Hindawi Publishing Corporation
Modelling and Simulation in Engineering
Volume 2008, Article ID 343940, 8 pages
doi:10.1155/2008/343940

*Research Article*

# Stability Analysis of Neural Networks-Based System Identification

**Talel Korkobi, Mohamed Djemel, and Mohamed Chtourou**

*Research Unit on Intelligent Control, Design and Optimization of Complex Systems (ICOS), National Engineering School of Sfax (ENIS), University of Sfax, BP W, 3038 Sfax, Tunisia*

Correspondence should be addressed to Talel Korkobi, korkobi_talel@yahoo.fr

Received 28 January 2008; Revised 23 April 2008; Accepted 12 June 2008

Recommended by Petr Musilek

This paper treats some problems related to nonlinear systems identification. A stability analysis neural network model for identifying nonlinear dynamic systems is presented. A constrained adaptive stable backpropagation updating law is presented and used in the proposed identification approach. The proposed backpropagation training algorithm is modified to obtain an adaptive learning rate guarantying convergence stability. The proposed learning rule is the backpropagation algorithm under the condition that the learning rate belongs to a specified range defining the stability domain. Satisfying such condition, unstable phenomena during the learning process are avoided. A Lyapunov analysis leads to the computation of the expression of a convenient adaptive learning rate verifying the convergence stability criteria. Finally, the elaborated training algorithm is applied in several simulations. The results confirm the effectiveness of the CSBP algorithm.



## 1. Introduction

The last few decades have witnessed the use of artificial neural networks (ANNs) in many real-world applications and have offered an attractive paradigm for a broad range of adaptive complex systems. In recent years, ANNs have enjoyed a great deal of success and have proven useful in wide variety pattern recognition feature-extraction tasks. Examples include optical character recognition, speech recognition, and adaptive control, to name a few. To keep the pace with the huge demand in diversified application areas, many different kinds of ANN architecture and learning types have been proposed to meet varying needs as robustness and stability.

The area of system identification has received significant attention over the past decades and now it is a fairly mature field with many powerful methods available at the disposal of control engineers. Online system identification methods to date are based on recursive methods, such as least squares, for most systems that are expressed as linear in the parameters.

During the past few years, several authors [1–3] have suggested neural networks for nonlinear dynamical black-box modelling. The problem of designing a mathematical model of a process using only observed data has attracted much attention, both from an academic and an industrial point of view. Neural models can be used either as simulators or as models.

Recently, feedforward neural networks have been shown to obtain successful results in system identification and control [4]. Such neural networks are static input/output mapping schemes that can approximate a continuous function to an arbitrary degree of accuracy. Results have also been extended to recurrent neural networks [5, 6].

Recent results show that neural network technique seems to be very effective to identify a broad category of complex nonlinear systems when complete model information cannot be obtained. The Lyapunov approach has been used directly to obtain stable training algorithms for continuous-time neural networks [7–9]. The stability of neural networks can be found in [10, 11]. The stability of learning algorithms has been discussed in [6, 12].

It is well known that conventional identification algorithms are stable for ideal plants [13–15]. In the presence of disturbances or unmodeled dynamics, these adaptive procedures can go to instability easily. The lack of robustness in parameters identification was demonstrated in [10] and became a hot issue in 1980s. Several robust modification techniques were proposed in [13, 14]. The weight-adjusting algorithms of neural networks are a type of parameters identification; the normal-gradient algorithm is stable when

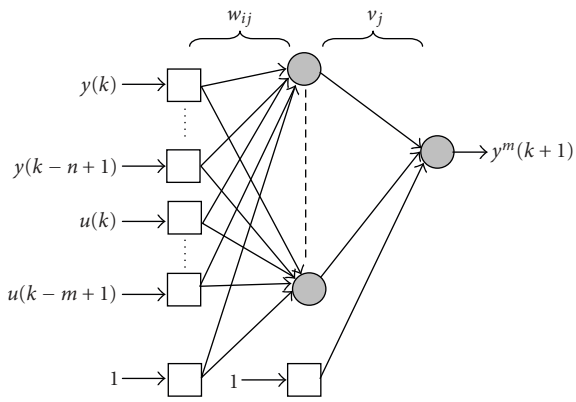


Figure 1: Feedforward neural model.

neural-network model can match the nonlinear plant exactly [6]. Generally, some modifications to the normal-gradient algorithm or backpropagation should be applied, such that the learning process is stable. For example, in [12, 16], some hard restrictions were added to the learning law, and in [11], the dynamic backpropagation has been modified with NLq stability constraints.

The paper is organized as follows. Section 2 describes the neural identifier structure considered in this paper and the usual backpropagation algorithm. In Section 3 and through a stability analysis, a constrained adaptive stable backpropagation algorithm (CSBP) is proposed to provide stable adaptive updating process. Three simulation examples give the effectiveness of the suggested algorithm in Section 4.

## 2. Preliminaries

The main concern of this section is to introduce the feedforward neural network, which is the adopted architecture, as well as some concepts of backpropagation training algorithm. Consider the following discrete-time input-output nonlinear system:

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

The neural model for the plant can be expressed as

$$y^m(k+1) = \hat{F}(Y(k), \theta), \tag{2}$$

where $Y(k) = (y(k), y(k-1), \ldots, y(k-n+1), u(k), u(k-1), \ldots, u(k-m+1))$, and $\theta = [W, V]^T$ is the weight parameter vector for the neural model.

