Journal of Software Engineering and Applications, 2012, 5, 225-248
http://dx.doi.org/10.4236/jsea.2012.54028 Published Online April 2012 (http://www.SciRP.org/journal/jsea)

Lyapunov-Based Dynamic Neural Network for Adaptive Control of Complex Systems

Farouk Zouari, Kamel Ben Saad, Mohamed Benrejeb

Unité de Recherche LARA Automatique, Ecole Nationale d'Ingénieurs de Tunis (ENIT), Tunis, Tunisia.
Email: zouari.farouk@gmail.com, {kamel.bensaad, mohamed.benrejeb}@enit.rnu.tn

Received December 14th, 2011; revised January 17th, 2012; accepted February 20th, 2012

ABSTRACT

In this paper, an adaptive neuro-control structure for complex dynamic system is proposed. A recurrent Neural Network is trained-off-line to learn the inverse dynamics of the system from the observation of the input-output data. The direct adaptive approach is performed after the training process is achieved. A Lyapunov-Base training algorithm is proposed and used to adjust on-line the network weights so that the neural model output follows the desired one. The simulation results obtained verify the effectiveness of the proposed control method.

Keywords: Complex Dynamical Systems; Lyapunov Approach; Recurrent Neural Networks; Adaptive Control

1. Introduction

For several decades, the problem of adaptive control of complex dynamic systems causes the interest of automation specialists. The use of Proportional-Integral-Derivative (PID) controllers is simple to perform, that give poor performance if there are uncertainties and nonlinearities in the system to be controlled. In several references like [1-3], neural networks are presented as tools to solve control problems due to their ability to model systems without analyzing them theoretically and their possessions a great capacity for generalization, which gives them a good robustness to noise [4].

Several strategies of the neural adaptive control exist which we quote: direct adaptive neural control, indirect adaptive neuronal control, adaptive neural internal model control, adaptive depth control based on feedforward neural networks, robust adaptive neural control, Feedback-Linearization based neural adaptive control, adaptive neural network model based nonlinear predictive control [5-13]. Each strategy has neural adaptive control architecture, the algorithms used during the calculation of the parameters and stability conditions. It has three types of neural adaptive control architectures. The first type of architecture consists of a neural controller and a system to be controlled. The second type of neural architecture includes a controller, a system to be controlled and his neural model. The third type of architecture is composed of a neuronal controller, one or more robustness filter, a system to be controlled and his neural model.

The adjustment of the model parameters and the controller is performed by neural learning algorithms that are based on the choice of the criterion to minimize, a minimization method and the theory of Lyapunov for stability and borniture of all signals existing. Several minimization methods exist which are presented: simple gradient method, gradient method with variable pitch, Newton method and Levenberg-Marquardt method [14].

The contribution of this paper is to propose an adaptive Lyapunov-Based control strategy for complex dynamic system. The control structure takes advantage of Artificial Neural Network (ANN) learning and generalization capabilities to achieve accurate speed tracking and estimation. ANN-Based controllers lack stability proofs in many control structure applications and tuning them in a cascaded control structure is a difficult task to undertake. Therefore, we proposed a Lyapunov stability-Based adaptation technique as an alternative to the conventional gradient-Based and heuristic tuning methods. Thus, the stability of the proposed approach is guaranteed by Lyapunov Stability direct method unlike many computational intelligence- Based controllers.

The different sections of this paper are organized as follows: in Section 2, we present the considered recurrent neural network and the proposed Lyapunov learning algorithm used for updating the weight parameters of the model system.

The proposed adaptive control approach training while a Lyapunov Stability-Based adaptation algorithm is detailed in Section 3. Numerical results are reported and discussed in Section 4, and a conclusion is drawn in Section 5.

2. Neural Network Modeling Approach

Neural network modeling of a system from samples affected by noise usually requires three steps. The first step is the choice of neural network architecture, that is to say, the number of neurons in the input layer which is a function of past values of the input and output, the number of hidden neurons, the number of neurons in the output layer neurons and the organization of them. The work [15,16] show that every continuous function can be approximated by a neural network with three layers, the activation functions of neurons are respectively the sigmoid function for hidden layer neurons and linear function for neurons in the output layer. There are two types of architectures of multilayer neural networks: neural networks, non-curly (static networks) and neural networks curly or recurrent (dynamic networks). Neural networks are non curly most used in the identification and control systems [17]. They may not be powerful enough to model complex dynamic systems with respect to neural networks curly. Different types of recurrent neural networks have been proposed and have been successfully applied in many fields [18-25]. The structure of fully connected recurrent neural networks which was proposed by Williams and Zipser [26], is most often used [27,28] because of its generality. The second step is learning or in other words, estimating the parameters of the network from examples of input-output system identification. The methods of learning are numerous and depend on several factors, including the choice of error function, the initialization of weights, and the selection of the learning algorithm and the stopping criteria of learning. Learning strategies were presented in several research works that we cite [29-31]. The third step is the validation of the neural network obtained using the testing criteria for measuring performance. Most of these tests require a data set that was not used in learning. Such a test set or validation should, if possible, cover the same range of operation given that all learning.

2.1. Architecture of the Recurrent Neural Network

In this work, we consider a recurrent neural network (**Figure 1**) for identification of complex dynamic systems to a single input and single output. The architecture of these networks is composed of two parts: a linear model the linear behavior of the system and a non-linear approach to nonlinear dynamics.

where:

$\hat{y}(k)$ is the output of the neural network at time k,

u and y are respectively the input and output system to identify,

$f_1(x) = \frac{e^{-x}-1}{1+e^{-x}}$ and $f_2(x) = x$ are the activation functions of neurons,

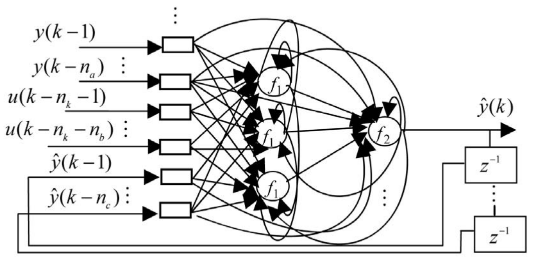


Figure 1. Architecture of the considered neural network.

n_h the number of neurons in the hidden layer respectively of the model and controller,

The coefficients of the vector of parameters of the neural model w are decomposed into 7 groups, formed respectively by:

$w^1 = \begin{bmatrix} w^1_{11} & \cdots & w^1_{1n_r} \\ \vdots & \ddots & \vdots \\ w^1_{n_h1} & \cdots & w^1_{n_hn_r} \end{bmatrix}$ the weights between neurons in the input layer and neurons in the hidden layer,

$w^2 = \begin{bmatrix} w^2_{11} \\ \vdots \\ w^2_{n_h1} \end{bmatrix}$ the bias of neurons in the hidden layer,

$w^3 = \begin{bmatrix} w^3_{11} \cdots w^3_{1n_h} \end{bmatrix}$ the weights between neurons in the hidden layer neurons and output layer,

$w^4 = \begin{bmatrix} w^4_{11} \end{bmatrix}$ the bias of neuron in the output layer,

$w^5 = \begin{bmatrix} w^5_{11} \cdots w^5_{1n_r} \end{bmatrix}$ the weights between neurons of input layer neurons and output layer,

$w^6 = \begin{bmatrix} w^6_{11} & \cdots & w^6_{1n_h} \\ \vdots & \ddots & \vdots \\ w^6_{n_h1} & \cdots & w^6_{n_hn_h} \end{bmatrix}$ the weights between neurons in the hidden layer,

$w^7 = \begin{bmatrix} w^7_{11} \end{bmatrix}$ back weight of neuron in the output layer,

$x^h = \begin{bmatrix} x^h_{11} \\ \vdots \\ x^h_{n_h1} \end{bmatrix}$ the outputs of the hidden layer of neural model,

$n_r = n_a + n_b + n_c$ number of neurons in the input layer.

$$\begin{aligned}\psi(k) &= \big[y(k-1),\cdots,y(k-n_a),u(k-n_k-1),\\ &\quad \cdots,u(k-n_k-n_b),\hat{y}(k-1),\cdots,\hat{y}(k-n_c)\big]^T \\ &= \big[\psi_1(k),\cdots,\psi_{n_r}(k)\big]^T\end{aligned} \tag{1}$$

The vector of parameters of the neural model is defined as:

$$w = \left[w^1_{11}, \cdots, w^1_{n_h n_r}, w^2_{11}, \cdots, w^2_{n_h 1}, w^3_{11}, \cdots, w^3_{1n_h}, w^4_{11}, w^5_{11}, \cdots, w^5_{1n_r}, w^6_{11}, \cdots, w^6_{n_h n_h}, w^7_{11} \right]^T \quad (2)$$

The output of neural model $\hat{y}(k)$ is given by:

$$\hat{y}(k) = w^7_{11}(k)\hat{y}(k-1) + \sum_{j=1}^{n_h} w^3_{1j}(k) x^h_j(k) + \sum_{i=1}^{n_r} w^5_{1i}(k)\psi_i(k) + w^4_{11}(k) \quad (3)$$

such as:

$$s_j(k) = \sum_{m=1}^{n_r} w^1_{jm}(k)\psi_m(k) + \sum_{m=1}^{n_h} w^6_{jm}(k) x^h_m(k-1) + w^2_{j1}(k) \quad (4)$$

$$x^h_j(k) = f_1(s_j(k)) \quad (5)$$

$$s(k) = \left[s_1(k), \cdots, s_{n_h}(k) \right]^T \quad (6)$$

$$x^h(k) = \left[x^h_1(k), \cdots, x^h_{n_h}(k) \right]^T \quad (7)$$

The neural model of the system can be expressed by the following expression:

$$\hat{y}(k) = f\big(y(k-1), \cdots, y(k-n_a), u(k-n_k-1), \cdots, u(k-n_k-n_b), \hat{y}(k-1), \cdots, \hat{y}(k-n_c), w(k)\big) \quad (8)$$

2.2. Proposed Lyapunov-Base Learning Algorithms

Several Lyapunov stability-based ANN learning techniques are also proposed to insure the ANNs' convergence and stability [32,33]. In this section we present three learning procedures of neural network.

***Theorem* 1.** *The learning procedure of a neural network can be given by the following equation*:

$$w(k+1) = w(k) + \frac{\lambda}{\gamma}\left(\frac{1}{1+\frac{\beta}{\gamma}\left\| \frac{\partial \hat{y}(k)}{\partial w(k)} \right\|^2} \right) e(k) \frac{\partial \hat{y}(k)}{\partial w(k)} \quad (9)$$

such as:

$$0 < \lambda \le \gamma \le \beta \quad (10)$$

$$e(k) = y(k) - \hat{y}(k) \quad (11)$$

$\| \; \|$ *is the Euclidean norm.*

Proof:

Considering the following quadratic criterion:

$$J_r(k) = \frac{\lambda}{2}(e(k))^2 + \frac{\beta}{2}(\Delta e(k))^2 + \frac{\gamma}{2}(\Delta w(k))^2 \quad (12)$$

$$\Delta e(k) = e(k) - e(k-1) \cong e(k+1) - e(k) \quad (13)$$

$$\Delta w(k) = w(k) - w(k-1) \cong w(k+1) - w(k) \quad (14)$$

The learning procedure is to adjust the coefficients of the neural networks considered by minimizing the criterion $J_r(k)$; it is necessary to solve the equation:

$$\frac{\partial J_r(k)}{\partial w(k)} = 0 \quad (15)$$

therefore:

$$\begin{aligned} w(k+1) &= w(k) - \frac{\lambda}{\gamma} e(k) \frac{\partial e(k)}{\partial w(k)} - \frac{\beta}{\gamma}\Delta e(k) \frac{\partial e(k)}{\partial w(k)} \\ &= w(k) + \frac{\lambda}{\gamma} e(k) \frac{\partial \hat{y}(k)}{\partial w(k)} + \frac{\beta}{\gamma}\Delta e(k) \frac{\partial \hat{y}(k)}{\partial w(k)} \end{aligned} \quad (16)$$

According to reference [34], the difference in error due to learning can be calculated by:

$$\begin{aligned} e(k+1) &= e(k) + \Delta e(k) \\ &= e(k) + \left[\frac{\partial e(k)}{\partial w(k)} \right]^T (\Delta w(k)) \end{aligned} \quad (17)$$

The term $\Delta e(k)$ is then written as follows:

$$\begin{aligned} \Delta e(k) &= \left[\frac{\partial e(k)}{\partial w(k)} \right]^T (\Delta w(k)) \\ &= \left[\frac{\partial e(k)}{\partial w(k)} \right]^T \left(-\frac{\lambda}{\gamma} e(k) \frac{\partial e(k)}{\partial w(k)} - \frac{\beta}{\gamma}\Delta e(k) \frac{\partial e(k)}{\partial w(k)} \right) \\ &= -\frac{\lambda}{\gamma} e(k) \left\| \frac{\partial e(k)}{\partial w(k)} \right\|^2 - \frac{\beta}{\gamma}\Delta e(k) \left\| \frac{\partial e(k)}{\partial w(k)} \right\|^2 \end{aligned} \quad (18)$$

Therefore:

$$\Delta e(k) = -\frac{\frac{\lambda}{\gamma} e(k) \left\| \frac{\partial e(k)}{\partial w(k)} \right\|^2}{1 + \frac{\beta}{\gamma}\left\| \frac{\partial e(k)}{\partial w(k)} \right\|^2} \quad (19)$$

According to Equations (16) and (19), we can write:

$$w(k+1)$$

$$= w(k) - \frac{\lambda}{\gamma} e(k)\frac{\partial e(k)}{\partial w(k)} - \frac{\beta}{\gamma}\left(\frac{-\frac{\lambda}{\gamma}e(k)\left\|\frac{\partial e(k)}{\partial w(k)}\right\|^2}{1+\frac{\beta}{\gamma}\left\|\frac{\partial e(k)}{\partial w(k)}\right\|^2}\right)\frac{\partial e(k)}{\partial w(k)}$$

$$= w(k) - \frac{\lambda}{\gamma} e(k)\frac{\partial e(k)}{\partial w(k)}\left(1-\frac{\beta}{\gamma}\left(\frac{\left\|\frac{\partial e(k)}{\partial w(k)}\right\|^2}{1+\frac{\beta}{\gamma}\left\|\frac{\partial e(k)}{\partial w(k)}\right\|^2}\right)\right)$$

$$= w(k) - \frac{\lambda}{\gamma}\left(\frac{1}{1+\frac{\beta}{\gamma}\left\|\frac{\partial e(k)}{\partial w(k)}\right\|^2}\right)e(k)\frac{\partial e(k)}{\partial w(k)}$$

$$= w(k) + \frac{\lambda}{\gamma}\left(\frac{1}{1+\frac{\beta}{\gamma}\left\|\frac{\partial \hat{y}(k)}{\partial w(k)}\right\|^2}\right)e(k)\frac{\partial \hat{y}(k)}{\partial w(k)} \qquad (20)$$

The parameters λ, β, γ are chosen so that the neural model of the system must be stable. In our case, the stability analysis is based on the famous Lyapunov approach [35]. It is well known that the purpose of identification is to have a zero gap between the output of the system and that of the neural model. Three Lyapunov candidate functions are proposed:

The first candidate Lyapunov function is defined by:

$$V(t) = \frac{1}{2}\left(e(t)\right)^2 \qquad (21)$$

The function $V(t)$ satisfies the following conditions: $V(t)$ is continuous and differentiable.

$$V(t) = 0 \ \text{ si } \ e(t) = 0.$$

$$V(t) > 0, \ \forall e(t) \neq 0.$$

The neural model is stable in the sense of Lyapunov if $\dot{V}(t) \leq 0$ or simply $\Delta V(k) \leq 0$.

