Available online at www.elixirpublishers.com (Elixir International Journal)

Control Engineering

Elixir Control Engg. 55A (2013) 13050-13064

ISSN: 2229-712X

Robust neuronal adaptive control for a class of uncertain nonlinear complex dynamical multivariable systems

Farouk Zouari

LAboratoire de Recherche en Automatique (LARA),École Nationale d'Ingénieurs de Tunis (ENIT), B.P. 37, 1002 Tunis.

ARTICLE INFO

Article history:

Received: 3 May 2012;

Received in revised form:

13 February 2013;

Accepted: 14 February 2013;

Keywords

Neural adaptive controls;
Robust neural adaptive control;
Neural indirect adaptive control;
Lyapunov theory;
Uncertain nonlinear multivariable systems.

ABSTRACT

In this paper, we proposed the development of neural adaptive controls to ensure the robustness of uncertain nonlinear multivariable systems. We used two techniques: Robust neural adaptive control and neural indirect adaptive control. The study of the stability and robustness of both techniques was performed by Lyapunov theory. To validate these techniques and discover their effectiveness, a simulation example was considered. The simulation results obtained by these two control techniques have shown the effects of disturbance compensation, good performance tracking data paths and stability control systems. Comparative studies between these two techniques show that the neural indirect adaptive control cannot mitigate the effect of disturbances compared to the robust neural adaptive control.

Introduction

In recent decades, the robust neuronal adaptive control of complex nonlinear dynamic systems has been studied in several research works which we quote [1-6]. It is used in several industrial applications and particularly in the cases where we are confronted with complex nonlinear dynamics and inaccuracies due to uncertainties attached to the system to be controlled. The use of the method of robustification is essential to improve the tracking performance and ensure the robustness of the closed loop system in front of the structural uncertainties and external disturbances. This control technology adds to the main control signal a supervisory signal by sliding mode or of H_∞ type.

Several studies of robustification of sliding mode adaptive neural control are based on the use of adaptive neural networks for modeling the process or to calculate the desired control law [7-11]. Generally, the control laws were derived from the examination of stability. Overall, the constructed command ensures stability and good tracking performance. The disadvantage of sliding mode adaptive neural control is the existence of the sign function that causes sudden and rapid changes of the control signal, which can excite the high frequency of the process and cause damage it. Many solutions have been proposed in literature in particular Lie Slotine and added a transition band around the sliding surface to transform the sign function in saturation and thus remove the abrupt changes [12].

Several research works have studied the technique H_∞ like [13-16]. This technique aims at determining the tracking performances based on a criterion connecting on the one hand the norms of prosecution errors and on the other hand the desired level of disturbance attenuation. This criterion can be interpreted in the state space by obtaining a positive definite matrix unique solution of the Riccati equation.

The contribution of this paper is to propose two adaptive controls neuronal structures of nonlinear dynamical multivariable systems rested on the theory of Lyapunov. The architecture and learning algorithm of these two neural adaptive control structures require the modeling of system to be controlled, that is to say the determination of its state equations using the concepts of neural networks. In this sense, we proposed linearization technique inputs- outputs of the system to be controlled based on neural networks. This technique consists in finding linear relationships between inputs and outputs of the system. The neural model of the system obtained online by this linearization technique is used to calculate the commands laws. In fact, the first proposed control structure which is the neural indirect adaptive control uses the Jacobian matrix of the neural model during the calculation of the parameters of neural controller. by cons the second proposed control structure which is the robust neural adaptive control, use the state equations derived from the neural model to determine the neuronal controller and add to main control signals supervision signals by the technique H_∞.

This document is organized as follows: In Section 2 we present the proposed architecture of recurrent neural network and its learning algorithm based on Lyapunov theory, used in the calculations of the model parameters of a complex nonlinear dynamic multivariable system. In Section 3 we show the architecture of the neuronal controller and its proposed learning algorithm in the structure of neuronal indirect adaptive control. The proposed structure of robust neural adaptive control by the technique H_∞ is described in Section 4. The numerical results and discussions of these two commands mentioned previously are presented in Section 5. Finally, the conclusion is given in Section 6.

Tele:
E-mail addresses: zouari.farouk@gmail.com

Neural network modeling approach

Neural network modeling of a system from samples affected by noise usually requires four steps- The first step is the choice of the architecture of the neural network, that is to say the number of neurons in the input layer which is a function of past values of inputs and outputs, the number of hidden layers, the number of neurons in each hidden layer, the number of neurons in the output layer, the activation functions of each neuron and organization of these neurons between themselves [17-28].

-The second step is the normalization or the transformation performed on the data inputs-outputs to distribute them uniformly and adapt them to an acceptable level for the neural network [29-31]. All data values must be between $[0,1]$ or $[-1,1]$.- The third step is learning or in other words the calculation of network parameters from samples inputs-outputs system to be identified [32-34].- The fourth step is the validation of the neural network obtained by using the tests measurements performances criteria.

Proposed architecture of recurrent neural network

The figure 1 shows the architecture of the neural network used during the identification phase of an uncertain and perturbed nonlinear complex dynamic multivariable system (with p inputs and p outputs). The structure of this neural network is composed of three parts: two linear parts to model the behavior linear of the system and a nonlinear part to approximate the nonlinear dynamics.

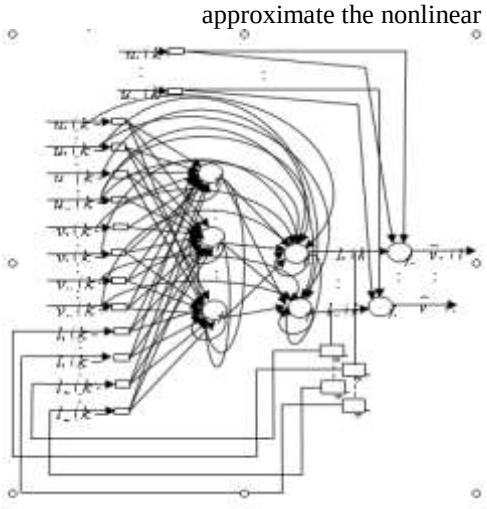


Figure 1- Proposed architecture of the neural network

with:

$\hat{y}(k)=\left[\hat{y}_1(k),\text{L},\hat{y}_p(k)\right]^T$ is the vector of the neural network outputs at instant k.

$u(k-1)=\left[u_1(k-1),\text{L},u_p(k-1)\right]^T$ is the vector of the system inputs.

$y(k)=\left[y_1(k),\text{L},y_p(k)\right]^T$ is the vector of the system outputs.

$$U(k-2)=\left[u_1(k-2),\text{L},u_1(k-n_{b1}),\text{L},u_j(k-2),\right.$$
$$\left.\text{L},u_j(k-n_{bj}),\text{L},u_p(k-2),\text{L},u_p(k-n_{bp})\right]^T$$

such as $2\le n_{bj}$, $1<j<p$ (1)

$$Y(k-1)=\left[y_1(k-1),\text{L},y_1(k-n_{a1}),\text{L},y_j(k-1),\right.$$
$$\left.\text{L},y_j(k-n_{aj}),\text{L},y_p(k-1),\text{L},y_p(k-n_{ap})\right]^T$$

such as $1\le n_{aj}$, $1<j<p$ (2)

$\chi(k)=\left[l_1(k),\text{L},l_p(k)\right]^T$ is the vector of the second hidden layer outputs of neural model

$$Y_r(k-1)=\left[l_1(k-1),\text{L},l_1(k-n_{c1}),\text{L},l_j(k-1),\right.$$
$$\left.\text{L},l_j(k-n_{cj}),\text{L},l_p(k-1),\text{L},l_p(k-n_{cp})\right]^T$$

such as $1\le n_{cj}$, $1<j<p$ (3)

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

n_h the number of neurons in the first hidden layer.

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

$$w^1=\begin{bmatrix} w^1_{11} & \text{L} & w^1_{1n_r} \\ \text{M} & \text{O} & \text{M} \\ w^1_{n_h1} & \text{L} & w^1_{n_hn_r} \end{bmatrix}$$ the weights between neurons of the input layer and neurons of the first hidden layer,

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

$$w^3=\begin{bmatrix} w^3_{11} & \text{L} & w^3_{1n_h} \\ \text{M} & \text{O} & \text{M} \\ w^3_{p1} & \text{L} & w^3_{pn_n} \end{bmatrix}$$ the weights between neurons of the first hidden layer and neurons of the second hidden layer,

$$w^4=\begin{bmatrix} w^4_{11} \\ \text{M} \\ w^4_{p1} \end{bmatrix}$$ the bias of neurons in the second hidden layer,

$$w^5=\begin{bmatrix} w^5_{11} & \text{L} & w^5_{1n_r} \\ \text{M} & \text{O} & \text{M} \\ w^5_{p1} & \text{L} & w^5_{pn_r} \end{bmatrix}$$ the weights between neurons of the input layer and neurons of the second hidden layer,

$$w^6=\begin{bmatrix} w^6_{11} & \text{L} & w^6_{1n_h} \\ \text{M} & \text{O} & \text{M} \\ w^6_{n_h1} & \text{L} & w^6_{n_hn_h} \end{bmatrix}$$ the weights between neurons of the first hidden layer,

$$w^7=\begin{bmatrix} w^7_{11} & \text{L} & w^7_{1p} \\ \text{M} & \text{O} & \text{M} \\ w^7_{p1} & \text{L} & w^7_{pp} \end{bmatrix}$$ the weights between neurons of the second hidden layer,

$w^8 = \begin{bmatrix} w^8_{11} & \text{L} & w^8_{1p} \\ \text{M} & \text{O} & \text{M} \\ w^8_{p1} & \text{L} & w^8_{pp} \end{bmatrix}$ the weights between neurons of the input layer and output layer.

$x^h(k) = \begin{bmatrix} x^h_{11}(k) \\ \text{M} \\ x^h_{n_h 1}(k) \end{bmatrix}$ the first hidden layer outputs of neural model,

$$n_r = \sum_{j=1}^{p} n_{aj} + \sum_{j=1}^{p} n_{cj} + \sum_{j=1}^{p} n_{bj} - p$$

The vector of the neural model parameters is as follows:

$$w = \left[w^1_{11}, \text{L}\ , w^1_{n_h n_r}, w^2_{11}, \text{L}\ , w^2_{n_h 1}, w^3_{11}, \text{L}\ , w^3_{pn_h}, w^4_{11}, \text{L}\ , w^4_{p1}, w^5_{11}, \text{L}\ , w^5_{pn_r}, w^6_{11}, \text{L}\ , w^6_{n_h n_h}, w^7_{11}, \text{L}\ , w^7_{pp}, w^8_{11}, \text{L}\ , w^8_{pp} \right]^T \quad (5)$$

$$\psi(k) = \left[(U(k-2))^T, (Y(k-1))^T, (Y_r(k-1))^T \right]^T = \left[\psi_1(k), \text{L}\ , \psi_{n_r}(k) \right]^T \quad (6)$$

The vector of the first hidden layer outputs is in the following form:

$$x^h(k) = \begin{bmatrix} x^h_1(k) \\ \text{M} \\ x^h_{n_h}(k) \end{bmatrix} = \begin{bmatrix} f_1(s_1(k)) \\ \text{M} \\ f_1(s_{n_h}(k)) \end{bmatrix} \quad (7)$$

Such as:

$$S(k) = \begin{bmatrix} s_1(k) \\ \text{M} \\ s_{n_h}(k) \end{bmatrix} = \begin{bmatrix} w^1_{11}(k) & \text{L} & w^1_{1n_r}(k) \\ \text{M} & 0 & \text{M} \\ w^1_{n_h 1}(k) & \text{L} & w^1_{n_h n_r}(k) \end{bmatrix} \begin{bmatrix} \psi_1(k) \\ \text{M} \\ \psi_{n_r}(k) \end{bmatrix} + \begin{bmatrix} w^6_{11}(k) & \text{L} & w^6_{1n_h}(k) \\ \text{M} & 0 & \text{M} \\ w^6_{n_h 1}(k) & \text{L} & w^6_{n_h n_h}(k) \end{bmatrix} \begin{bmatrix} x^h_1(k-1) \\ \text{M} \\ x^h_{n_h}(k-1) \end{bmatrix} + \begin{bmatrix} w^2_{11}(k) \\ \text{M} \\ w^2_{n_h 1}(k) \end{bmatrix} \quad (8)$$

