International Journal of Automation and Computing 16(2), April 2019, 213-225
DOI: 10.1007/s11633-017-1062-2

Robust Neural Control of Discrete Time Uncertain Nonlinear Systems Using Sliding Mode Backpropagation Training Algorithm

Imen Zaidi Mohamed Chtourou Mohamed Djemel
Control & Energy Management Laboratory, National School of Sfax Engineers, University of Sfax, Sfax 3038, Tunisia

Abstract: This work deals with robust inverse neural control strategy for a class of single-input single-output (SISO) discrete-time nonlinear system affected by parametric uncertainties. According to the control scheme, in the first step, a direct neural model (DNM) is used to learn the behavior of the system, then, an inverse neural model (INM) is synthesized using a specialized learning technique and cascaded to the uncertain system as a controller. In previous works, the neural models are trained classically by backpropagation (BP) algorithm. In this work, the sliding mode-backpropagation (SM-BP) algorithm, presenting some important properties such as robustness and speedy learning, is investigated. Moreover, four combinations using classical BP and SM-BP are tested to determine the best configuration for the robust control of uncertain nonlinear systems. Two simulation examples are treated to illustrate the effectiveness of the proposed control strategy.

Keywords: Discrete time uncertain nonlinear systems, neural modelling, sliding mode, backpropagation (BP) algorithm, robust neural control.

1 Introduction

Robust control of discrete time uncertain nonlinear systems is a challenging task. Many studies have been proposed towards finding a controller that guarantees stability, robustness and satisfactory tracking performance. These ones include H_∞ control[1, 2], robust adaptive control[3, 4], robust predictive control[5, 6] and sliding mode control[7, 8]. Moreover, neural networks have been proven useful and effective for the modeling and the control of a wide class of uncertain nonlinear systems due to their universal approximation capabilities. Thus, during the last decade, several studies dealing with the neural modeling of nonlinear system affected by uncertainties have been proposed[9−13]. Furthermore, robust control based on neural networks has attracted an ever increasing interest[14−17]. For example, in [18], a stable robust neural adaptive control scheme was proposed to achieve performance for a class of unknown single-input single-output (SISO) system with external disturbances. Later, Wang et al.[19] designed a new adaptive neuro-fuzzy sliding mode controller with tracking performance for uncertain nonlinear systems to attenuate the effects of unmodeled dynamics, disturbances and approximation errors. Internal model control (IMC) is also considered as a robust control technique. Indeed, Deng et al.[20] presented a novel neural internal model control for unknown non-affine discrete-time multi-input multi-output (MIMO) processes under model mismatch and disturbances. Of the many architectures associated with the neural based control schemes, the principal used type is the feedforward neural network (FNN)[21, 22] which is often trained by the popular backpropagation (BP) learning algorithm[23]. Several improvements were introduced to speed up and robustify this algorithm[24−27]. However, the performance of neural networks structure trained by backpropagation algorithm is not completely robust in the presence of strong external disturbances and parametric variations. The high performance of sliding mode control (SMC) in handling uncertainties and imprecision has motivated its use in training FNN[28]. Sira-Ramirez and Colina-Morles[29] proposed one of the earliest studies that suggests the use of sliding mode for adaptive learning in Adaline neural networks. This approach was further extended in [30] by removing the requirement of a priori knowledge of the upper bounds of bounded signals. In addition, this approach was extended to a more general class of multilayer networks with scalar output[31, 32].

The first sliding mode learning algorithm considered for training FNN was proposed by Parma et al.[33] It differs from the algorithms presented in [29−32], by the use of separate sliding surfaces for each network layer. Parma et al.[34] developed an approach that is slightly different from the one suggested in [33] in order to speed up the convergence of the standard backpropagation algorithm.

Motivated by the results presented in [34], a robust training method of a direct neural model (DNM) and an inverse neural model (INM) will be proposed in this paper for the identification and the control of a class of single-input single-output discrete-time nonlinear systems affected

Research Article
Manuscript received November 20, 2015; accepted August 4, 2016; published online April 19, 2017
Recommended by Guest Editor Ding-Li Yu

by parameter uncertainties.

Indeed, the use of the sliding mode-backpropagation (SM-BP) algorithm in this work is not only to achieve high speed learning but also to develop a robust neural control strategy for the uncertain nonlinear system. According to the control scheme, a DNM is firstly used to emulate the behavior of the system, then an INM is synthesized and cascaded to the uncertain system as a robust controller. In the design step of the INM and the DNM networks, the specialized learning approach has been considered. The training of the DNM then the INM structures has been accomplished by four combinations of the two learning techniques BP and SM-BP (BP-BP, SMBP-BP, BP-SMBP and SMBP-SMBP). The rest of paper is organized as follows. Section 2 presents neural models training using BP and SM-BP. In Section 3, different combinations of the presented learning algorithms are tested to determine the best configuration for the robust neural control of an uncertain nonlinear system. In Section 4, simulation examples are provided to show the proposed control strategy′s performance and comparisons between the two training algorithms are shown. Finally, conclusions are given in Section 5.

2 Neural modelling

Consider the following single-input single-output uncertain nonlinear system expressed by

$$y(k+1) = F[y(k), \cdots, y(k-n+1), u(k), u(k-1), \cdots, u(k-m+1), p] \quad (1)$$

where $y(k)$ and $u(k)$ denote respectively the output and the input of the system, n is the order of $y(k)$, m is the order of $u(k)$, F is an unknown nonlinear function to be estimated by a neural network and p is an uncertain parameters vector. A DNM is used to learn the behavior of system (1).

2.1 Direct neural model

The DNM builds a nonlinear function that estimates the output of the system through old data of its inputs and outputs. Two approaches often discussed in the literature are the series parallel model and the parallel one[13]. In this work, we are interested in series parallel model.

2.2 Inverse neural model

The inverse neural model can be generally presented in the following form:

$$u(k) = F^{-1}[y(k+1), \cdots, y(k-n+1), u(k), \cdots, u(k-m+1), p]. \quad (2)$$

To develop the INM, a neural network is trained to approximate the inverse behavior of the system.

2.3 Neural models training

The learning processes of the DNM and the INM is accomplished, firstly through the BP algorithm and secondly by the SM-BP one.

2.3.1 DNM training

The block diagram of the DNM training process is shown in Fig. 1. The output of the DNM for system (1) can be expressed as

$$y_m(k+1) = \hat{F}[y(k), \cdots, y(k-n+1), u(k-1), \cdots, u(k-m+1), p] \quad (3)$$

where y_m and $\hat{F}$ denote respectively the output of the DNM and the estimate of F.

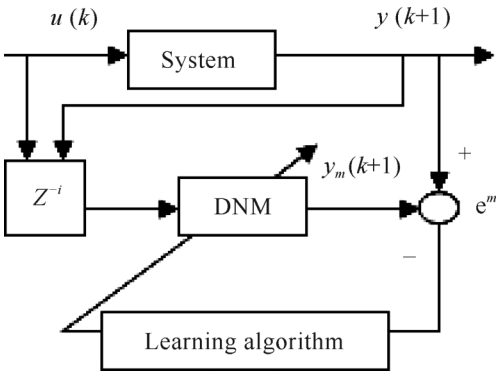

Fig. 1 DNM training

The weights of the DNM are adjusted to minimize the cost function expressed as

$$J = \frac{1}{2}[\mathrm{e}^m]^2 \quad (4)$$

where $\mathrm{e}^m = y(k+1) - y_m(k+1)$ is the error between the output of the system $y(k+1)$ and the one of the DNM $y_m(k+1)$.

BP algorithm for training the DNM

For the output layer, the following weight adjustment rule is defined as

$$W_{jh}^m(k+1) = W_{jh}^m(k) + \varepsilon.\mathrm{e}^m(k+1).f^{m'}[V_j^m(k)].Y_{Hh}^m(k) \quad j = 1,\ h = 1, \cdots, N_c^m. \quad (5)$$

The index m refers to DNM′s parameters, W_{jh}^m represents the weight between the output node j and the hidden node h, $f^{m'}$ denotes the derivative of the output activation function, V_j^m is the global input of the output node j, Y_{Hh}^m is the output of the hidden node h, ε is the learning rate and N_c^m represents the number of neurons in the hidden layer.

For the hidden layer, the weight adjustment is described as

$$Z_{hi}^m(k+1) = Z_{hi}^m(k) + \varepsilon.\mathrm{e}^m(k+1).f^{m'}[V_j^m(k)].W_{jh}^m(k).f_H^{m'}[R_h^m(k)].T_i^m(k) \quad \text{with} \quad i = 1, \cdots, (n+m+1) \quad (6)$$

where Z_{hi}^m represents the weight between the hidden node h and the input node i, $f_H^{m'}$ denotes the derivative of the hidden activation function, R_h^m is the global input of the hidden node h and T_i^m is the input of the input node i.

SM-BP algorithm for training the DNM

The SM-BP equations as defined by Parma et al.[33, 34] are presented by the following equations. In these equations, the sliding surfaces $S_j^m(k)$ and $S_{Hh}^m(k)$, based on sliding mode theory (SMT), are designed to force the weight trajectories into the sliding manifold.