A typical multilayer feedforward neural network is shown in Figure 1, where $I_j$ is the $j$th hidden neuron input, $O_j$ is the $j$th hidden neuron output $i$, $j$, and $k$ indicate neurons, $w_{ij}$ is the weight between neuron $i$ and neuron $j$, while $v_j$ is the weight between neuron $j$ and output neuron. For all neurons, the nonlinear activation function is defined as

$$f(x) = \frac{1}{1+e^{-x}}. \tag{3}$$

The output $y_m(k)$ of the considered NN is

$$\begin{aligned} I_j &= \sum_{i=1}^{n+m+1} x_i w_{ij}, \quad O_j = f(I_j), \quad j = 1, \ldots, N, \\ I_k &= \sum_{j=1}^{N} O_j v_j, \quad y^m(k) = f(I_k), \quad k = 1. \end{aligned} \tag{4}$$

Training the neural model is to adjust the weight parameters so that it emulates the nonlinear plant dynamics. Input-output training is obtained from the operation history of the plant.

Using the gradient descent, the weight connecting neuron $i$ to neuron $j$ is updated as

$$\begin{aligned} w_{ij}(k+1) &= w_{ij}(k) - \varepsilon \cdot \frac{\partial J(k)}{\partial w_{ij}(k)}, \\ v_j(k+1) &= v_j(k) - \varepsilon \cdot \frac{\partial J(k)}{\partial v_j(k)}, \end{aligned} \tag{5}$$

where $J(k) = (1/2)[y(k+1) - y^m(k+1)]^2$, and $\varepsilon$ is the learning rate. The partial derivatives are calculated with respect to the vectors of weights $W$ and $V$,

$$\begin{aligned} \frac{\partial J(k)}{\partial v_j(k)} &= f'(I_j)\cdot(y(k+1) - y^m(k+1))O_j, \\ \frac{\partial J(k)}{\partial w_{ij}(k)} &= f'(I_j)\left[\sum_{j=1}^{L} f'(I_j)(y(k+1) - y^m(k+1))v_j\right]x_i. \end{aligned} \tag{6}$$

Backpropagation algorithm has become the most popular one for training of the multilayer perceptron [1]. Generally, some modifications to the normal-gradient algorithm or backpropagation should be applied, such that the learning process is stable. For example, in [12, 16], some hard restrictions were added in the learning law, and in [11], the dynamic backpropagation was modified with stability constraints.

The research on modified algorithms of feedforward neural networks is becoming a challenging field. These researches involve the development of heuristic techniques, which arise out of a study of the distinctive performance of standard backpropagation algorithm. These heuristic techniques include such ideas as varying the learning rate [17], using momentum [18], and rescaling variables [19].

## 3. Stability Analysis and CSBP Algorithm Formulation

In the literature, the Lyapunov synthesis [4, 5] consists of the selection of a positive function candidate $V$ which leads to the computation of an adaptation law insuring its decrescence, that is, $\dot{V} \leq 0$ for continuous systems and $\Delta V(k) = V(k+1) - V(k) \leq 0$ for discrete-time systems. Under these assumptions, the function $V$ is called Lyapunov function and guarantees the stability of the system. Our objective is the determination of a stabilizing adaptation law

ensuring the stability of the identification scheme presented in what follows and the boundness of the output signals. The stability of the learning process in an identification approach leads to a better modelling and a guaranteed reached performance. The proposed learning rule is the backpropagation algorithm adopting a constrained learning rate. Satisfying such condition, unstable phenomena during the learning process are avoided. This problem has been treated in the literature of neural identification so this work is considered as a solution for extended problems. The originality of this work consists of the constraints themselves. In fact, a choice of the learning rate with respect to the proposed constraints ensures an efficient stable identification which is not the case adopting an arbitrary learning rate especially when it does not belong to the specified stability domain. In the proposed one and through the original calculation results, the learning rate is iterative and computed instantaneously with respect to the elaborated constraints. The following assumptions are made for system (1).

*Assumption 1.* The unknown nonlinear function $F(\cdot)$ is continuous and differentiable.

*Assumption 2.* System output $y(k)$ can be measured and its initial values are assumed to be in a compact set $\Omega_0$.

**Theorem 1.** *The stability in Lyapunov sense of the identification scheme is guaranteed for a learning rate verifying the following inequality:*

$$0 \le \varepsilon \le \frac{2tr(J_\theta \cdot \theta)}{\|J_\theta\|^2}, \tag{7}$$

*where* $J_\theta = [\partial J/\partial W(k), \partial J/\partial V(k)]$ *denotes the gradient of* $J$ *with respect to* $\theta$.

*Proof.* Considering the Lyapunov function

$$V_L(k) = \mathrm{tr}\left(\widetilde{\theta}^T(k) \cdot \widetilde{\theta}(k)\right), \tag{8}$$

where $\mathrm{tr}(\cdot)$ denotes the matrix trace operation, $\widetilde{\theta}(k) = \theta(k) - \theta^*$, and $\theta^*$ denotes the optimal value of the weight vector parameters.

