the overall system for the indiret adaptive neural control of the plant using the neural network controller (NNC) and the neural network model (NNM) is shown in Figure 1. In this study, u is generated by NNC. The control problem is to determine a control input u confined to the plant such that the output y_p follows the desired value y_d as close as possible.

For the control design, the plant can be described as a discrete-time simple-input simple-output (SISO) unknown nonlinear systems:

$$y_p(k+1) = \psi\left[y_p(k), y_p(k-1), \ldots, y_p(k-n_a+1), u(k), u(k-1), \ldots, u(k-n_b+1)\right] \quad (1)$$

where y_p is the output of the process, u is its input, and n_a and n_b are the output and input orders respectively. $\psi[\ldots]$ is a nonlinear function. k represents the system time delay.

Both neural model and controller are feedforward neural networks having one hidden layer shown in Figure 2. The proposed algorithm uses the fusion of Kohonen algorithm [16] and gradient descent method [17].

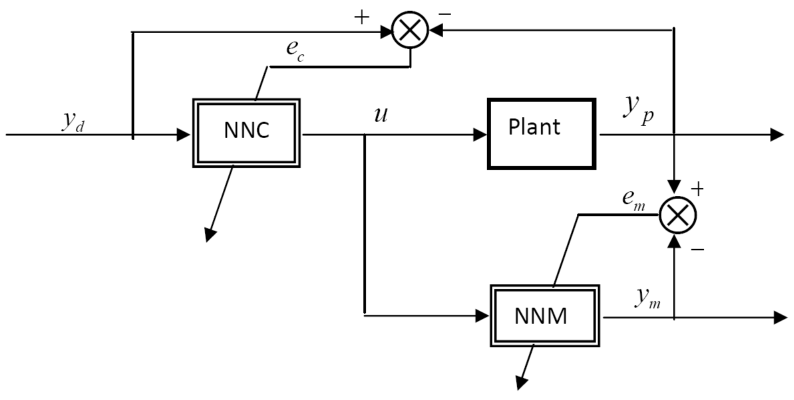

FIGURE 1. Control block diagram of the overall system

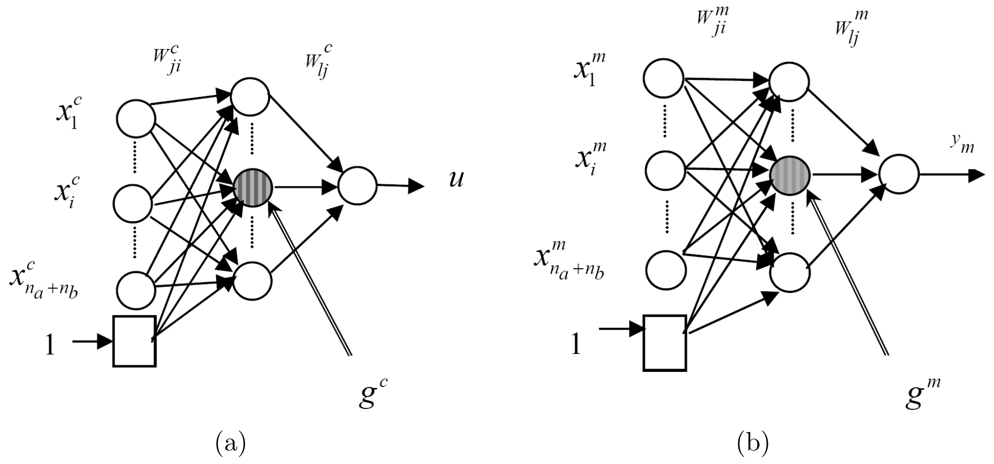

FIGURE 2. Design of the neural networks: (a) neural network controller; (b) neural network model

The on-line adjustment of the two neural networks is summarized as follows.