The term $\Delta V(k)$ is given by the following equation:

$$\Delta V(k) = \frac{1}{2}\left(\left(e(k+1)\right)^2 - \left(e(k)\right)^2\right) \qquad (22)$$

From Equation (22), $\Delta V(k)$ may be as follows:

$$\begin{aligned}\Delta V(k) &= \frac{1}{2}\left(\left(e(k)+\Delta e(k)\right)^2 - \left(e(k)\right)^2\right)\\ &= \Delta e(k)\left(e(k)+\frac{1}{2}\Delta e(k)\right)\\ &= \Delta e(k)e(k)+\frac{1}{2}\left(\Delta e(k)\right)^2\end{aligned} \qquad (23)$$

From the above equations, we obtain:

$$\Delta V(k) = \frac{-\frac{\lambda}{\gamma}\left(e(k)\right)^2\left\|\frac{\partial e(k)}{\partial w(k)}\right\|^2}{1+\frac{\beta}{\gamma}\left\|\frac{\partial e(k)}{\partial w(k)}\right\|^2}\left(1-\frac{\frac{\lambda}{\gamma}\left\|\frac{\partial e(k)}{\partial w(k)}\right\|^2}{2\left(1+\frac{\beta}{\gamma}\left\|\frac{\partial e(k)}{\partial w(k)}\right\|^2\right)}\right) \qquad (24)$$

like:

$$\frac{-\frac{\lambda}{\gamma}\left(e(k)\right)^2\left\|\frac{\partial e(k)}{\partial w(k)}\right\|^2}{1+\frac{\beta}{\gamma}\left\|\frac{\partial e(k)}{\partial w(k)}\right\|^2} \leq 0 \qquad (25)$$

The proposed neural model is stable in the sense of Lyapunov if and only if:

$$1-\frac{\frac{\lambda}{\gamma}\left\|\frac{\partial e(k)}{\partial w(k)}\right\|^2}{2\left(1+\frac{\beta}{\gamma}\left\|\frac{\partial e(k)}{\partial w(k)}\right\|^2\right)} \geq 0 \qquad (26)$$

noting that:

$$d = \max\left(\left\|\frac{\partial e(k)}{\partial w(k)}\right\|\right) = \max\left(\left\|\frac{\partial \hat{y}(k)}{\partial w(k)}\right\|\right) \qquad (27)$$

Therefore, we will have:

$$1-\frac{\frac{\lambda}{\gamma}\left\|\frac{\partial e(k)}{\partial w(k)}\right\|^2}{2\left(1+\frac{\beta}{\gamma}\left\|\frac{\partial e(k)}{\partial w(k)}\right\|^2\right)} \geq 1-\frac{\frac{\lambda}{\gamma}d^2}{2\left(1+\frac{\beta}{\gamma}d^2\right)} \geq 0 \qquad (28)$$

The stability condition becomes:

$$\frac{\lambda-2\beta}{\gamma} \leq \frac{2}{d^2} \qquad (29)$$

The second candidate Lyapunov function is:

$$V(k) = \frac{1}{2}\left(e(k)\right)^2 + \frac{1}{2}\left(\Delta e(k)\right)^2 \qquad (30)$$

Given that:

$$\Delta V(k)=e(k)\left(\frac{\partial e(k)}{\partial w(k)}\right)^T(\Delta w(k))+(\Delta e(k))\left(\frac{\partial e(k)}{\partial w(k)}\right)^T(\Delta w(k)) \quad (31)$$

Using Equations (19) and (20), the above relation becomes:

$$\Delta V(k)=e(k)(\Delta e(k))+(\Delta e(k))^2 =\frac{-\frac{\lambda}{\gamma}(e(k))^2\left\|\frac{\partial e(k)}{\partial w(k)}\right\|^2}{1+\frac{\beta}{\gamma}\left\|\frac{\partial e(k)}{\partial w(k)}\right\|^2}\left(1-\frac{\frac{\lambda}{\gamma}\left\|\frac{\partial e(k)}{\partial w(k)}\right\|^2}{\left(1+\frac{\beta}{\gamma}\left\|\frac{\partial e(k)}{\partial w(k)}\right\|^2\right)}\right) \quad (32)$$

Then the second stability condition is:

$$\frac{\lambda-\beta}{\gamma}\le\frac{1}{d^2} \quad (33)$$

The third and last candidate Lyapunov function is:

$$V(k)=\frac{1}{2}(e(k))^2+\frac{1}{2}\frac{\gamma}{\lambda}(\Delta e(k))^2 \quad (34)$$

The term $\Delta V(k)$ is as follows:

$$\Delta V(k)=e(k)\left(\frac{\partial e(k)}{\partial w(k)}\right)^T\cdot(\Delta w(k))+\frac{\gamma}{\lambda}(\Delta e(k))\left(\frac{\partial e(k)}{\partial w(k)}\right)^T(\Delta w(k))$$
$$=e(k)(\Delta e(k))+\frac{\gamma}{\lambda}(\Delta e(k))^2$$
$$=\frac{-\frac{\lambda}{\gamma}(e(k))^2\left\|\frac{\partial e(k)}{\partial w(k)}\right\|^2}{1+\frac{\beta}{\gamma}\left\|\frac{\partial e(k)}{\partial w(k)}\right\|^2}\left(1-\frac{\left\|\frac{\partial e(k)}{\partial w(k)}\right\|^2}{1+\frac{\beta}{\gamma}\left\|\frac{\partial e(k)}{\partial w(k)}\right\|^2}\right) \quad (35)$$

The third stability condition is:

$$\left(1+\frac{\beta}{\gamma}\left\|\frac{\partial e(k)}{\partial w(k)}\right\|^2\right)-\left\|\frac{\partial e(k)}{\partial w(k)}\right\|^2>0 \quad (36)$$

therefore:

$$\frac{\beta}{\gamma}\ge 1-\frac{1}{d^2}\ge 1-\frac{1}{\left\|\frac{\partial e(k)}{\partial w(k)}\right\|^2} \quad (37)$$

To meet the three conditions of stability of Lyapunov candidate functions proposed parameters λ, β and γ must verify:

$$\frac{\lambda}{\gamma}\le 1\le\frac{\beta}{\gamma}+\frac{1}{d^2} \quad (38)$$

$$\frac{\beta}{\gamma}\ge 1\ge 1-\frac{1}{d^2} \quad (39)$$

then:

$$\lambda\le\gamma\le\beta \quad (40)$$

***Theorem* 2.** *The parameters of the neural network can be adjusted using the following equation*:

$$w(k+1)=\left(1-\frac{1}{2\left(1+\left\|\frac{\partial\hat{y}(k)}{\partial w(k)}\right\|^2\right)}\right)w(k)+\frac{1}{2\left(1+\left\|\frac{\partial\hat{y}(k)}{\partial w(k)}\right\|^2\right)}e(k)\frac{\partial\hat{y}(k)}{\partial w(k)} \quad (41)$$

***Proof*:**

Using the following Lyapunov function:

$$V(k)=\frac{1}{2}(e(k))^2+\frac{1}{2}(\Delta e(k))^2+\frac{1}{2}\|w(k)\|^2+\frac{1}{2}\|\Delta w(k)\|^2 \quad (42)$$

The learning procedure of the neural network is stable if:

$$\Delta V(k)=(e(k))(\Delta e(k))+\|\Delta e(k)\|^2+(w(k))^T(\Delta w(k))+\|\Delta w(k)\|^2\le 0 \quad (43)$$

Using the above equation, we can write:

$$\Delta V(k)=\|\Delta w(k)\|^2+\|\Delta e(k)\|^2+(w(k))^T(\Delta w(k))+(e(k))(\Delta e(k))=-r \quad (44)$$

such as $r\ge 0$

therefore:

$$\|\Delta w(k)\|^2+\|\Delta e(k)\|^2+(w(k))^T(\Delta w(k))+(e(k))(\Delta e(k))+r$$
$$=\|\Delta w(k)\|^2\left(1+\left\|\frac{\Delta e(k)}{\Delta w(k)}\right\|^2\right)+\left(w(k)+e(k)\frac{\Delta e(k)}{\Delta w(k)}\right)^T(\Delta w(k))+r=0 \quad (45)$$

For Equation (45) has a unique solution requires that:

$$\left\| w(k)+e(k)\frac{\Delta e(k)}{\Delta w(k)} \right\|^2 - 4\left(1+\left\|\frac{\Delta e(k)}{\Delta w(k)}\right\|^2\right) r = 0 \quad (46)$$

then:

$$r = \frac{\left\| w(k)+e(k)\frac{\Delta e(k)}{\Delta w(k)} \right\|^2}{4\left(1+\left\|\frac{\Delta e(k)}{\Delta w(k)}\right\|^2\right)} \quad (47)$$

The term $\Delta w(k)$ can be written as follows:

$$\Delta w(k) = -\frac{\left(w(k)+e(k)\frac{\Delta e(k)}{\Delta w(k)}\right)}{2\left(1+\left\|\frac{\Delta e(k)}{\Delta w(k)}\right\|^2\right)} \quad (48)$$

For a very small variation, we can write Equation (48):

$$\Delta w(k) = -\frac{\left(w(k)+e(k)\frac{\partial e(k)}{\partial w(k)}\right)}{2\left(1+\left\|\frac{\partial e(k)}{\partial w(k)}\right\|^2\right)} \quad (49)$$

therefore:

$$\begin{aligned} w(k+1) &= w(k) - \frac{\left(w(k)+e(k)\frac{\partial e(k)}{\partial w(k)}\right)}{2\left(1+\left\|\frac{\partial e(k)}{\partial w(k)}\right\|^2\right)} \\ &= \left(1-\frac{1}{2\left(1+\left\|\frac{\partial e(k)}{\partial w(k)}\right\|^2\right)}\right) w(k) - \frac{1}{2\left(1+\left\|\frac{\partial e(k)}{\partial w(k)}\right\|^2\right)} e(k)\frac{\partial e(k)}{\partial w(k)} \\ &= \left(1-\frac{1}{2\left(1+\left\|\frac{\partial \hat{y}(k)}{\partial w(k)}\right\|^2\right)}\right) w(k) + \frac{1}{2\left(1+\left\|\frac{\partial \hat{y}(k)}{\partial w(k)}\right\|^2\right)} e(k)\frac{\partial \hat{y}(k)}{\partial w(k)} \end{aligned} \quad (50)$$

***Theorem* 3.** *The updating of the neural network parameters can be made by the following equation*:

$$\begin{aligned} w(k+1) = &\left(1-\frac{\alpha}{2\left(1+\left\|\frac{\partial \hat{y}(k)}{\partial w(k)}\right\|^2\right)}\right) w(k) \\ &+ \frac{\alpha}{2\left(1+\left\|\frac{\partial \hat{y}(k)}{\partial w(k)}\right\|^2\right)} e(k)\frac{\partial \hat{y}(k)}{\partial w(k)} \\ &+ (1-\alpha)\Delta w(k) \end{aligned} \quad (51)$$

with: $0<\alpha<1$

Proof:

From Equations (14) and (41), we can write:

$$\begin{aligned} w(k+1) &= w(k)+\Delta w(k+1) \\ &\cong w(k)+\Delta w(k) \\ &\cong w(k)+\alpha\Delta w(k+1)+(1-\alpha)\Delta w(k) \\ &= \left(1-\frac{\alpha}{2\left(1+\left\|\frac{\partial \hat{y}(k)}{\partial w(k)}\right\|^2\right)}\right) w(k) \\ &\quad + \frac{\alpha}{2\left(1+\left\|\frac{\partial \hat{y}(k)}{\partial w(k)}\right\|^2\right)} e(k)\frac{\partial \hat{y}(k)}{\partial w(k)} \\ &\quad + (1-\alpha)\Delta w(k) \end{aligned} \quad (52)$$

The choice of initial synaptic weights and biases can affect the speed of convergence of the learning algorithm of the neural network [36-47]. According to [48], the weights can be initialized by a random number generator with a uniform distribution between $-\theta$ and θ or a normal distribution $N\left(0,\theta^2\right)$.

For weights with uniform distribution:

$$\theta \le \bar{s}\sqrt{\frac{3}{(n_r+1)\left(1+\sum_{m=1}^{n_r}\left(\psi_m(0)\right)^2\right)}} \quad (53)$$

For weight with a normal distribution:

$$\theta \le \bar{s}\sqrt{\frac{1}{(n_r+1)\left(1+\sum_{m=1}^{n_r}\left(\psi_m(0)\right)^2\right)}} \quad (54)$$

where: $\bar{s} \approx 2.29$

2.3. Organizational of the Learning Algorithm of Neural Model

The proposed Lyapunov-Based used to training dynamic model of the system is presented by the flowchart in **Figure 2**, reads as follows:

Step 1:

We fix the desired square error δ_0, the parameters (n_a, n_b, n_c, n_k), the number of samples N, the maximum number of iterations itr, the number of neurons in the first hidden layer n_h.

The weights $w(0,0)$ are initialized by a random number generator with a normal distribution between $-\theta$ and θ.

where:

$$\theta \le \bar{s}\sqrt{\frac{1}{(n_r+1)^2(\psi_e)^2}} \le \bar{s}\sqrt{\frac{1}{(n_r+1)\left(1+\sum_{m=1}^{n_r}(\psi_m(0))^2\right)}} \quad (55)$$

$$\psi_e = \max_{1\le k\le N}\left(\max_{1\le m\le n_r}(\psi_m(k))\right) \quad (56)$$

Initialize:

- the output of the neural network

$$\hat{y}(0,0) = 0 \quad (57)$$

- the vector of outputs of the hidden layer:

$$x^h(0,0) = [0\cdots 0]^T \quad (58)$$

- the vector potentials of neurons in the hidden layer:

$$s(0,0) = [0\cdots 0]^T \quad (59)$$

- the input vector of the neural network:

$$\psi(0,0) = [y(0),\cdots,y(0),u(0),\cdots,u(0),0\cdots 0]^T \quad (60)$$

Step 2:

initialize:

$$w(0,k) = w(k-1) \quad (61)$$

$$\hat{y}(0,k) = \hat{y}(k-1) \quad (62)$$

$$x^h(0,k) = x^h(k-1) \quad (63)$$

$$s(0,k) = s(k-1) \quad (64)$$

Step 3:

Consider an input vector network

$$\psi(k) = [y(k-1),\cdots,y(k-n_a),u(k-n_k-1), \cdots,u(k-n_k-n_b),\hat{y}(k-1),\cdots,\hat{y}(k-n_c)]^T$$

and the desired value for output $y(k)$.

Step 4:

Calculate the output of the neural network $\hat{y}(i,k)$.

Step 5:

Calculate the difference between the system output and the model $e(i,k)$.

Step 6:

Calculate the square error $J_r(i,k)$.