The computation of the $\Delta V_L(k)$ expression leads to

$$\Delta V_L(k) = V_L(k+1) - V_L(k), \tag{9}$$

where

$$\begin{aligned} V_L(k+1) &= \mathrm{tr}(\widetilde{W}^T(k+1)\widetilde{W}(k+1)) + \widetilde{V}^T(k+1)\widetilde{V}(k+1), \\ V_L(k) &= \mathrm{tr}(\widetilde{W}^T(k)\widetilde{W}(k)) + \widetilde{V}^T(k)\widetilde{V}(k). \end{aligned} \tag{10}$$

The adopted adaptation law is the gradient algorithm. We have

$$\begin{aligned} \widetilde{\theta}(k+1) &= \theta(k) - \varepsilon \cdot J_\theta - \theta^*, \\ \widetilde{W}(k+1) &= W(k) - \varepsilon \frac{\partial J}{\partial W(k)} - W^*, \\ \widetilde{V}(k+1) &= V(k) - \varepsilon \frac{\partial J}{\partial V(k)} - V^*, \end{aligned} \tag{11}$$

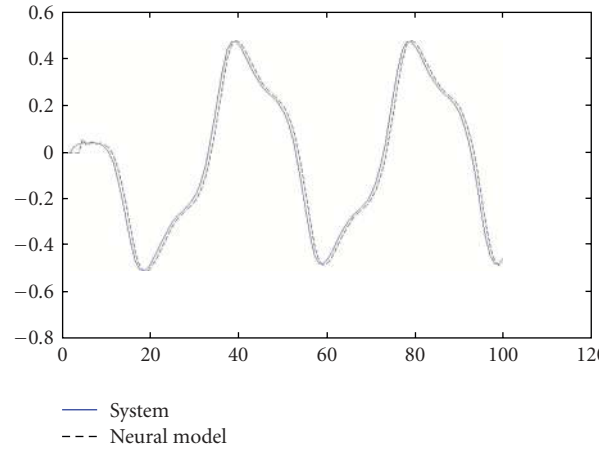


Figure 2: Evolution of the system output and the neural model output ($\varepsilon \in$ stability domain).

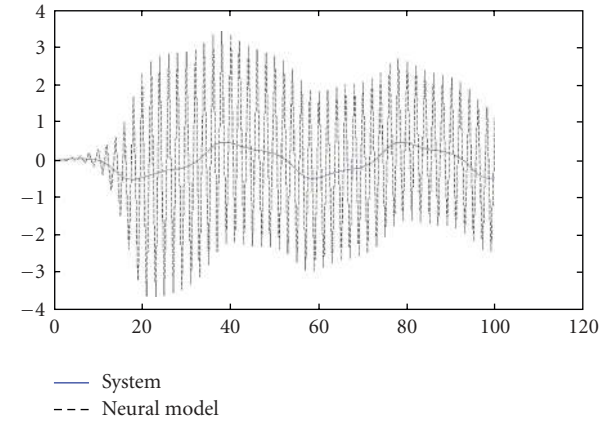


Figure 3: Evolution of the system output and the neural model output ($\varepsilon \notin$ stability domain).

where the partial derivatives are expressed as

$$\begin{aligned} \frac{\partial J}{\partial W(k)} &= \frac{\partial J}{\partial y^m(k+1)} \frac{\partial y^m(k+1)}{\partial W(k)}, \\ \frac{\partial J}{\partial V(k)} &= \frac{\partial J}{\partial y^m(k+1)} \frac{\partial y^m(k+1)}{\partial V(k)}. \end{aligned} \tag{12}$$

The partial derivatives are given through

$$\begin{aligned} &\left[\frac{\partial J}{\partial W_{ij}(k)}\right] \\ &= \left[y^m(k+1)(1 - y^m(k+1)) \cdot e(k) \cdot \sum_{j=1}^{N} V_{j1} \cdot O_j \cdot (1 - O_j)\right] x_i, \\ &\left[\frac{\partial J}{\partial V_j(k)}\right] = \left[y^m(k+1)(1 - y^m(k+1)) \cdot e(k) \cdot O_j\right]. \end{aligned} \tag{13}$$

Let $A$ and $B$ be defined as follows:

$$\begin{aligned} A &= \mathrm{tr}(\widetilde{W}^T(k+1)\widetilde{W}(k+1)) - \mathrm{tr}(\widetilde{W}^T(k)\widetilde{W}(k)), \\ B &= \widetilde{V}^T(k+1)\widetilde{V}(k+1) - \widetilde{V}^T(k)\widetilde{V}(k). \end{aligned} \tag{14}$$