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

$$\begin{bmatrix} \hat{y}_1(k) \\ \text{M} \\ \hat{y}_p(k) \end{bmatrix} = \begin{bmatrix} w^7_{11}(k) & \text{L} & w^7_{1p}(k) \\ \text{M} & 0 & \text{M} \\ w^7_{p1}(k) & \text{L} & w^7_{pp}(k) \end{bmatrix} \begin{bmatrix} l_1(k-1) \\ \text{M} \\ l_p(k-1) \end{bmatrix} + \begin{bmatrix} w^3_{11}(k) & \text{L} & w^3_{1n_h}(k) \\ \text{M} & 0 & \text{M} \\ w^3_{p1}(k) & \text{L} & w^3_{pn_h}(k) \end{bmatrix} \begin{bmatrix} x^h_1(k) \\ \text{M} \\ x^h_{n_h}(k) \end{bmatrix} + \begin{bmatrix} w^5_{11}(k) & \text{L} & w^5_{1n_r}(k) \\ \text{M} & 0 & \text{M} \\ w^5_{p1}(k) & \text{L} & w^5_{pn_r}(k) \end{bmatrix} \begin{bmatrix} \psi_1(k) \\ \text{M} \\ \psi_{n_r}(k) \end{bmatrix} + \begin{bmatrix} w^4_{11}(k) \\ \text{M} \\ w^4_{p1}(k) \end{bmatrix} + \begin{bmatrix} w^8_{11}(k) & \text{L} & w^8_{1p}(k) \\ \text{M} & 0 & \text{M} \\ w^8_{p1}(k) & \text{L} & w^8_{pp}(k) \end{bmatrix} \begin{bmatrix} u_1(k-1) \\ \text{M} \\ u_p(k-1) \end{bmatrix} \quad (9)$$

Assuming that:

$$\begin{bmatrix} h_1(\psi(k)) \\ \text{M} \\ h_p(\psi(k)) \end{bmatrix} = \begin{bmatrix} w^7_{11}(k) & \text{L} & w^7_{1p}(k) \\ \text{M} & 0 & \text{M} \\ w^7_{p1}(k) & \text{L} & w^7_{pp}(k) \end{bmatrix} \begin{bmatrix} x_1(k-1) \\ \text{M} \\ x_p(k-1) \end{bmatrix} + \begin{bmatrix} w^3_{11}(k) & \text{L} & w^3_{1n_h}(k) \\ \text{M} & 0 & \text{M} \\ w^3_{p1}(k) & \text{L} & w^3_{pn_h}(k) \end{bmatrix} \begin{bmatrix} x^h_1(k) \\ \text{M} \\ x^h_{n_h}(k) \end{bmatrix} + \begin{bmatrix} w^5_{11}(k) & \text{L} & w^5_{1n_r}(k) \\ \text{M} & 0 & \text{M} \\ w^5_{p1}(k) & \text{L} & w^5_{pn_r}(k) \end{bmatrix} \begin{bmatrix} \psi_1(k) \\ \text{M} \\ \psi_{n_r}(k) \end{bmatrix} + \begin{bmatrix} w^4_{11}(k) \\ \text{M} \\ w^4_{p1}(k) \end{bmatrix} \quad (10)$$

Equation (9) then becomes:

$$\begin{bmatrix} \hat{y}_1(k) \\ \text{M} \\ \hat{y}_p(k) \end{bmatrix} = \begin{bmatrix} h_1(\psi(k)) \\ \text{M} \\ h_p(\psi(k)) \end{bmatrix} + \begin{bmatrix} w^8_{11}(k) & \text{L} & w^8_{1p}(k) \\ \text{M} & O & \text{M} \\ w^8_{p1}(k) & \text{L} & w^8_{pp}(k) \end{bmatrix} \begin{bmatrix} u_1(k-1) \\ \text{M} \\ u_p(k-1) \end{bmatrix} \quad (11)$$

Equation (11) can be rewritten in the following state representation:

$$\begin{cases} x_i(k+1) = x_{i+1}(k),\ i = n_j, \text{L}\ , m_j - 1,\ j = 1, \text{L}\ , p,\ q_s = \max\left(n_{b1}, \text{L}\ , n_{bp}, n_{a1}, \text{L}\ , n_{ap}, n_{c1}, \text{L}\ , n_{cp}\right), n_j = \left((j-1)*q_s + 1\right),\ m_j = (j*q_s) \\ x_{m_j}(k+1) = h_j(\psi(k+q_s)) + \sum_{i=1}^{p} w^8_{ji}(k+q_s) u_i(k+q_s-1) \\ \hat{y}_j(k) = x_{n_j}(k) \end{cases} \quad (12)$$

Where:

$$\begin{cases} n = p * q_s \\ x(k) = \left[x_1(k), \text{K}\ , x_n(k) \right]^T \in \mathbb{R}^n \end{cases} \quad (13)$$

We can write equation (12) as follows:

$$\begin{cases} x(k+1) = Ax(k) + B\left[H(k+q_s) + g(k+q_s) u(k+q_s-1) \right] \\ \hat{y}(k) = C^T x(k) \end{cases} \quad (14)$$

With:

$$\begin{cases} A = \begin{bmatrix} M_1 & 0 & \text{L} & 0 \\ 0 & \text{O} & \text{O} & \text{M} \\ \text{M} & \text{O} & \text{O} & 0 \\ 0 & \text{L} & 0 & M_p \end{bmatrix} \\ A \in \mathbb{R}^{n \times n} \\ \forall j = 1, \text{L}\ , p \\ M_j = \begin{bmatrix} 0 & 1 & 0 & \text{L} & 0 \\ 0 & 0 & 1 & \text{O} & \text{M} \\ \text{M} & \text{O} & \text{O} & \text{O} & 0 \\ 0 & \text{L} & 0 & 0 & 1 \\ 0 & \text{L} & 0 & \text{L} & 0 \end{bmatrix} \\ M_j \in \mathbb{R}^{q_s \times q_s} \end{cases} \quad (15)$$

$$\begin{cases} B = \begin{bmatrix} b_1 & 0 & \text{L} & 0 \\ 0 & \text{O} & \text{O} & \text{M} \\ \text{M} & \text{O} & \text{O} & 0 \\ 0 & \text{L} & 0 & b_p \end{bmatrix} \\ B \in \mathbb{R}^{n \times p} \\ \forall j = 1, \text{L}\ p \\ b_j = \begin{bmatrix} 0 \\ \text{M} \\ 0 \\ 1 \end{bmatrix} \\ b_j \in \mathbb{R}^{q_s} \end{cases} \quad (16)$$

$$\begin{cases} C = \begin{bmatrix} c_1 & 0 & \text{L} & 0 \\ 0 & \text{O} & \text{O} & \text{M} \\ \text{M} & \text{O} & \text{O} & 0 \\ 0 & \text{L} & 0 & c_p \end{bmatrix} \\ C \in \mathbb{R}^{n \times p} \\ \forall j = 1, \text{L}\ , p \\ c_j = \begin{bmatrix} 1 \\ 0 \\ \text{M} \\ 0 \end{bmatrix} \\ c_j \in \mathbb{R}^{q_s} \end{cases} \quad (17)$$

$$\begin{cases} H(k+q_s)=\begin{bmatrix} h_1\left(\psi(k+q_s)\right) \\ \text{M} \\ h_p\left(\psi(k+q_s)\right) \end{bmatrix} \\ H(k+q_s)\in ¡^{\,p} \\ \forall j=1,\text{L}\ ,p \\ h_j\left(\psi(k+q_s)\right)\in ¡ \end{cases} \quad (18)$$

$$\begin{cases} g\left(k+q_s\right)=\begin{bmatrix} w^8{}_{11}\left(k+q_s\right) & \text{L} & w^8{}_{1p}\left(k+q_s\right) \\ \text{M} & \text{O} & \text{M} \\ w^8{}_{p1}\left(k+q_s\right) & \text{L} & w^8{}_{pp}\left(k+q_s\right) \end{bmatrix} \\ g(k+q_s)\in ¡^{\,p\times p} \\ \forall i=1,\text{L}\ ,p \quad j=1,\text{L}\ ,p \\ w^8{}_{ij}\left(k+q_s\right)\in ¡ \end{cases} \quad (19)$$

Define the modeling errors by:

$$e_i\left(k\right)=y_i\left(k\right)-\hat{y}_i(k),\quad i=1,\text{L}\ \ p \quad (20)$$

and the modeling errors vector of all states is defined by:

$$e\left(k\right)=y_y\left(k\right)-x\left(k\right) \quad (21)$$

with:

$$y_y\left(k\right)=\left[y_1\left(k\right),\text{L}\ ,y_1\left(k+q_s-1\right),\text{L}\ ,y_j\left(k\right),\text{L}\ ,y_j\left(k+q_s-1\right),\text{L}\ y_p\left(k\right),\text{L}\ ,y_p\left(k+q_s-1\right)\right]^T\in ¡^{\,n}$$

$$,\ 1<j<p \quad (22)$$

Combining (14) and (21), the dynamic of modeling errors is then given by:

$$\begin{cases} e\left(k+1\right)=Ae\left(k\right)+B\left[-H\left(k+q_s\right)-g\left(k+q_s\right)u\left(k+q_s-1\right)+y\left(k+q_s\right)\right] \\ e_{moo}\left(k\right)=C^Te\left(k\right) \end{cases} \quad (23)$$

where :

$$e_{moo}\left(k\right)=\left[e_1\left(k\right),\text{L}\ ,e_p\left(k\right)\right]^T \quad (24)$$

Normalization techniques of data

There are two techniques of normalization:

-Min-Max normalization:

This technique performs a linear transformation on the original data so that all values are in the interval $\left[a,b\right]$. The formula of the normalization min-max is the following:

$$\Upsilon'=\frac{\Upsilon-\Upsilon_{\min}}{\Upsilon_{\max}-\Upsilon_{\min}}\left(b-a\right)+a \quad (25)$$

with :

Υ is the data value to normalize.

Υ' is the new data value after the normalization, $\Upsilon'\in\left[a,b\right]$.

$\Upsilon_{\min}$ and $\Upsilon_{\max}$ are respectively the minimum and the maximum of data value to normalize .

- Normalization by decimal scaling:

The data are normalized by the following formula:

$$\Upsilon'=\frac{\Upsilon}{10^{\kappa}} \quad (26)$$

where :

κ is the smallest integer such as $\max\left(\left|\Upsilon'\right|\right)<1$.

2.3 Learning algorithm of neural network

In this section, we propose a theorem 1 that can be used during the learning phase of a neural network.

Theorem 1:

The learning procedure of a neural network may be given by the following equation:

$$w(k+1)=\left(1-\frac{\varpi_0\gamma}{2\left(\eta+\sum_{i=1}^{p}\beta_i\left\|\frac{\partial\hat{y}_i(k)}{\partial w(k)}\right\|^2\right)}\right)w(k)+\frac{\varpi_0\sum_{i=1}^{p}\lambda_i\frac{\partial\hat{y}_i(k)}{\partial w(k)}e_i(k)}{2\left(\eta+\sum_{i=1}^{p}\beta_i\left\|\frac{\partial\hat{y}_i(k)}{\partial w(k)}\right\|^2\right)}+\sum_{i=1}^{m}\varpi_i\Delta w(k-i) \quad (27)$$

such as :

$$m\geq 1 \quad (28)$$

$$\varpi_i>0\ \ \forall i\in\left[0,m\right] \quad (29)$$

$$\sum_{i=0}^{m}\varpi_i=1 \quad (30)$$

$$\gamma>0 \quad (31)$$

$$\eta>0 \quad (32)$$

$$0<\lambda_i\ \forall i\in\left[1,p\right] \quad (33)$$

$$\beta_i>0\ \ \forall i\in\left[1,p\right] \quad (34)$$

$$\gamma<\eta \quad (35)$$

$$\sum_{i=1}^{p}\lambda_i<\eta \quad (36)$$

$\|\ \|$ the Euclidean norm.

T designates the transpose operator.

Proof:

Using the following Lyapunov function:

$$V\left(k\right)=\sum_{i=1}^{p}\frac{\lambda_i}{2}\left(e_i\left(k\right)\right)^2+\sum_{i=1}^{p}\frac{\beta_i}{2}\left(\Delta e_i\left(k\right)\right)^2+\frac{\gamma}{2}\left\|w\left(k\right)\right\|^2+\frac{\eta}{2}\left\|\Delta w\left(k\right)\right\|^2 \quad (37)$$

with :

$$\Delta e_i\left(k\right)=e_i\left(k\right)-e_i\left(k-1\right) \quad (38)$$

$$\Delta w\left(k\right)=w\left(k\right)-w\left(k-1\right) \quad (39)$$

The learning procedure of the neural network is stable if:

$$\Delta V\left(k\right)=\sum_{i=1}^{p}\lambda_i\left(e_i\left(k\right)\right)\left(\Delta e_i\left(k\right)\right)+\sum_{i=1}^{p}\beta_i\left(\Delta e_i\left(k\right)\right)^2+\gamma\left(w\left(k\right)\right)\left(w\left(k\right)\right)^T+\eta\left\|\Delta w\left(k\right)\right\|^2\leq 0 \quad (40)$$

Using the previous equation, we can write:

$$\Delta V\left(k\right)=\sum_{i=1}^{p}\lambda_i\left(e_i\left(k\right)\right)\left(\Delta e_i\left(k\right)\right)+\sum_{i=1}^{p}\beta_i\left(\Delta e_i\left(k\right)\right)^2+\gamma\left(\Delta w\left(k\right)\right)^T\left(w\left(k\right)\right)+\eta\left\|\Delta w\left(k\right)\right\|^2=-\alpha_1 \quad (41)$$

Such as: $\alpha_1\geq 0$.