Step 7:

Adjust the vector of network parameters $w(i,k)$ using one of the three following relations:

$$w(i,k) = \left(1 - \frac{1}{2\left(1+\left\|\frac{\partial\hat{y}(i-1,k)}{\partial w(i-1,k)}\right\|^2\right)}\right)w(i-1,k) + \frac{1}{2\left(1+\left\|\frac{\partial\hat{y}(i-1,k)}{\partial w(i-1,k)}\right\|^2\right)}e(i-1,k)\frac{\partial\hat{y}(i-1,k)}{\partial w(i-1,k)} \quad (65)$$

$$w(i,k) = w(i-1,k) + \frac{\lambda}{\gamma}\left(\frac{1}{1+\frac{\beta}{\gamma}\left\|\frac{\partial\hat{y}(i-1,k)}{\partial w(i-1,k)}\right\|^2}\right)e(i-1,k)\frac{\partial\hat{y}(i-1,k)}{\partial w(i-1,k)} \quad (66)$$

with $0 < \lambda \le \gamma \le \beta$

$$w(i,k) = \left(1 - \frac{\alpha}{2\left(1+\left\|\frac{\partial\hat{y}(i-1,k)}{\partial w(i-1,k)}\right\|^2\right)}\right)w(i-1,k) + \frac{\alpha}{2\left(1+\left\|\frac{\partial\hat{y}(i-1,k)}{\partial w(i-1,k)}\right\|^2\right)}e(i-1,k)\frac{\partial\hat{y}(i-1,k)}{\partial w(i-1,k)} + (1-\alpha)\Delta w(i-1,k) \quad (67)$$

with $0 < \alpha < 1$

Step 8:

If the number of iterations $i = itr$ or $J_r(i,k) \le \delta_0$, proceed to Step 9.

Otherwise, increment *i* and return to Step 4.

Step 9:

Save:

- the weights of the network at time k:

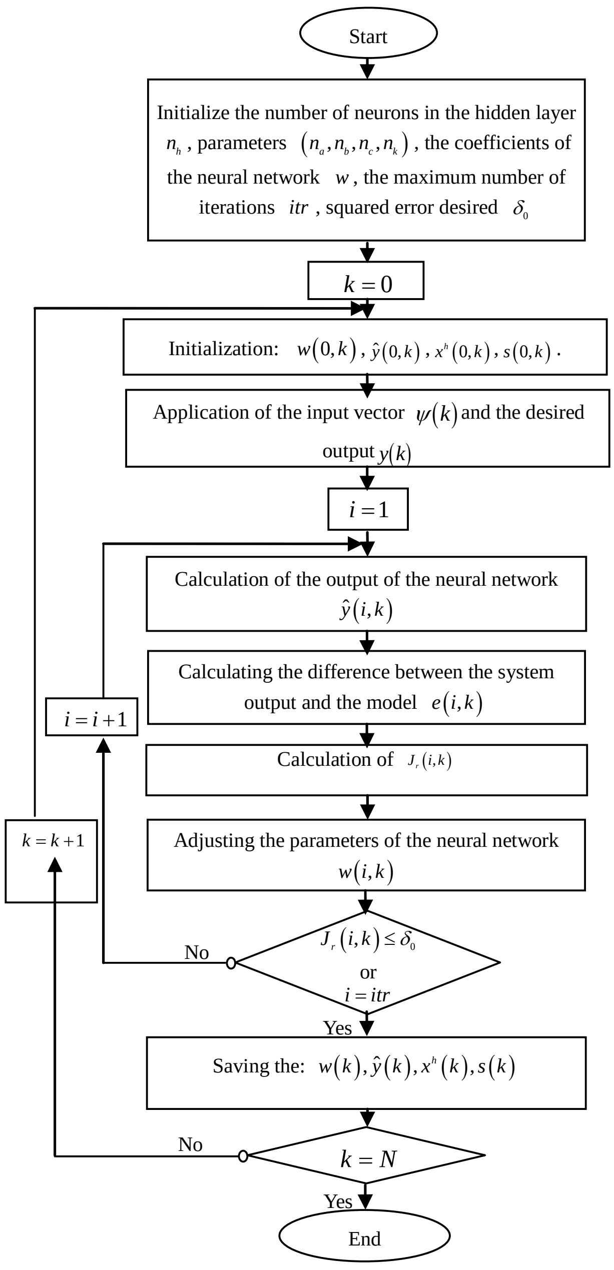

Start
Initialize the number of neurons in the hidden layer n_h, parameters (n_a, n_b, n_c, n_k), the coefficients of the neural network w, the maximum number of iterations itr, squared error desired δ_0
$k = 0$
Initialization: $w(0,k)$, $\hat{y}(0,k)$, $x^h(0,k)$, $s(0,k)$.
Application of the input vector $\psi(k)$ and the desired output $y(k)$
$i = 1$
Calculation of the output of the neural network $\hat{y}(i,k)$
Calculating the difference between the system output and the model $e(i,k)$
Calculation of $J_r(i,k)$
Adjusting the parameters of the neural network $w(i,k)$
$J_r(i,k) \le \delta_0$ or $i = itr$
No
Yes
$i = i + 1$
Saving the: $w(k), \hat{y}(k), x^h(k), s(k)$
$k = N$
No
Yes
$k = k + 1$
End

Figure 2. Flowchart of the learning algorithm of the neural network.

$$w(k) = w(i,k) \tag{68}$$

- the output of the neural network:

$$\hat{y}(k) = \hat{y}(i,k) \tag{69}$$

- the vector of outputs of the hidden layer:

$$x^h(k) = x^h(i,k) \tag{70}$$

- the vector potentials of neurons in the hidden layer:

$$s(k) = s(i,k) \tag{71}$$

Step 10:

If $k = n$, proceed to Step11.
Otherwise, increment k and return to Step 2.
Step 11:
Stop learning.
The flowchart of this algorithm is given in **Figure 2**.

2.4. Validation Tests of the Neuronal Model

The neuronal model obtained from the estimation of its parameters is valid strictly used for the experiment. So check it is compatible with other forms of input in order to properly represent the system operation to identify. Most static tests of model validation are based on the criterion of Nash, on the auto-correlation of residuals, based on cross-correlation between residues and other inputs to the system. According to [49], the Nash criterion is given by the following equation:

$$Q = 100\% \left(1 - \frac{\sum_{k=1}^{N} \left(y(k) - \hat{y}(k) \right)^2}{\sum_{k=1}^{N} \left(y(k) - \left(\frac{1}{N} \sum_{k=1}^{N} y(k) \right) \right)^2} \right) \tag{72}$$

N is the number of samples.

In [50-52], the correlation functions are:
- autocorrelation function of residuals:

$$R_{ee}(\tau) = \frac{\sum_{k=1}^{N-\tau} \left(e(k) - \left(\frac{1}{N} \sum_{k=1}^{N} e(k) \right) \right) \left(e(k-\tau) - \left(\frac{1}{N} \sum_{k=1}^{N} e(k) \right) \right)}{\sum_{k=1}^{N} \left(e(k) - \left(\frac{1}{N} \sum_{k=1}^{N} e(k) \right) \right)^2} \tag{73}$$

- crosscorrelation function between the residuals and the previous entries:

$$R_{ue}(\tau) = \frac{\sum_{k=1}^{N-\tau} \left(u(k) - \left(\frac{1}{N} \sum_{k=1}^{N} u(k) \right) \right) \left(e(k-\tau) - \left(\frac{1}{N} \sum_{k=1}^{N} e(k) \right) \right)}{\sqrt{\sum_{k=1}^{N} \left(u(k) - \left(\frac{1}{N} \sum_{k=1}^{N} u(k) \right) \right)^2} \sqrt{\sum_{k=1}^{N} \left(e(k) - \left(\frac{1}{N} \sum_{k=1}^{N} e(k) \right) \right)^2}} \tag{74}$$

Ideally, if the model is validated, the results of correlation tests and the Nash criterion following results:

$$\hat{R}_{ee}(\tau) = \begin{cases} 1, \tau = 0 \\ 0, \tau \neq 0 \end{cases}, \quad R_{ue}(\tau) = 0 \ \forall \tau \ \text{ and } \ Q = 100\% .$$

Typically, we verify that $Q \cong 100\%$ and the functions $\hat{R}$ are null for the interval $\tau \in [-20, 20]$ with a confidence interval 95%, that is to say that:

$$-\frac{1.96}{\sqrt{N}} \le \hat{R} \le \frac{1.96}{\sqrt{N}}.$$

3. Adaptive Control of Complex Dynamic Systems

In this section, we propose a structure of neural adaptive control of a complex dynamic system and three learning algorithms of a neuronal controller.

3.1. Structure of the Proposed Adaptive Control

In this work, the architecture of the proposed adaptive control is given in **Figure 3**.

The considered neural network is first trained off-line to learn the inverse dynamics of the considered system from the input-output data. The model following adaptive control approach is performed after the training process is achieved. The proposed Lyapunov-Base training algorithm is used to adjust the considered neural network weights so that the neural model output follows the desired one.

3.2. Learning Algorithms of Neural Controller

Three learning algorithms of the neural controller are proposed.

***Theorem* 4.** *Learning the neuronal controller may be effected by the following equation*:

$$\begin{aligned} wc(k+1) &= wc(k) \\ &+\frac{\lambda_1}{\gamma_1}\left(\frac{1}{1+\frac{\beta_1}{\gamma_1}\left\|\frac{\partial\hat{y}(k)}{\partial wc(k)}\right\|^2+\frac{\alpha_1}{\gamma_1}\left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2}\right) \\ &\cdot e_c(k)\frac{\partial\hat{y}(k)}{\partial wc(k)} \end{aligned} \quad (75)$$

with:

$$\lambda_1,\beta_1,\gamma_1,\alpha_1>0 \quad (76)$$

$$\begin{cases} \frac{\beta_1}{\gamma_1}\ge 1 \\ \frac{\alpha_1}{\gamma_1}\ge 1 \\ \frac{\lambda_1}{\gamma_1}\le 1 \end{cases} \quad (77)$$

r is the reference signal.

$$\begin{aligned} e_c(k) &= r(k)-y(k) \\ &\approx r(k)-\hat{y}(k) \end{aligned} \quad (78)$$

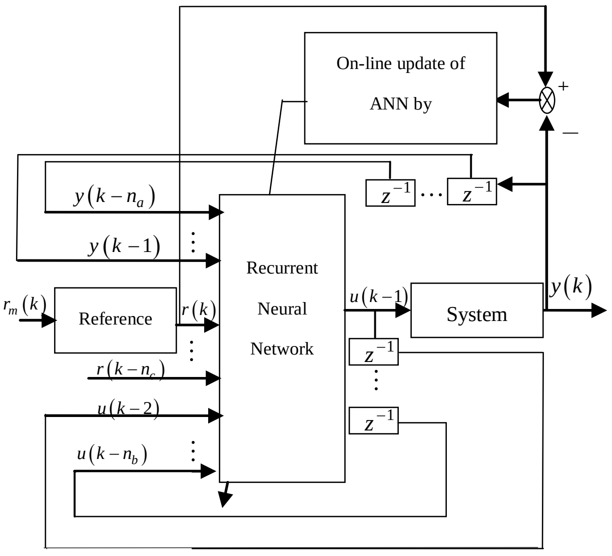


Figure 3. Structure of the proposed adaptive control.

$$\begin{aligned} wc = \Big[& wc^1_{11},\cdots,wc^1_{n_h n_r},wc^2_{11},\cdots,wc^2_{n_h 1}, \\ & wc^3_{11},\cdots,wc^3_{1n_h},wc^4_{11},wc^5_{11}, \\ & \cdots,wc^5_{1n_r},wc^6_{11},\cdots,wc^6_{n_h n_h},wc^7_{11}\Big]^T \end{aligned}$$

are the weight of the neuronal controller.

Proof:

The control system consists of using an optimization digital non-linear algorithm to minimize the following criterion:

$$\begin{aligned} J_c(k) &= \frac{\lambda_1}{2}\left(e_c(k)\right)^2+\frac{\beta_1}{2}\left(\Delta e_c(k)\right)^2+\frac{\gamma_1}{2}\left(\Delta wc(k)\right)^2 \\ &+\frac{\alpha_1}{2}\left(\Delta u(k-1)\right)^2 \end{aligned} \quad (79)$$

with:

$$\begin{aligned} \varphi(k) &= \left[\hat{y}(k-1),\cdots,\hat{y}(k-n_a),u(k-2),\cdots,u(k-n_b),\right. \\ &\left. r(k),\cdots,r(k-n_c)\right]^T=\left[\varphi_1(k),\cdots,\varphi_{n_r}(k)\right]^T \end{aligned} \quad (80)$$

$$\begin{aligned} u(k-1) &= wc^7_{11}u(k-2)+\sum_{j=1}^{n_h}wc^3_{1j}xc^h_j(k) \\ &+\sum_{i=1}^{n_r}wc^5_{1i}\varphi_i(k)+wc^4_{11} \end{aligned} \quad (81)$$

$$xc^h(k)=\left[xc^h_1(k),\cdots,xc^h_{n_h}(k)\right]^T \quad (82)$$

$$xc^h_j(k)=f_1\left(sc_j(k)\right) \quad (83)$$

$$s_c(k)=\left[sc_1(k),\cdots,sc_{n_h}(k)\right]^T \quad (84)$$

$$sc_j(k) = \sum_{m=1}^{n_r} wc_{jm}^1 \varphi_m(k) + \sum_{m=1}^{n_h} wc_{jm}^6 xc_m^h(k-1) + wc_{j1}^2 \tag{85}$$

$$\Delta u(k-1) = u(k-1) - u(k-2) \tag{86}$$

$$\Delta e_c(k) = e_c(k) - e_c(k-1) \tag{87}$$

The minimum of criterion $J_c(k)$ is reached when:

$$\frac{\partial J_c(k)}{\partial wc(k)} = 0 \tag{88}$$

The solution of Equation (88) calculates the weight of the neuronal controller as follows:

$$wc(k+1) = wc(k) - \frac{\lambda_1}{\gamma_1} e_c(k) \frac{\partial e_c(k)}{\partial wc(k)} - \frac{\beta_1}{\gamma_1} \Delta e_c(k) \frac{\partial e_c(k)}{\partial wc(k)} - \frac{\alpha_1}{\gamma_1} \Delta u(k-1) \frac{\partial u(k-1)}{\partial wc(k)} \tag{89}$$

The term $\Delta e_c(k)$ defined by:

$$\begin{aligned}
\Delta e_c(k) &= \left(\frac{\partial e_c(k)}{\partial wc(k)}\right)^T \Delta wc(k) \\
&= \left(\frac{\partial e_c(k)}{\partial wc(k)}\right)^T \left(-\frac{\lambda_1}{\gamma_1} e_c(k) \frac{\partial e_c(k)}{\partial wc(k)} - \frac{\beta_1}{\gamma_1} \Delta e_c(k) \frac{\partial e_c(k)}{\partial wc(k)} - \frac{\alpha_1}{\gamma_1} \Delta u(k-1) \frac{\partial u(k-1)}{\partial wc(k)}\right) \\
&= -\frac{\lambda_1}{\gamma_1} e_c(k) \left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2 - \frac{\beta_1}{\gamma_1} \Delta e_c(k) \left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2 - \frac{\alpha_1}{\gamma_1} \Delta u(k-1) \left(\frac{\partial e_c(k)}{\partial wc(k)}\right)^T \left(\frac{\partial u(k-1)}{\partial wc(k)}\right) \\
&= -\frac{\lambda_1}{\gamma_1} e_c(k) \left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2 - \frac{\beta_1}{\gamma_1} \Delta e_c(k) \left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2 - \frac{\alpha_1}{\gamma_1} \Delta u(k-1) \left(\frac{\partial e_c(k)}{\partial u(k-1)}\right)^T \left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2 \\
&= -\frac{\lambda_1}{\gamma_1} e_c(k) \left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2 - \frac{\beta_1}{\gamma_1} \Delta e_c(k) \left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2 - \frac{\alpha_1}{\gamma_1} \Delta e_c(k) \left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2
\end{aligned} \tag{90}$$

therefore:

$$\Delta e_c(k) = \frac{-\frac{\lambda_1}{\gamma_1} e_c(k) \left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2}{1 + \frac{\beta_1}{\gamma_1}\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2 + \frac{\alpha_1}{\gamma_1}\left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2} \tag{91}$$