The $\Delta V_L(k)$ expression is calculated as

$$\begin{aligned}\Delta V(k) &= A + B \\ &= \operatorname{tr}\left(\varepsilon^2\left[\frac{\partial J}{\partial W^T(k)}\frac{\partial J}{\partial W(k)}\right] - 2\varepsilon\frac{\partial J}{\partial W(k)}\widetilde{W}(k)\right) \\ &\quad + \left(\varepsilon^2\left[\frac{\partial J}{\partial V^T(k)}\frac{\partial J}{\partial V(k)}\right] - 2\varepsilon\frac{\partial J}{\partial V(k)}\widetilde{V}(k)\right) \\ &= \left(\varepsilon^2\sum_{i,j}\left[\frac{\partial J}{\partial W_{ij}(k)}\right]^2 - 2\varepsilon\operatorname{tr}\left(\frac{\partial J}{\partial W_{ij}(k)}\widetilde{W}^T(k)\right)\right. \\ &\quad \left. + \left(\varepsilon^2\sum_{j}\left[\frac{\partial J}{\partial V_j(k)}\right]^2 - 2\varepsilon\frac{\partial J}{\partial V(k)}\widetilde{V}^T(k)\right)\right) \\ &= \varepsilon^2\left(\sum_{i,j}\left[\frac{\partial J}{\partial W_{ij}(k)}\right]^2 + \sum_{j}\left[\frac{\partial J}{\partial V_j(k)}\right]^2\right) \\ &\quad - 2\varepsilon\left(\frac{\partial J}{\partial V(k)}\widetilde{V}^T(k) + \operatorname{tr}\left(\frac{\partial J}{\partial W(k)}\widetilde{W}^T(k)\right)\right) \\ &\le \varepsilon^2\left(\sum_{i,j}\left[\frac{\partial J}{\partial W_{ij}(k)}\right]^2 + \sum_{j}\left[\frac{\partial J}{\partial V_j(k)}\right]^2\right) \\ &\quad - 2\varepsilon\left(\frac{\partial J}{\partial V(k)}V^T(k) + \operatorname{tr}\left(\frac{\partial J}{\partial W(k)}W^T(k)\right)\right) \\ &\le \alpha\cdot\varepsilon^2 - 2\cdot\beta\cdot\varepsilon,\end{aligned} \tag{15}$$

where

$$\begin{aligned}\alpha &= \sum_{i,j}\left[\frac{\partial J}{\partial W_{ij}(k)}\right]^2 + \sum_{j}\left[\frac{\partial J}{\partial V_j(k)}\right]^2 = \|J_\theta\|^2, \\ \beta &= \frac{\partial J}{\partial V(k)}V^T(k) + \operatorname{tr}\left(\frac{\partial J}{\partial W(k)}W^T(k)\right) = \operatorname{tr}(J_\theta\cdot\theta).\end{aligned} \tag{16}$$

The stability condition $\Delta V_L(k) \le 0$ is satisfied only if

$$\alpha\cdot\varepsilon^2 - 2\cdot\beta\cdot\varepsilon \le 0. \tag{17}$$

Solving this $\varepsilon$, second-degree equation leads to the establishment of the result presented in (7); $\Delta V(k) \le 0$ if $\varepsilon$ satisfies the following condition:

$$0 \le \varepsilon \le \varepsilon_s, \tag{18}$$

where $\varepsilon_s = 2\operatorname{tr}(J_\theta\cdot\theta)/\|J_\theta\|^2$.

Using the expressions of $J_\theta$ and $\theta$, we obtain

$$\varepsilon_s = \frac{2[\operatorname{tr}(H_1\cdot W^T(k) + H_2\cdot V^T(k))]}{D + \sum_j([y^m(k+1)(1-y^m(k+1))\cdot e(k)\cdot O_j])^2}, \tag{19}$$

where $H_1$ denotes $\{[y^m(k+1)(1-y^m(k+1))\cdot e(k)\cdot\sum_{j=1}^{N}V(j,1)\cdot O_j\cdot(1-O_j)]x_i\}_{j\in[1\cdots m]}^{i\in[1\cdots n]}$, $H_2$ denotes $\{[y^m(k+1)(1-y^m(k+1))\cdot e(k)\cdot O_j]\}_{j\in[1\cdots m]}$, and $D$ denotes $\sum_{i,j}([y^m(k+1)(1-y^m(k+1))\cdot e(k)\cdot\sum_{j=1}^{N}V(j,1)\cdot O_j\cdot(1-O_j)]x_i)^2$.

The previous result is useful in the case of a backpropagation adaptation law adopting the same learning rate for training in all the neural network architecture. An extension is made in the next section. This extension consists in the fact of considering two different constrained learning rates which improve the efficiency of the first elaborated algorithm. □

**Theorem 2.** *Let $\underline{\varepsilon_I} = [\varepsilon_W, \varepsilon_V]^T$ be the learning rates for the tuning parameters of the neural identifier $\theta = [W, V]^T$ and let $\lambda$ be defined as*

$$\begin{aligned}\lambda &= [\lambda_1, \lambda_2]^T, \ \textit{where} \\ \lambda_1 &= \frac{\partial y^m(k)}{\partial W(k)}, \\ \lambda_2 &= \frac{\partial y^m(k)}{\partial V(k)}.\end{aligned} \tag{20}$$

*Then asymptotic convergence is guaranteed if the learning rates are chosen to satisfy*

$$\varepsilon_W \prec \frac{2}{(\lambda_{1\max})^2}, \qquad \varepsilon_V \prec \frac{2}{(\lambda_{2\max})^2}, \tag{21}$$