Therefore:

$$\left\|\Delta w(k)\right\|^2\left(\eta+\sum_{i=1}^{p}\beta_i\left\|\frac{\partial e_i(k)}{\partial w(k)}\right\|^2\right)+\left(\Delta w(k)\right)^T\left(\gamma w(k)+\sum_{i=1}^{p}\lambda_i\left(e_i(k)\right)\left(\frac{\partial e_i(k)}{\partial w(k)}\right)\right)+\alpha_1=0 \quad (42)$$

For equation (42) has a unique solution, it is necessary that:

$$\alpha_1 = \frac{\left\| \gamma w(k) + \sum_{i=1}^{p} \lambda_i \left(e_i(k)\right) \left(\frac{\partial e_i(k)}{\partial w(k)} \right) \right\|^2}{4\left(\eta + \sum_{i=1}^{p} \beta_i \left\| \frac{\partial e_i(k)}{\partial w(k)} \right\|^2 \right)} \quad (43)$$

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

$$\Delta w(k) = -\frac{\left(\gamma w(k) + \sum_{i=1}^{p} \lambda_i \left(e_i(k)\right) \left(\frac{\partial e_i(k)}{\partial w(k)} \right) \right)}{2\left(\eta + \sum_{i=1}^{p} \beta_i \left\| \frac{\partial e_i(k)}{\partial w(k)} \right\|^2 \right)} \quad (44)$$

Like:

$$\lim_{z \to 1} \sum_{i=0}^{m} \varpi_i z^{-i-1} = 1 \quad (45)$$

We can write:

$$\begin{aligned} \Delta w(k+1) ; & \left(\sum_{i=0}^{m} \varpi_i z^{-i-1} \right) \Delta w(k+1) \\ & = \varpi_0 \Delta w(k) + \sum_{i=1}^{m} \varpi_i \Delta w(k-i) \\ & = \left(-\frac{\varpi_0 \gamma}{2\left(\eta + \sum_{i=1}^{p} \beta_i \left\| \frac{\partial e_i(k)}{\partial w(k)} \right\|^2 \right)} \right) w(k) - \frac{\varpi_0 \sum_{i=1}^{p} \lambda_i \left(\frac{\partial e_i(k)}{\partial w(k)} \right) \left(e_i(k)\right)}{2\left(\eta + \sum_{i=1}^{p} \beta_i \left\| \frac{\partial e_i(k)}{\partial w(k)} \right\|^2 \right)} + \sum_{i=1}^{m} \varpi_i \Delta w(k-i) \\ & = \left(-\frac{\varpi_0 \gamma}{2\left(\eta + \sum_{i=1}^{p} \beta_i \left\| \frac{\partial \hat{y}_i(k)}{\partial w(k)} \right\|^2 \right)} \right) w(k) + \frac{\varpi_0 \sum_{i=1}^{p} \lambda_i \left(\frac{\partial \hat{y}_i(k)}{\partial w(k)} \right) \left(e_i(k)\right)}{2\left(\eta + \sum_{i=1}^{p} \beta_i \left\| \frac{\partial \hat{y}_i(k)}{\partial w(k)} \right\|^2 \right)} + \sum_{i=1}^{m} \varpi_i \Delta w(k-i) \end{aligned} \quad (46)$$

therefore :

$$w(k+1) = \left(1 - \frac{\varpi_0 \gamma}{2\left(\eta + \sum_{i=1}^{p} \beta_i \left\| \frac{\partial \hat{y}_i(k)}{\partial w(k)} \right\|^2 \right)} \right) w(k) + \frac{\varpi_0 \sum_{i=1}^{p} \lambda_i \left(\frac{\partial \hat{y}_i(k)}{\partial w(k)} \right) \left(e_i(k)\right)}{2\left(\eta + \sum_{i=1}^{p} \beta_i \left\| \frac{\partial \hat{y}_i(k)}{\partial w(k)} \right\|^2 \right)} + \sum_{i=1}^{m} \varpi_i \Delta w(k-i) \quad (47)$$

The choice of the initial synaptic weights and the bias can influence the convergence speed of the learning algorithm of neural network [35-46]. According to [47], the weights can be initialized by a random number generator with a uniform distribution between $-\theta$ and θ or a normal distribution $N\left(0, \theta^2\right)$.- If the weights are initialized by a random number generator with a uniform distribution: $N\left(0, \theta^2\right)$.

- If the weights are initialized by a random number generator with a uniform distribution:

$$\theta \le \bar{s} \sqrt{\frac{3}{(n_r + 1)\left(1 + \sum_{m=1}^{n_r} \left(\psi_m(0) \right)^2 \right)}} \quad (48)$$

- If the weights are initialized by a random number generator with a normal distribution:

$$\theta \le \bar{s} \sqrt{\frac{1}{(n_r + 1)\left(1 + \sum_{m=1}^{n_r} \left(\psi_m(0) \right)^2 \right)}} \quad (49)$$

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

Validation tests of the neural model

Most validation tests use a set of samples inputs-outputs which have not been used in learning. Such a test set or validation should if possible cover the same range operating as the set of training samples.

The residues $e_i(k)$, $i = 1, \mathrm{L}\ , p$ obtained from the estimated model parameters represent non-measurable disturbance presented within the system. The residues must constitute independent random sequences thus assimilating the prediction errors to white noise. Various tests called whiteness tests residues were developed to validate this property. These validation tests of a model are based on analysis of prediction errors, on the Nash-Sutcliffe criterion, on the auto-correlation of residues, on the cross-correlation function between the residues and other inputs in the system [31, 48-52].

The Nash-Sutcliffe criterion relating to on each output is given by the following relation:

$$Q_i = 100\% \left(1 - \frac{\sum_{k=1}^{N} \left(y_i(k) - \hat{y}_i(k) \right)^2}{\sum_{k=1}^{N} \left(y_i(k) - \left(\frac{1}{N} \sum_{k=1}^{N} y_i(k) \right) \right)^2} \right) \qquad i = 1, \mathrm{L}\ , p \quad (50)$$

The correlation functions are:

- Autocorrelation functions of the residues:

$$R_{e_i e_i}(\tau) = \frac{\sum_{k=1}^{N-\tau} \left(e_i(k) - \left(\frac{1}{N} \sum_{k=1}^{N} e_i(k) \right) \right) \left(e_i(k-\tau) - \left(\frac{1}{N} \sum_{k=1}^{N} e_i(k) \right) \right)}{\sum_{k=1}^{N} \left(e_i(k) - \left(\frac{1}{N} \sum_{k=1}^{N} e_i(k) \right) \right)^2} \qquad i = 1, \mathrm{L}\ , p \quad (51)$$

- Cross-correlation function between the residues and previous inputs:

$$R_{u_i e_j}(\tau) = \frac{\sum_{k=1}^{N-\tau} \left(u_i(k) - \left(\frac{1}{N} \sum_{k=1}^{N} u_i(k) \right) \right) \left(e_j(k-\tau) - \left(\frac{1}{N} \sum_{k=1}^{N} e_j(k) \right) \right)}{\sqrt{\sum_{k=1}^{N} \left(u_i(k) - \left(\frac{1}{N} \sum_{k=1}^{N} u_i(k) \right) \right)^2} \sqrt{\sum_{k=1}^{N} \left(e_j(k) - \left(\frac{1}{N} \sum_{k=1}^{N} e_j(k) \right) \right)^2}} \qquad i = 1, \mathrm{L}\ , p \quad j = 1, \mathrm{L}\ , p \quad (52)$$

N is the number of samples.

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

$$R_{e_i e_i}(\tau) = \begin{cases} 1, \tau = 0 \\ 0, \tau \neq 0 \end{cases}, \ R_{u_i e_j}(\tau) = 0 \ \forall \tau \ \text{ and } \ Q_i = 100\% \ ,$$

$$i = 1, \mathrm{L}\ , p \ , \ j = 1, \mathrm{L}\ , p \quad (53)$$

Typically, it is verified that $Q_i \cong 100\%$ and the functions R are null for the interval $\tau \in [-20, 20]$ with a confidence interval 95%, that is to say:

$$-\frac{1.96}{\sqrt{N}} \le R \le \frac{1.96}{\sqrt{N}} \quad (54)$$

- Average error on each output is defined as follows:

$$ARE_i = \frac{1}{N} \sum_{k=1}^{N} e_i(k), \quad i = 1, \mathrm{L}\ , p \quad (55)$$

- Mean absolute error on each output is:

$$AARE_i = \frac{1}{N} \sum_{k=1}^{N} \left| e_i(k) \right|, \quad i = 1, \mathrm{L}\ , p \quad (56)$$

- Root mean square error on each output:

$$RMS_i = \sqrt{\frac{1}{N}\sum_{k=1}^{N}\left(e_i(k)\right)^2}, \quad i = 1,\mathrm{L}\ , p \tag{57}$$

- MAE (Mean Absolute Error):

$$MEA = \frac{1}{(p*N)}\sum_{i=1}^{p}\sum_{k=1}^{N}\left|e_i(k)\right| \tag{58}$$

The desired value of ARE_i , $AARE_i$, RMS_i **and** MEA is zero.

Neuronal indirect adaptive control

In this section, we propose a neuronal indirect adaptive control structure of a complex dynamic multivariable system (with

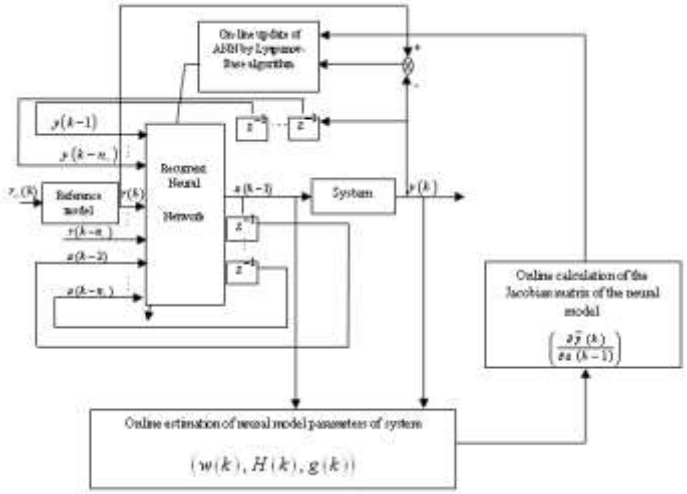


Figure 2. Structure of the neuronal indirect adaptive control

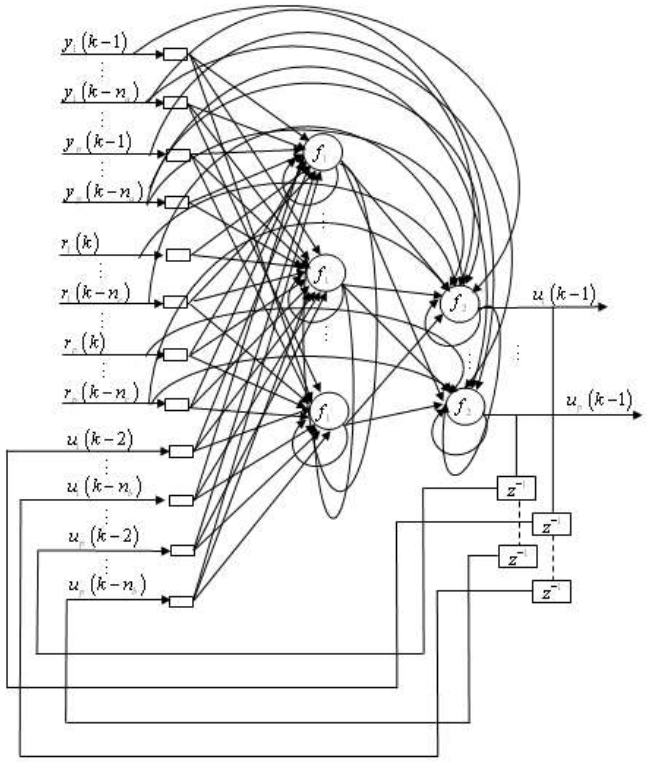


Figure 3- Proposed architecture of the neural controller

Learning algorithm of neural controller

The learning algorithm of neural controller of a nonlinear complex uncertain and perturbed multivariate system (with p inputs and p outputs) can use Theorem 2.