For the node j from the output layer, the sliding surface is defined as[35]

$$S_j^m(k) = X_{2j}^m(k) + C_0.X_{1j}^m(k) \tag{7}$$

where j is the output node, $C_0 > 0$ and

$$X_{1j}^m(k) = [y(k) - y_m(k)].f^{m'}[V_j^m(k)] \tag{8}$$

$$X_{2j}^m(k) = X_{1j}^m(k) - X_{1j}^m(k-1). \tag{9}$$

For the node h from the hidden layer, the following sliding surface is defined as

$$S_{Hh}^m(k) = X_{2Hh}^m(k) + C_{0H}.X_{1Hh}^m(k) \tag{10}$$

where h is the hidden node, $C_{0H} > 0$ and

$$X_{1Hh}^m(k) = X_{1j}^m(k).W_{jh}^m(k).f_H^{m'}[R_h^m(k)] \tag{11}$$

$$X_{2Hh}^m(k) = X_{1Hh}^m(k) - X_{1Hh}^m(k-1). \tag{12}$$

Thus, the weights update equations based on SM-BP are given as

$$\Delta W_{jh}^m(k) = \alpha_m.\mathrm{sgn}[S_j^m(k)].|X_{1j}^m(k)|.Y_{Hh}^m(k) \tag{13}$$

$$\Delta Z_{hi}^m(k) = \beta_m.\mathrm{sgn}[S_{Hh}^m(k)].|X_{1Hh}^m(k)|.T_i^m(k) \tag{14}$$

where $\alpha_m > 0$, $\beta_m > 0$, and defined as follows:

The limits for gain $\boldsymbol{\alpha_m}$ and $\boldsymbol{\beta_m}$

The upper limits of α_m and β_m can be obtained from the sliding mode condition. For a given surface S, delimited by the training data and the network topology, the upper limits for the gains α_m and β_m can be easily obtained[36]. According to Utkin[35], the condition for existence of sliding mode and system stability is defined by

$$S\frac{\mathrm{d}s}{\mathrm{d}t} < 0. \tag{15}$$

For discrete time case, Sarpturk et al.[37] defined (16), instead of (15) as the necessary and sufficient condition to guarantee the sliding manifold.

$$|S(k)| < |S(k-1)|. \tag{16}$$

For the output layer, the substitution of (8) and (9) in (7), results in the equation as

$$\begin{aligned} S_j^m(k) =& f^{m'}[V_j^m(k)].[C_0(y(k) - y_m(k)) + \\ & (\Delta y(k) - \Delta y_m(k))] + \Delta f^{m'}[V_j^m(k)]. \\ & [y(k-1) - y_m(k-1)]. \end{aligned} \tag{17}$$

For the general case, $\Delta f(k) = f(k) - f(k-1)$.

Considering (13) and using the approximation $\frac{\partial y_m}{\partial W_{jh}^m} = \frac{\Delta y_m}{\Delta W_{jh}^m}$, (18) could be obtained as

$$\begin{aligned} \Delta y_m(k) =& f^{m'}[V_j^m(k)].\alpha_m.\mathrm{sgn}[S_j^m(k)]. \\ & |X_{1j}^m(k)|.[Y_{Hh}^m(k)]^2. \end{aligned} \tag{18}$$

Substituting (18) into (17) yields

$$S_j^m(k) = -\alpha_m.\mathrm{sgn}[S_j^m(k)].B_j(k) + A_j(k) \tag{19}$$

where $B_j(k)$ and $A_j(k)$ are defined by

$$B_j(k) = [f^{m'}(V_j^m(k)).Y_{Hh}^m(k)]^2.|X_{1j}^m(k)| \tag{20}$$

$$\begin{aligned} A_j(k) =& f^{m'}[V_j^m(k)].[C_0(y(k) - y_m(k)) + \\ & \Delta y(k)] + \Delta f^{m'}[V_j^m(k)].[y(k-1) - \\ & y_m(k-1)]. \end{aligned} \tag{21}$$

The application of the condition expressed by (16) for the sliding surface (19) gives the limits of α_m defined by[34, 38]

$$0 < \alpha_m < \min\Big\{\frac{|A_j(k)|}{B_j(k)}, \frac{|A_j(k-1)| - |A_j(k)|}{B_j(k-1) - B_j(k)}\Big\}. \tag{22}$$

For the hidden layer, the substitution of (11) into (12) leads to (23).

$$\begin{aligned} X_{2Hh}^m(k) =& f_H^{m'}[R_h^m(k)].[(\Delta y(k) - \Delta y_m(k)). \\ & f^{m'}(V_j^m(k)).W_{jh}^m(k) + (y(k-1) - \\ & y_m(k-1)).(\Delta f^{m'}(V_j^m(k)).W_{jh}^m(k) + \\ & f^{m'}(V_j^m(k)).\Delta W_{jh}^m(k) - \Delta f^{m'}(V_j^m(k)). \\ & \Delta W_{jh}^m(k))] + \Delta f_H^{m'}[R_h^m(k)].[(y(k-1) - \\ & y_m(k-1)).f^{m'}(V_j^m(k-1)).W_{jh}^m(k-1)]. \end{aligned} \tag{23}$$

Considering (14) and using the approximation $\frac{\partial y_m}{\partial Z_{hi}^m} = \frac{\Delta y_m}{\Delta Z_{hi}^m}$, (24) could be obtained as

$$\begin{aligned} \Delta y_m(k) =& f^{m'}[V_j^m(k)].W_{jh}^m(k).f_H^{m'}[R_h^m(k)].\beta_m. \\ & \mathrm{sgn}[S_{Hh}^m(k)].|X_{1Hh}^m(k)|.[T_i^m(k)]^2. \end{aligned} \tag{24}$$

Substituting (11), (23) and (24) into (10) leads to (25) as

$$S_{Hh}^m(k) = -\beta_m.\mathrm{sgn}[S_{Hh}^m(k)].D_{Hh}(k) + N_{Hh}(k) \tag{25}$$

where $D_{Hh}(k)$ and $N_{Hh}(k)$ are defined by

$$\begin{aligned} D_{Hh}^m(k) =& [f^{m'}(V_j^m(k)).W_{jh}^m(k).f_H^{m'}(R_h^m(k)) \\ & T_i^m(k)]^2.|X_{1Hh}^m(k)| \end{aligned} \tag{26}$$

$$\begin{aligned} N_{Hh}^m(k) =& f_H^{m'}[(R_h^m(k))].[\Delta y(k).f^{m'}(V_j^m(k)). \\ & W_{jh}^m(k) + C_{0H}.X_{1j}^m(k).W_{jh}^m(k) + (y(k-1) - \\ & y_m(k-1)).(\Delta f^{m'}(V_j^m(k)).W_{jh}^m(k) + \\ & f^{m'}(V_j^m(k)).\Delta W_{jh}^m(k) - \Delta f^{m'}(V_j^m(k)). \\ & \Delta W_{jh}^m(k))] + \Delta f_H^{m'}[(R_h^m(k))].[(y(k-1) - \\ & y_m(k-1))f^{m'}(V_j^m(k-1)).W_{jh}^m(k-1)]. \end{aligned} \tag{27}$$

The application of the condition expressed by (16) for the sliding surface (25) gives the limits of β_m expressed as[34, 38]

$$0 < \beta_m < \min\left\{\frac{|N_{Hh}(k)|}{D_{Hh}(k)}, \frac{|N_{Hh}(k-1)| - |N_{Hh}(k)|}{D_{Hh}(k-1) - D_{Hh}(k)}\right\}. \tag{28}$$

The limit values of α_m and β_m given by (22) and (28) guarantee the existence and the convergence of sliding surfaces described by (7) and (10). The weight updating rules defined by (13) and (14) can be used to train the DNM taking into consideration these limits for the gains. Any values within the two ranges are acceptable to guarantee the convergence of sliding surfaces and, consequently, the convergence of the training algorithm[34].

2.3.2 INM training

In the literature, the most common proposed process for training inverse neural models are the generalized method and the specialized one[39] shown in Figs. 2 and 3, respectively.

Generalized training process

In the generalized training, the INM is trained offline to minimize the following criterion:

$$J = \frac{1}{2}[u(k) - u^{\mathrm{e}}(k)]^2. \tag{29}$$

Such structure has two major drawbacks. The first one relates to the approximation procedure which does not take into account the desired control signals, while for the second one, an incorrect inverse model will be obtained if the system parameters are variable[40].

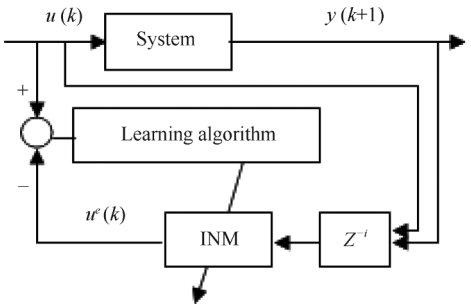

Fig. 2 Generalized method for INM training

To overcome such weaknesses, the specialized training approach is adopted in this work.

Specialized training process

To train the INM, the specialized training structure[39] is considered. Indeed, based on the DNM, which gives good representation of the system after satisfactory training, the INM is trained as shown in Fig. 3.

The cost function to be minimized in the training step is given by

$$J_c = \frac{1}{2}[\mathrm{e}^c]^2 \tag{30}$$

where $\mathrm{e}^c = y^d(k+1) - y(k+1)$ is the error between the output of the DNM $y(k+1)$ and the desired one $y^d(k+1)$.