*where*

$$\begin{aligned}\lambda_{1\max} &= \left\|\frac{\partial y^m(k)}{\partial W(k)}\right\|_2 \\ &= \sqrt{\max\left(\left[\frac{\partial y^m(k)}{\partial W(k)}\right]^T\cdot\left[\frac{\partial y^m(k)}{\partial W(k)}\right]\right)}, \\ \lambda_{2\max} &= \left\|\frac{\partial y^m(k)}{\partial V(k)}\right\|_2 \\ &= \sqrt{\max\left(\left[\frac{\partial y^m(k)}{\partial V(k)}\right]^T\cdot\left[\frac{\partial y^m(k)}{\partial V(k)}\right]\right)}.\end{aligned} \tag{22}$$

**Lemma 1.** *If the learning rates are chosen as $\varepsilon_W = \varepsilon_V = \varepsilon$, then one has the convergence condition*

$$\varepsilon \prec \frac{1}{(\lambda_{1\max})^2} + \frac{1}{(\lambda_{2\max})^2}. \tag{23}$$

*Proof.* Considering the Lyapunov function

$$V_L(k) = \frac{1}{2}e^2(k), \tag{24}$$

where

$$e(k) = y(k) - y^m(k). \tag{25}$$

The computation of the $\Delta V_L(k)$ expression leads to

$$\begin{aligned}\Delta V_L(k) &= V_L(k+1) - V_L(k), \\ \Delta V_L(k) &= \frac{1}{2}(e^2(k+1) - e^2(k)) \\ &= \Delta e(k)\left[e(k) + \frac{1}{2}\Delta e(k)\right],\end{aligned} \tag{26}$$

where $\Delta e(k) = [\partial e(k)/\partial\theta(k)]^T\Delta\theta(k)$.

The expression of $\Delta W(k)$ is given by

$$\begin{aligned}\Delta\theta(k) &= \theta(k+1) - \theta(k), \\ \Delta W_I(k) &= \underline{\varepsilon_I}\cdot e(k)\cdot\frac{\partial y^m(k)}{\partial\theta(k)}.\end{aligned} \tag{27}$$

Substituting the expression of $\Delta e(k)$ in $\Delta V_L(k)$, we have

$$\begin{aligned}
\Delta V_L(k) &= \left[\frac{\partial e(k)}{\partial \theta(k)}\right]^T \underline{\varepsilon_I} e(k) \frac{\partial y^m(k)}{\partial \theta(k)} \\
&\quad \cdot \left\{ e(k) + \frac{1}{2}\left[\frac{\partial e(k)}{\partial \theta(k)}\right]^T \underline{\varepsilon_I} e(k) \frac{\partial y^m(k)}{\partial \theta(k)} \right\} \\
&= -\left[\frac{\partial y^m(k)}{\partial \theta(k)}\right]^T \underline{\varepsilon_I} e(k) \frac{\partial y^m(k)}{\partial \theta(k)} \\
&\quad \cdot \left\{ e(k) - \frac{1}{2}\left[\frac{\partial y^m(k)}{\partial \theta(k)}\right]^T \underline{\varepsilon_I} e(k) \frac{\partial y^m(k)}{\partial \theta(k)} \right\} \\
&= -\left[\frac{\partial y^m(k)}{\partial \theta(k)}\right]^T \underline{\varepsilon_I} e^2(k) \frac{\partial y^m(k)}{\partial \theta(k)} \\
&\quad \cdot \left\{ 1 - \frac{1}{2}\left[\frac{\partial y^m(k)}{\partial \theta(k)}\right]^T \underline{\varepsilon_I} \frac{\partial y^m(k)}{\partial \theta(k)} \right\} \\
&= -e^2(k) \cdot \left\{ \frac{1}{2}\left[\frac{\partial y^m(k)}{\partial V(k)}\right]^T \varepsilon_V \left[\frac{\partial y^m(k)}{\partial V(k)}\right] \right. \\
&\qquad \times \left( 2 - \left[\frac{\partial y^m(k)}{\partial V(k)}\right]^T \varepsilon_V \left[\frac{\partial y^m(k)}{\partial V(k)}\right] \right) \\
&\qquad + \frac{1}{2}\left[\frac{\partial y^m(k)}{\partial W(k)}\right]^T \varepsilon_W \left[\frac{\partial y^m(k)}{\partial W(k)}\right] \\
&\qquad \left. \times \left( 2 - \left[\frac{\partial y^m(k)}{\partial W(k)}\right]^T \varepsilon_W \left[\frac{\partial y^m(k)}{\partial W(k)}\right] \right) \right\} \\
&= -e^2(k) \cdot \left\{ \frac{1}{2}\varepsilon_V \left\| \left[\frac{\partial y^m(k)}{\partial V(k)}\right] \right\|^2 \right. \\
&\qquad \times \left( 2 - \varepsilon_V \left\| \left[\frac{\partial y^m(k)}{\partial V(k)}\right] \right\|^2 \right) \\
&\qquad + \frac{1}{2}\varepsilon_W \left\| \left[\frac{\partial y^m(k)}{\partial W(k)}\right] \right\|^2 \\
&\qquad \left. \times \left( 2 - \varepsilon_W \left\| \left[\frac{\partial y^m(k)}{\partial W(k)}\right] \right\|^2 \right) \right\}, \\
\Delta V_L(k) &\le 0,
\end{aligned} \tag{28}$$

so

$$2 - \varepsilon_W \left\| \frac{\partial y^m(k)}{\partial W(k)} \right\|^2 \le 0, \qquad 2 - \varepsilon_V \left\| \frac{\partial y^m(k)}{\partial V(k)} \right\|^2 \le 0. \tag{29}$$

Finally, when we define the matrix norm $\|\cdot\|_2$ by

$$\|\rho\|_2 = \sqrt{\max\left(\rho^T \cdot \rho\right)}, \tag{30}$$

the theorem results are established.