Using the above equations, the relationship giving the vector $wc(k+1)$ minimizing the criterion $J_c(k)$ can be written as follows:

$$\begin{aligned}
& wc(k+1) \\
&= wc(k) - \frac{\lambda_1}{\gamma_1} e_c(k) \frac{\partial e_c(k)}{\partial wc(k)} \\
&\quad - \frac{\beta_1}{\gamma_1} \left(\frac{-\frac{\lambda_1}{\gamma_1} e_c(k) \left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2}{1 + \frac{\beta_1}{\gamma_1}\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2 + \frac{\alpha_1}{\gamma_1}\left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2} \right) \frac{\partial e_c(k)}{\partial wc(k)} \\
&\quad - \frac{\alpha_1}{\gamma_1} \left(\frac{-\frac{\lambda_1}{\gamma_1} e_c(k) \left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2}{1 + \frac{\beta_1}{\gamma_1}\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2 + \frac{\alpha_1}{\gamma_1}\left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2} \right) \frac{\partial e_c(k)}{\partial wc(k)} \\
&= wc(k) - \frac{\lambda_1}{\gamma_1} e_c(k) \frac{\partial e_c(k)}{\partial wc(k)} \\
&\quad \cdot \left(1 - \frac{\frac{\beta_1}{\gamma_1}\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2}{1 + \frac{\beta_1}{\gamma_1}\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2 + \frac{\alpha_1}{\gamma_1}\left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2} - \frac{\frac{\alpha_1}{\gamma_1}\left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2}{1 + \frac{\beta_1}{\gamma_1}\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2 + \frac{\alpha_1}{\gamma_1}\left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2} \right) \\
&= wc(k) - \frac{\lambda_1}{\gamma_1} e_c(k) \frac{\partial e_c(k)}{\partial wc(k)} \\
&\quad \cdot \left(\frac{1}{1 + \frac{\beta_1}{\gamma_1}\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2 + \frac{\alpha_1}{\gamma_1}\left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2} \right)
\end{aligned} \tag{92}$$

It is necessary to check the stability of this procedure to adjust the weight of the correction before applying. In this case, the candidate Lyapunov function may be as follows:

$$V(k)=\frac{1}{2}\left(e_c(k)\right)^2+\frac{1}{2}\left(\Delta e_c(k)\right)^2 \tag{93}$$

According to Equation (32), the term $\Delta V(k)$ is written as follows:

$$\Delta V(k)=\Delta e_c(k)e_c(k)+\left(\Delta e_c(k)\right)^2=\frac{-\frac{\lambda_1}{\gamma_1}\left(e_c(k)\right)^2\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2}{1+\frac{\beta_1}{\gamma_1}\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2+\frac{\alpha_1}{\gamma_1}\left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2}\left(1-\frac{\frac{\lambda_1}{\gamma_1}\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2}{1+\frac{\beta_1}{\gamma_1}\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2+\frac{\alpha_1}{\gamma_1}\left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2}\right) \tag{94}$$

For the procedure to adjust the parameters of the controller is stable, it must:

$$1-\frac{\frac{\lambda_1}{\gamma_1}\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2}{1+\frac{\beta_1}{\gamma_1}\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2+\frac{\alpha_1}{\gamma_1}\left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2}\geq 0 \tag{95}$$

then:

$$1\geq\frac{\lambda_1-\beta_1}{\gamma_1}\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2-\frac{\alpha_1}{\gamma_1}\left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2 \tag{96}$$

The second condition for stability is obtained by the following Lyapunov function:

$$V(k)=\frac{1}{2}\left(e_c(k)\right)^2+\frac{\gamma_1}{2\lambda_1}\left(\Delta e_c(k)\right)^2 \tag{97}$$

From the Equation (35), we can write:

$$\begin{aligned}\Delta V(k)&=e_c(k)\left(\Delta e_c(k)\right)+\frac{\gamma_1}{\lambda_1}\left(\Delta e_c(k)\right)^2\\&=\frac{-\frac{\lambda_1}{\gamma_1}\left(e_c(k)\right)^2\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2}{1+\frac{\beta_1}{\gamma_1}\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2+\frac{\alpha_1}{\gamma_1}\left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2}\left(1-\frac{\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2}{1+\frac{\beta_1}{\gamma_1}\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2+\frac{\alpha_1}{\gamma_1}\left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2}\right)\end{aligned} \tag{98}$$

The learning algorithm parameters of the controller are stable if:

$$1\geq\left(1-\frac{\beta_1}{\gamma_1}\right)\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2-\frac{\alpha_1}{\gamma_1}\left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2 \tag{99}$$

Using the following Lyapunov function:

$$V(k)=\frac{1}{2}\left(e_c(k)\right)^2+\frac{\gamma_1}{2\lambda_1}\left(\Delta e_c(k)\right)^2+\frac{\alpha_1}{2\gamma_1}\left(\Delta u(k-1)\right)^2 \tag{100}$$

The adjustment procedure is stable if the parameters:

$$\begin{aligned}&\Delta V(k)\\&=\left(e_c(k)\right)\left(\frac{\partial e_c(k)}{\partial wc}\right)^T\left(\Delta wc(k)\right)+\frac{\gamma_1}{\lambda_1}\left(\Delta e_c(k)\right)\left(\frac{\partial e_c(k)}{\partial wc}\right)^T\left(\Delta wc(k)\right)+\frac{\alpha_1}{\lambda_1}\left(\Delta u(k-1)\right)\left(\frac{\partial u(k-1)}{\partial e_c(k)}\right)^T\left(\Delta e_c(k)\right)\\&=\left(e_c(k)\right)\left(\frac{\partial e_c(k)}{\partial wc}\right)^T\left(\Delta wc(k)\right)+\frac{\gamma_1}{\lambda_1}\left(\Delta e_c(k)\right)\left(\frac{\partial e_c(k)}{\partial wc}\right)^T\left(\Delta wc(k)\right)+\frac{\alpha_1}{\lambda_1}\left\|\frac{\partial u(k-1)}{\partial e_c(k)}\right\|^2\left(\Delta e_c(k)\right)^2\\&=\frac{-\frac{\lambda_1}{\gamma_1}\left(e_c(k)\right)^2\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2}{1+\frac{\beta_1}{\gamma_1}\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2+\frac{\alpha_1}{\gamma_1}\left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2}\left(1-\frac{\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2}{1+\frac{\beta_1}{\gamma_1}\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2+\frac{\alpha_1}{\gamma_1}\left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2}-\frac{\frac{\alpha_1}{\gamma_1}\left\|\frac{\partial u(k-1)}{\partial e_c(k)}\right\|^2\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2}{1+\frac{\beta_1}{\gamma_1}\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2+\frac{\alpha_1}{\gamma_1}\left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2}\right)\leq 0\end{aligned} \tag{101}$$

The third stability condition is:

$$1+\frac{\beta_1}{\gamma_1}\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2-\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|\geq 0 \tag{102}$$

Assuming:

$$d_1=\max\left(\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|\right)=\max\left(\left\|\frac{\partial \hat{y}(k)}{\partial wc(k)}\right\|\right) \tag{103}$$

$$L_1=\max\left(\left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|\right) \tag{104}$$

Therefore:

$$\frac{\beta_1}{\gamma_1}\geq 1-\left(\frac{1}{d_1}\right)^2\geq 1-\frac{1}{\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2} \tag{105}$$

The fourth condition for stability is obtained by the following Lyapunov function:

$$V(k)=\frac{1}{2}\left(e_c(k)\right)^2+\frac{\beta_1}{2\lambda_1}\left(\Delta e_c(k)\right)^2+\frac{\gamma_1}{2\lambda_1}\left(\Delta u(k-1)\right)^2 \tag{106}$$

The term $\Delta V(k)$ is as follows:

$$\begin{aligned}
&\Delta V(k)\\
&=\left(e_c(k)\right)\left(\frac{\partial e_c(k)}{\partial wc(k)}\right)^T\left(\Delta wc(k)\right)+\frac{\gamma_1}{\lambda_1}\left(\Delta e_c(k)\right)\left(\frac{\partial e_c(k)}{\partial wc(k)}\right)^T\left(\Delta wc(k)\right)+\frac{\alpha_1}{\lambda_1}\left(\Delta u(k-1)\right)\left(\frac{\partial u(k-1)}{\partial e_c(k)}\right)\left(\Delta e_c(k)\right)\\
&=\left(e_c(k)\right)\left(\frac{\partial e_c(k)}{\partial wc(k)}\right)^T\left(\Delta wc(k)\right)+\frac{\gamma_1}{\lambda_1}\left(\Delta e_c(k)\right)\left(\frac{\partial e_c(k)}{\partial wc(k)}\right)^T\left(\Delta wc(k)\right)+\frac{\alpha_1}{\lambda_1}\left\|\frac{\partial u(k-1)}{\partial e_c(k)}\right\|^2\left(\Delta e_c(k)\right)^2\\
&=\frac{-\frac{\lambda_1}{\gamma_1}\left(e_c(k)\right)^2\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2}{1+\frac{\beta_1}{\gamma_1}\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2+\frac{\alpha_1}{\gamma_1}\left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2}\left(1-\frac{-\frac{\beta_1}{\gamma_1}\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2}{1+\frac{\beta_1}{\gamma_1}\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2+\frac{\alpha_1}{\gamma_1}\left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2}-\frac{\left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2}{1+\frac{\beta_1}{\gamma_1}\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2+\frac{\alpha_1}{\gamma_1}\left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2}\right)
\end{aligned} \tag{107}$$

For the learning algorithm is stable, it must:

$$1+\frac{\alpha_1}{\gamma_1}\left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2-\left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2\geq 0 \tag{108}$$

therefore:

$$\frac{\alpha_1}{\gamma_1}\geq 1\geq 1-\left(\frac{1}{L_1}\right)^2\geq 1-\frac{1}{\left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2} \tag{109}$$

According to Equations (96), (105) and (109), we can write:

$$\frac{1}{\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2}+\frac{\beta_1}{\gamma_1}+\frac{\alpha_1}{\gamma_1}\frac{\left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2}{\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2}\geq 1+\frac{\alpha_1}{\gamma_1}\frac{\left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2}{\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2}\geq 1\geq\frac{\lambda_1}{\gamma_1} \tag{110}$$

The stability conditions can be so:

$$\begin{cases}\dfrac{\beta_1}{\gamma_1}\geq 1\\ \dfrac{\alpha_1}{\gamma_1}\geq 1\\ \dfrac{\lambda_1}{\gamma_1}\leq 1\end{cases} \tag{111}$$

The term $\frac{\partial e_c(k)}{\partial wc(k)}$ is calculated by the following equations:

$$\begin{aligned}\frac{\partial e_c(k)}{\partial wc(k)}&=\frac{\partial e_c(k)}{\partial u(k+n_k-1)}\frac{\partial u(k+n_k-1)}{\partial u(k-1)}\frac{\partial u(k-1)}{\partial wc(k)}\\ &\approx-\frac{\partial \hat{y}(k)}{\partial u(k+n_k-1)}\frac{\partial u(k+n_k-1)}{\partial u(k-1)}\frac{\partial u(k-1)}{\partial wc(k)}\end{aligned} \tag{112}$$

$$\frac{\partial \hat{y}(k)}{\partial u(k+n_k-1)} = w^5_{1n_a+1}(k) + \sum_{j=1}^{n_h} w^3_{1j}(k)\frac{\partial x^h_j(k)}{\partial u(k+n_k-1)} \tag{113}$$

$$\frac{\partial x^h_j(k)}{\partial u(k+n_k-1)} = f_1'\left(s_j(k)\right)w^1_{jn_a+1}(k) \tag{114}$$

***Theorem* 5.** *The procedure for adjusting the parameters of neuronal controller can be described by the following equation*:

$$\begin{aligned} &wc(k+1) \\ &= \left(1 - \frac{1}{2\left(1+\left\|\frac{\partial \hat{y}(k)}{\partial wc(k)}\right\|^2 + \left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2\right)}\right) wc(k) \\ &+ \frac{1}{2\left(1+\left\|\frac{\partial \hat{y}(k)}{\partial wc(k)}\right\|^2 + \left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2\right)} e_c(k)\frac{\partial \hat{y}(k)}{\partial wc(k)} \end{aligned} \tag{115}$$

Proof:

From the following Lyapunov function:

$$\begin{aligned} V(k) &= \frac{1}{2}\left(e_c(k)\right)^2 + \frac{1}{2}\left(\Delta e_c(k)\right)^2 \\ &+ \frac{1}{2}\left(\Delta u(k-1)\right)^2 + \frac{1}{2}\left\|wc(k)\right\|^2 + \frac{1}{2}\left\|\Delta wc(k)\right\|^2 \end{aligned} \tag{116}$$

The procedure for adjusting the parameters of the neuronal controller is stable if:

$$\begin{aligned} \Delta V(k) &= \left(\Delta e_c(k)\right)^2 + \left(\Delta e_c(k)\right)\left(e_c(k)\right) \\ &+ \left(\Delta u(k-1)\right)^2 + \left\|\Delta wc(k)\right\|^2 \\ &+ \left(wc(k)\right)^T\left(\Delta wc(k)\right) = -r_1 \end{aligned} \tag{117}$$

such as: $r_1 \geq 0$

Equation (117) becomes:

$$\begin{aligned} &\left\|\Delta wc(k)\right\|^2\left(1+\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2 + \left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2\right) \\ &+ \left(wc(k) + \frac{\partial e_c(k)}{\partial wc(k)}e_c(k)\right)^T\left(\Delta wc(k)\right) + r_1 = 0 \end{aligned} \tag{118}$$

If the above equation has a unique solution, the term r_1 is as follows:

$$r_1 = \frac{\left\|wc(k) + e_c(k)\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2}{4\left(1+\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2 + \left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2\right)} \tag{119}$$

The equation for adjusting the parameters of the neuronal controller can be written:

$$\Delta wc(k) = -\frac{wc(k) + e_c(k)\left(\frac{\partial e_c(k)}{\partial wc(k)}\right)}{2\left(1+\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2 + \left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2\right)} \tag{120}$$

therefore:

$$\begin{aligned} &wc(k+1) \\ &= \left(1 - \frac{1}{2\left(1+\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2 + \left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2\right)}\right) wc(k) \\ &- \frac{1}{2\left(1+\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2 + \left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2\right)} e_c(k)\frac{\partial e_c(k)}{\partial wc(k)} \\ &= \left(1 - \frac{1}{2\left(1+\left\|\frac{\partial \hat{y}(k)}{\partial wc(k)}\right\|^2 + \left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2\right)}\right) wc(k) \\ &+ \frac{1}{2\left(1+\left\|\frac{\partial \hat{y}(k)}{\partial wc(k)}\right\|^2 + \left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2\right)} e_c(k)\frac{\partial \hat{y}(k)}{\partial wc(k)} \end{aligned} \tag{121}$$

Theorem 6. *The procedure for adjusting controller parameters can be made by the following equation*:

$$\begin{aligned} &wc(k+1) \\ &= \left(1 - \frac{\alpha}{2\left(1+\left\|\frac{\partial \hat{y}(k)}{\partial wc(k)}\right\|^2 + \left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2\right)}\right) wc(k) \\ &+ \frac{\alpha}{2\left(1+\left\|\frac{\partial \hat{y}(k)}{\partial wc(k)}\right\|^2 + \left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2\right)} e_c(k)\frac{\partial \hat{y}(k)}{\partial wc(k)} \\ &+ (1-\alpha)\Delta wc(k) \end{aligned} \tag{122}$$

Proof:

Using the Equation (121), we may write:

$$\begin{aligned}
&wc(k+1)\\
&= wc(k)+\Delta wc(k+1)\\
&\cong wc(k)+\Delta wc(k)\\
&\cong wc(k)+\alpha\Delta wc(k+1)+(1-\alpha)\Delta wc(k)\\
&=\left(1-\frac{\alpha}{2\left(1+\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2+\left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2\right)}\right)wc(k)\\
&-\frac{\alpha}{2\left(1+\left\|\frac{\partial e_c(k)}{\partial wc(k)}\right\|^2+\left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2\right)}e_c(k)\frac{\partial e_c(k)}{\partial wc(k)}\\
&+(1-\alpha)\Delta wc(k)\\
&=\left(1-\frac{\alpha}{2\left(1+\left\|\frac{\partial \hat{y}(k)}{\partial wc(k)}\right\|^2+\left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2\right)}\right)wc(k)\\
&+\frac{\alpha}{2\left(1+\left\|\frac{\partial \hat{y}(k)}{\partial wc(k)}\right\|^2+\left\|\frac{\partial u(k-1)}{\partial wc(k)}\right\|^2\right)}e_c(k)\frac{\partial \hat{y}(k)}{\partial wc(k)} \qquad (123)\\
&+(1-\alpha)\Delta wc(k)
\end{aligned}$$

Flowchart of the learning algorithm of the neural controller

Once the modeling phase is completed, the calculation of parameters of neuronal controller is carried through the following steps:

Step 1:

We fix the desired square error δ_0, the parameters (n_a, n_b, n_c, n_k), the number of samples *N*, the maximum number of iterations itr, the number of neurons in the hidden layer n_h.