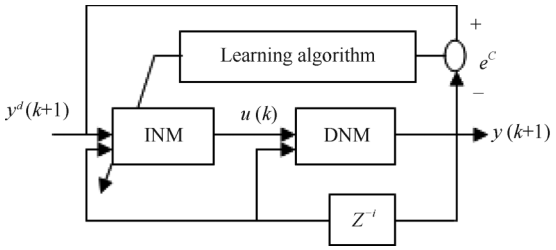

Fig. 3 Specialized method for INM training

BP algorithm for the training of the INM

For the output layer, the weight adjustment is computed as follows:

$$W_{jh}^c(k+1) = W_{jh}^c(k) - \varepsilon.\frac{\partial J_c(k+1)}{\partial y(k+1)}.\frac{\partial y(k+1)}{\partial W_{jh}^c(k)} \tag{31}$$

where W_{jh}^c is the weight between the output node j and the hidden node h of the INM.

$$\frac{\partial J_c(k+1)}{\partial y(k+1)} = -[y^d(k+1) - y(k+1)] \tag{32}$$

$$\frac{\partial y(k+1)}{\partial W_{jh}^c(k)} = \frac{\partial y(k+1)}{\partial u(k)}.\frac{\partial u(k)}{\partial W_{jh}^c(k)} \tag{33}$$

$$\frac{\partial y(k+1)}{\partial u(k)} = \gamma_h^m(k) = \sum_{h=1}^{N_c^m - 1} f^{m'}[V_j^m(k)].\, W_{jh}^m(k).f_{Hh}^{m'}[R_h^m(k)].Z_{h1}^m(k) \tag{34}$$

$$\frac{\partial u(k)}{\partial W_{jh}^c(k)} = f^{c'}[V_j^c(k)].Y_{Hh}^c(k). \tag{35}$$

The index c refers to INM′s parameters, $f^{c'}$ denotes the derivative of the output activation function, V_j^c is the global input of the output node j, Y_{Hh}^c represents the output of the hidden node h and N_c^m is the number of the DNM′s hidden neurons.

For the hidden layer, the weight adjustment is defined as

$$Z_{hi}^c(k+1) = Z_{hi}^c(k) - \varepsilon.\frac{\partial J_c(k+1)}{\partial y(k+1)}.\frac{\partial y(k+1)}{\partial Z_{hi}^c(k)} \tag{36}$$

where Z_{hi}^c is the INM′s weight between the hidden node h and the input node i.

$$\frac{\partial y(k+1)}{\partial Z_{hi}^c(k)} = \frac{\partial y(k+1)}{\partial u(k)}.\frac{\partial u(k)}{\partial Z_{hi}^c(k)} \tag{37}$$

$$\frac{\partial u(k)}{\partial Z_{hi}^c(k)} = f^{c'}[V_j^c(k)].W_{jh}^c(k).f_H^{c'}[R_H^{c'}(k)].T_i^c(k) \tag{38}$$

where $f_H^{c'}$ is the derivative of the hidden activation function, R_H^c denotes the global input of the hidden node h and T_i^c represents the input of the input node i of the INM.

SM-BP algorithm for the training of the INM

Based on the SM-BP algorithm, updating rules for adjusting the INM′s weights are expressed by the following equations:

Let the sliding surface for the node j from the output layer be defined by

$$S_j^c(k) = X_{2j}^c(k) + C_1.X_{1j}^c(k) \tag{39}$$

where j is the output node, $C_1 > 0$.

$$X_{1j}^c(k) = [y^d(k+1) - y(k+1)].\gamma_h^m(k).f^{c'}[V_j^c(k)] \tag{40}$$

where the term $\gamma_h^m(k)$ has been defined by (34).

$$X_{2j}^c(k) = X_{1j}^c(k) - X_{1j}^c(k-1). \tag{41}$$

For the node h from the hidden layer, the sliding surface is expressed as follows:

$$S_{Hh}^c(k) = X_{2Hh}^c(k) + C_{1H}.X_{1Hh}^c(k) \tag{42}$$

where h is the hidden node, $C_{1H} > 0$.

$$X_{1Hh}^c(k) = X_{1j}^c(k).W_{jh}^c(k).f_H^{c'}[R_h^c(k)] \tag{43}$$

$$X_{2Hh}^c(k) = X_{1Hh}^c(k) - X_{1Hh}^c(k-1). \tag{44}$$

Thus, the weights update equations of the INM based on the SM-BP algorithm are given as

$$\Delta W_{jh}^c(k) = \alpha_c.\mathrm{sgn}[S_j^c(k)].|X_{1j}^c(k)|.Y_{Hh}^c(k) \tag{45}$$

$$\Delta Z_{hi}^c(k) = \beta_c.\mathrm{sgn}[S_{Hh}^c(k)].|X_{1Hh}^c(k)|.T_i^c(k). \tag{46}$$

The limits for gain $\boldsymbol{\alpha_c}$ and $\boldsymbol{\beta_c}$

For the output layer, substituting (40) and (41) into (39) leads to (47).

$$\begin{aligned} S_j^c(k) =& f^{c'}[V_j^c(k)].\gamma_h^m(k).[C_1.(y^d(k+1)- \\ & y(k+1)) + (\Delta y^d(k+1) - \Delta y(k+1))]+ \\ & [y^d(k) - y(k)].[f^{c'}(V_j^c(k)).\Delta\gamma_h^m(k)+ \\ & \Delta f^{c'}(V_j^c(k)).\gamma_h^m(k) - \Delta f^{c'}(V_j^c(k)). \\ & \Delta\gamma_h^m(k)]. \end{aligned} \tag{47}$$

Considering (45) and using the approximation $\frac{\partial y}{\partial W_{jh}^c} = \frac{\Delta y}{\Delta W_{jh}^c}$, (48) is obtained as

$$\begin{aligned} \Delta y(k+1) =& \gamma_h^m(k).f^{c'}[V_j^c(k)].\alpha_c.\mathrm{sgn}[S_j^c(k)]. \\ & |X_{1j}^c(k)|.[Y_{Hh}^c(k)]^2. \end{aligned} \tag{48}$$

Substituting (48) into (47) yields

$$S_j^c(k)) = -\alpha_c.\mathrm{sgn}[S_j^c(k)].F_j(k) + E_j(k) \tag{49}$$

where $F_j(k)$ and $E_j(k)$ are defined by

$$F_j(k) = [f^{c'}[V_j^c(k)].\gamma_h^m(k).Y_{Hh}^c(k)]^2.|X_{1j}^c(k)| \tag{50}$$

$$\begin{aligned} E_j(k) =& f^{c'}[V_j^c(k)].\gamma_h^m(k).[C_1.(y^d(k+1)- \\ & y(k+1)) + \Delta y^d(k+1)] + [y^d(k) - y(k)]. \\ & [f^{c'}(V_j^c(k)).\Delta\gamma_h^m(k) + \Delta f^{c'}(V_j^c(k)). \\ & \gamma_h^m(k) - \Delta f^{c'}(V_j^c(k)).\Delta\gamma_h^m(k)]. \end{aligned} \tag{51}$$

The application of the condition (16) for the sliding surface given by (49) gives the limit values of α_c as

$$0 < \alpha_c < \min\Big\{\frac{|E_j(k)|}{F_j(k)}, \frac{|E_j(k-1)| - |E_j(k)|}{F_j(k-1) - F_j(k)}\Big\}. \tag{52}$$

For the hidden layer, the substitution of (43) into (44) yields

$$\begin{aligned} X_{2Hh}^c(k) =& f_H^{c'}[R_h^c(k)].[(\Delta y^d(k+1) - \Delta y(k+1)). \\ & f^{c'}(V_j^c(k)).\gamma_h^m(k).W_{jh}^c(k) + (y^d(k) - y(k)). \\ & (f^{c'}(V_j^c(k)).\Delta\gamma_h^m(k).W_{jh}^c(k) + \Delta f^{c'}(V_j^c(k)). \\ & \gamma_h^m(k).W_{jh}^c(k) - \Delta f^{c'}(V_j^c(k)).\Delta\gamma_h^m(k). \\ & W_{jh}^c(k) + f^{c'}(V_j^c(k)).\gamma_h^m(k).\Delta W_{jh}^c(k)- \\ & \Delta f^{c'}(V_j^c(k)).\gamma_h^m(k).\Delta W_{jh}^c(k)- \\ & f^{c'}(V_j^c(k)).\Delta\gamma_h^m(k).\Delta W_{jh}^c(k)+ \\ & \Delta f^{c'}(V_j^c(k)).\Delta\gamma_h^m(k).\Delta W_{jh}^c(k)]+ \\ & \Delta f_H^{c'}[R_h^c(k)].[y^d(k) - y(k)].\gamma_h^m(k-1). \\ & f^{c'}(V_j^c(k-1)).W_{jh}^c(k-1). \end{aligned} \tag{53}$$

Considering (46) and using the approximation $\frac{\partial y}{\partial Z_{hi}^c} = \frac{\Delta y}{\Delta Z_{hi}^c}$, (54) is obtained as

$$\begin{aligned} \Delta y(k+1) =& \delta_h^m(k).f^{c'}[V_j^c(k)]W_{jh}^c(k).f_H^{c'}[R_h^c(k)]. \\ & \beta_c.\mathrm{sgn}[S_{Hh}^c(k)].|X_{1Hh}^c(k)|.[T_i^c(k)]^2. \end{aligned} \tag{54}$$