Let X^c be the input vector of NNC:

$$\begin{aligned} X^c(k) &= \left[x_1^c(k), \ldots, x_{n_a+n_b}^c(k)\right] \\ &= [y_p(k), \ldots, y_p(k-n_a+1), y_d(k+1), u(k-1), \ldots, u(k-n_b+1)] \end{aligned} \tag{2}$$

First, the input X^c is generated. After that, find the index g^c of the winner neuron using the minimum distance Euclidean criterion:

$$g^c = \text{argmin}_j \left\{\left\|X^c - W_j^c\right\|\right\}, \quad \forall j \in \{1 \ldots n_a+n_b\} \tag{3}$$

Next, we have to define a neighborhood set around the winner. There after the output of neural network controller is given by the following equation:

$$u(k) = f\left(\sum_j W_{lj}^{N(g^c,d)}(k) \cdot \left[f\left(\sum_i W_{ji}^{N(g^c,d)}(k) \cdot x_i^c + b_j^c(k)\right)\right] + b^c(k)\right) \tag{4}$$

where the neighborhood $N(g^c, d)$ contains the indices for all of the neurons that lie within a radius d of the winning neuron g^c, $W_{ji}^{N(g^c,d)}$ are the weights of the winner neuron and its neighbors in the hidden layer and $W_{lj}^{N(g^c,d)}$ are the weights of the winner neuron and its neighbors in the output layer. Thereafter, the signal u is applied to the process.

Let

$$\begin{aligned} X^m(k) &= \left[x_1^m(k), \ldots, x_{n_a+n_b}^m(k)\right] \\ &= [y_p(k), \ldots, y_p(k-n_a+1), u(k), \ldots, u(k-n_b+1)] \end{aligned} \tag{5}$$

Similarly, the input X^m is generated. After that, find the index g^m of the winner neuron using the minimum distance Euclidean criterion:

$$g^m = \operatorname{argmin}_j \left\{ \left\| X^m - W_j^m \right\| \right\}, \quad \forall j \in \{1 \ldots n_a + n_b\} \tag{6}$$

Next, we have to define a neighborhood set around the winner. There after the output of neural network model is given by the following equation:

$$y_m(k+1) = f\left(\sum_j W_{lj}^{N(g^m,d)}(k) \left[f\left(\sum_i W_{ji}^{N(g^m,d)}(k) \cdot x_i^m + b_j^m(k) \right) \right] + b^m(k) \right) \tag{7}$$

where the neighborhood $N(g^m, d)$ contains the indices for all of the neurons that lie within a radius d of the winning neuron g^m, $W_{ji}^{N(g^m,d)}$ are the weights of the winner neuron and its neighbors in the hidden layer and $W_{lj}^{N(g^m,d)}$ are the weights of the winner neuron and its neighbors in the output layer.

The error function used to update the parameters of the neural network model is defined as follows:

$$E_m = \frac{1}{2}(e_m)^2 = \frac{1}{2}\left(y_p(k) - y_m(k)\right)^2 \tag{8}$$

For this on-line scheme, the weights of the controller neural network are adapted using the performances criterion defined as follows:

$$E_c = \frac{1}{2}(e_c)^2 = \frac{1}{2}\left(y_d(k) - y_m(k)\right)^2 \tag{9}$$

The weights update of the NNM and NNC is accomplished according the following steps:

Step 1: Modify the hidden weights $W_{ji}^{N(g^m,d)}$ of the NNM in order to move the weights of the winner neuron and its neighbors to the input vector by the Kohonen learning rule:

$$\begin{aligned}\Delta W_{ji}^{N(g^m,d)}(k) &= \alpha^m(k) \cdot \left(x_i^m - W_{ji}^{N(g^m,d)}(k) \right) \\ &= \alpha_0 \cdot \exp\left(-\frac{k}{T}\right) \cdot \left(x_i^m - W_{ji}^{N(g^m,d)}(k) \right)\end{aligned} \tag{10}$$

where α_0 is its initial value, T is the total number of iterations and k is the current iteration.