The stability condition $\Delta V_L(k) \le 0$ is satisfied only if

$$\varepsilon_W \prec \frac{2}{(\lambda_{1\max})^2}, \qquad \varepsilon_V \prec \frac{2}{(\lambda_{2\max})^2}, \tag{31}$$

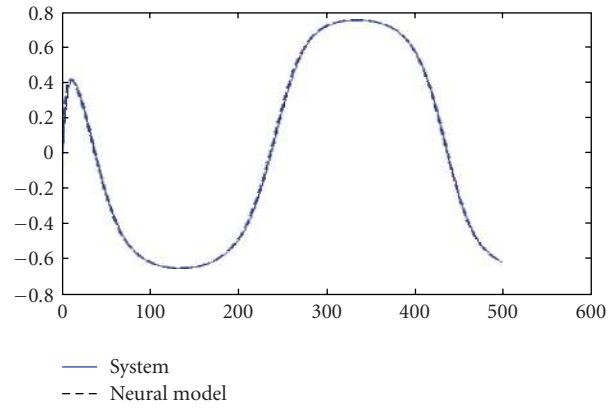


Figure 4: Evolution of the system output and the neural model output ($\varepsilon \in$ stability domain).

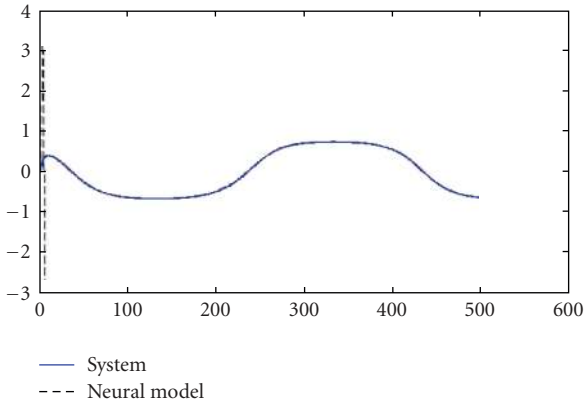


Figure 5: Evolution of the system output and the neural model output ($\varepsilon \notin$ stability domain).

where

$$\begin{aligned}
\lambda_{1\max} &= \left\| \frac{\partial y^m(k)}{\partial W(k)} \right\|_2 = \sqrt{\max\left( \left[\frac{\partial y^m(k)}{\partial W(k)}\right]^T \cdot \left[\frac{\partial y^m(k)}{\partial W(k)}\right] \right)}, \\
\lambda_{2\max} &= \left\| \frac{\partial y^m(k)}{\partial V(k)} \right\|_2 = \sqrt{\max\left( \left[\frac{\partial y^m(k)}{\partial V(k)}\right]^T \cdot \left[\frac{\partial y^m(k)}{\partial V(k)}\right] \right)}, \\
\lambda_{1\max} &= \left\| \frac{\partial y^m(k)}{\partial W(k)} \right\|_2 = \sqrt{\max\left( \left[\frac{\partial y^m(k)}{\partial W(k)}\right]^T \cdot \left[\frac{\partial y^m(k)}{\partial W(k)}\right] \right)}, \\
\lambda_{2\max} &= \left\| \frac{\partial y^m(k)}{\partial V(k)} \right\|_2 = \sqrt{\max\left( \left[\frac{\partial y^m(k)}{\partial V(k)}\right]^T \cdot \left[\frac{\partial y^m(k)}{\partial V(k)}\right] \right)}.
\end{aligned} \tag{32}$$

□

*Remark 1.* Through simulations, learning rates are chosen belonging to the defined learning rates stability range to prove the effectiveness of the proposed CSBP algorithm. The learning rate which guarantees convergence corresponds to

$$\varepsilon_W = \frac{2}{\phi + (\lambda_{1\max})^2}, \qquad \varepsilon_V = \frac{2}{\phi + (\lambda_{2\max})^2}, \tag{33}$$

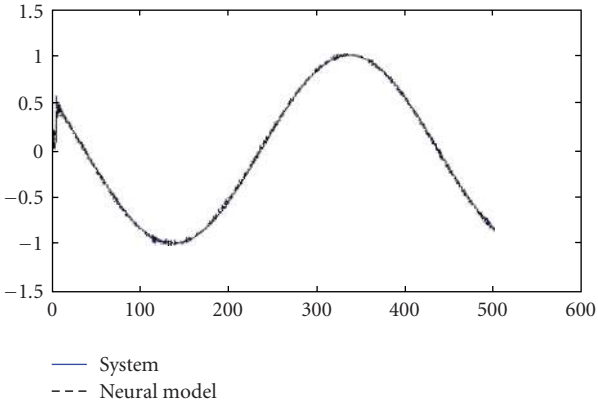


Figure 6: Evolution of the system output and the neural model output ($\varepsilon \in$ stability domain).

where $\phi$ is a small value guarantying the convergence stability condition.