Substituting (53) and (54) into (42) leads to

$$[S_{Hh}^c(k)] = -\beta_c.\mathrm{sgn}[S_{Hh}^c(k)].P_{Hh}(k) + M_{Hh}(k) \tag{55}$$

where $P_{Hh}(k)$ and $M_{Hh}(k)$ are given by

$$\begin{aligned} P_{Hh}(k) =& [\gamma_h^m(k).f^{c'}(V_j^c(k)).W_{jh}^c(k).f_H^{c'}(R_h^c(k)). \\ & T_i^c(k)]^2.|X_{1Hh}^c(k)| \end{aligned} \tag{56}$$

$$\begin{aligned} M_{Hh}(k) =& f_H^{c'}[R_h^c(k)].[C_{1H}.X_{1j}^c(k).W_{jh}^c(k)+ \\ & \Delta y^d(k+1).f^{c'}(V_j^c(k)).\gamma_h^m(k).W_{jh}^c(k)+ \\ & (y^d(k) - y(k)).(f^{c'}(V_j^c(k)).\Delta\gamma_h^m(k).W_{jh}^c(k)+ \\ & \Delta f^{c'}(V_j^c(k)).\gamma_h^m(k).W_{jh}^c(k) - \Delta f^{c'}(V_j^c(k)). \\ & \Delta\gamma_h^m(k).W_{jh}^c(k) + f^{c'}(V_j^c(k)).\gamma_h^m(k). \\ & \Delta W_{jh}^c(k) - \Delta f^{c'}(V_j^c(k)).\gamma_h^m(k).\Delta W_{jh}^c(k)- \\ & f^{c'}(V_j^c(k)).\Delta\gamma_h^m(k).\Delta W_{jh}^c(k)+ \\ & \Delta f^{c'}(V_j^c(k)).\Delta\gamma_h^m(k).\Delta W_{jh}^c(k)]+ \\ & \Delta f_H^{c'}[R_h^c(k)].[y^d(k) - y(k)].\gamma_h^m(k-1). \\ & f^{c'}(V_j^c(k-1)).W_{jh}^c(k-1). \end{aligned} \tag{57}$$

The application of the condition (16) for the sliding surface expressed by (55) gives the limit values of β_c

$$0 < \beta_c < \min\Big\{\Big|\frac{M_{Hh}(k)}{P_{Hh}(k)}\Big|, \frac{|M_{Hh}(k-1)| - |M_{Hh}(k)|}{P_{Hh}(k-1) - P_{Hh}(k)}\Big\}. \tag{58}$$

The limit values of α_c and β_c given by (52) and (58) guarantee the existence and the convergence of sliding surfaces described in (39) and (42). The update equations (45) and (46) can be used to train the INM taking into consideration these limits for the gains.

3 Neural control strategy

The direct inverse control (DIC)[39], as shown in Fig. 4, is the adopted strategy to control the uncertain nonlinear system. Indeed, the INM must be trained before using it as a controller for the system.

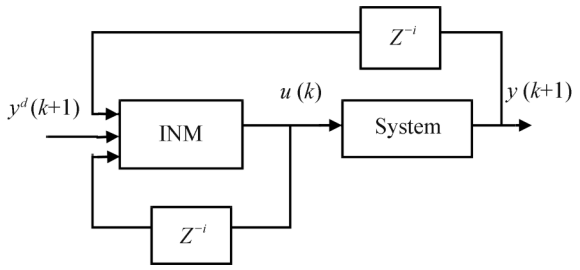

Fig. 4 Direct inverse control structure

3.1 INM synthesis

According to the control scheme, the DNM of the system is trained in a first step, then based on this later the INM is trained. Different combinations of the above presented learning algorithms are tested to determine the best one for the control of uncertain nonlinear systems.

Table 1 Different combinations of the INM synthesis

	DNM training algorithm	INM training algorithm
1st combination	BP	BP
2nd combination	SM-BP	BP
3rd combination	BP	SM-BP
4th combination	SM-BP	SM-BP

3.2 Control steps

After satisfactory training, the synthesized INM is applied as a controller for the uncertain nonlinear system. This controller must guarantee that the output of the system follows the desired one despite the existence of the parametric variations.

4 Simulation results

In this section, simulation results are presented and discussed in order to illustrate the performance and the robustness of the SM-BP algorithm for the modelling and the control and to compare it with conventional BP algorithm. Two examples are considered. The first example is a numerical one described by a recurrent nonlinear equation inspired from [21] and the second example is surge tank model taken from [41] and [42].

4.1 First example

Consider the nonlinear uncertain system given by (59) which is a modified version of the one presented in [21]:

$$y(k+1) = \frac{a(k).y(k)}{1+b(k).y^2(k)} + c(k).u^3(k). \tag{59}$$

The variations of the parameters a, b and c are given by the following equations:

$$\begin{cases} a(k) = 1 - 0.75\sin\left(\dfrac{2\pi k}{140}\right) \\ b(k) = 1 - 0.5\sin\left(\dfrac{2\pi k}{140}\right) \\ c(k) = 1 - 0.05\sin\left(\dfrac{2\pi k}{140}\right). \end{cases} \tag{60}$$

The variables $u(k)$ and $y(k)$ indicate respectively the input and the output of the system at the instant k. Fig. 5 illustrates the variations of the variables a, b and c.

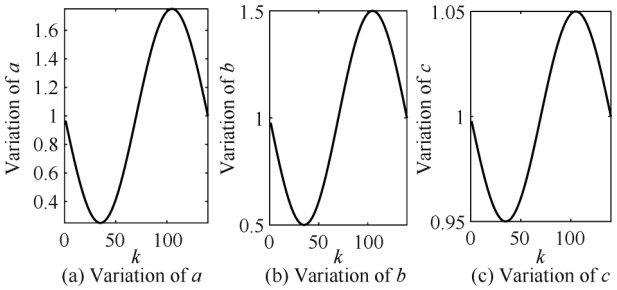

Fig. 5 Variations of variables a, b and c for the training set

4.1.1 Neural models synthesis

According to the control structure, the development of the DNM and the INM of the system is the preliminary step. These models will be used to determine the controller for the uncertain nonlinear system. The training of the neural models is assured first through the BP algorithm and secondly by SM-BP algorithm. FNN is the adopted architecture for neural modelling.

DNM synthesis

The development of the DNM consists in training a FNN to reproduce the dynamics of the uncertain system. The input vector of the DNM is composed by $y(k)$ and $u(k)$, $y_m(k+1)$ is the output. The input is a signal with an amplitude distributed over the interval $[0, 2]$. In order to ensure compromise between the quality of modeling and the time of convergence, the choice of the DNM′s parameters has been done after several simulations. In fact, the number of neurons in the hidden layer is $N_c^m = 5$. The database is divided into two parts, one serves to training $N_{tr} = 140$ and the other to testing $N_{ts} = 70$. The activation function of the hidden and the output layer is the sigmoid one. Since both algorithms BP and SM-BP consider in this work fixed learning rate, several simulation results were carried out in order to find the best values for the two algorithms. Table 2 shows the training parameters used in this simulation. The deduction of the boundaries of α_m and β_m is

not shown here, nevertheless, the values chosen are within these boundaries.

Table 2 DNM training parameters (first example)

Algorithms	BP	SM-BP
Parameters	$\varepsilon = 0.4$	$\alpha_m = 6$ $\beta_m = 2$ $C = C_H = 1$

Fig. 6 shows the evolution of the mean squared error model MSE_1 for the two algorithms.

$$MSE_1 = \frac{1}{2}\sum_{k=1}^{N_{tr}}[y(k+1) - y_m(k+1)]^2. \quad (61)$$

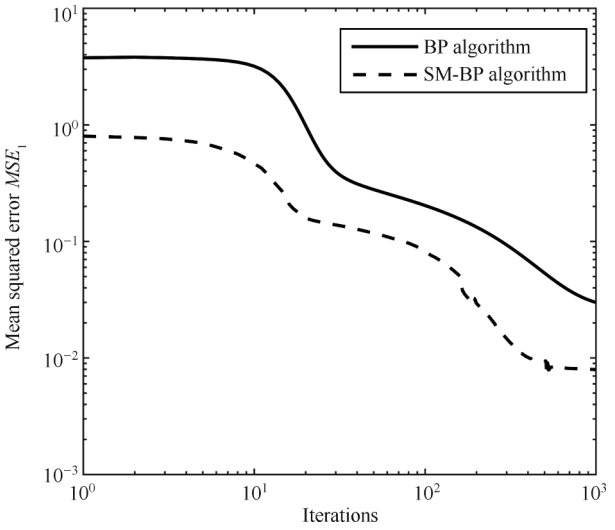

Fig. 6 Evolution of DNM training criterion for the SM-BP versus BP (first example)

It is to be noted that SM-BP algorithm presents better convergence than the BP one.

INM synthesis

After satisfactory training of the DNM, it is used to train the INM. The different combinations of the above presented learning algorithms given by Table 1 have been tested to determine the best one for the control of the uncertain system.

The input vector of the INM is composed by the desired output $y^d(k+1)$ and the output of the neural model $y(k)$, the hidden layer contains five hidden neurons $N_c = 5$ and $u(k)$ is the output. The data used for the training and the testing consist of $N_{tr} = 140$ and $N_{ts} = 70$ elements respectively. The activation function of the hidden layer and the output layer is sigmoid one. INM training parameters are given by Table 3.