Step 2: Calculate the output weights $W_{lj}^{N(g^m,d)}$ of NNM using gradient descent method:

$$\begin{aligned}\Delta W_{lj}^{N(g^m,d)}(k) = &-\varepsilon \frac{\partial E_m(w)}{\partial W_{lj}^{N(g^m,d)}} \\ = &-\varepsilon \cdot \left(y_m(k) - y_p(k)\right) \cdot f'\left(\sum_j W_{lj}^{N(g^m,d)} \cdot f\left(\sum W_{ji}^{N(g^m,d)} \cdot x_i^m \right) \right) \\ &\cdot f\left(\sum W_{ji}^{N(g^m,d)} \cdot x_i^m \right)\end{aligned} \tag{11}$$

Step 3: Modify the hidden weights $W_{ji}^{N(g^c,d)}$ of the NNC in order to move the weights of the winning neuron and its neighbors to the input data by the Kohonen learning rule:

$$\Delta W_{ji}^{N(g^c,d)}(k) = \alpha^c(k) \cdot \left(x_i^c - W_{ji}^{N(g^c,d)}(k) \right) = \alpha_0 \cdot \exp\left(-\frac{k}{T}\right) \cdot \left(x_i^c - W_{ji}^{N(g^c,d)}(k) \right) \tag{12}$$

Step 4: Calculate the output weights $W_{lj}^{N(g^c,d)}$ of the NNC using gradient descent method:

$$\Delta W_{lj}^{N(g^c,d)}(k) = -\varepsilon \cdot \left(y_m(k) - y_d(k)\right) J_H(k) \frac{\partial u(k)}{\partial W_{lj}^{N(g^c,d)}(k-1)} \tag{13}$$

where

$$\frac{\partial u(k)}{\partial W_{lj}^{N(g^c,d)}(k-1)} = u(k)\cdot(1-u(k))\cdot f\left(\sum_i W_{ji}^{N(g^c,d)}(k)\cdot x_i^c\right) \tag{14}$$

Moreover, J_H in (13) is defined as:

$$\begin{aligned} J_H(k) &= \frac{\partial y_m(k+1)}{\partial u(k)} \\ &= \sum_j W_{kj}^{N(g^m,d)}\cdot W_{ji}^{N(g^m,d)}\cdot f'\left(\sum_j W_{lj}^{N(g^m,d)}\cdot f\left(\sum W_{ji}^{N(g^m,d)}\cdot x_i^m\right)\right) \\ &\quad \cdot f'\left(\sum W_{ji}^{N(g^m,d)}\cdot x_i^m\right) \end{aligned} \tag{15}$$

3. **Simulation Results.**

To test the performance of the proposed adaptive neural control with hybrid algorithm, and to compare it with adaptive neural control using BP algorithm, simulations of the two nonlinear plants are carried out.

3.1. **The first example.**

Consider the following nonlinear system [1]:

$$y(k+1) = \frac{y(k)}{1+y^2(k)} + A(k)\cdot u^3(k) \tag{16}$$

where

$$A(k) = a_0\cdot\left(1+\sin\left(\frac{\pi\cdot k}{100}\right)\right) \tag{17}$$

For the tracking problem, we consider the desired output to be smooth multi-step signal or sine wave. The structures of both NNC and NNM consist of 2 inputs, 5 hidden neurons and one output neuron. The initial weights of both neural networks are selected randomly between -2 and 2. The parameters used in simulation are: $\varepsilon = 0.5$, $\alpha_0 = 0.8$, $a_0 = 0.5$, $d = 2$.

Figure 3 shows the desired output y_d and the system output y in the case of a multi-step signal. The results of sine wave signal are shown in Figure 4.