The weights wc are initialized by a random number generator with a normal distribution between $-\theta_1$ and θ_1.

where:

$$\theta_1 \le \overline{s}\sqrt{\frac{1}{(n_r+1)^2(\varphi_e)^2}} \qquad (124)$$

with:

$$\varphi_e = \max_{1\le k\le N}\left(\max_{1\le m\le n_r}(\varphi_m(k))\right) \qquad (125)$$

Step 2:

Initialize:

$$wc(0,k)=wc(k-1) \qquad (126)$$

$$u(0,k)=u(k-1) \qquad (127)$$

$$xc^h(0,k)=xc^h(k-1) \qquad (128)$$

$$s_c(0,k)=s_c(k-1) \qquad (129)$$

Step 3:

Consider an input vector of the network

$$\varphi(k)=\left[\hat{y}(k-1),\cdots,\hat{y}(k-n_a),u(k-2),\ldots,u(k-n_b),r(k),\cdots,r(k-n_c)\right]^T$$

and the reference signal $r(k)$.

Step 4:

Calculate the output of the neuronal controller $u(i,k-1)$.

Step 5:

Calculate the output of the neural model $\hat{y}(i,k)$.

Step 6:

Calculating the difference between the reference signal and the output of neural model $e_c(i,k)$.

Step 7:

Calculate the square error $J_c(i,k)$.

Step 8:

Adjust the vector of network parameters $wc(i,k)$ using one of the three following relations:

$$\begin{aligned}
&wc(i,k)\\
&=\left(1-\frac{1}{2\left(1+\left\|\frac{\partial \hat{y}(i-1,k)}{\partial wc(i-1,k)}\right\|^2+\left\|\frac{\partial u(i-1,k-1)}{\partial wc(i-1,k)}\right\|^2\right)}\right)\\
&\cdot wc(i-1,k) \qquad (130)\\
&+\frac{1}{2\left(1+\left\|\frac{\partial \hat{y}(i-1,k)}{\partial wc(i-1,k)}\right\|^2+\left\|\frac{\partial u(i-1,k-1)}{\partial wc(i-1,k)}\right\|^2\right)}\\
&\cdot e_c(i-1,k)\frac{\partial \hat{y}(i-1,k)}{\partial wc(i-1,k)}
\end{aligned}$$

$$\begin{aligned}
&wc(i,k)=wc(i-1,k)\\
&+\frac{\lambda_1}{\gamma_1}\left(\frac{1}{1+\frac{\beta_1}{\gamma_1}\left\|\frac{\partial \hat{y}(i-1,k)}{\partial wc(i-1,k)}\right\|^2+\frac{\alpha_1}{\gamma_1}\left\|\frac{\partial u(i-1,k-1)}{\partial wc(i-1,k)}\right\|^2}\right) \qquad (131)\\
&\cdot e_c(i-1,k)\frac{\partial \hat{y}(i-1,k)}{\partial wc(i-1,k)}
\end{aligned}$$

with $\frac{\beta_1}{\gamma_1} \geq 1$, $\frac{\alpha_1}{\gamma_1} \geq 1$, $\frac{\lambda_1}{\gamma_1} \leq 1$

$$wc(i,k) = \left(1 - \frac{\alpha}{2\left(1 + \left\|\frac{\partial \hat{y}(i-1,k)}{\partial wc(i-1,k)}\right\|^2 + \left\|\frac{\partial u(i-1,k-1)}{\partial wc(i-1,k)}\right\|^2\right)}\right) \cdot wc(i-1,k) + \frac{\alpha}{2\left(1 + \left\|\frac{\partial \hat{y}(i-1,k)}{\partial wc(i-1,k)}\right\|^2 + \left\|\frac{\partial u(i-1,k-1)}{\partial wc(i-1,k)}\right\|^2\right)} \cdot e_c(i-1,k)\frac{\partial \hat{y}(i-1,k)}{\partial wc(i-1,k)} + (1-\alpha)\Delta wc(i-1,k) \quad (132)$$

Step 9:

If the number of iterations $i = itr$ or $J_c(i,k) \leq \delta_0$, proceed to Step 10.

Otherwise, increment i and return to Step 4.

Step 10:

Save:

- the weights of the network at time k:

$$wc(k) = wc(i,k) \quad (133)$$

- the output of the neuronal controller:

$$u(k-1) = u(i,k-1) \quad (134)$$

- the vector of outputs of the hidden layer:

$$xc^h(k) = xc^h(i,k) \quad (135)$$

- the vector potentials of neurons in the hidden layer:

$$s_c(k) = s_c(i,k) \quad (136)$$

Step 11:

If $k = N$, proceed to Step 12.

Otherwise, increment k and return to Step 2.

Step 12:

Stop Learning.

These steps are represented by the following flowchart, **Figure 4**.

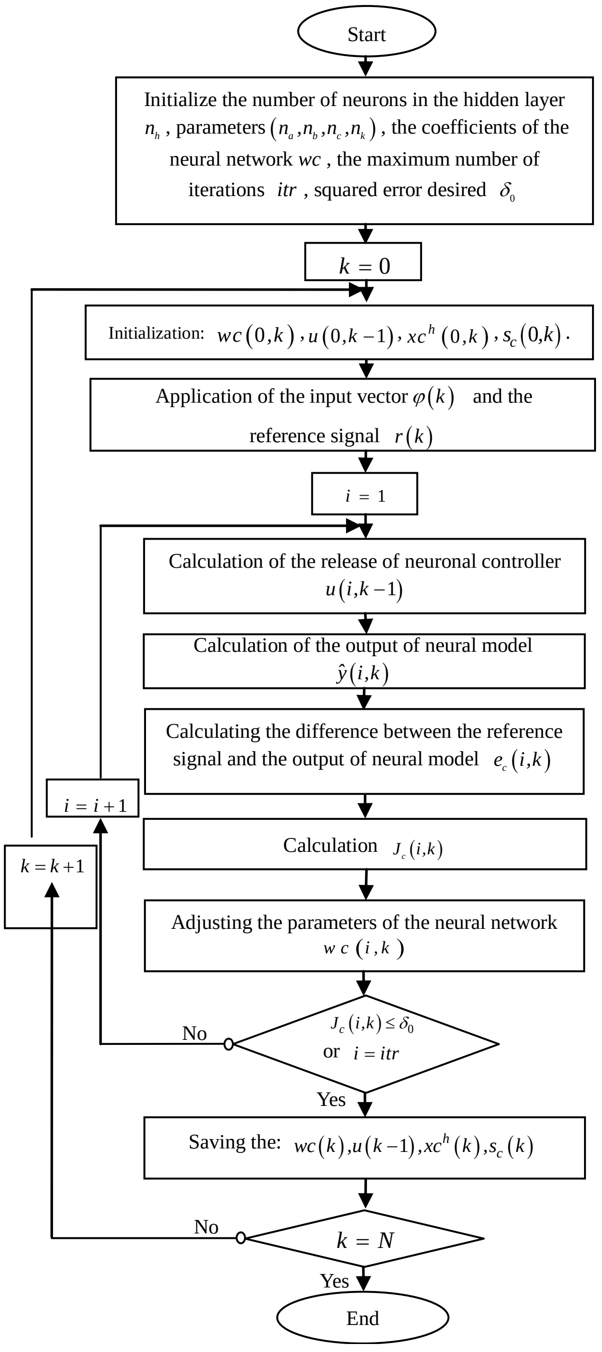

Start
Initialize the number of neurons in the hidden layer n_h, parameters (n_a, n_b, n_c, n_k), the coefficients of the neural network wc, the maximum number of iterations itr, squared error desired δ_0
$k = 0$
Initialization: $wc(0,k)$, $u(0,k-1)$, $xc^h(0,k)$, $s_c(0,k)$.
Application of the input vector $\varphi(k)$ and the reference signal $r(k)$
$i = 1$
Calculation of the release of neuronal controller $u(i,k-1)$
Calculation of the output of neural model $\hat{y}(i,k)$
Calculating the difference between the reference signal and the output of neural model $e_c(i,k)$
$i = i+1$
Calculation $J_c(i,k)$
$k = k+1$
Adjusting the parameters of the neural network $wc(i,k)$
No
$J_c(i,k) \leq \delta_0$ or $i = itr$
Yes
Saving the: $wc(k), u(k-1), xc^h(k), s_c(k)$
No
$k = N$
Yes
End

Figure 4. Flowchart of the proposed Lyapunov-Base learning algorithm of the controller neural network.

4. Numerical Results and Discussion

Let consider the nonlinear system described by the following equation of state:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = \left(10\cos(u) - \eta\sqrt{|x_1|}\right)^2 - \mu x_2 - \tau x_1 + \varepsilon \\ y = x_1 \end{cases} \quad (137)$$

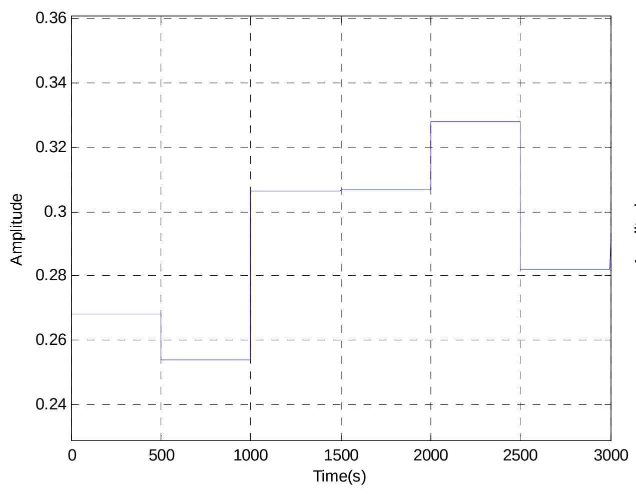
with:

u and y are respectively the input and output system.

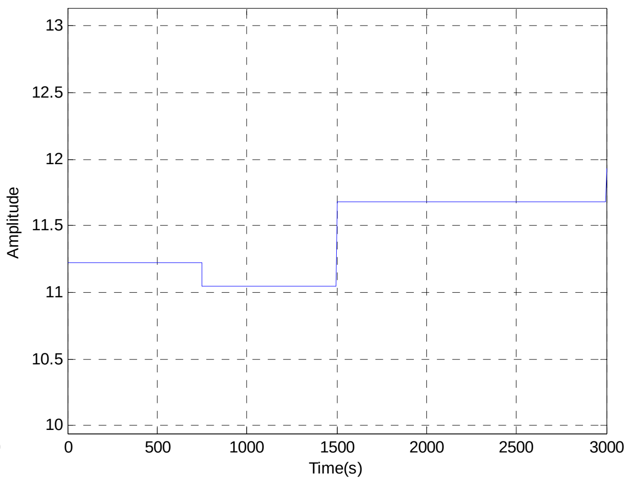
ε is a noise such as $|\varepsilon| < 0.1$.

The **Figure 5** shows the evolution of system parameters (η, τ and μ).

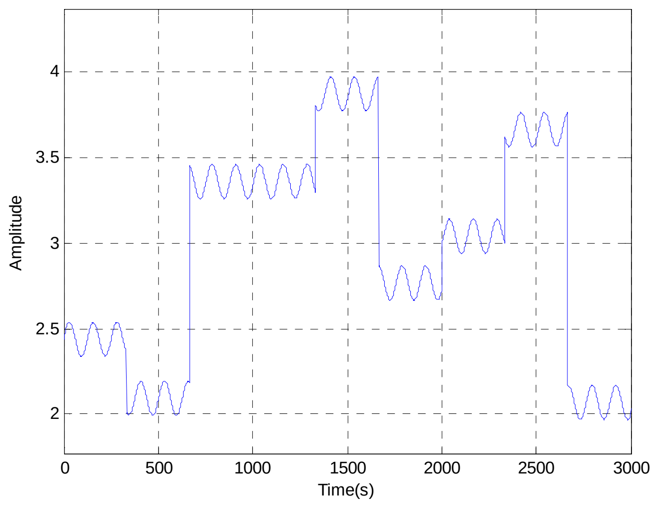
The sequences of input and output those used to calculate the parameters of the neural model are shown in **Figure 6**. These sequences show the system response to



(a)

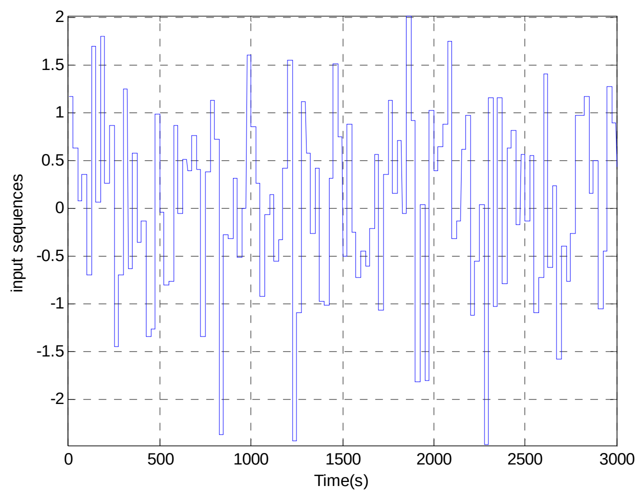


(b)

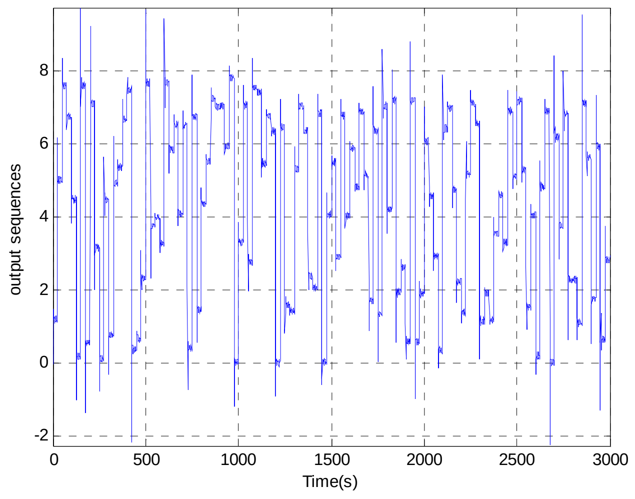


(c)

Figure 5. Evolution of the system parameters: (a) Parameter η; (b) Parameter τ; (c) Parameter μ.