Table 3 INM training parameters for different combinations of the INM synthesis (first example)

1st combination	2nd combination	3rd combination	4th combination
$\varepsilon = 0.5$	$\varepsilon = 0.5$	$\alpha_c = 3.5$ $\beta_c = 6$ $C_1 = C_{1H} = 1$	$\alpha_c = 3$ $\beta_c = 9.8$ $C_1 = C_{1H} = 1$

The evolution of the mean squared error MSE_2 for the different combinations is shown by Fig. 7 where

$$MSE_2 = \frac{1}{2}\sum_{k=1}^{N_{tr}}[y^d(k+1) - y_m(k+1)]^2. \quad (62)$$

It is noted from Fig. 7 that the third and the fourth combinations present the best convergence properties.

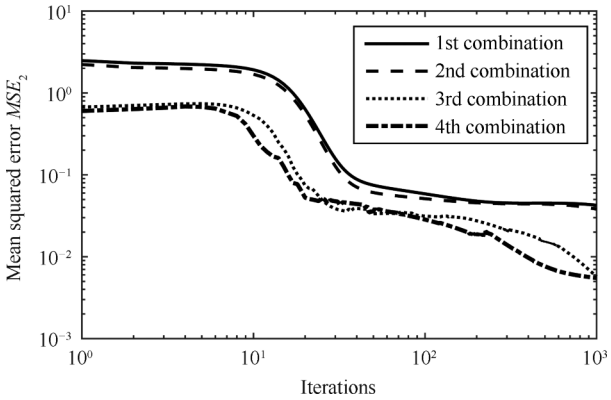

Fig. 7 Evolution of INM training criterion of the different combinations (first example)

4.1.2 Control results

After satisfactory training, the synthesized INM is placed in cascade with the plant to be controlled, thus it is used as a neural controller of the uncertain nonlinear system. The evolution of the system output for the parametric variations given by (60) and the desired output are illustrated by Fig. 8. It is clear from Fig. 8 that the neural controller synthesized by the 4th combination presents the best performance despite the existence of the parametric variations.

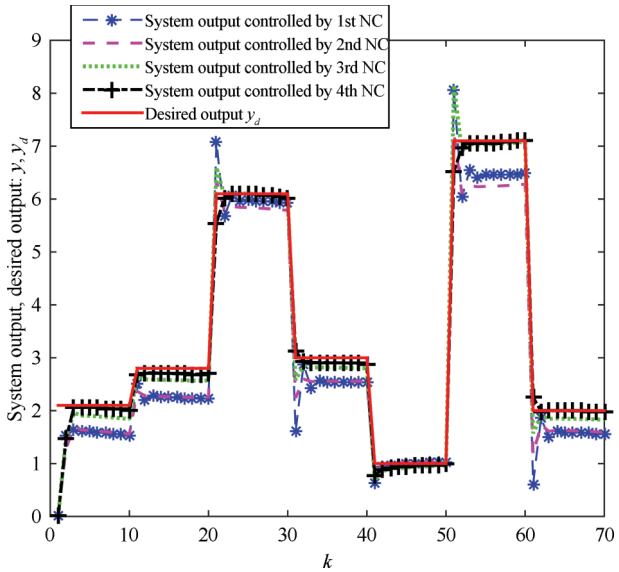

Fig. 8 Evolution of the system output controlled by the different neural controllers and the desired output (parametric variations of training set)

Fig. 9 shows the different control signals. In order to test

the robustness of the synthesized neural controllers, other variations of the parameters a, b and c have been considered as given by Fig. 10. The evolution of the system output and the desired one are shown in Fig. 11.

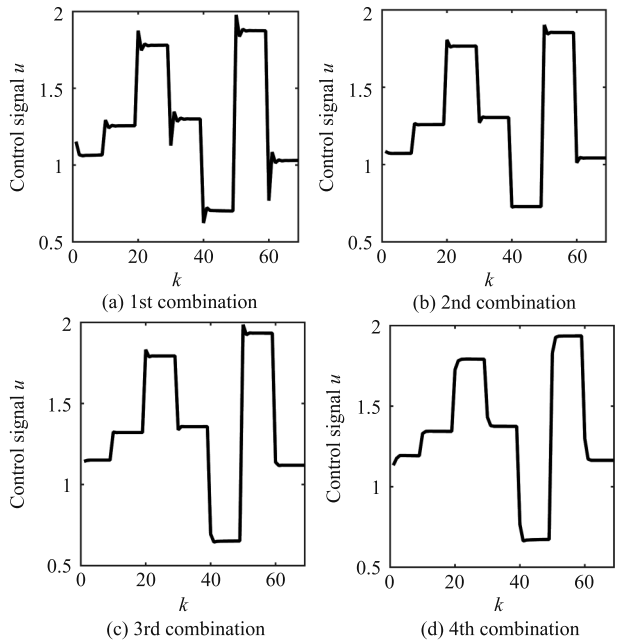

Fig. 9 Evolution of the control signals (parametric variations of training set)

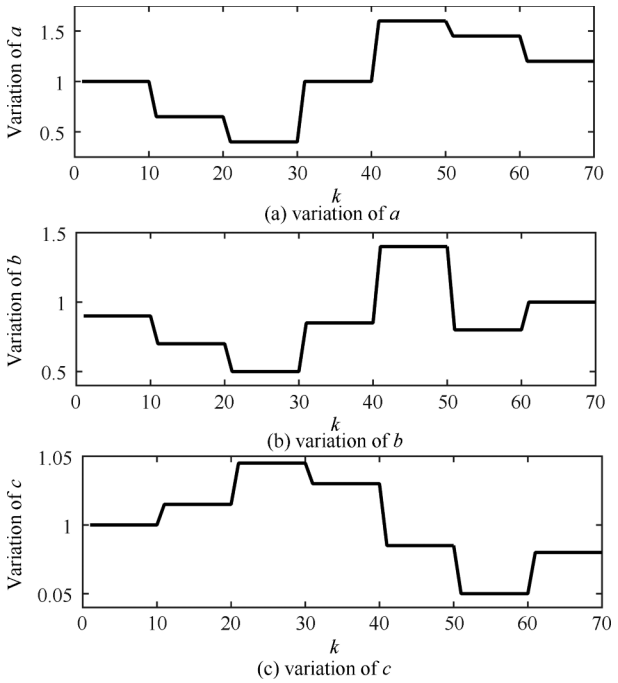

Fig. 10 Variation of variables a, b and c for the validation set

Fig. 12 presents the control signals provided by the INMs trained by the different combinations.

To show the performance of the different neural controllers, we propose to calculate the error between the desired output and the system one as

$$E = \frac{1}{70}\sum_{k=1}^{70}[y^d(k) - y(k)]^2 \tag{63}$$

where E and E_1 are respectively the errors between the desired output and the system one for the parametric variation given by Figs. 5 and 10. According to the obtained simulation results, we note that the SM-BP has really improved the standard BP. Thus, it is clear that the control process in case of training through the SM-BP is better than that of the BP. Indeed, referring to Table 4, the performance of the system controlled with the neural controller synthesized through the fourth combination (training process based on SM-BP for the DNM and the INM) is the best one.

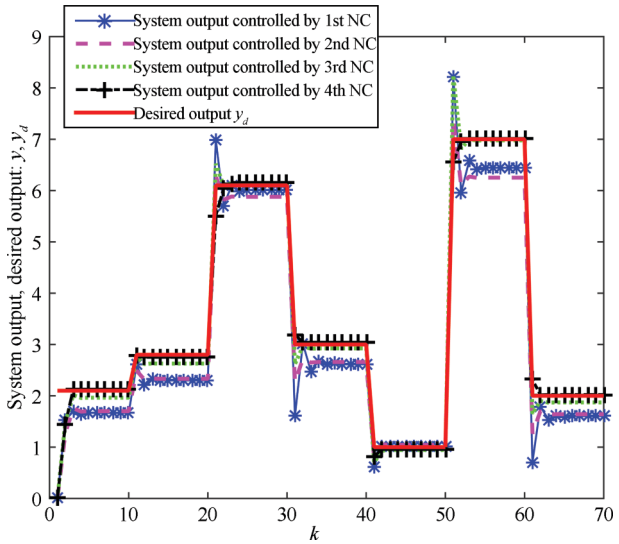

Fig. 11 Evolution of the system output controlled by the different neural controllers and the desired output (parametric variations of validation set)

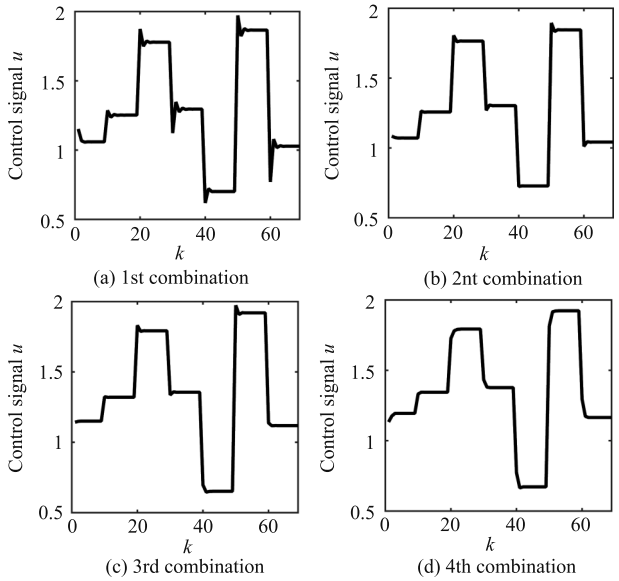

Fig. 12 Evolution of the control signals (parametric variations of validation set)