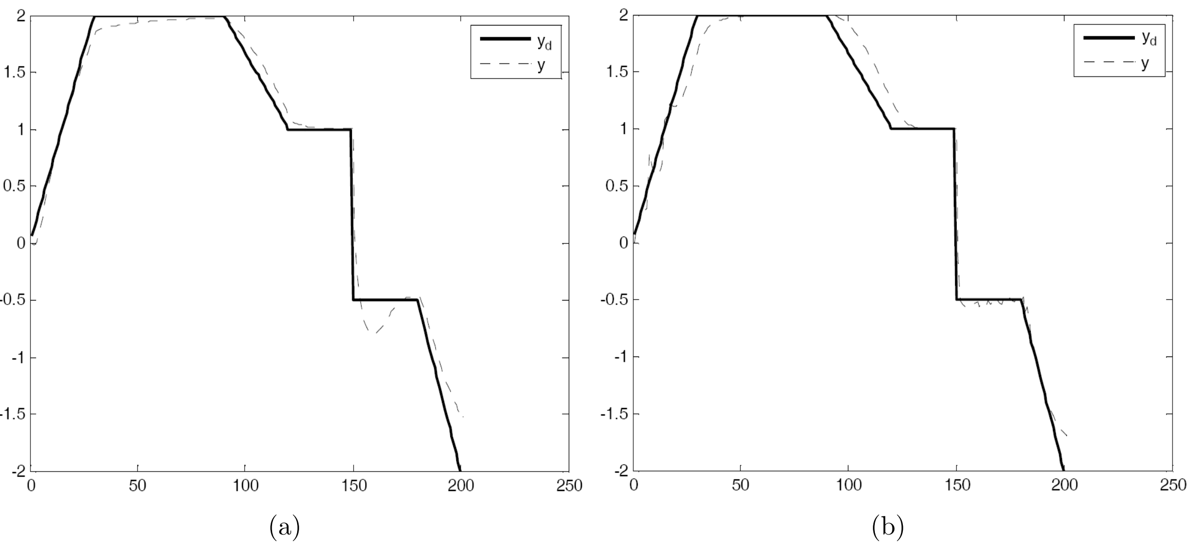

FIGURE 3. Evolution of y and y_d for Example 1 (multi-step signal): (a) BP method and (b) hybrid method

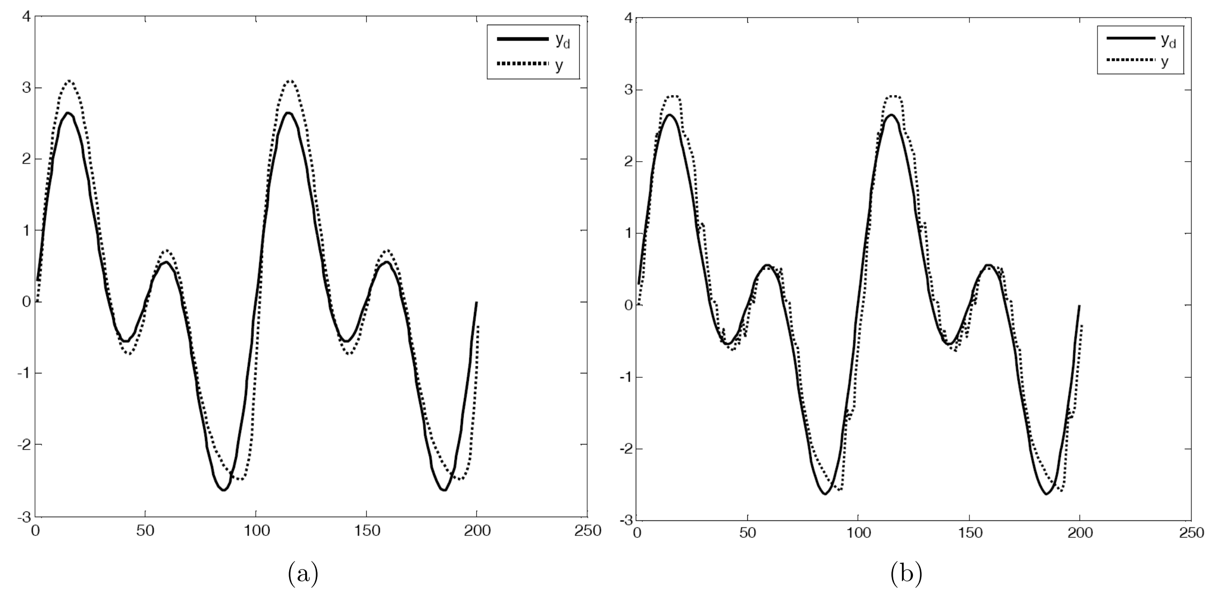

FIGURE 4. Evolution of y and y_d for Example 1 (sine wave signal): (a) BP method and (b) hybrid method