Theorem 2:

The learning of the neural controller can be made by the following equation:

$$wc(k+1) = \left(1 - \frac{\varpi c_0 \gamma c}{2\left(\eta c + \sum_{i=1}^{p}\beta c_i \left\|\frac{\partial \hat{y}_i(k)}{\partial wc(k)}\right\|^2 + \sum_{i=1}^{p}\mu c_i \left\|\frac{\partial u_i(k-1)}{\partial wc(k)}\right\|^2\right)}\right) wc(k) + \frac{\varpi c_0 \sum_{i=1}^{p}\lambda c_i \frac{\partial \hat{y}_i(k)}{\partial w(k)} ec_i(k)}{2\left(\eta c + \sum_{i=1}^{p}\beta c_i \left\|\frac{\partial \hat{y}_i(k)}{\partial wc(k)}\right\|^2 \sum_{i=1}^{p}\mu c_i \left\|\frac{\partial u_i(k-1)}{\partial wc(k)}\right\|^2\right)} + \sum_{i=1}^{mc}\varpi c_i \Delta wc(k-i) \tag{59}$$

Such as:

$$mc \geq 1 \tag{60}$$

$$\varpi c_i > 0 \ \ \forall i \in [0, mc] \tag{61}$$

$$\sum_{i=0}^{mc}\varpi c_i = 1 \tag{62}$$

$$\gamma c > 0 \tag{63}$$

$$\eta c > 0 \tag{64}$$

$$0 < \lambda c_i \ \forall i \in [1, p] \tag{65}$$

$$\beta c_i > 0 \ \ \forall i \in [1, p] \tag{66}$$

$$\gamma c < \eta c \tag{67}$$

$$\sum_{i=1}^{p}\lambda c_i < \eta c \tag{68}$$

$$\mu c_i \geq 0 \ \forall i \in [1, p] \tag{69}$$

$$ec_i(k) = r_i(k) - y_i(k) \qquad \forall i \in [1, p] \tag{70}$$

$$n_{rc} = p\left(n_a + n_b + n_c\right) \tag{71}$$

$$wc = \Big[wc^1{}_{11}, \mathrm{L}\ , wc^1{}_{n_{hc} n_{rc}}, wc^2{}_{11}, \mathrm{L}\ , wc^2{}_{n_{hc} 1}, wc^3{}_{11}, \mathrm{L}\ , wc^3{}_{pn_{hc}}, wc^4{}_{11}, \mathrm{L}\ , wc^4{}_{p1}, wc^5{}_{11}, \mathrm{L}\ , wc^5{}_{pn_{rc}}, wc^6{}_{11}, \mathrm{L}\ , wc^6{}_{n_{hc} n_{hc}}, wc^7{}_{11}, \mathrm{L}\ wc^7{}_{pp} \Big]^T$$

is the weights vector of neural controller.

n_{hc} is the number of neurons in the hidden layer of neural controller.

Proof:

From the following Lyapunov function:

$$V(k) = \sum_{i=1}^{n_3}\frac{\lambda c_i}{2}\left(ec_i(k)\right)^2 + \sum_{i=1}^{n_3}\frac{\beta c_i}{2}\left(\Delta ec_i(k)\right)^2 + \frac{\gamma c}{2}\left\|wc(k)\right\|^2 + \frac{\eta c}{2}\left\|\Delta wc(k)\right\|^2 + \sum_{i=1}^{n_3}\frac{\mu c_i}{2}\left(\Delta u_i(k-1)\right)^2 \tag{72}$$

The adjustment parameters procedure of the neural controller is stable if:

$$\Delta V(k) = \sum_{i=1}^{n_3}\lambda c_i\left(ec_i(k)\right)\left(\Delta e_i(k)\right) + \sum_{i=1}^{n_3}\beta c_i\left(\Delta ec_i(k)\right)^2 + \gamma c\left(\Delta wc(k)\right)^T\left(wc(k)\right) + \eta c\left\|\Delta wc(k)\right\|^2 + \sum_{i=1}^{n_3}\mu c_i\left(\Delta u_i(k-1)\right)^2 = -\alpha_2 \tag{73}$$

with :

$$\alpha_2 \geq 0$$

Equation (73) then becomes:

$$\|\Delta wc(k)\|^2\left(\eta c+\sum_{i=1}^{n_s}\beta c_i\left\|\frac{\partial ec_i(k)}{\partial wc(k)}\right\|^2+\sum_{i=1}^{n_u}\mu c_i\left\|\frac{\partial u_i(k-1)}{\partial wc(k)}\right\|^2\right)+(\Delta wc(k))^T\left(\gamma c(wc(k))+\sum_{i=1}^{n_s}\lambda c_i(ec_i(k))\left(\frac{\partial ec_i(k)}{\partial wc(k)}\right)\right)+\alpha_2=0 \quad (74)$$

If the previous equation has a unique solution, the term α_2 is as follows:

$$\alpha_2=\frac{\left\|\sum_{i=1}^{n_s}\lambda c_i(ec_i(k))\left(\frac{\partial ec_i(k)}{\partial wc(k)}\right)+\gamma c(wc(k))\right\|^2}{4\left(\eta c+\sum_{i=1}^{n_s}\beta c_i\left\|\frac{\partial ec_i(k)}{\partial wc(k)}\right\|^2+\sum_{i=1}^{n_u}\mu c_i\left\|\frac{\partial u_i(k-1)}{\partial wc(k)}\right\|^2\right)} \quad (75)$$

The adjustment parameters equation of the controller neural can be written:

$$\Delta wc(k)=-\frac{\left(\gamma c(wc(k))+\sum_{i=1}^{n_s}\lambda c_i(ec_i(k))\left(\frac{\partial ec_i(k)}{\partial wc(k)}\right)\right)}{2\left(\eta c+\sum_{i=1}^{n_s}\beta c_i\left\|\frac{\partial ec_i(k)}{\partial wc(k)}\right\|^2+\sum_{i=1}^{n_u}\mu c_i\left\|\frac{\partial u_i(k-1)}{\partial wc(k)}\right\|^2\right)} \quad (76)$$

Like:

$$\lim_{z\to 1}\sum_{i=0}^{m}\varpi c_i z^{-i-1}=1 \quad (77)$$

We can write:

$$\Delta wc(k+1);\left(\sum_{i=0}^{m}\varpi c_i z^{-i-1}\right)\Delta wc(k+1)=-\frac{\varpi c_0\left(\gamma c(wc(k))+\sum_{i=1}^{n_s}\lambda c_i(ec_i(k))\left(\frac{\partial ec_i(k)}{\partial wc(k)}\right)\right)}{2\left(\eta c+\sum_{i=1}^{n_s}\beta c_i\left\|\frac{\partial ec_i(k)}{\partial wc(k)}\right\|^2+\sum_{i=1}^{n_u}\mu c_i\left\|\frac{\partial u_i(k-1)}{\partial wc(k)}\right\|^2\right)}+\sum_{i=1}^{m}\varpi c_i\Delta wc(k-i) \quad (78)$$

therefore :

$$wc(k+1)=wc(k)+\Delta wc(k+1)=\left(1-\frac{\varpi c_0\gamma c}{2\left(\eta c+\sum_{i=1}^{n_s}\beta c_i\left\|\frac{\partial ec_i(k)}{\partial wc(k)}\right\|^2+\sum_{i=1}^{n_u}\mu c_i\left\|\frac{\partial u_i(k-1)}{\partial wc(k)}\right\|^2\right)}\right)wc(k)-\frac{\varpi c_0\left(\sum_{i=1}^{n_s}\lambda c_i(ec_i(k))\left(\frac{\partial ec_i(k)}{\partial wc(k)}\right)\right)}{2\left(\eta c+\sum_{i=1}^{n_s}\beta c_i\left\|\frac{\partial ec_i(k)}{\partial wc(k)}\right\|^2+\sum_{i=1}^{n_u}\mu c_i\left\|\frac{\partial u_i(k-1)}{\partial wc(k)}\right\|^2\right)}+\sum_{i=1}^{m}\varpi c_i\Delta wc(k-i) \quad (79)$$

If all conditions are met proper identification, the neural model outputs $\hat{y}(k)$ are good approximations of the system outputs $y(k)$, which allow writing:

$$y_i(k);\ \hat{y}_i(k)\quad \forall i=1,\mathrm{L}\ ,p \quad (80)$$

Equation (79) then becomes:

$$wc(k+1)=\left(1-\frac{\varpi c_0\gamma c}{2\left(\eta c+\sum_{i=1}^{n_s}\beta c_i\left\|\frac{\partial \hat{y}_1(k)}{\partial wc(k)}\right\|^2+\sum_{i=1}^{n_u}\mu c_i\left\|\frac{\partial u_i(k-1)}{\partial wc(k)}\right\|^2\right)}\right)wc(k)+\frac{\varpi c_0\left(\sum_{i=1}^{n_s}\lambda c_i(ec_i(k))\left(\frac{\partial \hat{y}_1(k)}{\partial wc(k)}\right)\right)}{2\left(\eta c+\sum_{i=1}^{n_s}\beta c_i\left\|\frac{\partial \hat{y}_1(k)}{\partial wc(k)}\right\|^2+\sum_{i=1}^{n_u}\mu c_i\left\|\frac{\partial u_i(k-1)}{\partial wc(k)}\right\|^2\right)}+\sum_{i=1}^{m}\varpi c_i\Delta wc(k-i) \quad (81)$$

The calculation of the term $\frac{\partial \hat{y}_i(k)}{\partial wc(k)}$ can be determined through the neural model as follows:

$$\begin{bmatrix}\frac{\partial \hat{y}_1(k)}{\partial wc(k)}\\ \mathrm{M}\\ \frac{\partial \hat{y}_p(k)}{\partial wc(k)}\end{bmatrix}=\begin{bmatrix}\frac{\partial \hat{y}_1(k)}{\partial u_1(k-1)} & \mathrm{L} & \frac{\partial \hat{y}_1(k)}{\partial u_p(k-1)}\\ \mathrm{M} & \mathrm{O} & \mathrm{M}\\ \frac{\partial \hat{y}_p(k)}{\partial u_1(k-1)} & \mathrm{L} & \frac{\partial \hat{y}_p(k)}{\partial u_p(k-1)}\end{bmatrix}\begin{bmatrix}\frac{\partial u_1(k-1)}{\partial wc(k)}\\ \mathrm{M}\\ \frac{\partial u_p(k-1)}{\partial wc(k)}\end{bmatrix} \quad (82)$$

We can write:

$$\frac{\partial \hat{y}_i(k)}{\partial u_j(k-1)};\ \frac{\hat{y}_i(k)-\hat{y}_i(k-1)}{u_j(k-1)-u_j(k-2)}\quad ,i=1,\mathrm{K}\ ,p,\ j=1,\mathrm{K}\ ,p \quad (83)$$

Robust neuronal adaptive control

The structure of the proposed robust neural adaptive control is given in Figure 4.

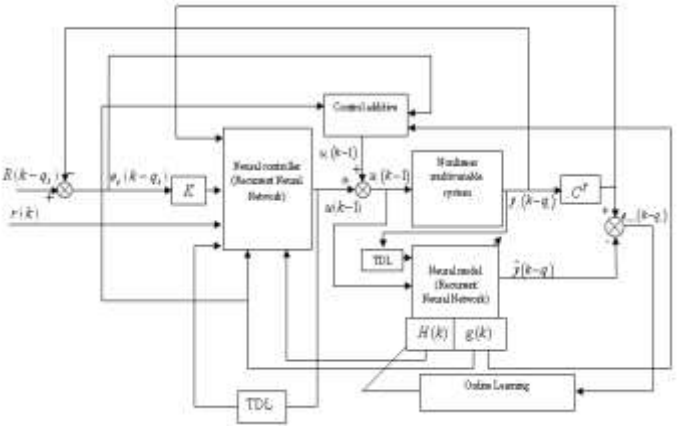


Figure 4. Proposed structure of robust neuronal adaptive control

With:

TDL : designates delays.

The vectors of reference signals are:

$$r(k)=\left[r_1(k),\mathrm{K}\ ,r_p(k)\right]^T \quad (84)$$

$$R(k)=\left[r_1(k),\mathrm{K},r_1(k+q_s-1),\mathrm{K},r_j(k),\mathrm{K},r_j(k+q_s-1),\mathrm{K},r_p(k),\mathrm{K},r_p(k+q_s-1)\right]^T,\ 1<j<p \quad (85)$$

The command applied to the system is given by:

$$u_c(k-1)=u(k-1)+u_r(k-1) \quad (86)$$

The architecture of the neuronal controller (Figure 5) is deduced from the architecture of the neural model (Figure 1).