Table 4 Comparative results

Neural controller	E	E_1
1	0.137 7	0.116 7
2	0.123 4	0.094 3
3	0.026 3	0.022 5
4	**0.010 5**	**0.008 6**

4.2 Second example

A level control problem for a surge tank[41, 42] is presented in this section as shown in Fig. 13. The differential equation representing the dynamic of the surge tank can be given by

$$\frac{\mathrm{d}h}{\mathrm{d}t} = \frac{-\bar{d}\sqrt{2gh(t)}}{A(h(t))} + \frac{\bar{c}}{A(h(t))}q_e(t) \tag{64}$$

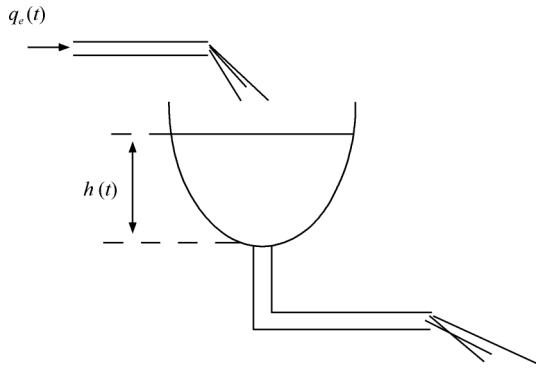

Fig. 13 Surge tank

where $q_e(t)$ is the input flow (control input), $h(t)$ is the liquid level (the output of the system), $A(h(t))$ is the cross-sectional area of the tank, $g = 9.8\,\mathrm{m/s^2}$ is the gravitational acceleration, $\bar{c} \in [0.9, 1]$ is the "clogging factor" for a filter in the pump actuator: if $\bar{c} = 1$, the filter is clean so there is no clogging and $\bar{d} > 0$ is the parameter related to the diameter of the output pipe. Let $h_d(t)$ be the desired liquid level in the tank and it is assumed that $h(0) = 1$. Furthermore, the area of the tank is given as $A(h(t)) = \bar{a}h^2(t) + \bar{b}$, assume also that $\bar{a}$ and $\bar{b}$ are unknown but that $\bar{a} \in [a_1, a_2]$, $\bar{b} \in [b_1, b_2]$. $a_1 > 0$, $b_1 > 0$, where $a_1 = 0.4 \times 10^{-3}$, $a_2 = 3.6 \times 10^{-3}$, $b_1 = 0.15$ and $b_2 = 0.25$ are are all fixed. Using Euler approximation, the discrete-time uncertain nonlinear system is

$$h(k+1) = h(k) + T\frac{-\bar{d}(k)\sqrt{2gh(k)}}{\bar{a}(k)h^2(k) + \bar{b}(k)} + \frac{\bar{c}(k)}{\bar{a}(k)h^2(k) + \bar{b}(k)}q_e(k). \tag{65}$$

$T = 0.1\,\mathrm{s}$ denotes the sampling period[41]. Assume that the variations of the variables $\bar{a}$, $\bar{b}$, $\bar{c}$ and $\bar{d}$ are given by the equations as

$$\begin{cases} \bar{a}(k) = 0.002 - 0.001\,6\sin\left(\dfrac{2\pi k}{550}\right) \\ \bar{b}(k) = 0.2 + 0.05\sin\left(\dfrac{2\pi k}{550}\right) \\ \bar{c}(k) \in [0.9\ ,\ 1] \\ \bar{d}(k) \in [0.8\ ,\ 1]. \end{cases} \tag{66}$$

4.2.1 Neural models synthesis

DNM synthesis

The DNM of the above presented system has two inputs $q_e(k)$ and $h(k)$, one hidden layer with four hidden neurons $N_c^m = 4$ and a single output $h_m(k+1)$. The input flow rate is constrained within the limits of $4-14\,\mathrm{m^3/s}$. The liquid level as the input flow is also constrained within the limits of $0.8-10\,\mathrm{m}$. The training samples and the testing ones contained $N_{tr} = 550$ and $N_{ts} = 300$ elements respectively. Table 5 shows the training parameters used in this simulation.

Table 5 DNM training parameters (second example)

Algorithms	BP	SM-BP
Parameters	$\varepsilon = 0.5$	$\alpha_m = 3$ $\beta_m = 2.9$ $C = C_H = 1$

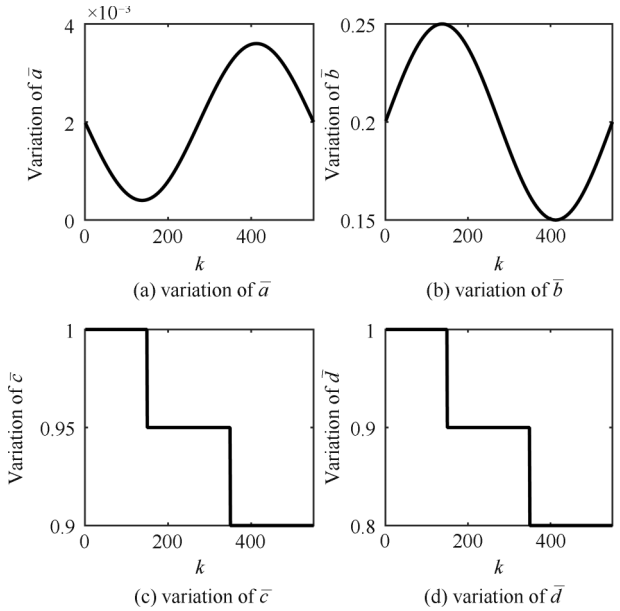

Fig. 14 Variation of the variables $\bar{a}$, $\bar{b}$, $\bar{c}$ and $\bar{d}$ for the training set

Fig. 15 shows the evolution of the mean squared error model MSE_1 for the two algorithms.

It is to be noted that SM-BP presents better convergence than the BP.

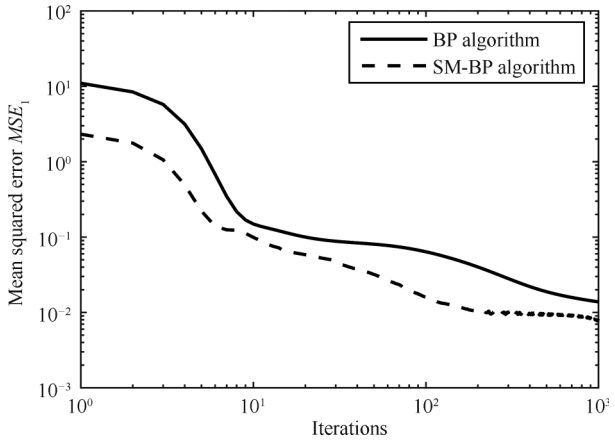

Fig. 15 Evolution of DNM training criterion for the SM-BP versus BP (second example)

INM synthesis

Assume that we know the desired trajectory a priori, and suppose that $h_d(k+1) \in [1, 9]$. Indeed, based on the DNM which gives acceptable performance after the training phase, the INM is trained. The different combinations of the above presented learning algorithms given by Table 1 have been tested to determine the best one that is able to force the output of the system following the desired liquid level under the parametric variations. The input vector of the INM is composed of the desired output $h_d(k+1)$ and the output of the neural model $h(k)$, the hidden layer contains four hidden neurons $N_c = 4$ and $q_e(k)$ is the output. The training samples and the testing ones contain $N_{tr} = 450$ and $N_{ts} = 300$ elements respectively. INM training parameters are given by Table 6.

Table 6 INM training parameters for different combinations of the INM synthesis (second example)

1st combination	2nd combination	3rd combination	4th combination
		$\alpha_c = 4$	$\alpha_c = 6.6$
$\varepsilon = 0.12$	$\varepsilon = 0.13$	$\beta_c = 37$	$\beta_c = 39.8$
		$C_1 = C_{1H} = 1$	$C_1 = C_{1H} = 1$

The evolution of the mean squared error MSE_2 for the different combinations is illustrated by Fig. 16.

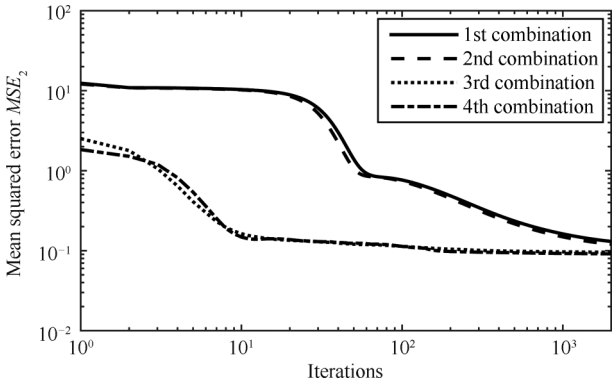

Fig. 16 Evolution of INM training criterion of the different combinations (second example)

It is noted from Fig. 16 that the third and the fourth combinations present the best convergence properties.

4.2.2 Control results

After satisfactory training, the trained INM is used as a neural controller for the uncertain nonlinear system. The evolution of the tank liquid level for the different parametric variations given by (66) and the desired liquid level are illustrated by Fig. 17.