3.2. **The second example.** Consider a continuous stirred tank reactor (CSTR) system model. A polynomial ARMA model for this system can be represented as follows [18]:

$$y(k+1) = \theta_0 + \theta_1 u(k) + \theta_2 y(k) + \theta_3 u^3(k) + \theta_4 y(k-1)u(k-1)u(k) \tag{18}$$

The parameters are given by:

$$\theta_0 = 0.558 \quad \theta_1 = 0.538 \quad \theta_2 = 0.116 \quad \theta_3 = -0.127 \quad \theta_4 = -0.034$$

y is the output of the process and the input u is the dilution rate around the operating point. The objective is to control y by manipulating u. The system is excited by a control input u of random amplitude in the range $[0, 1]$. The neuro-controller NNC consists of four input neurons, 5 hidden neurons and one output neuron. The NNC and the NNM used here have the same structure. The parameters used in simulation are: $\varepsilon = 0.5$, $\alpha_0 = 0.8$, $d = 2$. The classical approach by using backpropagation algorithm and our proposed approach have been applied to the control of above plant.

Figure 5 shows the desired output y_d and the system output y in the case of a multi-step signal. The results of sine wave signal are shown in Figure 6.

3.3. **Discussion and results.** The results as seen in Figures 5 and 6 show good asymptotic set point tracking with minimal offsets at all points. From Table 1, it can be seen that the proposed method achieves a better accuracy when compared with BP algorithm because the error criterion Ec has been reduced for the proposed method. In addition, the training time of the proposed algorithm is less than the training time of traditional method. Inclusion of neighborhood approach in training algorithm of neural networks implies minimum number of parameters and speeds up the convergence of neural controller.

4. **Conclusion.** In this paper, we investigated a hybrid algorithm in the context of indirect adaptive control of nonlinear system. Simulation results show that the proposed indirect adaptive control based on neural network using hybrid algorithm improves the principal characteristic of the controller for a nonlinear system compared to the standard backpropagation. This algorithm consists of the use of Kohonen algorithm coupled with gradient descent method. One of the future works is the study of the stability of the proposed control method to extend the performance of our methodology.