From equation (23), we can write:

$$u(k+q_s-1)=\left(g(k+q_s)\right)^{-1}\left(-H(k+q_s)-Ke(k)+y(k+q_s)\right) \quad (87)$$

For equation (87) is realizable, the matrix $g(k)$ must be invertible.

Such as:

The matrix $K=\left[K_1,\mathrm{L}\ ,K_p\right]^T\in \mathbb{R}^{p\times n}$ is calculated so that the matrix $(A-BK)$ has all its eigenvalues strictly less than 1.

The neural controller equation is then:

$$u(k-1)=\left(g(k)\right)^{-1}\left(r(k)-H(k)-Ke_c(k-q_s)\right) \quad (88)$$

It is also assumed:

$$\left(g(k)\right)^{-1}=\left[G_1(k),\mathrm{L}\ ,G_p(k)\right]^T \quad (89)$$

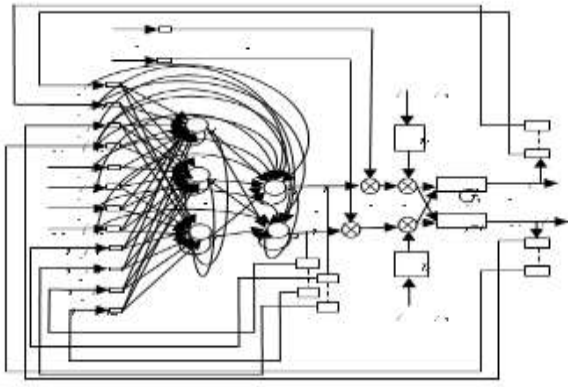


Figure 5- Proposed Architecture of the neural controller

In reality, there are always modeling errors during the identification phase and disturbance which may affect the system. The equation of the dynamics of tracking errors without the additive component of control may be as follows:

$$e_c(k+1-q_s)=(A-BK)e_c(k-q_s)+B\left[\varepsilon_l(k-q_s)\right] \quad (90)$$

With $\varepsilon_l(k)$ represents the set of disturbances and uncertainty estimates

$$\varepsilon_l(k);\ e_{moo}(k) \quad (91)$$

If we put:

$$A_0=A-BK \quad (92)$$

$$e_r(k)=C^Te_c(k) \quad (93)$$

The transfer function between the term $\varepsilon_l(z)$ and prosecution errors $e_r(z)$ is as follows:

$$H_{e_r,\varepsilon_l}(z)=C^T\left(zI-A_0\right)^{-1}B \quad (94)$$

Theorem 3:

The additive component u_r can be calculated using techniques based on optimization H_∞. It can be given by:

$$u_r(k-1)=-\left(g(k)\right)^{-1}\frac{1}{\alpha_3}B^TP\left(e_c(k-q_c)\right) \quad (95)$$

where :

$$\left\|H_{e_r,\varepsilon_l}(z)\right\|_\infty\le\gamma_2 \qquad \text{such as } \gamma_2>0 \quad (96)$$

α_3 is a positive scalar and P a symmetric positive definite matrix verifying the following Riccati equation:

$$P(A_0-I)+(A_0-I)^TP+CC^T-\left(\frac{1}{(\gamma_2)^2}-\frac{2}{\alpha_3}\right)PBB^TP+Q=0 \quad (97)$$

With: $Q>0$ and $\dfrac{1}{(\gamma_2)^2}-\dfrac{2}{\alpha_3}\le 0$

***Proof*:**

The dynamic of the system prosecution errors with the additive command u_r is in the following form:

$$\begin{aligned}e_c(k+1-q_s)&=(A-BK)e_c(k-q_s)+B\left[\varepsilon_l(k-q_s)+g(k)u_r(k-1)\right]\\&=A_0\,e_c(k-q_s)+B\,\varepsilon_l(k-q_s)+Bg(k)u_r(k-1)\end{aligned} \quad (98)$$

To verify the stability of the system, define the following candidate Lyapunov function:

$$V(k)=\frac{1}{2}\left(e_c(k-q_s)\right)^TP\left(e_c(k-q_s)\right) \quad (99)$$

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

$$\begin{aligned}\Delta V(k)&=\frac{1}{2}\left(\Delta e_c(k-q_s)\right)^TP\left(e_c(k-q_s)\right)+\frac{1}{2}\left(e_c(k-q_s)\right)^TP\left(\Delta e_c(k-q_s)\right)\\&=\frac{1}{2}\left(e_c(k-q_s)\right)^T\left[(A_0-I)^TP+P(A_0-I)\right]\left(e_c(k-q_s)\right)+\left(e_c(k-q_s)\right)^TPB\left[\varepsilon_l(k-q_s)+g(k)u_r(k-1)\right]\end{aligned} \quad (100)$$

Using the Riccati equation (97), we obtain:

$$\begin{aligned}\Delta V(k)=&-\frac{1}{2}\left(e_c(k-q_s)\right)^TQ\left(e_c(k-q_s)\right)-\frac{1}{2}\left(e_c(k-q_s)\right)^T\left(\frac{1}{(\gamma_2)^2}-\frac{2}{\alpha_3}\right)PBB^TP\left(e_c(k-q_s)\right)\\&-\frac{1}{2}\left(e_c(k-q_s)\right)^TCC^T\left(e_c(k-q_s)\right)+\left(e_c(k-q_s)\right)^TPBg(k)u_r(k-1)\\&+\left(e_c(k-q_s)\right)^TPB\left(\varepsilon_l(k-q_s)\right)\end{aligned} \quad (101)$$

Replacing the value of $u_r(k-1)$ in (95), equation (101) becomes:

$$\begin{aligned}\Delta V(k)\le&-\frac{1}{2}\left(e_c(k-q_s)\right)^TQ\left(e_c(k-q_s)\right)-\frac{1}{2(\gamma_2)^2}\left(e_c(k-q_s)\right)^TPBB^TP\left(e_c(k-q_s)\right)\\&-\frac{1}{2}\left(e_c(k-q_s)\right)^TCC^T\left(e_c(k-q_s)\right)+\left(e_c(k-q_s)\right)^TPB\left(\varepsilon_l(k-q_s)\right)\end{aligned} \quad (102)$$

If we assume:

$$\begin{aligned}a=&-\frac{1}{2}\left(e_c(k-q_s)\right)^TQ\left(e_c(k-q_s)\right)-\frac{1}{2(\gamma_2)^2}\left(e_c(k-q_s)\right)^TPBB^TP\left(e_c(k-q_s)\right)\\&-\frac{1}{2}\left(e_c(k-q_s)\right)^TCC^T\left(e_c(k-q_s)\right)+\left(e_c(k-q_s)\right)^TPB\left(\varepsilon_l(k-q_s)\right)\\=&-\frac{1}{2}\left(e_c(k-q_s)\right)^TQ\left(e_c(k-q_s)\right)-\frac{1}{2}\left(e_c(k-q_s)\right)^TCC^T\left(e_c(k-q_s)\right)+\frac{1}{2}(\gamma_2)^2\left\|\varepsilon_l(k-q_s)\right\|^2\\&-\frac{1}{2}\left(\frac{1}{(\gamma_2)^2}\left(e_c(k-q_s)\right)^TPBB^TP\left(e_c(k-q_s)\right)-2\frac{\left(e_c(k-q_s)\right)^TPB}{\gamma_2}\gamma_2\left(\varepsilon_l(k-q_s)\right)+(\gamma_2)^2\left\|\varepsilon_l(k-q_s)\right\|^2\right)\end{aligned} \quad (103)$$

According to the previous equation, we can write:

$$\Delta V(k)\le a\le-\frac{1}{2}\left(e_c(k-q_s)\right)^TCC^T\left(e_c(k-q_s)\right)+\frac{1}{2}(\gamma_2)^2\left\|\varepsilon_l(k-q_s)\right\|^2 \quad (104)$$

then:

$$V(k)-V(0)\le-\frac{1}{2}\sum_{i=0}^{k}\left(e_c(i-q_s)\right)^TCC^T\left(e_c(i-q_s)\right)+\frac{1}{2}(\gamma_2)^2\sum_{i=0}^{k}\left\|\varepsilon_l(i-q_s)\right\|^2 \quad (105)$$

As $V(k)\ge 0$, in this case inequality (105) can be written as follows:

$$\frac{1}{2}\sum_{i=0}^{k}\left(e_c(i-q_s)\right)^TCC^T\left(e_c(i-q_s)\right)\le\frac{1}{2}(\gamma_2)^2\sum_{i=0}^{k}\left\|\varepsilon_l(i-q_s)\right\|^2+V(0) \quad (106)$$

If $V(0)=0$, we can write:

$$\frac{\sum_{i=0}^{k}\left\|e_r(i-q_s)\right\|^2}{\sum_{i=0}^{k}\left\|\varepsilon_l(i-q_s)\right\|^2}\le(\gamma_2)^2 \quad (107)$$

Finally, we can write:

$$\left\| H_{e_r,\varepsilon_l}(z) \right\|_{\infty} \le \gamma_2 \qquad (108)$$

Numerical results and discussion:

Either the nonlinear system described by the equations system:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = -\theta_6 x_2 - \theta_3 x_1 + \left(\theta_1 \sin(u_1) - \theta_2 x_1\right)^2 + \varepsilon_1 \\ \dot{x}_3 = x_4 \\ \dot{x}_4 = \left(\theta_5 + 2\dot{\theta}_5 + \ddot{\theta}_5 - \theta_5\theta_3\right)x_1 + \theta_5\left(\theta_1 \sin(u_1) - \theta_2 x_1\right)^2 + \left(2\dot{\theta}_5 + 2\theta_5 - \theta_5\theta_6\right)x_2 \\ \qquad - x_3 - 3x_4 + \left(\theta_4 + 2\dot{\theta}_4 + \ddot{\theta}_4\right)u_2 + \left(2\dot{\theta}_4 + 2\theta_4\right)\dot{u}_2 + \theta_4\ddot{u}_2 + \varepsilon_2 \\ y_1 = x_1 \\ y_2 = x_3 \end{cases} \qquad (109)$$

with :

u_1 and u_2 are the system inputs.

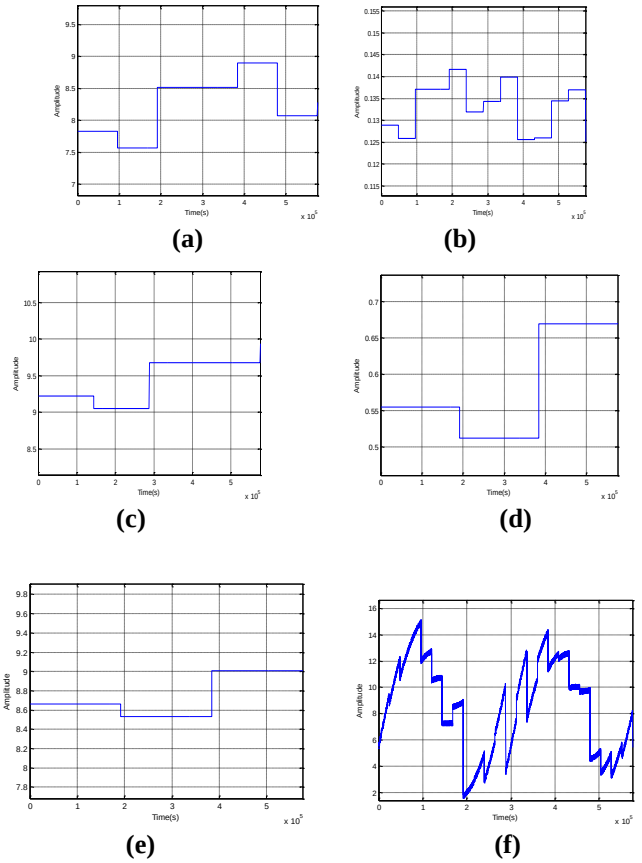
y_1 and y_2 are the system outputs.

ε_1 and ε_2 are noises such as:

$$|\varepsilon_1| \le \max(|y_1|) \qquad (110)$$

$$|\varepsilon_2| \le 10 \max(|y_1|) \qquad (111)$$

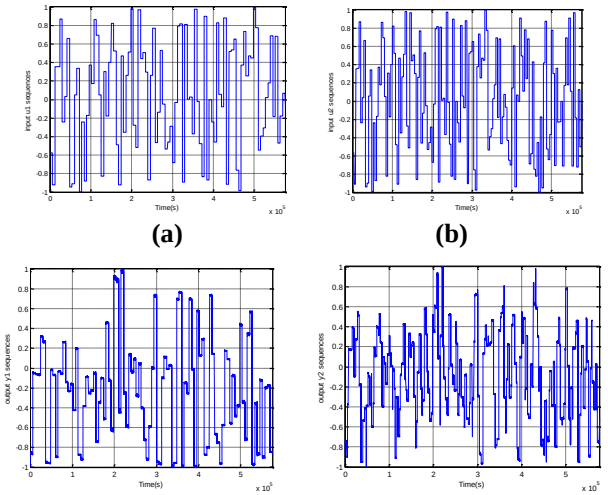
Figure 6 shows the evolution of the system parameters.



