New Conditions for Lyapunov Stability Of A Hybrid Learning Based Indirect Adaptive Neural Control for Nonlinear Systems

Fadwa Damak[1], Mounir Ben Nasr[2] , Mohamed Chtourou[3]

[2] Department of Industrial computer science, ENET'COM, University of Sfax, Tunisia,
[1,3] Department of Electrical Engineering, ENIS, University of Sfax,Tunisia.

Abstract: In this study, novel criteria for both asymptotic and exponential stability within a hybrid learning-based adaptive neural control framework are presented. These necessary and sufficient conditions are employed to adaptively tune the learning rates of the Neural Model (NM) and the Neural Controller (NC), thereby guaranteeing the stability of the indirect adaptive neural control system. Theoretical analysis together with simulation outcomes confirm the efficiency and robustness of the proposed adaptive control.

Keywords: Indirect adaptive control, Neural network, Lyapunov methods, Stability analysis, On line hybrid learning.

1. Introduction

The design of adaptive control systems has been a major focus of research over the past decades. Various advanced adaptive strategies, including sliding mode control, robust adaptive control, backstepping, and decentralized control, have been developed to enhance control system performance [1–4]. Due to their universal approximation capabilities, neural networks have also been widely applied in the adaptive control of nonlinear systems. Adaptive neural controllers are generally classified into two main approaches. Direct Adaptive Control estimates the controller parameters directly, while Indirect Adaptive Control first adjusts the model parameters and then tunes the controller parameters [5–19]. A typical neural network-based control scheme employs multilayer feedforward networks trained using the well-known backpropagation (BP) algorithm, which relies on gradient descent optimization. Despite its popularity, the standard BP algorithm and its variants exhibit limitations, such as slow convergence and potential instability, which can hinder global convergence. To address these issues, numerous studies have proposed adaptive neural control methods grounded in Lyapunov stability theory [20–27]. These approaches derive adaptive laws for updating network weights based on a quadratic Lyapunov function, ensuring the stability of the controlled system. This paper extends prior work by investigating the stability of a hybrid learning-based indirect adaptive neural control scheme, supported by theoretical analysis and simulation studies. The proposed scheme employs two neural networks: a neural model (NM) for system identification and a neural controller (NC**)** for control. Using the Lyapunov synthesis approach, the parameters of both networks are tuned online via a hybrid learning algorithm that combines the Kohonen rule with gradient descent, integrating the strengths of both methods. Furthermore, two strategies are proposed for selecting appropriate learning rates for NM and NC, ensuring the stability of the indirect adaptive control scheme.

The paper is structured as follows: Section 2 describes the design of the indirect adaptive neural network controller. Sections 3 and 4 present two methods for adjusting the learning rates of NM and NC to guarantee stability. Simulation results are provided in Section 5, followed by conclusions and references in Sections 6 and 7.

2. Structure of the Indirect Adaptive Control Approach

2.1. Description

The indirect adaptive control scheme employed in this study, illustrated in Figure 1, consists of two neural networks: a neural model (NM) and a neural controller (NC). In this framework, the control signal is produced by the NC. The main objective of the control problem is to compute an input for the plant that ensures the system output closely tracks the desired reference signal.

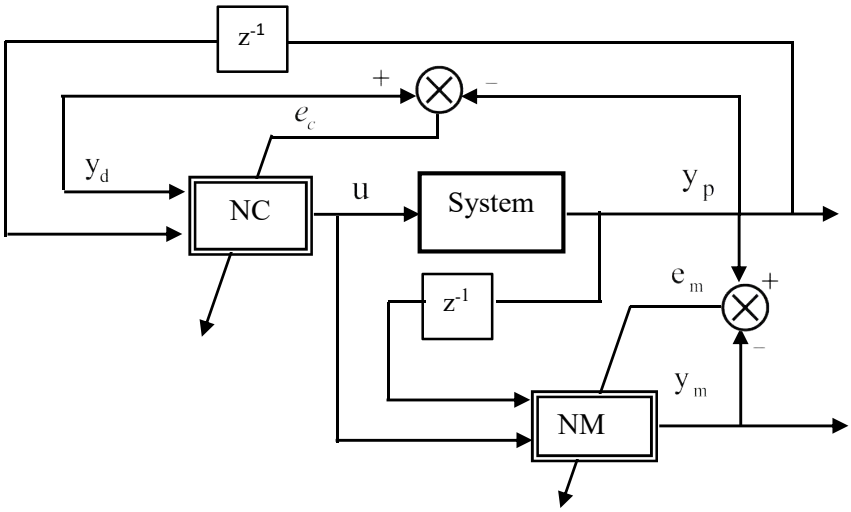


Figure.1. Shematic of Indirect adaptive control system.