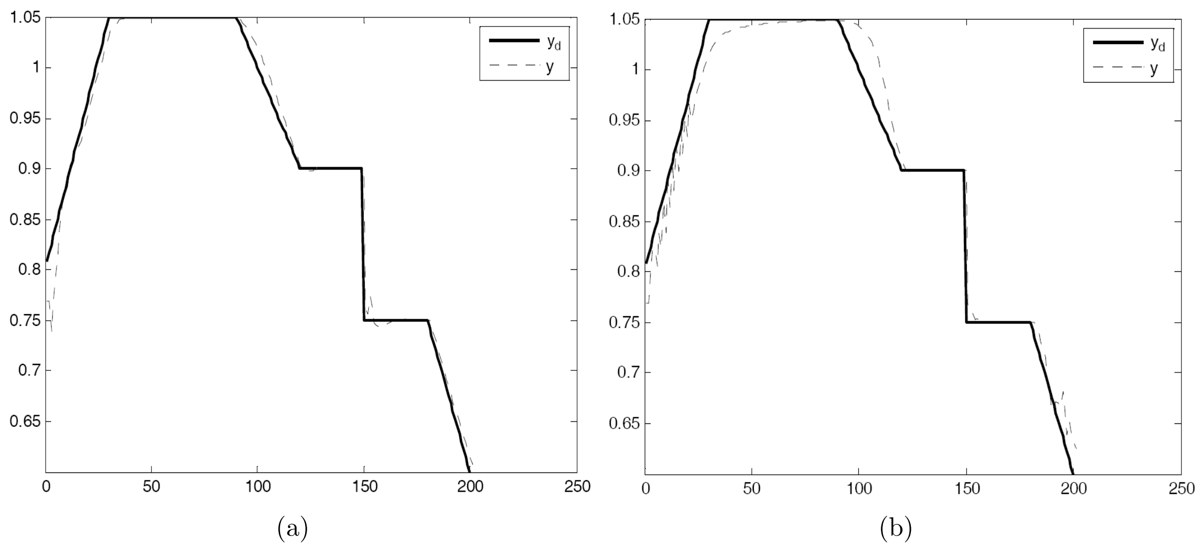

FIGURE 5. Evolution of y and y_d for Example 2 (multi-step signal): (a) BP method and (b) hybrid method

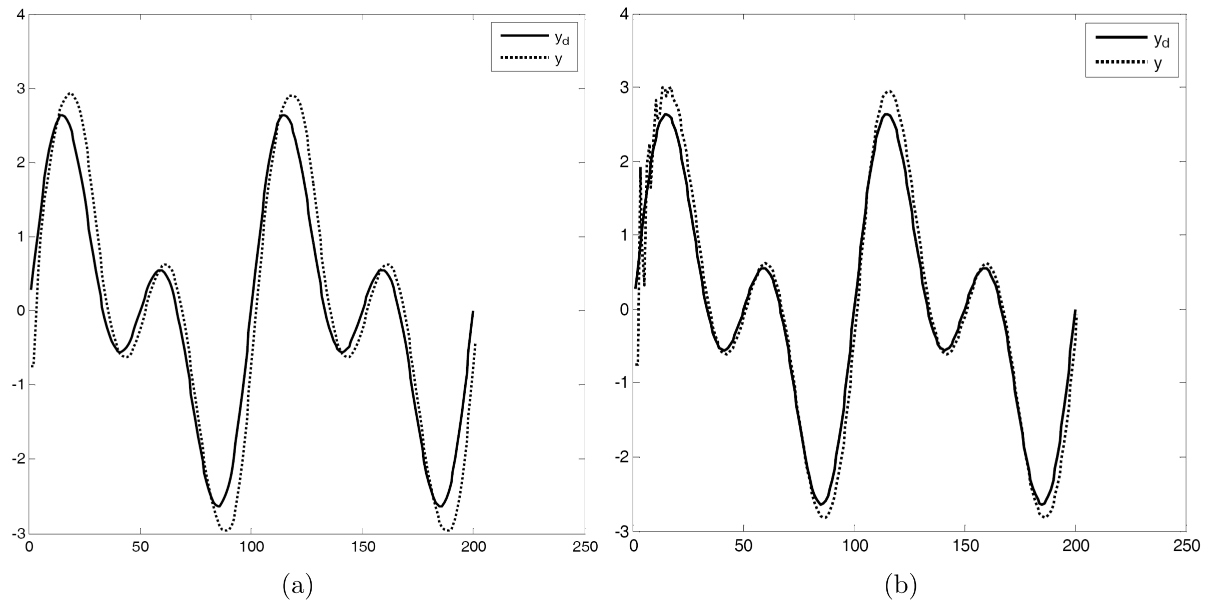

FIGURE 6. Evolution of y and y_d for Example 2 (sine wave signal): (a) BP method and (b) hybrid method

TABLE 1. Comparative results of learning methods

Example	Desired trajectory	Learning method	Ec (Criterion)	Training time (s)
Example 1	Multi-step	Backpropagation	0.123	41.823
		Proposed	0.080	37.975
	Sine wave	Backpropagation	0.18	39.5
		Proposed	0.09	29.6
Example 2	Multi-step	Backpropagation	0.254	66.246
		Proposed	0.190	50.237
	Sine wave	Backpropagation	0.30	46
		Proposed	0.23	40

Acknowledgment. The authors thank the reviewers for their helpful and constructive comments.