(a) (b) (c) (d) (e) (f)

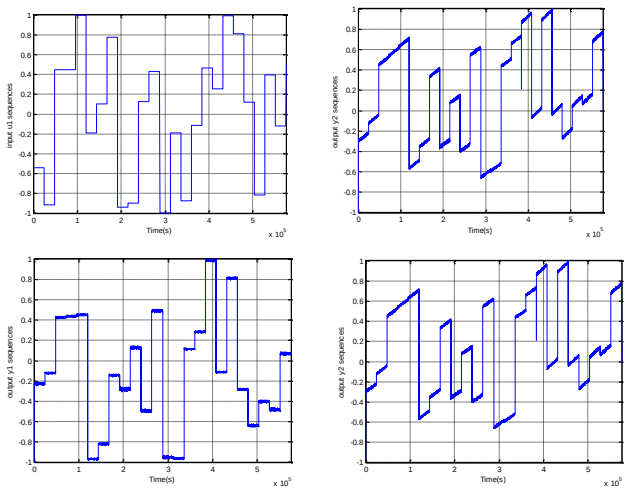
Figure 6 – Evolution of system parameters: (a) parameter θ_1 ; (b) parameter θ_2 ; (c) parameter θ_3 ; (d) parameter θ_4 ; (e) parameter θ_5 ; (f) parameter θ_6

The figures 7 and 8 represent respectively the training sequences and assessment performance sequences (or test sequences) which are normalized by the technical of Min-Max normalization. The neural model structure of the studied system is given in Figure 2. For online learning of this model we used Theorem 1. The maximum number of iterations is 1000 during this learning phase.



(a) (b) (c) (d)

Figure 7 – Training sequences: (a) control input u_1 ; (b) control input u_2 ; (c) desired output y_1 ; (d) desired output



(c) (d)

Tables 1 and 2 show the obtained test results from different candidate neural models. Note that to obtain a neural model Tables 1 and 2 show the of the system studied of a satisfactory accuracy, it requires that: $m = 2$, $n_{b1} = 2, n_{b2} = 2$, $n_{a1} = 2, n_{a2} = 2$, $n_{c1} = 1$, $n_{c2} = 1$, $n_h = 8$, $\lambda_1 = 0.7$, $\lambda_2 = 0.8$, $\beta_1 = 2$, $\beta_2 = 2.4$, $\eta = 1.8$, $\varpi_0 = 0.96$, $\varpi_1 = 0.025$, $p = 2$, $\gamma = 0.4$ and $\varpi_2 = 0.015$.

Tableau 1. Evolution of MEA of different candidate neural models in the case $m = 1$

The autocorrelation functions of residuals and cross-correlation functions between inputs and residues (figure 9) are within the confidence intervals, validating the use of neural network of characteristics ($n_{b1} = 2, n_{b2} = 2$, $n_{a1} = 2, n_{a2} = 2$, $n_{c1} = 1$, $n_{c2} = 1$, $n_h = 8$) as a model of the studied system.

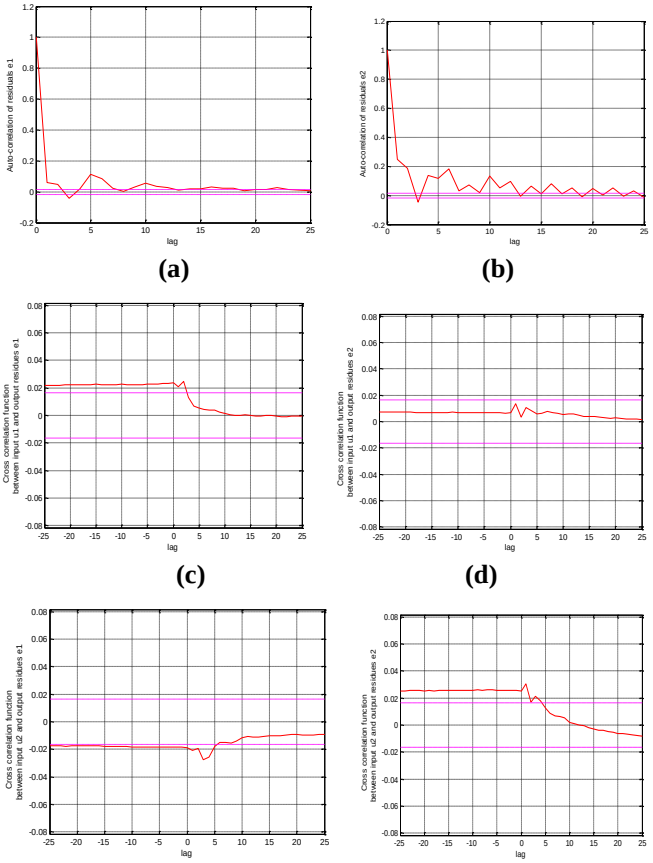


Figure 9 – Validation tests of the chosen neural model: (a) Autocorrelation function of the prediction error e_1; (b) Autocorrelation function of the prediction error e_2 ; (c) Cross-correlation function between the input u_1 and the residues e_1; (d) Cross-correlation function between the input u_1 and the residues e_2; (e) Cross-correlation function between the input u_2 and the residues e_1; (f) Cross-correlation function between the input u_2 and the residues e_2.

After the determination phase of a neural network capable to best approximate the desired relationships of inputs-outputs of the studied system, the proposed structures of neural adaptive control (indirect adaptive control and robust neural adaptive control by the technique H_∞) are applied to this system.

First, we will control the system by neural indirect adaptive control. The structure of this command is that presented in Figure 2. The architecture of the neural controller is that given by Figure 3. The online calculation procedure of the controller parameters uses the theorem 2. The maximum number of iterations is 1000 during the phase of the calculation of the parameters. The evolution of mean values of the absolute differences between reference signals and outputs of the system as a function to the parameters (n_a, n_b, n_c and n_{hc}) are **presented in Tables 3 and 4.**

From the results of these evolutions, the chosen neural controller has the characteristics:

$$\begin{cases} n_a = 2 \\ n_b = 2 \\ n_c = 1 \\ n_{hc} = 7 \end{cases} \quad (112)$$

and during learning of this controller, the chosen parameters (λc_1, λc_2, βc_1, βc_2, ηc, ϖc_0, ϖc_1, ϖc_2, m) are as follows:

$$\begin{cases} \lambda c_1 = 0.8 \\ \lambda c_2 = 0.9 \\ \beta c_1 = 2.1 \\ \beta c_2 = 2.7 \\ \eta c = 1.8 \\ \gamma c = 1.7 \\ m = 2 \\ \varpi c_0 = 0.96 \\ \varpi c_1 = 0.021 \\ \varpi c_2 = 0.019 \end{cases} \quad (113)$$

The average value of absolute differences between reference signals and the outputs of the system is given by the following equation:

$$VME = \frac{1}{p}\sum_{i=1}^{P}\left(\frac{1}{N}\sum_{k=1}^{N}\left|ec_i(k)\right|\right) = \frac{1}{p}\sum_{i=1}^{P}\left(\frac{1}{N}\sum_{k=1}^{N}\left|r_i(k)-y_i(k)\right|\right) \quad (114)$$

The results obtained by the proposed neural indirect adaptive control applied to the system are defined in Figures 10, 11 and 12.

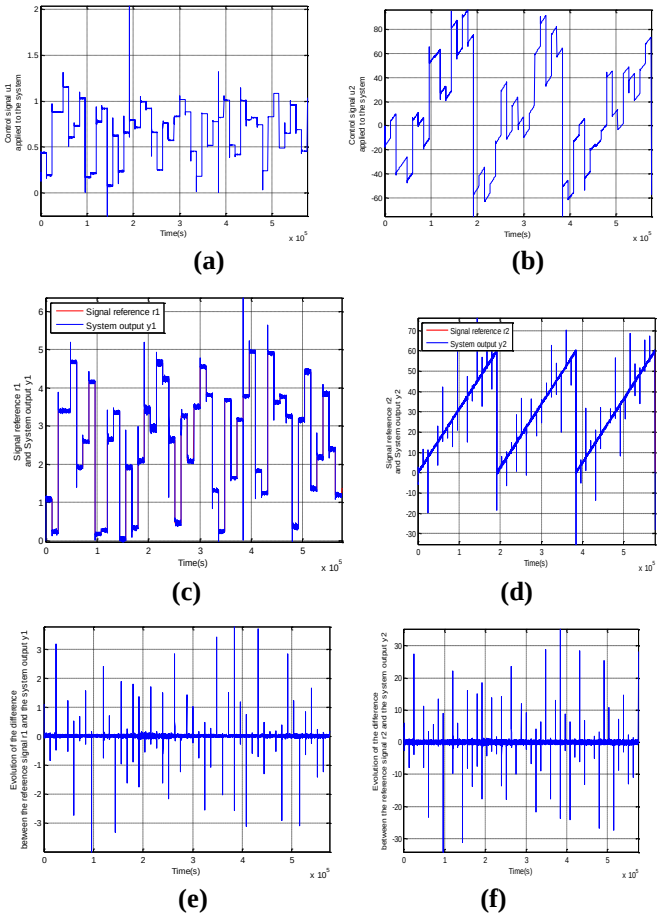
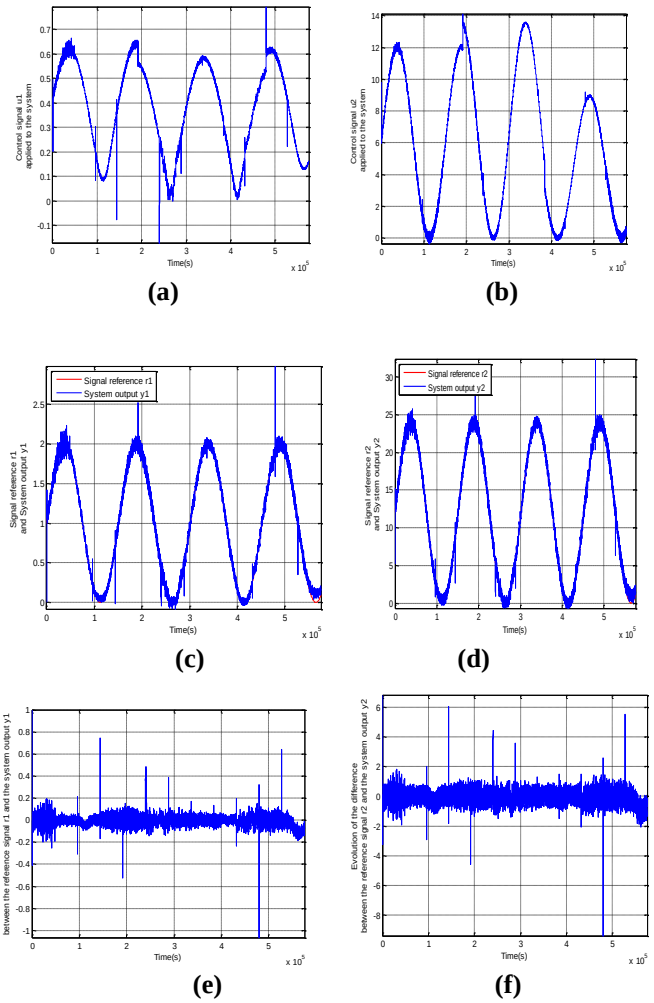
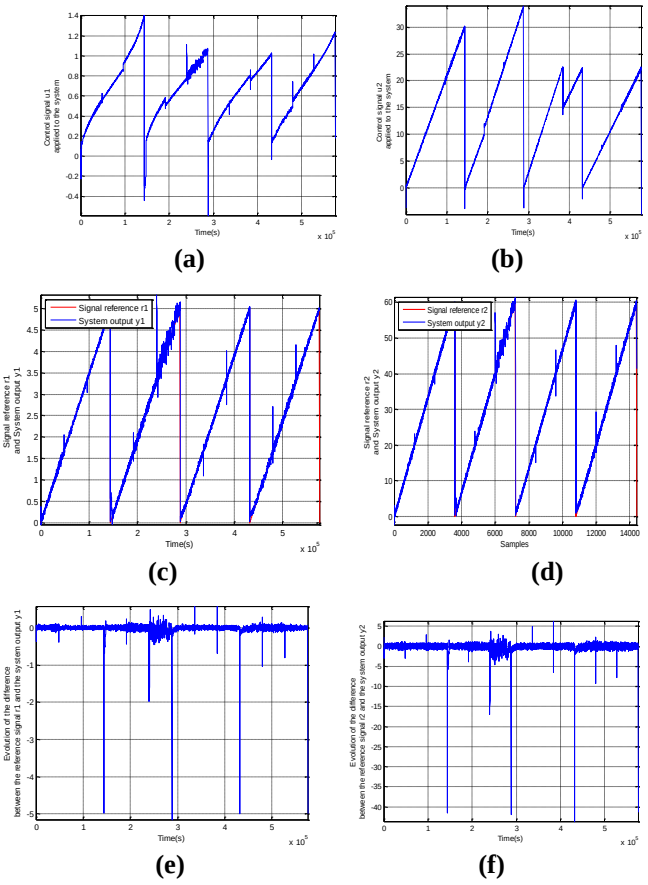


Figure 10 – Results of neuronal indirect adaptive control where r_1 and r_2 are respectively a signal with random uniformly distributed amplitudes and a triangular signal: (a) control signal u_1 applied to the system ; (b) control signal u_2 applied to the system; (c) reference signal r_1 and the system output y_1 ; (d) reference signal r_2 and the system output y_2 ; (e) Evolution of the difference between the reference signal r_1 and the system output y_1 ; (f) Evolution of the difference between the reference signal r_2 and the system output y_2 .