Considered a nonlinear system described by:

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

Where y_p is the system output, u is the control input, n_a and n_b denote the output and input orders, respectively. The function F represents the system's nonlinear dynamics, and the term k accounts for the system time delay.

2.2.The hybrid training algorithm

Both the neural model and the neural controller are implemented using three-layer neural networks (Figures 2 and 3). The networks are trained online with a hybrid learning algorithm that combines the Kohonen rule [29] and gradient descent [28], leveraging the fast self-organization of Kohonen learning with the precise weight adjustment of gradient descent. All neurons employ the sigmoid activation function, ensuring smooth nonlinear mapping capabilities. This hybrid approach improves convergence speed and stability compared to using gradient descent alone

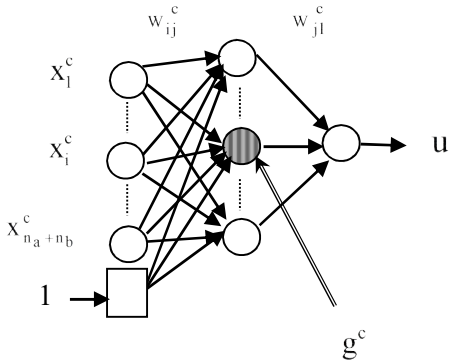


Figure 2. Neural controller.

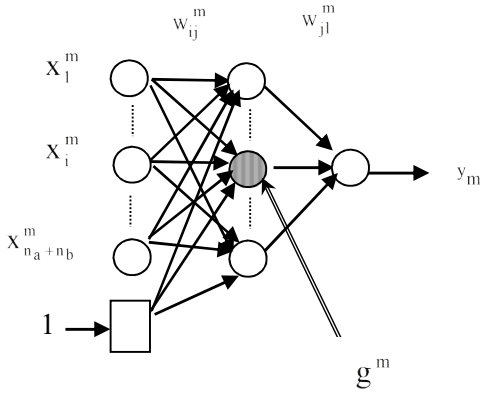


Figure 3. Neural model.

The online hybrid learning procedure for the two neural networks can be summarized as follows:

Let X^c denote the input vector of Neural Controller (NC):

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

First, the input vector is generated. Then the index g^c of the winning neuron is identified using the minimum distance Euclidean criterion:

$$g^c = \operatorname{argmin}_j \left\{ \left\| X^c - W_j^c \right\| \right\}, \quad \forall j \in \left\{ 1 \cdots p^c \right\} \quad (3)$$

where p^c indiates the total number of neurons contained in hidden layer of the neural controller

After determining the winning neuron, a corresponding neighborhood region is defined. The output of the neural controller is then obtained according to the following expression:

$$u(k) = f\left(\sum_{j \in V^c} W_{jl}^c(k) . \left[f\left(\sum_i W_{ij}^c(k) . x_i^c + \theta_j^c(k) \right) \right] + \theta_l^c(k) \right) \quad (4)$$

Here, V^c defines the neighborhood region surrounding the winning neuron g^c, while W_{ij}^c and W_{jl}^c correspond to the synaptic the weights of the hidden and output layers .the computed control signal u is subsequently fed into the process.

Let $X^m(k) = \left[x^m{}_1(k), \ldots, x_{n_a+n_b}^m(k) \right] = \left[y_p(k), \ldots, y_p(k-n_a+1), u(k), \ldots u(k-n_b+1) \right]$ (5)

Similarly, the input X^m is constructed, after which the index g^m of the winning neuron is selected according to the minimum distance Euclidean criterion:

$$g^m = \operatorname{argmin}_j \left\{ \left\| X^m - W_j^m \right\| \right\}, \quad \forall j \in \left\{ 1 \cdots p^m \right\} \quad (6)$$

Here p^m indicates the total number of neurons in the hidden layer of the Neural Model (NM).

Subsequently, a neighborhood surrounding the winning neuron is determined, and the neural network model output is obtained according to the following expression:

$$y_m(k+1) = f\left(\sum_{j \in V^m} W_{jl}^m(k) \left[f\left(\sum_i W_{ij}^m(k) . x_i^m + \theta_j^m(k) \right) \right] + \theta_l^m(k) \right) \quad (7)$$

In this context, V^m specifies the neighborhood region surrounding the winning neuron g^m, and W_{ij}^m and W_{jl}^m correspond to the synaptic weights of the hidden and output layers.

The error function for neural model, which is to be minimized, is expressed as follows:

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

In the case of neural controller, the error function to be minimized is expressed as follows:

$$E_c = \frac{1}{2}\left(e_c\right)^2 = \frac{1}{2}\left(y_d(k+1) - y_p(k+1)\right)^2 \quad (9)$$

The NM weights are then updates by equation (10) and (11).