REFERENCES

[1] K. S. Narendra and K. Parthasarathy, Identification and control of dynamical systems using neural networks, *IEEE Trans. Neural Networks*, vol.1, no.1, pp.4-27, 1990.

[2] B. Widrow and E. Walach, *Adaptive Inverse Control*, Prentice-Hall, Englewood Cliffa, NJ, 1996.

[3] G. J. Jeon and I. Lee, Neural network indirect adaptive control with fast learning algorithm, *Neurocomputing*, vol.13, pp.185-199, 1996.

[4] G. P. Liu, V. Kadirkamanathan and S. A. Billings, Variable neural networks for adaptive control of nonlinear system, *IEEE Trans. Systems, Man and Cybernetic*, vol.29, no.1, pp.34-43, 1999.

[5] D. Wang and P. Bao, Enhancing the estimation of plant jacobian for adaptive neural inverse control, *Neurocomputing*, vol.34, pp.99-115, 2000.

[6] H. Mekki, M. Chtourou and N. Derbel, Variable structure neural networks for adaptive control of nonlinear systems using the stochastic approximation, *Simulation Modelling Practice and Theory*, vol.14, pp.1000-1009, 2006.

[7] Z. Wang, Y. Zhang and H. Fang, Neural adaptive control for a class of nonlinear system with unknown deadzonem, *Neural Computing & Applications*, vol.17, pp.339-345, 2008.

[8] M. Chen, S. S. Ge and B. V. E. How, Robust adaptive neural network control for a class of uncertain MIMO nonlinear systems with input nonlinearities, *IEEE. Trans. Neural Networks*, vol.21, no.5, pp.796-812, 2010.

[9] S. C. Tong, Y. M. Li and H. G. Zhang, Adaptive neural network decentralized backstepping output-feedback control for nonlinear large scale systems with time delays, *IEEE Trans. Neural Networks*, vol.22, no.7, pp.1073-1086, 2011.

[10] T. Zhao and F. Fan, Neural network-based adaptive output control for MIMO non-affine systems, *Neural Computing & Applications*, vol.21, no.1, pp.145-151, 2012.

[11] H. J. Rong and G. S. Zhao, Direct adaptive neural control of nonlinear systems with exteme learning machine, *Neural Computing & Applications*, vol.22, pp.577-586, 2013.

[12] G. Cui, T. Jiao, Y. Wei, G. Song and Y. Chu, Adaptive neural control of stochastic nonlinear systems with multiple time-varying delays and input saturation, *Neural Computing & Applications*, vol.25, nos.3-4, pp.779-791, 2014.

[13] Y. J. Liu, J. Li, S. Tong and C. L. P. Chen, Neural network control-based adaptive learning design for nonlinear systems with full-state constraints, *IEEE Trans. Neural Networks*, vol.27, no.7, pp.1562-1571, 2016.

[14] Z. Chen, Z. Li and C. L. Philip, Adaptive neural control of uncertain MIMO nonlinear systems with state and input constraints, *IEEE Trans. Neural Networks*, vol.28, no.3, pp.1318-1330, 2017.

[15] M. B. Nasr and M. Chtourou, Neural network control of nonlinear dynamic systems using hybrid algorithm, *Applied Soft Computing*, vol.24, pp.423-431, 2014.

[16] T. Kohonen, The self-organizing map, *Proc. of the IEEE*, vol.78, no.9, pp.1464-1480, 1990.

[17] D. E. Rumelhart, G. E. Hinton and R. J. Williams, Learning internal representations by error propagation, *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, vol.1, pp.318-362, 1986.

[18] E. Hernandez and Y. Arkun, Stability of nonlinear polynomial ARMA models and their inverse, *International Journal of Control*, vol.63, pp.885-906, 1996.