## 4. Simulation Results

In this section, two discrete time systems are considered to demonstrate the effectiveness of the result discussed below.

*4.1. First-Order System.* The considered system is a famous one in the literature of neural adaptive control and identification. The system is described by the following recurrent equation [2]:

$$y(k+1) = \frac{y(k)}{1+y(k)^2} + u(k)^3. \tag{34}$$

For the neural model, a three-layer NN was selected with two inputs, three hidden and one output nodes. Sigmoidal activation functions were employed in all the nodes.

The weights are initialized to small random values. The learning rate is evaluated at each iteration through (21). It is also recognized that the training performs very well when the learning rate is small.

As input signal, a sinusoidal one is chosen in which the expression is defined by

$$u(k) = 0.5 \cdot \cos\left(0.05k\pi + \frac{\pi}{5}\right). \tag{35}$$

The simulations are realized in the two cases during 120 iterations. Two learnning rate values are fixed in and out of the learning rate range presented in (7).

Simulation results are given through Figures 2 and 3.

Figures 2 and 3 show that if the learning rate belongs to the range defined in (7), the stability of the identification scheme is guaranteed. It is shown through this simulation that the identification objectives are satisfied. Out of this variation domain of the learning rate, the identification is instable and the identification objectives are unreachable.

*4.2. Second-Order System.* An example is used to illustrate the effectiveness of the proposed constrained updating law. Consider a nonlinear discrete time plant

$$y(k) = 50\tanh[\phi(k-1)] + 0.5u(k-1), \tag{36}$$

where $\phi(k-1) = 2.10^2(((24+y(k-1))/3)y(k-1) - 8(u^2(k-1)/(1+u^2(k-1)))y(k-2))$.

The process dynamic is interesting. In fact it has the behaviour of a first-order lowpass filter for input signal amplitude about 0.1, the behaviour of a linear second-order system in the case of small amplitudes ($0.1 < |u| < 0.5$), and the behaviour of a nonlinear second-order system in the case of great input amplitudes ($0.5 < |u| < 5$) [20].

For the neural model, a three-layer NN was selected with three inputs, three hidden and one output nodes. Sigmoidal activation functions were employed in all the nodes.

The weights are initialized to small random values. The learning rate parameter is computed instantaneously. As input signal, a sinusoidal one is chosen which the expression is defined by

$$u(k) = 0.5 \cdot \cos\left(0.005k\pi + \frac{\pi}{3}\right). \tag{37}$$

The simulations are realized in the two cases during 120 iterations. Two learning rates values are fixed in and out of the learning rate range presented in (7).

Simulation results are given through Figures 4 and 5.

The simulation results, through Figures 4 and 5, show that a learning rate arbitrarily chosen out of the predefined stability domain leads to an unstable identification of the considered system; however, a specified learning rate belonging to the range verifying stability condition ensures the tracking capability and the stability of the identification scheme

*Example 1* (identification of semiconductor manufacturing process). This example illustrates the advantage and effectiveness of our approach-on-line self-tuning property (stability). We consider here the SISO simple first-order linear process of the form [21]

$$y(k+1) = \phi y(k) + \alpha + \beta u(k) + N(k+1), \tag{38}$$

where $\alpha$ and $\beta$ are process parameters, $\phi$ is the autoregressive coefficient, and $N$ denotes the noise term that follows an ARMA process:

$$N(k) = \frac{1-cz^{-1}}{1-wz^{-1}} \cdot r(k). \tag{39}$$

In this simulation, $r(k)$ is a uniform distribution and the system parameters are chosen as

$$\begin{gathered} \alpha = 2, \quad \beta = 2, \quad \phi = 0.1, \quad c = 0.7, \\ w = 1.0, \quad r(k) = 1 - 2 \cdot \text{rand}\ (k). \end{gathered} \tag{40}$$

Here, the current output of the plant depends on four previous outputs and four previous inputs. In this case, the feedforward neural network, with four input nodes for

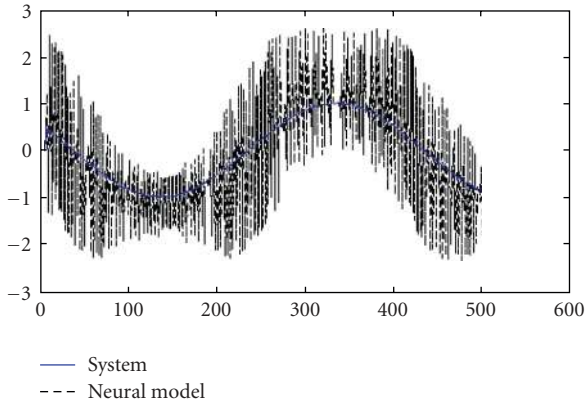


Figure 7: Evolution of the system output and the neural model output ($\varepsilon \notin$ stability domain).

feeding the appropriate past values of $y(k)$ and $u(k)$ were used. In this paper, only four values are fed into the FFNN to determine the output $y^m(k)$. In training the FFNN, we used 100 epochs. The testing input signal is used to determine the identification results and is given by (37).