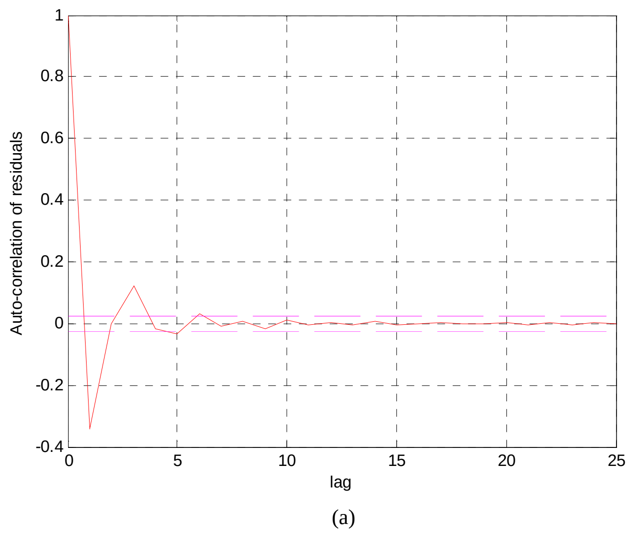


(a)



(b)

Figure 6. Training data-pattern: (a) Input sequences; (b) Output sequences.



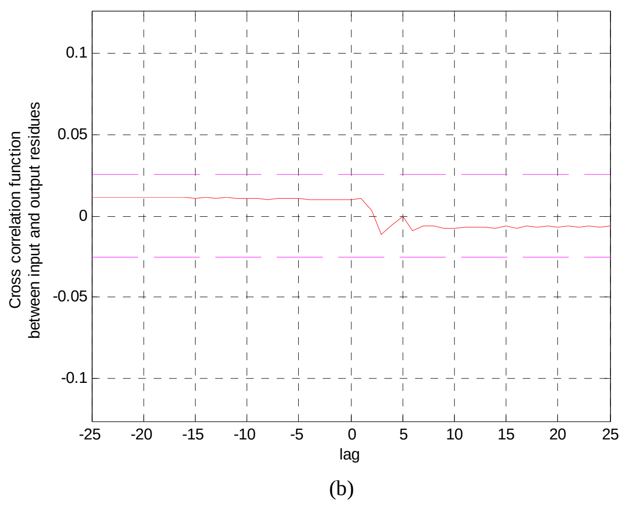
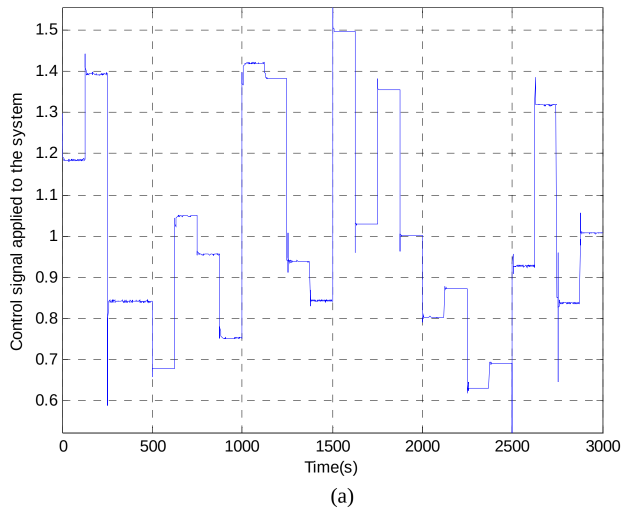
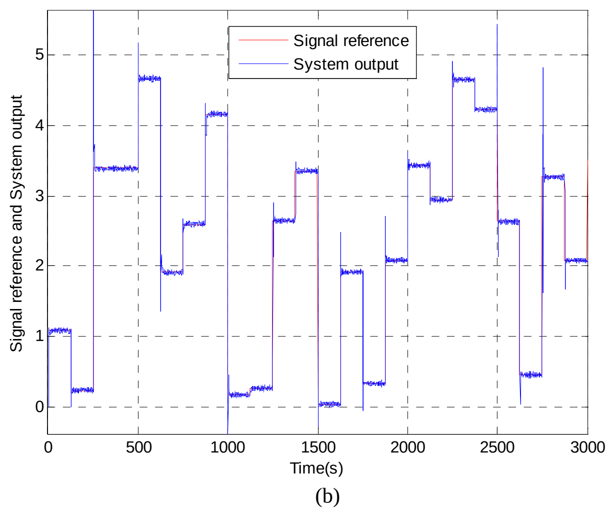
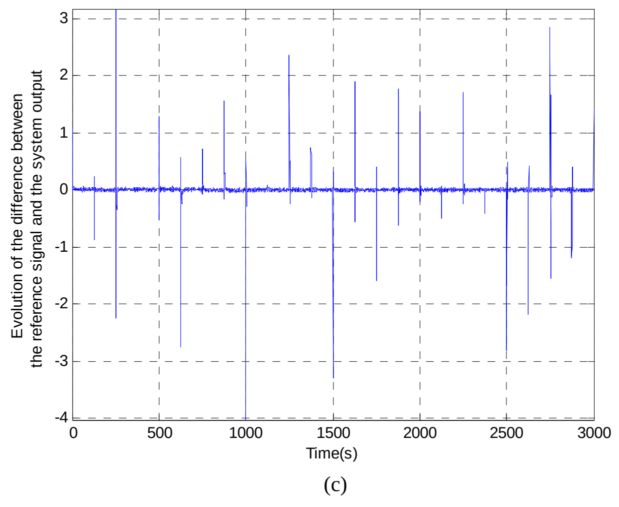


Figure 7. Validation tests of the model: (a) Auto-correlation of residuals; (b) Cross correlation function between input and output residues.







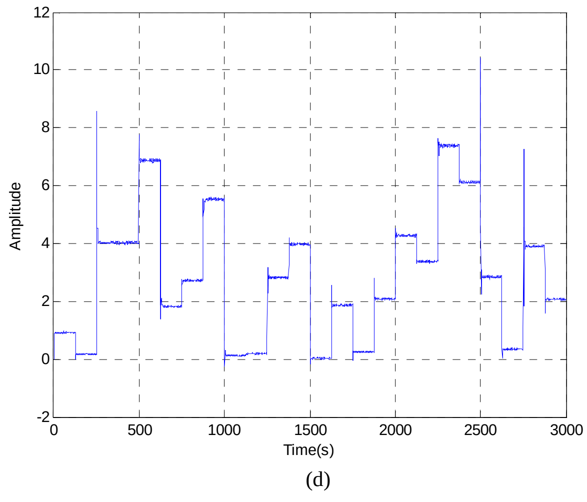
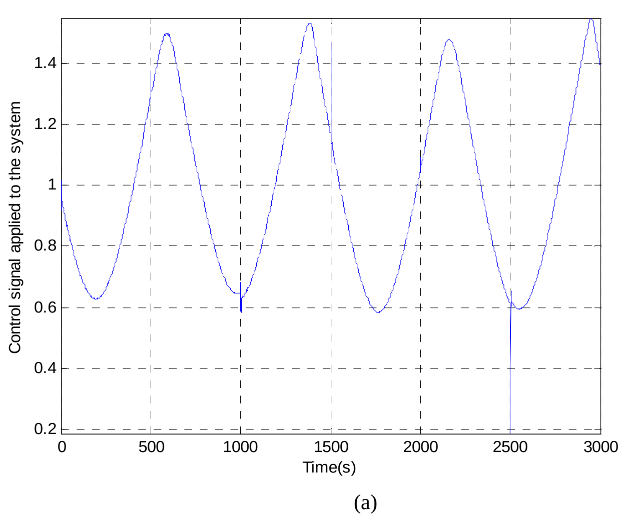
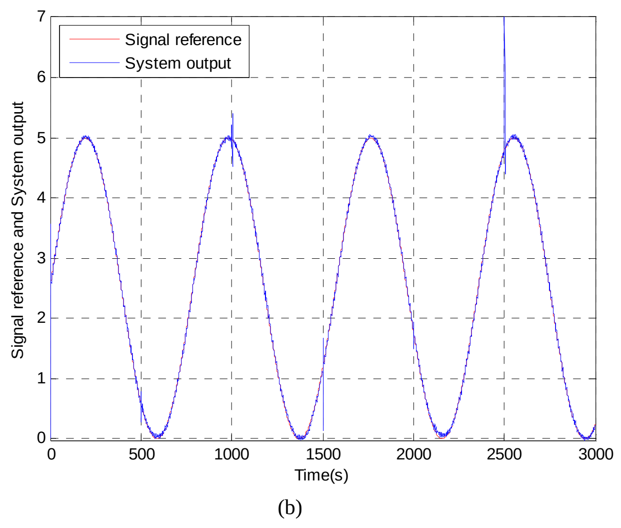
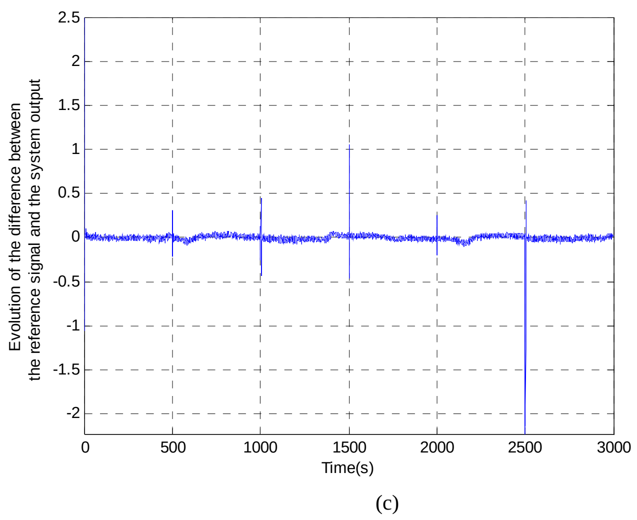


Figure 8. Results of adaptive control system in the case of a reference signal amplitude random uniform distribution: (a) Control signal applied to the system; (b) Response of the system; (c) Evolution of the difference between the reference signal and the system output; (d) Sensitivity of the process.







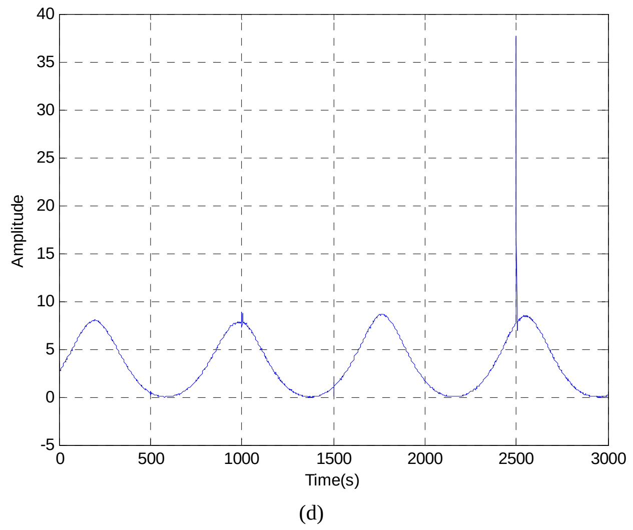
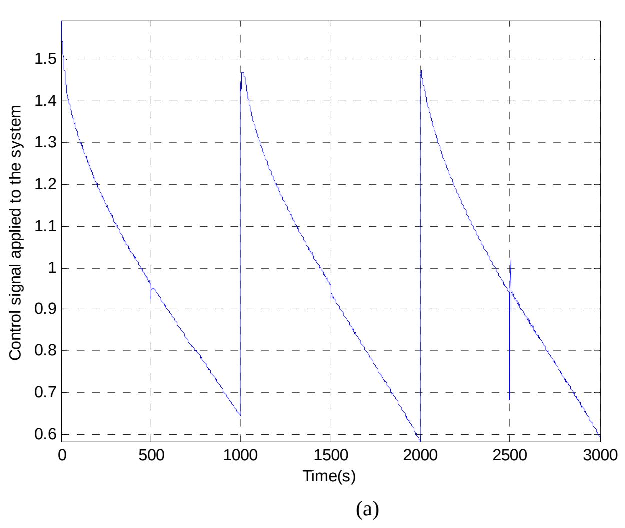
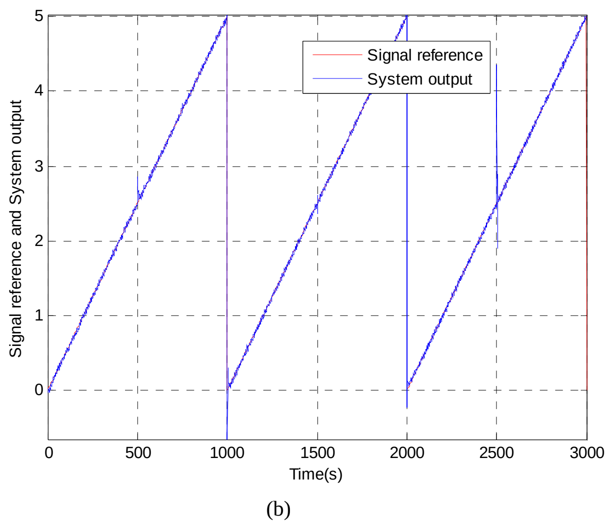
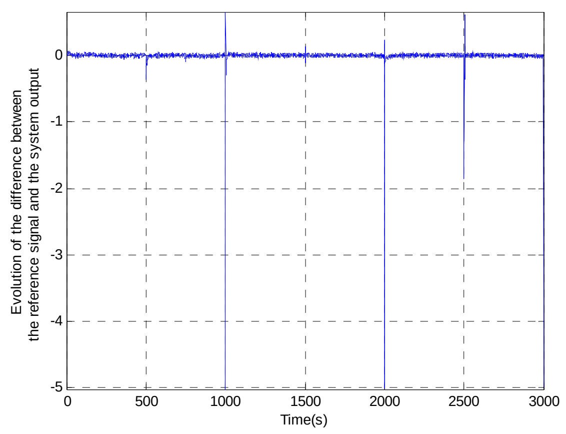


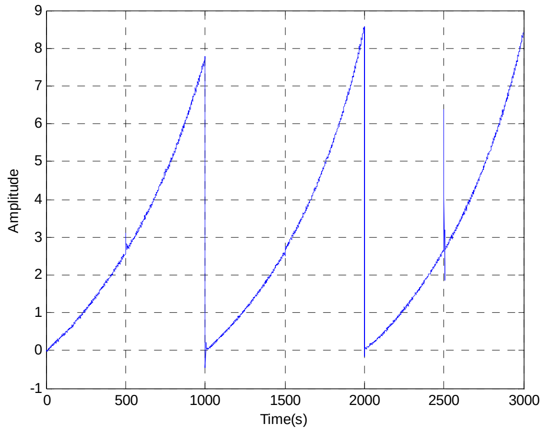
Figure 9. Results of adaptive control system in the case of a sinusoidal reference signal: (a) Control signal applied to the system; (b) Response of the system; (c) Evolution of the difference between the reference signal and the system output; (d) Sensitivity of the process.