(a) **(b)** **(c)** **(d)** **(e)** **(f)**

Figure 11 – Results of neuronal indirect adaptive control where r_1 and r_2 are respectively a sinusoidal signal and a sinusoidal signal : (a) Control signal u_1 applied to the system ; (b) Control signal u_2 applied to the system ; (c) reference signal r_1 and the system output y_1 ; (d) reference signal r_2 and the system output y_2 ; (e) Evolution of the difference between the reference signal r_1 and the system output y_1 ; (f) Evolution of the difference between the reference signal r_2 and the system output y_2



(a) **(b)** **(c)** **(d)** **(e)** **(f)**

Figure 12 – Results of neuronal indirect adaptive control where r_1 and r_2 are respectively a triangular signal and a triangular signal: (a) control signal u_1 applied to the system; (b) Control signal u_2 applied to the system; (c) reference signal r_1 and the system output y_1 ; (d) reference signal r_2 and the system output y_2 ; (e) Evolution of the difference between the reference signal r_1 and the system output y_1 ; (f) Evolution of the difference between the reference signal r_2 and the system output y_2 .

Secondly, the proposed robust neural adaptive control by the technique H_∞ is applied to the system using the following procedure:

- We chose:

$$\begin{cases} \gamma_2 = 0.1 \\ \alpha_3 = 0.02 \end{cases} \tag{115}$$

- The offline calculation of the matrix A, B, C, K, P, Q :

Based on the neural identification results of the system, we have:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}, C = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \tag{116}$$

So that all the eigenvalues of matrix A_0 are less than 1:

$$K = \begin{bmatrix} -0.002 & 0.05 & 1 & 0.01 \\ 0 & 0.007 & 0 & 0.003 \end{bmatrix} \quad (117)$$

The resolution of the Riccati equation (97) gives:

$$P = \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 10 \end{bmatrix} \quad (118)$$

If :

$$Q = \begin{bmatrix} 10 & -10.2 & 0 & 0 \\ -10.2 & 21 & 0 & 0.17 \\ 0 & 0 & 19 & -10 \\ 0 & 0.17 & -10 & 20.6 \end{bmatrix} \quad (119)$$

The figures (13), (14) and (15) show the results of the robust neural adaptive control by technical H_∞. We notice that the specified constraint of attenuation is verified:

- where r_1 and r_2 are respectively a signal with random uniformly distributed amplitudes and a triangular signal:

$$\sum_{i=0}^{N} \left\| e_r(i) \right\|^2 = 450.37 \quad (120)$$

$$\sum_{i=0}^{k} \left\| \varepsilon_l(i) \right\|^2 = 5629.82 \quad (121)$$

$$\left\| H_{e_r,\varepsilon_l}(z) \right\|_\infty ; \ 0.08 \le 0.1 \quad (122)$$

- Where r_1 and r_2 are respectively a sinusoidal signal and a sinusoidal signal:

$$\sum_{i=0}^{N} \left\| e_r(i) \right\|^2 = 516.23 \quad (123)$$

$$\sum_{i=0}^{k} \left\| \varepsilon_l(i) \right\|^2 = 5735.82 \quad (124)$$

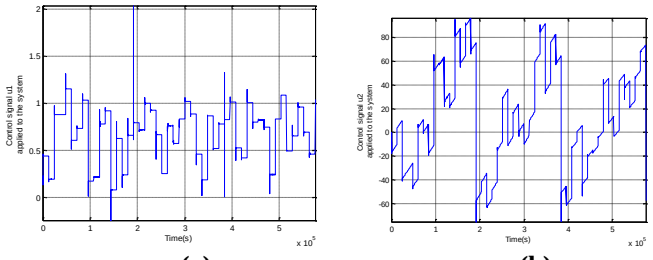
$$\left\| H_{e_r,\varepsilon_l}(z) \right\|_\infty ; \ 0.09 \le 0.1 \quad (125)$$

- Where r_1 and r_2 are respectively a triangular signal and a triangular signal:

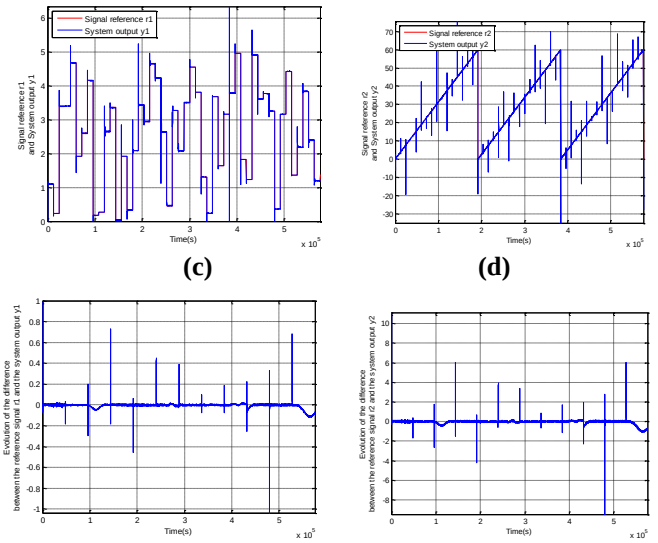
$$\sum_{i=0}^{N} \left\| e_r(i) \right\|^2 = 325.17 \quad (126)$$

$$\sum_{i=0}^{k} \left\| \varepsilon_l(i) \right\|^2 = 5419.82 \quad (127)$$

$$\left\| H_{e_r,\varepsilon_l}(z) \right\|_\infty ; \ 0.06 \le 0.1 \quad (128)$$

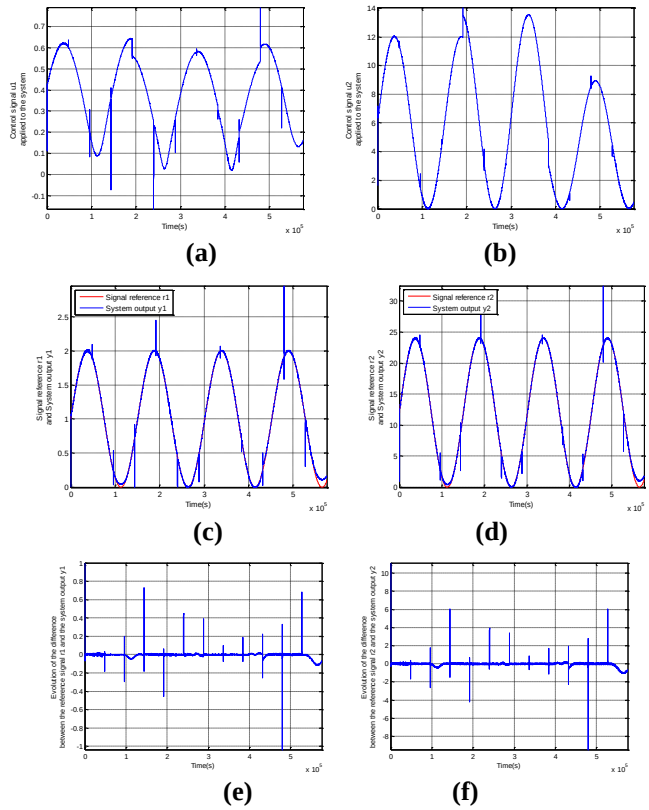


(a) **(b)**



(c) **(d)** **(e)** **(f)**

Figure 13 – Results of the robust neural adaptive control by technique H_∞ where r_1 and r_2 are respectively a signal with random uniformly distributed amplitudes and a triangular signal : (a) control signal u_1 applied to the system; (b) Control signal u_2 applied to the system; (c) reference signal r_1 and the system output y_1 ; (d) reference signal r_2 and the system output y_2 ; (e) Evolution of the difference between the reference signal r_1 and the system output y_1 ; (f) Evolution of the difference between the reference signal r_2 and the system output y_2 .



(a) **(b)** **(c)** **(d)** **(e)** **(f)**

Figure 14 – Results of the robust neural adaptive control by technique H_∞ where r_1 and r_2 are respectively a sinusoidal signal and a sinusoidal signal : (a) Control signal u_1 applied to

the system; (b) Control signal u_2 applied to the system; (c) reference signal r_1 and system output y_1 ; (d) reference signal r_2 and system output y_2 ; (e) Evolution of the difference between the reference signal r_1 and system output y_1 ; (f) Evolution of the difference between the reference signal r_2 and the system output y_2

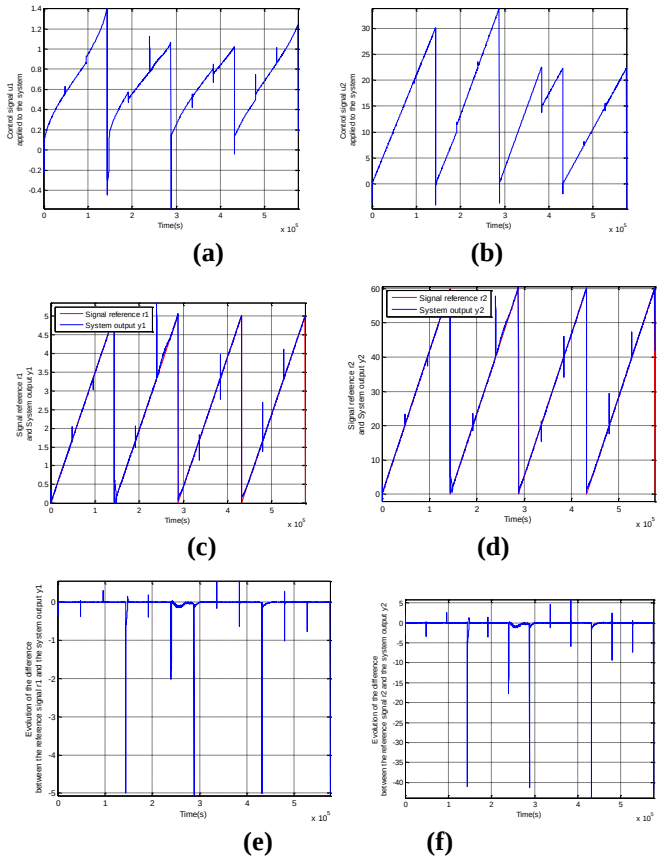


Figure 15 – Results of the robust neural adaptive control by technique H_∞ where r_1 and r_2 are respectively a triangular signal and a triangular signal : (a) Control signal u_1 applied to the system; (b) Control signal u_2 applied to the system; (c) Reference signal r_1 and system output y_1 ; (d) Reference signal r_2 and system output y_2 ; (e) Evolution of the difference between the reference signal r_1 and the system output y_1 ; (f) Evolution of the difference between the reference signal r_2 and the system output y_2 .

Based on the results of the proposed neural adaptive controls, we can conclude:

- The control signals are bounded.
- Abrupt changes of system parameters involve sudden changes of the amplitudes of commands laws and the outputs of the controlled system.
- The proposed neural adaptive control guarantees the stability of control structures and show robustness in the presence of parameter changes of the controlled system.
- Robust neural adaptive control by technique H_∞ reduces the effect of disturbances and / or uncertainties compared with neural indirect adaptive control.