The weights are initialized to very small random values. The learning rate parameter is calculated each iteration.

The simulations are realized in the two cases. Two learning rates values are fixed in and out of the learning rate range presented in (21) (see Figures 6 and 7).

In order to compare the performances of different learning rate, rules are chosen in and out the learning rate stability range.

Adopting an adaptive constrained learning rate inside the stability domain, faster convergence, stability, and tracking capability are guaranteed.

## 5. Conclusion

To avoid unstable phenomenon during the learning process, constrained stable backpropagation algorithm is proposed (CSBP). A stable adaptive updating process is guaranteed. A Lyapunov analysis is made in order to extract new updating formulations which contain a set of inequality constraints. Both the convergence rate and the tracking capability of the CSBP algorithm are mainly determined by the learning rate. For a larger learning rate, one has the faster convergence but the poorer tracking capability; while for a smaller learning rate, one gets the slower convergence but the better tracking capability. With The CSBP algorithm, faster convergence, stability, and tracking capability are guaranteed. The applicability and the effectiveness of the approach presented are proved through simulation examples.

## References

[1] L. Chen and K. S. Narendra, "Identification and control of a nonlinear discrete-time system based on its linearization: a unified framework," *IEEE Transactions on Neural Networks*, vol. 15, no. 3, pp. 663–673, 2004.

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

[3] J. D. Bošković and K. S. Narendra, "Comparison of linear, nonlinear and neural-network-based adaptive controllers for a class of fed-batch fermentation processes," *Automatica*, vol. 31, no. 6, pp. 817–840, 1995.

[4] W. Yu and X. Li, "Some stability properties of dynamic neural networks," *IEEE Transactions on Circuits and Systems I*, vol. 48, no. 2, pp. 256–259, 2001.

[5] Z.-P. Jiang and Y. Wang, "Input-to-state stability for discrete-time nonlinear systems," *Automatica*, vol. 37, no. 6, pp. 857–869, 2001.

[6] M. M. Polycarpou and P. A. Ioannou, "Learning and convergence analysis of neural-type structured networks," *IEEE Transactions on Neural Networks*, vol. 3, no. 1, pp. 39–50, 1992.

[7] S. S. Ge, C. C. Hang, T. H. Lee, and T. Zhang, *Stable Adaptive Neural Network Control*, Kluwer Academic Publishers, Boston, Mass, USA, 2001.

[8] E. B. Kosmatopoulos, M. M. Polycarpou, M. A. Christodoulou, and P. A. Ioannou, "High-order neural network structures for identification of dynamical systems," *IEEE Transactions on Neural Networks*, vol. 6, no. 2, pp. 422–431, 1995.

[9] W. Yu, A. S. Poznyak, and X. Li, "Multilayer dynamic neural networks for non-linear system on-line identification," *International Journal of Control*, vol. 74, no. 18, pp. 1858–1864, 2001.

[10] Z. Feng and A. N. Michel, "Robustness analysis of a class of discrete-time systems with applications to neural networks," in *Proceedings of the American Control Conference (ACC '99)*, vol. 5, pp. 3479–3483, San Diego, Calif, USA, June 1999.

[11] J. A. K. Suykens, J. Vandewalle, and B. L. R. De Moor, "$NL_q$ theory: checking and imposing stability of recurrent neural networks for nonlinear modeling," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2682–2691, 1997.

[12] L. Jin and M. M. Gupta, "Stable dynamic backpropagation learning in recurrent neural networks," *IEEE Transactions on Neural Networks*, vol. 10, no. 6, pp. 1321–1334, 1999.

[13] P. A. Ioannou and J. Sun, *Robust Adaptive Control*, Prentice-Hall, Upper Saddle River, NJ, USA, 1995.

[14] W. Yu, "Nonlinear system identification using discrete-time recurrent neural networks with stable learning algorithms," *Information Sciences*, vol. 158, pp. 131–147, 2004.

[15] E. Barnard, "Optimization for training neural nets," *IEEE Transactions on Neural Networks*, vol. 3, no. 2, pp. 232–240, 1992.

[16] B. Egardt, *Stability of Adaptive Controllers*, Lecture Notes in Control and Information Sciences 20, Springer, Berlin, Germany, 1979.

[17] R. A. Jacobs, "Increased rates of convergence through learning rate adaptation," *Neural Networks*, vol. 1, no. 4, pp. 295–307, 1988.

[18] T. P. Vogl, J. K. Mangis, A. K. Rigler, W. T. Zink, and D. L. Alkon, "Accelerating the convergence of the back-propagation method," *Biological Cybernetics*, vol. 59, no. 4-5, pp. 257–263, 1988.

[19] T. Tollenaere, "SuperSAB: fast adaptive back propagation with good scaling properties," *Neural Networks*, vol. 3, no. 5, pp. 561–573, 1990.

[20] C.-H. Lee and C. C. Teng, "Control of a nonlinear dynamic system via adaptive PID control scheme with saturation bound," *International Journal of Fuzzy Systems*, vol. 4, no. 4, pp. 922–927, 2002.

[21] E. Del Castillo and A. M. Hurwitz, "Run-to-run process control: literature review and extensions," *Journal of Quality Technology*, vol. 29, no. 2, pp. 184–196, 1997.