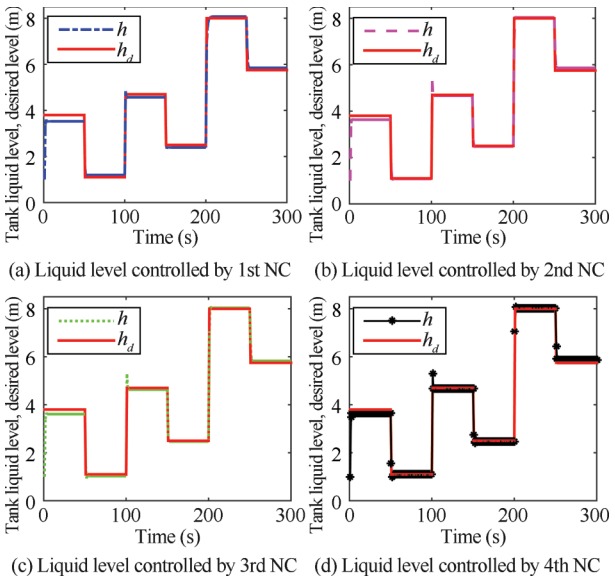

Fig. 17 Evolution of the tank liquid level controlled by different neural controllers and the desired liquid level (parametric variations of training set)

The input flow generated by different neural controllers is described by Fig. 18.

In order to test the robustness of the trained neural controllers, other variations of the parameters $\bar{a}$, $\bar{b}$, $\bar{c}$ and $\bar{d}$ have been considered as given by Fig. 19. The obtained results are shown in Fig. 20.

Fig. 21 illustrates the evolution of the input flow of the surge tank provided by the INM trained by the different combinations. It is clear from the obtained results shown in Figs. 20 and 21 that the neural controller trained through the first and the second combinations (based on BP training algorithm) present an oscillatory behavior under the parametric variations of $\bar{a}$, $\bar{b}$, $\bar{c}$ and $\bar{d}$ given by Fig. 19.

To show the performance of the different neural controllers, we propose to calculate the error between the tank liquid level and the desired one as

$$E' = \frac{1}{300}\sum_{k=1}^{300}[h_d(k) - h(k)]^2 \tag{67}$$

where E' and E'_1 are respectively the errors between the tank liquid level and the desired one for the parametric variation given by Figs. 14 and 19.

It can be seen from the simulation results of the second example that the control process in case of training through

the SM-BP provides better performance than that of the BP. Indeed, the neural controller synthesized through the fourth combination shows that is able to guarantee satisfactory tracking performance (the overshot is attenuated) and robustness in spite of the existence of parametric variations.

Table 7 Comparative results

Neural controller	E'	E'_1
1	0.014 8	0.029 4
2	0.013 5	0.024
3	0.008	0.01
4	**0.007 2**	**0.007 4**

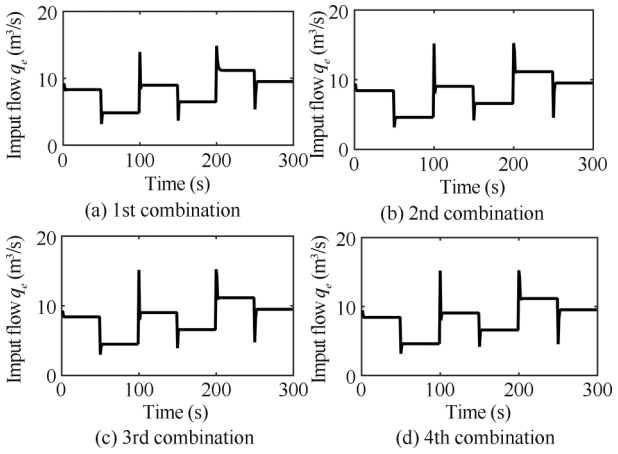

Fig. 18 Evolution of the input flow of the surge tank (parametric variations of training set)

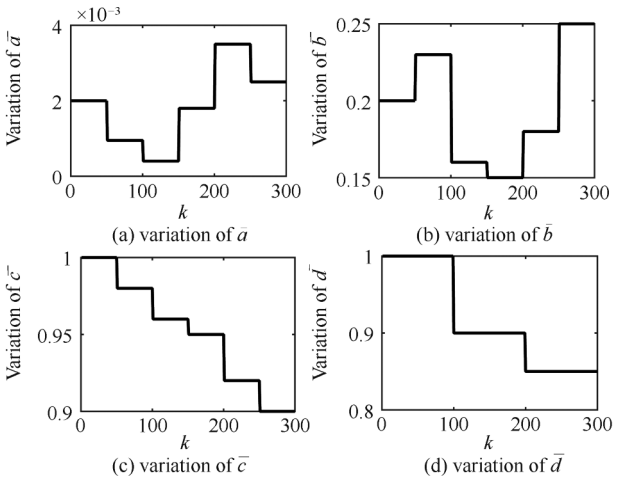

Fig. 19 Variation of the variables $\bar{a}$, $\bar{b}$, $\bar{c}$ and $\bar{d}$ for the validation set

5 Conclusions

In this paper, a robust neural control of uncertain nonlinear systems was presented. The base idea of this work was the synthesis of a robust neural controller based on training algorithm that combines standard BP and sliding mode theory. The applicability and the performance of the proposed control strategy were tested using two simulation examples. The above results showed the robustness of the developed neural controller under the parametric variations. As future work, the adaptation of the gains α and β will be considered and other neural control strategies will be studied to improve the control performance.

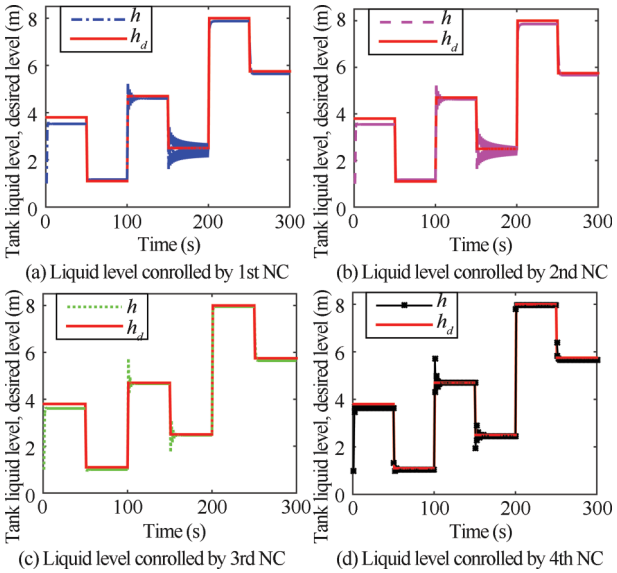

Fig. 20 Evolution of the tank liquid level controlled by different neural controllers and the desired liquid level (parametric variations of validation set)

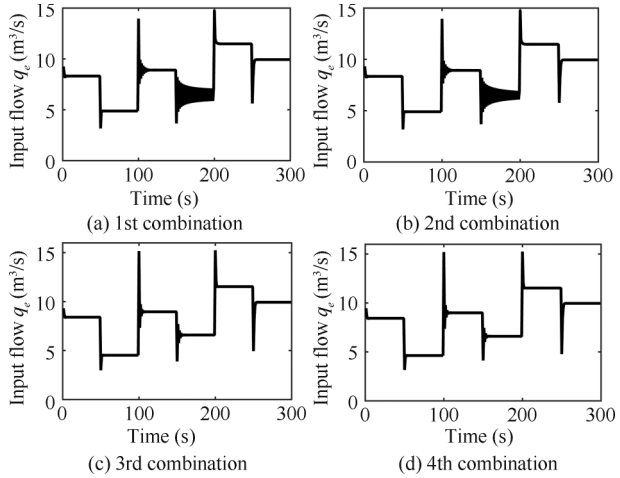

Fig. 21 Evolution of the input flow of the surge tank (parametric variations of validation set)

References

[1] Z. D. Wang, D. W. C. Ho, Y. R. Liu, X. H. Liu. Robust H_∞ control for a class of nonlinear discrete time-delay stochastic systems with missing measurements. *Automatica*, vol. 45, no. 3, pp. 684–691, 2009.

[2] L. H. Xie. Output feedback H_∞ control of systems with parameter uncertainty. *International Journal of Control*, vol. 63, no. 4, pp. 741–750, 1996.

[3] S. S. Ge, J. Wang. Robust adaptive tracking for time-varying uncertain nonlinear systems with unknown control coefficients. *IEEE Transactions on Automatic Control*, vol. 48, no. 8, pp. 1463–1469, 2003.

[4] W. M. Haddad, T. Hayakawa, V. Chellaboina. Robust adaptive control for nonlinear uncertain systems. *Automatica*, vol. 39, no. 3, pp. 551–556, 2003.

[5] D. L. Marruedo, T. Alamo, E. F. Camacho. Input-to-state stable MPC for constrained discrete-time nonlinear systems with bounded additive uncertainties. In *Proceedings of the 41st IEEE Conference on Decision and Control*, IEEE, Las Vegas, USA, pp. 4619–4624, 2002.

[6] D. L. Marruedo1, J. M. Bravo, T. Alamo, E. F. Camacho. Robust MPC of constrained discrete-time nonlinear systems based on uncertain evolution sets: Application to a CSTR model. In *Proceedings of International Conference on Control Applications*, IEEE, Glasgow, UK, pp. 657–662, 2002.

[7] X. Y. Lu, S. K. Spurgeon. Robust sliding mode control of uncertain nonlinear systems. *Systems & Control Letters*, vol. 32, no. 2, pp. 75–90, 1997.

[8] Y. Q. Xia, Z. Zhu, C. M. Li, H. J. Yang, Q. M. Zhu. Robust adaptive sliding mode control for uncertain discrete-time systems with time delay. *Journal of the Franklin Institute*, vol. 347, no. 1, pp. 339–357, 2010.