(c)



(d)

Figure 10. Results of adaptive control system in the case of a triangular reference signal: (a) Control signal applied to the system; (b) Response of the system; (c) Evolution of the difference between the reference signal and the system output; (d) Sensitivity of the process.

Table 1. Values of the Nash criterion of candidate neural models using Theorem 1 with (λ = 1, β = γ = 2).

Neuronal model parameters	$n_a=1$ $n_b=1$ $n_c=1$ $n_k=0$ $n_h=1$	$n_a=1$ $n_b=1$ $n_c=1$ $n_k=0$ $n_h=2$	$n_a=1$ $n_b=1$ $n_c=1$ $n_k=0$ $n_h=3$	$n_a=1$ $n_b=1$ $n_c=1$ $n_k=0$ $n_h=4$	$n_a=1$ $n_b=1$ $n_c=1$ $n_k=0$ $n_h=5$	$n_a=1$ $n_b=1$ $n_c=1$ $n_k=0$ $n_h=6$	$n_a=2$ $n_b=1$ $n_c=1$ $n_k=0$ $n_h=7$	$n_a=2$ $n_b=2$ $n_c=1$ $n_k=0$ $n_h=8$	$n_a=2$ $n_b=2$ $n_c=2$ $n_k=0$ $n_h=8$	$n_a=2$ $n_b=2$ $n_c=2$ $n_k=1$ $n_h=8$	$n_a=2$ $n_b=2$ $n_c=2$ $n_k=2$ $n_h=8$
Nash criterion	71%	74%	77%	79%	83%	88%	92%	94%	91%	89%	88%

Table 2. Values of the Nash criterion of candidate neural models using Theorem 2.

Neuronal model parameters	$n_a=1$ $n_b=1$ $n_c=1$ $n_k=0$ $n_h=1$	$n_a=1$ $n_b=1$ $n_c=1$ $n_k=0$ $n_h=2$	$n_a=1$ $n_b=1$ $n_c=1$ $n_k=0$ $n_h=3$	$n_a=1$ $n_b=1$ $n_c=1$ $n_k=0$ $n_h=4$	$n_a=1$ $n_b=1$ $n_c=1$ $n_k=0$ $n_h=5$	$n_a=1$ $n_b=1$ $n_c=1$ $n_k=0$ $n_h=6$	$n_a=2$ $n_b=1$ $n_c=1$ $n_k=0$ $n_h=7$	$n_a=2$ $n_b=2$ $n_c=1$ $n_k=0$ $n_h=8$	$n_a=2$ $n_b=2$ $n_c=2$ $n_k=0$ $n_h=8$	$n_a=2$ $n_b=2$ $n_c=2$ $n_k=1$ $n_h=8$	$n_a=2$ $n_b=2$ $n_c=2$ $n_k=2$ $n_h=8$
Nash criterion	72%	75%	76%	79%	84%	89%	91%	95.5%	92%	90%	88%

Table 3. Values of the Nash criterion of candidate neural models using Theorem 3 (α = 0.7).

Neuronal model parameters	$n_a=1$ $n_b=1$ $n_c=1$ $n_k=0$ $n_h=1$	$n_a=1$ $n_b=1$ $n_c=1$ $n_k=0$ $n_h=2$	$n_a=1$ $n_b=1$ $n_c=1$ $n_k=0$ $n_h=3$	$n_a=1$ $n_b=1$ $n_c=1$ $n_k=0$ $n_h=4$	$n_a=1$ $n_b=1$ $n_c=1$ $n_k=0$ $n_h=5$	$n_a=1$ $n_b=1$ $n_c=1$ $n_k=0$ $n_h=6$	$n_a=2$ $n_b=1$ $n_c=1$ $n_k=0$ $n_h=7$	$n_a=2$ $n_b=2$ $n_c=1$ $n_k=0$ $n_h=8$	$n_a=2$ $n_b=2$ $n_c=2$ $n_k=0$ $n_h=8$	$n_a=2$ $n_b=2$ $n_c=2$ $n_k=1$ $n_h=8$	$n_a=2$ $n_b=2$ $n_c=2$ $n_k=2$ $n_h=8$
Nash criterion	72%	74%	77%	80%	86%	89.5%	92%	97.4%	93%	90.2%	89.1%

a random signal of zero mean and variance 1.

The evolution of the Nash criterion of different candidate models of the system (**Tables 1-3**) can be concluded that $n_a=2$, $n_b=2$, $n_c=1$, $n_k=0$, $\alpha=0.7$, 8 neurons in the hidden layer use of Theorem 3 for the learning phase, is necessary and sufficient for a neuronal model of a satisfactory precision.

The autocorrelation functions of residuals and cross-correlation between input and residuals (**Figure 7**) are within the confidence intervals, thus validating the use of the network chosen as a model of the system studied.

After the learning phase of the neuronal model completed, the structure proposed of neural adaptive control is applied to the system. In this case, the learning algorithm of the neural controller uses Theorem 6. The results are presented in **Figures 8**, **9** and **10**. It appears from these figures that this control strategy provides satisfactory results. Indeed, the system follows the reference signal appropriately by responding to the objectives: rejection of disturbances, the control performance, robustness and system stability.

5. Conclusion

In this paper, we have proposed adaptive control structure for a complex dynamic system using a recurrent neural network. Before, the application of the proposed adaptive neuro control, the recurrent neural has been trained off-line to implement the inverse dynamic of the considered system using a proposed Lyapunov-Base system training algorithm. The simulation results obtained show the effectiveness of the recurrent neural network structure and its adaptation algorithm to simulate the inverse dynamics of the system, and to control it in closed loop with good tracking performance.

REFERENCES

[1] L. J. Chen and K. S. Narendra, "Nonlinear Adaptive Control Using Neural Networks and Multiple Models," *Automatica*, Vol. 37, No. 8, 2001, pp. 1245-1255. doi:10.1016/S0005-1098(01)00072-3

[2] A. Bagheri, T. Karimi and N. Amanifard, "Tracking Performance Control of a Cable Communicated Underwater Vehicle Using Adaptive Neural Network Controllers," *Applied Soft Computing*, Vol. 10, No. 3, 2001, pp. 908-918. doi:10.1016/j.asoc.2009.10.008

[3] D. L. Yu, T. K. Chang and D. W. Yu, "Adaptive Neural Model-Based Fault Tolerant Control for Multi-Variable Processes," *Engineering Applications of Artificial Intelligence*, Vol. 18, No. 4, 2005, pp. 393-411. doi:10.1016/j.engappai.2004.10.003

[4] J. M. Renders, "Algorithmes Génétiques et Réseaux de Neurones," Hermes Sciences Publicat, Paris, 1995.

[5] Y.-K. Choi, M.-J. Lee, S. Kim and Y.-C. Kay, "Design and Implementation of an Adaptive Neural Network Compensator for Control Systems," *IEEE Transactions on Industrial Electronics*, Vol. 48, No. 2, 2001, pp. 416-423. doi:10.1109/41.915421

[6] N. Magnus, *et al*., "Neural Networks for Modelling and Control of Dynamic Systems," Springer Berlin, Heidelberg, 2000.

[7] I. Kar and L. Behera, "Direct Adaptive Neural Control for Affine Nonlinear Systems," *Applied Soft Computing*, Vol. 9, No. 2, 2009, pp. 756-764. doi:10.1016/j.asoc.2008.10.001

[8] F. N. Koumboulis and N. D. Kouvakas, "Indirect Adaptive Neural Control for Precalcination in Cement Plants," *Mathematics and Computers in Simulation*, Vol. 60, No. 3-5, 2002, pp. 325-334. doi:10.1016/S0378-4754(02)00024-1

[9] Z. Nagy, S. Agachi and L. Bodizs, "Adaptive Neural Network Model Based Nonlinear Predictive Control of a Fluid Catalytic Cracking Unit," *Computer Aided Chemical Engineering*, Vol. 8, 2000, pp. 235-240. doi:10.1016/S1570-7946(00)80041-3

[10] T. T. Hu, J. H. Zhu and Z. Q. Sun, "Robust Adaptive Neural Control of a Class of MIMO Nonlinear Systems," *Tsinghua Science & Technology*, Vol. 12, No. 1, 2007, pp. 14-21. doi:10.1016/S1007-0214(07)70003-2

[11] H. Deng, H. X. Li and Y. H. Wu, "Feedback-Linearization-Based Neural Adaptive Control for Unknown Nonaffine Nonlinear Discrete-Time Systems," *IEEE Transactions on Neural Networks*, Vol. 19, No. 9, 2008, pp. 1615-1625.

[12] C. J. Yu, J. H. Zhu and Z. Q. Sun, "Adaptive Neural Network Internal Model Control for Tilt Rotor Aircraft Platform," *Advances in Natural Computation*, Vol. 3611, 2005, pp. 262-265. doi:10.1007/11539117_38

[13] S. Yang, W. Q. Qian, W. S. Yan and J. Li, "Adaptive Depth Control for Autonomous Underwater Vehicles Based on Feedforward Neural Networks," *International Journal of Computer Science & Applications*, Vol. 4, No. 3, 2007, pp. 107-118.

[14] M. Jalali-Heravi, M. Asadollahi-Baboli and P. Shahbazikhah, "QSAR Study of Heparanase Inhibitors Activity Using Artificial Neural Networks and Levenberg-Marquardt Algorithm," *European Journal of Medicinal Chemistry*, Vol. 43, No. 3, 2008, pp. 548-556. doi:10.1016/j.ejmech.2007.04.014

[15] K.-I. Funahashi, "On the Approximate Realization of Continuous Mapping by Neural Networks," *Neural networks*, Vol. 2, No. 3, 1989, pp. 183-192. doi:10.1016/0893-6080(89)90003-8

[16] G. Cybenko, "Approximation by Superposition of a Sigmoidal Function," *Mathematics of Control*, *Signal*, *and Systems*, Vol. 2, 1989, pp. 303-314.

[17] D. Psaltis, A. Sideris and A. A. Yamamura, "A Multilayered Neural Network Control," *IEEE Control Systems Magazine*, Vol. 8, No. 2, 1988, pp. 17-21. doi:10.1109/37.1868

[18] J. Baltersee and J. A. Chambers, "Nonlinear Adaptive Prediction of Speech with a Pipelined Recurrent Neural Network," *IEEE Transactions on Signal Processing*, Vol. 46, No. 8, 1998, pp. 2207-2216. doi:10.1109/78.705435

[19] D. G. Stavrakoudis and J. B. Theocharis, "Pipelined Recurrent Fuzzy Neural Networks for Nonlinear Adaptive Speech Prediction," *IEEE Transactions on Systems*, *Man*, *and Cybernetics*, (*Part B*): *Cybernetics*, Vol. 37, No. 5, 2007, pp. 1305-1320. doi:10.1109/TSMCB.2007.900516

[20] H. Q. Zhao and J. S. Zhang, "A Novel Adaptive Nonlinear Filter Based Pipelined Feedforward Second-Order Volterra Architecture," *IEEE Transactions Signal Processing*, Vol. 57, No. 1, 2009, pp. 237-246. doi:10.1109/TSP.2008.2007105

[21] P.-R. Chang and J.-T. Hu, "Optimal Nonlinear Adaptive Prediction and Modeling of MPEG Video in ATM Networks Using Pipelined Recurrent Neural Networks," *IEEE Journal of Selected Areas in Communications*, Vol. 15, No. 6, 1997, pp. 1087-1100. doi:10.1109/49.611161

[22] Y.-S. Chen, C.-J. Chang and Y.-L. Hsieh, "A Channel Effect Prediction-Based Power Control Scheme Using PRNN/ERLS for Uplinks in DS-CDMA Cellular Mobile Systems," *IEEE Transactions on Wireless Communications*, Vol. 5, No. 1, 2006, pp. 23-27.

doi:10.1109/TWC.2006.1576521

[23] D. P. Mandic and J. A. Chambers, "Toward an Optimal PRNN-Based Nonlinear Prediction," *IEEE Transactions on Neural Networks*, Vol. 10, No. 6, 1999, pp. 1435-1442. doi:10.1109/72.809088

[24] D. P. Mandic and J. A. Chambers, "On the Choice of Parameters of the Cost Function in Nested Modular RNN's," *IEEE Transactions on Neural Networks*, Vol. 11, No. 2, 2000, pp. 315-322. doi:10.1109/72.839003

[25] H. Q. Zhao and J. S. Zhang, "Pipelined Chebyshev Functional Link Artificial Recurrent Neural Network for Nonlinear Adaptive Filter," *IEEE Transactions on Systems, Man, and Cybernetics Part B*: *Cybernetics*, Vol. 40, No. 1, 2010, pp. 162-172. doi:10.1109/TSMCB.2009.2024313

[26] R. J. Williams and D. Zipser, "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks," *Neural Computation*, Vol. 1, No. 2, 1989, pp. 270-280. doi:10.1162/neco.1989.1.2.270

[27] B. A. Pearlmutter, "Dynamic Recurrent Neural Networks," *Technical Report CMU-CS*-88-191, Information Science and Technology Office, 1990.

[28] B. A. Pearlmutter, "Gradient Calculations for Dynamic Recurrent Neural Networks: A Survey," *IEEE Transactions on Neural Networks*, Vol. 6, No. 5, 1995, pp. 1212-1228. doi:10.1109/72.410363

[29] H. Al-Duwaish, M. N. Karim and V. Chandrasekar, "Use of Multilayer Feedforward Neural Networks in Identification and Control of Wiener Model," *IEEE Proceedings—Control Theory and Applications*, Vol. 143, No. 3, 1996, pp. 255-258. doi:10.1049/ip-cta:19960376

[30] K. S. Narendra and K. Parthasarathy, "Identification and Control of Dynamical Systems Using Neural Networks," *IEEE Transactions on Neural Networks*, Vol. 1, No. 1, 1990, pp. 4-27. doi:10.1109/72.80202

[31] C.-C. Ku and K. Y. Lee, "Diagonal Recurrent Neural Networks for Dynamic Systems Control," *IEEE Transactions on Neural Networks*, Vol. 6, No. 1, 1995, pp. 144-156. doi:10.1109/72.363441

[32] H. Chaoui and P. Sicard, "Adaptive Lyapunov-Based Neural Network Sensorless Control of Permanent Magnet Synchronous Machines," *Neural Computing & Applications*, Vol. 20, No. 5, 2010, pp. 717-727. doi:10.1007/s00521-010-0412-6

[33] R. A. Hooshmand and G. Isazadeh, "Application of Adaptive Lyapunov-Based UPFC Supplementary Controller by Neural Network Algorithm in Multi-Machine Power System," *Electrical Engineering* (*Archiv fur Elektrotechnik*), Vol. 91, No. 4-5, 2009, pp. 187-195. doi:10.1007/s00202-009-0132-z

[34] T. Yabuta and T. Yamada, "Learning Control Using Neural Networks," *Proceedings of* 1991 *IEEE International conference on Robotics and Automation*, Sacramento, 9-11 April 1991, pp. 740-745. doi:10.1109/ROBOT.1991.131673

[35] Y. H. Tan and A. van Cauwenberghe, "Nonlinear One-Step-Ahead Control Using Neural Networks: Control Strategy and Stability Design," *Automatica*, Vol. 32, No. 12, 1996, pp. 1667-1667. doi:10.1016/S0005-1098(96)80006-9

[36] T. Denoeux and R. Lengellé, "Initializing Back Propagation Networks with Prototypes," *Neural Networks*, Vol. 6, No. 3, 1993, pp. 351-363. doi:10.1016/0893-6080(93)90003-F

[37] G. P. Drago and S. Ridella, "Statistically Controlled Activation Weight Initialization (SCAWI)," *IEEE Transactions on Neural Networks*, Vol. 3, No. 4, 1992, pp. 627-631. doi:10.1109/72.143378

[38] J.-P. Martens, "A Stochastically Motivated Random Initialization of Pattern Classifying MLPs," *Neural Processing Letters*, Vol. 3, No. 1, 1996, pp. 23-29. doi:10.1007/BF00417786

[39] T. Masters, "Practical Neural Network Recipes in C++," Academic Press, Boston, 1993.