Conclusion

In this work, the purpose of the command is solving problems tracking given trajectories. The principal contribution of this work lies in developing new methodologies of adaptive control based on neural network. Two techniques of neural adaptive control have been proposed, developed and tested successfully. The first technique which is indirect neural adaptive control has the advantage of being simple to the use. It uses the neural model of the system to be controlled and Lyapunov theory for make online learning of neural controller and to maintain stability of the controlled system. On the other hand, this technique risks not to mitigate the effects of disturbance and therefore the controlled system cannot follow the trajectories of references of good performance. To solve this problem, we propose a robust neural adaptive control by the technique H_∞. In the second technique, the control law implemented is the sum of two terms. Indeed, the first term is an approximation of the inputs-outputs linearization control of the system to be controlled based on neural network. On the other hand, the second term is a term of robustification. The two proposed techniques of neural adaptive control ensure the stability of control structure and present robustness in the presence of parameter changes of the system to be controlled. Robust neural adaptive control by the technique H_∞ reduces the effect of disturbances and / or uncertainties compared with the neuronal indirect adaptive control. The robust neuronal adaptive control by the technique H_∞ allows to the controlled system outputs to follow correctly the reference signals relative to the adaptive control neuronal indirect.

Reference

[1] Ge, S.S., Jing Wang, "Robust adaptive neural control for a class of perturbed strict feedback nonlinear systems," *Neural Networks, IEEE Transactions on*, Vol. 13, No. 6, 2002, pp. 1409 - 1419.

[2] Ruliang Wang, Chaoyang Chen, "Robust adaptive neural control for a class of stochastic nonlinear systems," *Computational Intelligence and Security (CIS), 2010 International Conference on*, Nanning, 11-14 Dec. 2010, pp. 511 - 514.

[3] Hu, Tingliang, Zhu, Jihong, Sun, Zengqi, "Robust adaptive neural control of a class of nonlinear systems," *Tsinghua Science and Technology*, Vol. 12, No. 1, 2007, pp. 14 - 21.

[4] Mou Chen, Shuzhi Sam Ge, Bernard Voon Ee How, "Robust adaptive neural network control for a class of uncertain mimo nonlinear systems with input nonlinearities," *IEEE Transactions on Neural Networks*, Vol. 21, No. 5, 2010, pp. 796-812.

[5] Xiaoou Li and Wen Yu, "Robust adaptive control using neural networks and projection," *Advances in neural networks*, Vol. 3174, 2004.

[6] Y. Zuo, Y. N. Wang, Y. Zhang, Z. L. Shen, Z. S. Chen, J. Chen, Q. Y. Xie, "RBF neural network robust adaptive control for wind generator system," *MECHANIKA*, Vol. 17, No. 5, 2011, pp. 557-561.

[7] Lewis, F.L., Liu, K., Yesildirek, A, "Multilayer neural net robot controller with guaranteed tracking performance," *Intelligent Control, 1993., Proceedings of the 1993 IEEE International Symposium on*, Chicago, IL , USA, 25-27 Aug 1993 , pp. 225 - 231.

[8] Yangmin Li and Yugang Liu, "Sliding Mode Adaptive Neural-network Control for Nonholonomic Mobile Modular

Manipulators," *2nd International conference on Autonomous Robots and Agents*, Palmerston North, New Zealand, 13-15 Dec. 2004, pp. 511 - 514.
[9] Tairen Sun, Hailong Pei, Yongping Pan, Hongbo Zhou, Caihong Zhang, "Neural network-based sliding mode adaptive control for robot manipulators," *Neurocomputing*, Vol.74, No. 14–15, 2011, pp. 2377-2384.
[10] Tsung-Chih Lin, "Based on interval type-2 fuzzy-neural network direct adaptive sliding mode control for SISO nonlinear systems," *Communications in Nonlinear Science and Numerical Simulation*, Vol.15, No. 12, 2010, pp. *4084-4099*.
[11] A. Nied, S.I. Seleme Jr., G.G. Parma, B.R. Menezes, "On-line neural training algorithm with sliding mode control and adaptive learning rate," *Neurocomputing*, Vol.70, No. 16-18, 2007, pp. *2687-2691*.
[12] Slotine JJ. and Lie W, "Applied Nonlinear Control," *Prentice-Hall.Inc*, 1991.
[13] Yi Zuo, Yaonan Wang, Xinzhi Liu, Simon X. Yang, Lihong Huang, Xiru Wu, Zengyun Wang, "Neural network robust H_∞ tracking control strategy for robot manipulators," *Applied Mathematical Modelling*, Vol. 34, No. 7, 2010, pp. 1823-1838.
[14] Xingjian Jing, "An H_∞ control approach to robust learning of feedforward neural networks," *Neural Networks*, Vol.24, No. 7, 2011, pp. 759-766.
[15] Miyasato, Y., "Adaptive H_∞ formation control for Euler-Lagrange by utilizing neural network approximators," American Control Conference (ACC), June 29 2011-July 1 2011, San Francisco, CA, pp. 1753 - 1758.
[16] Sitthidet Vachirasricirikul, Issarachai Ngamroo, "Robust controller design of heat pump and plug-in hybrid electric vehicle for frequency control in a smart microgrid based on specified-structure mixed H_2 / H_∞ control technique," *Applied Energy*, Vol. 88, No. 11, 2011, pp. 3860-3868.
[17] Psaltis D., Sideris A.,Yamamura, A. A., "A multilayered neural network control," *IEEE Control Systems Magazine*, 1988, Vol. 8, No. 2, pp. 17-21.
[18] Baltersee J, Chambers JA, "Nonlinear adaptive prediction of speech with a pipelined recurrent neural network," *IEEE Trans Signal Process*, 1998, Vol. 46, No. 8, pp. 2207–2216.
[19] Stavrakoudis DG, Theocharis JB, "Pipelined recurrent fuzzy neural networks for nonlinear adaptive speech prediction," *IEEE Trans. Systems, Man and Cybernetics, (Part B)*, 2007, Vol. 37, No. 5, pp. 1305-1320.
[20] Haiquan Zhao, Jiashu Zhang, "A novel adaptive nonlinear filter based pipelined feedforward second-order Volterra architecture," *IEEE Trans Signal Process, 2009*, Vol. 57, No. 1, pp. 237–246.
[21] Po-Rong Chang, Jen-Tsung Hu, "Optimal nonlinear adaptive prediction and modeling of MPEG video in ATM networks using pipelined recurrent neural networks," *IEEE J Select Areas Commun*, 1997, Vol. 15, No. 6, pp. 1087–1100.
[22] Yih-Shen Chen, Chung-Ju Chang, Yi-Lin Hsieh, Nat. Chiao Tung Univ, Hsinchu, Taiwan, "A channel effect prediction-based power control scheme using PRNN/ERLS for uplinks in DS-CDMA cellular mobile systems," *IEEE Trans Wireless Commun*, 2006, Vol. 5, No. 1, pp. 23–27.
[23] Mandic DP, Chambers JA, Toward, "an optimal PRNN based nonlinear prediction," *IEEE Transactions on Neural Networks*, 1999, Vol. 10, No. 6, pp. 1435– 1442.
[24] Mandic DP, Chambers JA, "On the choice of parameters of the cost function in nested modular RNNs," *IEEE Transactions on Neural Networks*, 2000, Vol. 11, No. 2, pp. 315–322.
[25] Haiquan Zhao, Jiashu Zhang, "Pipelined Chebyshev functional link artificial recurrent neural network for nonlinear adaptive filter," *IEEE transactions on systems man and cybernetics Part B Cybernetics a publication of the IEEE Systems Man and Cybernetics Society*, 2010, Vol. 40, No. 1, pp. 162–172.
[26] Williams RJ, Zipser D, "A learning algorithm for continually running fully recurrent neural networks," *Neural Computation*, 1989, Vol. 1, No. 2, pp. 270–280.
[27] Barak A. Pearlmutter, "Dynamic recurrent neural networks," *Technical report CMU-CS-90-196*, 1990.
[28] B. A. Pearlmutter, "Gradient calculations for dynamic recurrent neural networks: A survey," *IEEE Transactions on Neural networks*, 1995, Vol. 6, No. 5, pp. 1212 – 1228.
[29] Irina Klevecka, Janis Lelis, "Pre-Processing of Input Data of Neural Networks: The Case of Forecasting Telecommunication Network Traffic," *Telektronikk*, 2008, Vol. 3, No. 4, pp. 168 – 178.
[30] Haykin, S., "Neural Networks - A Comprehensive Foundation," *2nd Ed.Prentice-Hall, Upper Saddle River, NJ*. 1998.
[31] Jalil Asadisaghandi,Pejman Tahmasebi, "Comparative evaluation of back-propagation neural network learning algorithms and empirical correlation for prediction of oil PVT propreties in Iran oilfields," *Journal of Petroleum Science and Engineering*, 2011, Vol. 78, No. 2, pp. *464-475*.
[32] H. Al-Duwaish, M.N. Karim, and V. Chandrasekar, "Use of multilayer feedforward neural networks in identification and control of wiener model," *IEE Proceedings, Control theory and applications*, 1996, Vol. 143, No. 3, pp. 255- 258.
[33] K. S. Narendra and K. Parthasarathy, "Identification and Control of Dynamical Systems Using Neural Networks," *IEEE Transactions on Neural Networks*, 1990, Vol. 1, No. 1, pp. 4-27.
[34] Chao-Chee Ku and Kwang Y. Lee, "Diagonal recurrent neural networks for dynamic systems control," *IEEE Transactions on Neural Networks*, 1995, Vol. 6, No. 1, pp. 144-156.
[35] T. Denoeux, R. Lengellé, "Initializing back propagation networks with prototypes," *Neural Networks*, 1993, Vol. 6, No. 3, pp. 351-363.
[36] G.P. Drago, S. Ridella, "Statistically controlled activation weight initialization (SCAWI)," *Neural Networks, IEEE Transactions on*, 1992, Vol. 3, No. 4, pp. 627-631.
[37] J.-P. Martens, "A stochastically motivated random initialization of pattern classifying MLPs," *Neural Processing Letters*, 1996, Vol. 3, No. 1, pp. 23-29.
[38] T. Masters, "Practical Neural Network Recipes in C++," *Academic Press, Boston*, 1993.
[39] D. Nguyen, B. Widrow, "Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights," *International Joint Conference on Neural Networks, San Diego, CA , USA*, 1990, Vol. 3, pp. 21-26.
[40] S. Osowski, "New approach to selection of initial values of weights in neural function approximation," *Electronics Letters*, 1993, Vol. 29, No. 3, pp. 313-315.
[41] J.F. Shepanski, "Fast learning in artificial neural systems: multilayer perceptron training using optimal estimation," *Neural Networks, IEEE International Conference on, San Diego, CA , USA*, 1988, Vol. 1, pp. 465-472.
[42] H. Shimodaira, "A weight value initialization method for improving learning performance of the back propagation algorithm in neural Networks," *Tools with Artificial Intelligence,*

Proceedings, Sixth International Conference on, New Orleans, LA, USA, 1994, pp. 672-675.
[43] Y.F. Yam, T.W.S. Chow, "Determining initial weights of feedforward neural networks based on least squares method," *Neural Processing Letters*, 1995, Vol. 2, No. 2, pp. 13-17.
[44] Y.F. Yam, T.W.S. Chow, C. T. Leung, "A new method in determining initial weights of feedforward neural networks for training enhancement," *Neurocomputing*, 1997, Vol. 16, No. 1, pp. 23-32.
[45] L.F.A. Wessels, E. Barnard, "Avoiding false local minima by proper initialization of connections," *Neural Networks, IEEE Transactions on*, 1992, Vol. 3, No. 6, pp. 899-905.
[46] N. Weymaere, J.P. Martens, "On the initialization and optimization of multilayer perceptrons," *Neural Networks, IEEE Transactions on*, 1994, Vol. 5, No. 5, pp. 738-751.
[47] Jim Y. F. Yam and Tommy W. S. Chow, "A weight initialization method for improving training speed in feedforward neural network," *Neurocomputing*, 2000, Vol. 30, No. 1-4, pp. 219-232.
[48] J.E. Nash, J.V. Sutcliffe, "River flow forecasting through conceptual models part I — A discussion of principles," *Journal of Hydrology*, 1970, Vol. 10, No. 3, pp. 282-290.
[49] S.A. Billings, Q.M. Zhu, "Nonlinear model validation using correlation tests," *International Journal of Control*, 1994, Vol. 60, No. 6, pp. 1107-1120.
[50] S.A. Billings, H.B. Jamaluddin, S. Chen, "Properties of neural networks with applications to modelling non-linear dynamical systems," *International Journal of Control*, 1992, Vol. 55, No. 1, pp. 193-224.
[51] Dokla, Mahmoud, Osman, Mohammed, UAE U, "Correlation of PVT Properties for UAE Crudes," *SPE Formation Evaluation*, 1992, Vol. 7, No. 1, pp. 41-46.
[52] Glaso, Oistein, SINTEF/NTH Trondheim, "Generalized Pressure-Volume-Temperature Correlations," *Journal of Petroleum Technology*, 1980, Vol. 32, No. 5, pp. 785-795.