[9] J. J. Garside, T. L. Ruchti, R. H. Brown. Using self-organizing artificial neural networks for solving uncertain dynamic nonlinear system identification and function modeling problems. In *Proceedings of the 31st IEEE Conference on Decision and Control*, IEEE, Tucson, USA, pp. 2716–2721, 1992.

[10] R. H. Brown, T. L. Ruchti, X. Feng. Artificial neural network identification of partially known dynamic nonlinear systems. In *Proceedings of the 32nd IEEE Conference on Decision and Control*, IEEE, USA, pp. 3694–3699, 1993.

[11] T. Tsuji, B. H. Xu, M. Kaneko. Adaptive control and identification using one neural network for a class of plants with uncertainties. *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans*, vol. 28, no. 4, pp. 496–505, 1998.

[12] A V. Topalov, O. Kaynak. Robust neural identification of robotic manipulators using discrete time adaptive sliding mode learning. In *Proceedings of the 16th IFAC World Congress*, IFAC, Czech Republic, 2005.

[13] Y. J. Shu, X. Liu, Y. J. Liu. Stability and passivity analysis for uncertain discrete-time neural networks with time-varying delay. *Neurocomputing*, vol. 173, pp. 1706–1714, 2016.

[14] X. Wang, T. S. Li, C. L. Philip Chen, B. Lin. Adaptive robust control based on single neural network approximation for a class of uncertain strict-feedback discrete-time nonlinear systems. *Neurocomputing*, vol. 138, pp. 325–331, 2014.

[15] Y. J. Liu, S. C. Tong. Adaptive NN tracking control of uncertain nonlinear discrete-time systems with nonaffine dead-zone input. *IEEE Transactions on Cybernetics*, vol. 45, no. 3, pp. 497–505, 2015.

[16] Y. J. Liu, L. Tang, S. C. Tong, C. L. Philip Chen, D. J. Li. Reinforcement learning design-based adaptive tracking control with less learning parameters for nonlinear discrete-time MIMO systems. *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 1, pp. 165–176, 2015.

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

[18] Q. G. Li, S. C. Tong, T. Y. Chai. H_∞ tracking of unknown nonlinear systems using neural network. In *Proceedings of International Symposium on Intelligent Control*, IEEE, Istanbul, pp. 199–204, 1997.

[19] W. Y. Wang, M. L. Chan, C. C. J. Hsu, T. T. Lee. H_∞ tracking-based sliding mode control for uncertain nonlinear systems via an adaptive fuzzy-neural approach. *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 32, no. 4, pp. 483–492, 2002.

[20] H. Deng, Z. Xu, H. X. Li. A novel neural internal model control for multi-input multi-output nonlinear discrete-time processes. *Journal of Process Control*, vol. 19, no. 8, pp. 1392–1400, 2009.

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

[22] F. Cubillos, H. Callejas, E. L. Lima, M. P. Vega. Adaptive control using a hybrid-neural model: Application to a polymerisation reactor. *Brazilian Journal of Chemical Engineering*, vol. 18, no. 1, pp. 113–120, 2001.

[23] D. E. Rumelhart, G. E. Hinton, R. J. Williams. Learning internal representations by error propagation. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, Cambridge, USA: MIT Press, pp. 318–362, 1986.

[24] Q. Song, J. Z. Xiao, Y. C. Soh. Robust backpropagation training algorithm for multilayered neural tracking controller. *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 1133–1141, 1999.

[25] C. C. Chuang, S. F. Su, C. C. Hsiao. The annealing robust backpropagation (ARBP) learning algorithm. *IEEE Transactions on Neural Networks*, vol. 11, no. 5, pp. 1067–1077, 2000.

[26] A. Rusiecki. Robust MCD-based backpropagation learning algorithm. In *Proceedings of the 9th International Conference on Artifical Intelligence and Soft Computing, Lecture Notes in Computer Science*, Springer, Zakopane, Poland, vol. 5097, pp. 154–163, 2008.

[27] A. Rusiecki. Fast robust learning algorithm dedicated to LMLS criterion. In *Proceedings of the 10th International Conference on Artifical Intelligence and Soft Computing, Lecture Notes in Computer Science*, Springer, Zakopane, Poland, vol. 6114, pp. 96–103, 2010.

[28] A. Nied, S. I. Seleme, Jr., G. G. Parma, B. R. Menezes. Online neural training algorithm with sliding mode control and adaptive learning rate. *Neurocomputing*, vol. 70, no. 16–18, pp. 2687–2691, 2007.

[29] H. Sira-Ramirez, E. Colina-Morles. A sliding mode strategy for adaptive learning in Adalines. *IEEE Transactions on Circuits and Systems — I: Fundamental Theory and Applications*, vol. 42, no. 12, pp. 1001–1012, 1995.

[30] X. H. Yu, Z. H. Ma, S. M. M. Rahman. Adaptive sliding mode approach for learning in a feedforward neural network. *Neural Computing and Applications*, vol. 7, no. 4, pp. 289–294, 1998.

[31] A. V. Topalov, O. Kaynak. A sliding mode strategy for adaptive learning in multilayer feedforward neural networks with a scalar output. In *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, IEEE, Washington, USA, pp. 1636–1641, 2003.

[32] A. V. Topalov, O. Kaynak, N. G. Shakev. Variable structure systems approach for on-line learning in multilayer artificial neural networks. In *Proceedings of the 29th Annual Conference of the IEEE Industrial Electronics Society*, IEEE, Roanoke, USA, pp. 2989–2994, 2003.

[33] G. G. Parma, B. R. Menezes, A. P. Barga. Sliding mode algorithm for training multilayer artificial neural networks. *Electronics Letters*, vol. 34, no. 1, pp. 97–98, 1998.

[34] G. G. Parma, B. R. Menezes, A. P. Barga. Improving backpropagation with sliding mode control. In *Proceedings of the 5th Brazilian Symposium on Neural Networks*, IEEE, Belo Horizonte, Brazil, pp. 8–13, 1998.

[35] V. I. Utkin. *Sliding Modes and Their Application in Variable Structure Systems*, Moscow, RUssia: MIR, 1978.

[36] A. V. Topalov, O. Kaynak. Neural network modeling and control of cement mills using a variable structure systems theory based on-line learning mechanism. *Journal of Process Control*, vol. 14, no. 5, pp. 581–589, 2004.

[37] S. Sarpturk, Y. Istefanopulos, O. Kaynak. On the stability of discrete-time sliding mode control systems. *IEEE Transactions on Automatic Control*, vol. 32, no. 10, pp. 930–932, 1987.

[38] G. G. Parma, B. R. Menezes, A. P. Barga. Sliding mode backpropagation: Control theory applied to neural network learning. In *Proceedings of the International Joint Conference on Neural Networks*, IEEE, Washington, USA, pp. 1774–1778, 1999.

[39] D. Psaltis, A. Sideris, A. A. Yamamura. A multilayered neural network controller. *IEEE Control Systems Magazine*, vol. 8, no. 2, pp. 17–21, 1988.

[40] S. Aoughellanet, T. Mohammedi, Y. Bouterfa. Reduced neural network models for dynamical systems. In *Proceedings of the 6th WSEAS International Conference on Neural Networks*, WSEAS, Lisbon, Portugal, pp. 26–29, 2005.

[41] K. M. Passino, S. Yurkovich. *Fuzzy Control*, Menlo Park, CA: Addison-Wesley Longman, 1998.

[42] K. M. Passino. *Biomimicry for Optimization, Control, and Automation*, London, UK: Springer-Verlag, 2005.

Imen Zaidi received the B. Eng. degree in electrical engineering, and the M. Sc. degree in automatic and industrial Informatics from the National Engineering School of Sfax, Tunisia in 2007 and 2008, respectively. Currently, she is a Ph. D. degree candidate in the Department of Electrical Engineering, National Engineering School of Sfax, Tunisia.

Her research interests include learning algorithms, artificial neural networks and their engineering applications and intelligent control.

E-mail: zaidi.imen@yahoo.fr (Corresponding author)
ORCID iD: 0000-0002-8873-3596

Mohamed Chtourou received the B. Eng. degree in electrical engineering from the National Engineering School of Sfax, Tunisia in 1989, the M. Sc. degree in automatic control from the National Institute of Applied Sciences of Toulouse, France in 1990, and the Ph. D. degree in process engineering from the National Polytechnic Institute of Toulouse, France in 1993. He is currently a professor in the Department of Electrical Engineering, National Engineering School of Sfax, Tunisia. He is the author and co-author of more than fifty papers in international journals and of more than 70 papers published in national and international conferences.

His research interests include learning algorithms, artificial neural networks and their engineering applications, fuzzy systems, and intelligent control.

E-mail: mohamed.chtourou@enis.rnu.tn

Mohamed Djemel received the B. Sc., the M. Sc. and Ph. D. degrees in electrical engineering from the Ecole Supérieure des Sciences Techniques de Tunis, Tunisia in 1987, 1989 and 1996, respectively. He joined the Tunisian University since 1990, where he held different positions involved in research and education. Currently, he is professor of automatic control at the Department of Electrical Engineering of National Engineering School of Sfax, Tunisia. He a member of several national and international conferences.

His research interests include the order reduction, the stability, the control and the advanced control of the complex systems.

E-mail: mohamed.djemel@enis.rnu.tn