[40] D. Nguyen and B. Widrow, "Improving the Learning Speed of 2-Layer Neural Networks by Choosing Initial Values of the Adaptive Weights," 1990 *IJCNN International Joint Conference on Neural Networks*, San Diego, 17-21 June 1990, pp. 21-26. doi:10.1109/IJCNN.1990.137819

[41] S. Osowski, "New Approach to Selection of Initial Values of Weights in Neural Function Approximation," *Electronics Letters*, Vol. 29, No. 3, 1993, pp. 313-315. doi:10.1049/el:19930214

[42] J. F. Shepanski, "Fast Learning in Artificial Neural Systems: Multilayer Perceptron Training Using Optimal Estimation," 1998 *IEEE International Conference on Neural Networks*, San Diego, 24-27 July 1988, pp. 465-472. doi:10.1109/ICNN.1988.23880

[43] H. Shimodaira, "A Weight Value Initialization Method for Improving Learning Performance of the Back Propagation Algorithm in Neural Networks," 1994 *Proceedings of Sixth International Conference on Tools with Artificial Intelligence*, New Orleans, 6-9 November 1994, pp. 672-675. doi:10.1109/TAI.1994.346429

[44] Y. F. Yam and T. W. S. Chow, "Determining Initial Weights of Feedforward Neural Networks Based on Least Squares Method," *Neural Processing Letters*, Vol. 2, No. 2, 1995, pp. 13-17. doi:10.1007/BF02312350

[45] Y. F. Yam, T. W. S. Chow and C. T. Leung, "A New Method in Determining Initial Weights of Feedforward Neural Networks for Training Enhancement," *Neurocomputing*, Vol. 16, No. 1, 1997, pp. 23-32. doi:10.1016/S0925-2312(96)00058-6

[46] L. F. A. Wessels and E. Barnard, "Avoiding False Local Minima by Proper Initialization of Connections," *IEEE Transactions on Neural Networks*, Vol. 3, No. 6, 1992, pp. 899-905. doi:10.1109/72.165592

[47] N. Weymaere and J.-P. Martens, "On the Initialization and Optimization of Multilayer Perceptrons," *IEEE Transactions on Neural Networks*, Vol. 5, No. 5, 1994, pp. 738-751. doi:10.1109/72.317726

[48] J. Y. F. Yam and T. W. S. Chow, "A Weight Initialization Method for Improving Training Speed in Feedforward Neural Network," *Neurocomputing*, Vol. 30, No. 1-4, 2000, pp. 219-232. doi:10.1016/S0925-2312(99)00127-7

[49] J. E. Nash and J. V. Sutcliffe, "River Flow Forecasting through Conceptual Models Part I—A Discussion of Principles," *Journal of Hydrology*, Vol. 10, No. 3, 1970, pp. 282-290. doi:10.1016/0022-1694(70)90255-6

[50] S. A. Billings and Q. M. Zhu, “Nonlinear Model Validation Using Correlation Tests,” *International Journal of Control*, Vol. 60, No. 6, 1994, pp. 1107-1120. doi:10.1080/00207179408921513

[51] S. A. Billings, H. B. Jamaluddin and S. Chen, “Properties of Neural Networks with Applications to Modelling Non-Linear Dynamical Systems,” *International Journal of Control*, Vol. 55, No. 1, 1992, pp. 193-224. doi:10.1080/00207179208934232

[52] F. Zouari, K. B. Saad and M. Benrejeb, “Adaptive Internal Model Control of DC-Motor Drive System Using Dynamic Neural Network,” *Journal of Software Engineering and Applications*, Vol. 5, No. 3, 2012, pp. 168-189. doi:10.4236/jsea.2012.53024

Appendixs

The calculation of the term $\frac{\partial\hat{y}(k)}{\partial w(k)}$ is performed by the following equations:
for the neuron in the output layer:

$$\frac{\partial\hat{y}(k)}{\partial w_{11}^{7}(k)}=\hat{y}(k-1)+w_{11}^{7}(k)\frac{\partial\hat{y}(k-1)}{\partial w_{11}^{7}(k)}+\sum_{i=n_a+n_b+1}^{n_r}w_{1i}^{5}(k)\frac{\partial\hat{y}(k+n_a+n_b-i)}{\partial w_{11}^{7}(k)}+\sum_{j=1}^{n_h}w_{1j}^{3}(k)\frac{\partial x_j^h(k)}{\partial w_{11}^{7}(k)} \quad (138)$$

$$\frac{\partial x_j^h(k)}{\partial w_{11}^{7}(k)}=f_1'(s_j(k))\left(\sum_{i=n_a+n_b+1}^{n_r}w_{ji}^{1}(k)\frac{\partial\hat{y}(k+n_a+n_b-i)}{\partial w_{11}^{7}(k)}+\sum_{i=1}^{n_h}w^{6}{}_{ji}(k)\frac{\partial x_i^h(k-1)}{\partial w_{11}^{7}(k)}\right) \quad (139)$$

$$\frac{\partial\hat{y}(k)}{\partial w_{1j}^{3}(k)}=x_j^h(k)+w_{1j}^{3}(k)\frac{\partial x^h{}_j(k)}{\partial w_{1j}^{3}}+w_{11}^{7}(k)\frac{\partial\hat{y}(k-1)}{\partial w_{1j}^{3}(k)}+\sum_{i=n_a+n_b+1}^{n_r}w_{1i}^{5}(k)\frac{\partial\hat{y}(k+n_a+n_b-i)}{\partial w_{1j}^{3}(k)} \quad (140)$$

$$\frac{\partial x_j^h(k)}{\partial w_{1j}^{3}(k)}=f_1'(s_j(k))\cdot\left(\sum_{i=n_a+n_b+1}^{n_r}w_{ji}^{1}(k)\frac{\partial\hat{y}(k+n_a+n_b-i)}{\partial w_{1j}^{3}(k)}+\sum_{m=1}^{n_h}w_{jm}^{6}(k)\frac{\partial x_m^h(k-1)}{\partial w_{1j}^{3}(k)}\right) \quad (141)$$

$$\frac{\partial\hat{y}(k)}{\partial w_{1m}^{5}(k)}=\psi_m(k)+\sum_{i=n_a+n_b+1}^{n_r}w_{1i}^{5}(k)\frac{\partial\hat{y}(k+n_a+n_b-i)}{\partial w_{1m}^{5}(k)}+\sum_{j=1}^{n_h}w_{1j}^{3}(k)\frac{\partial x_j^h(k)}{\partial w_{1m}^{5}}+w_{11}^{7}(k)\frac{\partial\hat{y}(k-1)}{\partial w_{1m}^{5}(k)} \quad (142)$$

$$\frac{\partial x_j^h(k)}{\partial w_{1j}^{5}(k)}=f_1'(s_j(k))\left(\sum_{i=n_a+n_b+1}^{n_r}w_{ji}^{1}(k)\frac{\partial\hat{y}(k+n_a+n_b-i)}{\partial w_{1j}^{5}(k)}+\sum_{m=1}^{n_h}w_{jm}^{6}(k)\frac{\partial x_m^h(k-1)}{\partial w_{1j}^{5}(k)}\right) \quad (143)$$

$$\frac{\partial\hat{y}(k)}{\partial w_{11}^{4}(k)}=1+w_{11}^{7}(k)\frac{\partial\hat{y}(k-1)}{\partial w_{11}^{4}(k)}+\sum_{i=n_a+n_b+1}^{n_r}w_{1i}^{5}(k)\frac{\partial\hat{y}(k+n_a+n_b-i)}{\partial w_{11}^{4}(k)}+\sum_{j=1}^{n_h}w_{1j}^{3}(k)\frac{\partial x_j^h(k)}{\partial w_{11}^{4}(k)} \quad (144)$$

$$\frac{\partial x_j^h(k)}{\partial w_{11}^{4}}=f_1'(s_j(k))\left(\sum_{i=n_a+n_b+1}^{n_r}w_{ji}^{1}(k)\frac{\partial\hat{y}(k+n_a+n_b-i)}{\partial w_{11}^{4}(k)}+\sum_{m=1}^{n_h}w_{jm}^{6}(k)\frac{\partial x_m^h(k-1)}{\partial w_{11}^{4}(k)}\right) \quad (145)$$

for a neuron in the hidden layer:

$$\frac{\partial\hat{y}(k)}{\partial w_{jm}^{6}(k)}=w_{11}^{7}(k)\frac{\partial\hat{y}(k-1)}{\partial w_{jm}^{6}}+\sum_{i=n_a+n_b+1}^{n_r}w_{1i}^{5}(k)\frac{\partial\hat{y}(k+n_a+n_b-i)}{\partial w_{jm}^{6}(k)}+\sum_{n=1}^{n_h}w_{1n}^{3}(k)\frac{\partial x_n^h(k)}{\partial w_{jm}^{6}(k)} \quad (146)$$

if $n=j$

$$i\frac{\partial x_j^h(k)}{\partial w_{jm}^{6}}=f_1'(s_j(k))\cdot\left(x_m^h(k-1)+\sum_{i=n_a+n_b+1}^{n_r}w_{ji}^{1}(k)\frac{\partial\hat{y}(k+n_a+n_b-i)}{\partial w_{jm}^{6}(k)}+\sum_{i=1}^{n_h}w_{ji}^{6}(k)\frac{\partial x_i^h(k-1)}{\partial w_{jm}^{6}(k)}\right) \quad (147)$$

if $n\neq j$

$$i\frac{\partial x_n^h(k)}{\partial w_{jm}^{6}(k)}=f_1'(s_n(k))\left(\sum_{i=n_a+n_b+1}^{n_r}w_{ni}^{1}(k)\frac{\partial\hat{y}(k+n_a+n_b-i)}{\partial w_{jm}^{6}(k)}+\sum_{i=1}^{n_h}w_{ni}^{6}(k)\frac{\partial x_i^h(k-1)}{\partial w_{jm}^{6}(k)}\right) \quad (148)$$

$$\frac{\partial\hat{y}(k)}{\partial w_{j1}^{2}(k)} = w_{11}^{7}(k)\frac{\partial\hat{y}(k-1)}{\partial w_{j1}^{2}(k)} + \sum_{i=n_a+n_b+1}^{n_r} w_{1i}^{5}(k)\frac{\partial\hat{y}(k+n_a+n_b-i)}{\partial w_{j1}^{2}(k)} + \sum_{n=1}^{n_h} w_{1n}^{3}(k)\frac{\partial x^{h}{}_{n}(k)}{\partial w_{j1}^{2}(k)} \quad (149)$$

if $n = j$

$$i\frac{\partial x_n^h(k)}{\partial w_{j1}^{2}(k)} = f_1'(s_j(k)) \cdot\left(\sum_{i=n_a+n_b+1}^{n_r} w_{ji}^{1}(k)\frac{\partial\hat{y}(k+n_a+n_b-i)}{\partial w_{j1}^{2}(k)} + \sum_{m=1}^{n_h} w_{jm}^{6}(k)\frac{\partial x_m^h(k-1)}{\partial w_{j1}^{2}(k)} + 1\right) \quad (150)$$

if $n \neq j$

$$i\frac{\partial x_n^h(k)}{\partial w_{j1}^{2}(k)} = f_1'(s_n(k)) \cdot\left(\sum_{i=n_a+n_b+1}^{n_r} w_{ni}^{1}(k)\frac{\partial\hat{y}(k+n_a+n_b-i)}{\partial w_{j1}^{2}(k)} + \sum_{m=1}^{n_h} w_{nm}^{6}(k)\frac{\partial x_m^h(k-1)}{\partial w_{j1}^{2}(k)}\right) \quad (151)$$

$$\frac{\partial\hat{y}(k)}{\partial w_{jm}^{1}(k)} = w_{11}^{7}(k)\frac{\partial\hat{y}(k-1)}{\partial w_{jm}^{1}(k)} + \sum_{i=n_a+n_b+1}^{n_r} w_{1i}^{5}(k)\frac{\partial\hat{y}(k+n_a+n_b-i)}{\partial w_{jm}^{1}(k)} + \sum_{n=1}^{n_h} w_{1n}^{3}(k)\frac{\partial x_n^h(k)}{\partial w_{jm}^{1}(k)} \quad (152)$$

if $n = j$

$$\frac{\partial x_j^h(k)}{\partial w_{jm}^{1}(k)} = f_1'(s_j(k))\Big(\psi_m(k) + \sum_{i=n_a+n_b+1}^{n_r} w_{ji}^{1}(k)\frac{\partial\hat{y}(k+n_a+n_b-i)}{\partial w_{jm}^{1}(k)} + \sum_{i=1}^{n_h} w_{ji}^{6}(k)\frac{\partial x_i^h(k-1)}{\partial w_{jm}^{1}(k)}\Big) \quad (153)$$

if $n \neq j$

$$\frac{\partial x_n^h(k)}{\partial w_{jm}^{1}(k)} = f_1'(s_n(k)) \cdot\left(\sum_{i=n_a+n_b+1}^{n_r} w_{ni}^{1}(k)\frac{\partial\hat{y}(k+n_a+n_b-i)}{\partial w_{jm}^{1}(k)} + \sum_{i=1}^{n_h} w_{ni}^{6}(k)\frac{\partial x_i^h(k-1)}{\partial w_{jm}^{1}(k)}\right) \quad (154)$$

From the above equations, we can write:

$$\frac{\partial\hat{y}(k-i)}{\partial w(k)} = 0 \ \ \forall i = 1,\cdots,n_a \quad (155)$$

$$\frac{\partial x_j^h(k-1)}{\partial w(k)} = 0 \ \ \forall j = 1,\cdots,n_h \quad 156)$$

It was therefore:

$$\frac{\partial\hat{y}(k)}{\partial w_{jm}^{1}(k)} = w_{1j}^{3}(k)\, f_1'(s_j(k))\psi_m(k) \quad (157)$$

$$\frac{\partial\hat{y}(k)}{\partial w_{j1}^{2}(k)} = w_{1j}^{3}(k)\, f_1'(s_j(k)) \quad (158)$$

$$\frac{\partial\hat{y}(k)}{\partial w_{1j}^{3}(k)} = x_j^h(k) \quad (159)$$

$$\frac{\partial\hat{y}(k)}{\partial w_{11}^{4}(k)} = 1 \quad (160)$$

$$\frac{\partial\hat{y}(k)}{\partial w_{1m}^{5}(k)} = \psi_m(k) \quad (161)$$

$$\frac{\partial\hat{y}(k)}{\partial w_{jm}^{6}(k)} = w_{1j}^{3}(k)\, f_1'(s_j(k))\left(x_m^h(k-1)\right) \quad (162)$$

$$\frac{\partial\hat{y}(k)}{\partial w_{11}^{7}(k)} = \hat{y}(k-1) \approx y(k-1) \quad (163)$$