Update the set of neighborhood neurons $W_{ij}^m \in V^m$ using the Kohonen learning rule:

$$\Delta W_{ij}^m(k) = \eta_{1m}(k) . (x_i^m - W_{ij}^m(k)) \quad (10)$$

$\eta_{1m}(k)$ is the learning rate at any time instant k.

Update the weights $W_{jl}^{m} \in V^{m}$ using gradient descent method:

$$\Delta W_{jl}^{m}(k) = -\eta_{2m}(k).\frac{\partial E_{m}(w)}{\partial W_{jl}^{m}}$$
$$= -\eta_{2m}(k)\cdot\left(y_{m}(k+1) - y_{p}(k+1)\right).f'\left(\sum_{j\in V^{m}} W_{jl}^{m}.f(\sum W_{ij}^{m}.x_{i}^{m})\right)\cdot f\left(\sum W_{ij}^{m}.x_{i}^{m}\right) \quad (11)$$

$\eta_{2m}(k)$ is the learning rate at any instant time k.

The NC weights are then updates by equation (12) and (13).

Modify the set of neighborhood neurons $W_{ij}^{c} \in V^{c}$ by the Kohonen learning rule:

$$\Delta W_{ij}^{c}(k) = \eta_{1c}(k).(x_{i}^{c} - W_{ij}^{c}(k)) \quad (12)$$

$\eta_{1c}(k)$ is the learning rate at any instant time k.

Calculate the weights $W_{jl}^{c} \in V^{c}$ using gradient descent method:

$$\Delta W_{jl}^{c}(k) = -\eta_{2c}.\left(y_{m}(k+1) - y_{d}(k+1)\right)J(k)\frac{\partial u(k)}{\partial W_{lj}^{c}(k-1)} \quad (13)$$

$\eta_{2c}(k)$ is the learning rate at any instant time k..

where $$\frac{\partial u(k)}{\partial W_{jl}^{c}(k-1)} = u(k)\cdot(1-u(k))\cdot f\left(\sum_{j\in V^{c}} W_{ij}^{c}(k).x_{i}^{c}\right) \quad (14)$$

Moreover J in (13) is defined as:

$$J(k) = \frac{\partial y_{m}(k+1)}{\partial u(k)} = \sum_{j\in V^{m}}\left(y_{m}.(1-y_{m})\right).W_{jl}^{m}.f'\left(\sum W_{ij}^{m}.x_{i}^{m}\right).W_{ij}^{m} \quad (15)$$

3. First Stability Approach

In this section, theorems and relevant proofs are presented to verify the Lyabunov Asmptotic Stability(LAS) convergence :

Theoreme 3.1 :

Let η_{1m} and η_{2m} are the learning rates for the tuning parameters of NM. Then the convergence is guaranted if the learning rates are chosen as :

$$\eta_{1m} = \min\left(\frac{2.\varepsilon_{1}.e_{m}}{P_{1}^{T}.\left(x_{i}^{m} - W_{ij}^{m}\right)}, \eta_{1m,max}\right) \text{ and } \eta_{2m} = \min\left(\frac{2.\varepsilon_{2}}{\left(P_{2}\right)^{2}}, \eta_{2m,max}\right) \quad (16)$$

Where ε_{s} and $\eta_{sm,max}$ are two predefined positive constants satisfying $0 < \varepsilon_{s} < 1$ and $\eta_{sm,max} = \max(\eta_{sm})$, respectively.

$s = 1$ and $s = 2$ denote the hidden layer and the output layer, respectively.

Proof of theorem 3.1 :

Define the Lyapunov function canditate :

$$V_m(k) = E_m(k) = \frac{1}{2}e_m^2(k) = \frac{1}{2}\left(y_p(k) - y_m(k)\right)^2 \tag{17}$$

Change of Lyapunov fonction is :

$$\Delta V_m(k) = \frac{1}{2}\left[e_m^2(k+1) - e_m^2(k)\right] \tag{18}$$

From $e(k+1) = e(k) + \Delta e(k)$ (19)

Then $\Delta V_m(k) = \Delta e_m(k)\left[e_m(k) + \frac{1}{2}\Delta e_m(k)\right]$ (20)

The change identification error with respect to the hidden layer is computed as :

$$\Delta e_{1m}(k) = \left(\frac{\partial e_m}{\partial W_{ij}^m}\right)^T \cdot \Delta W_{ij}^m = -\eta_{1m} \cdot P_1^T \cdot \left(x_i^m - W_{ij}^m(k)\right) \tag{21}$$

Where $P_1 = \frac{\partial y_m}{\partial W_{ij}^m} = \left(y_m \cdot (1 - y_m)\right) \cdot W_{jl}^m \cdot f'\left(\sum_{j \in V^m} W_{ij}^m . x_i\right) \cdot x_i^m$ (22)

Using (21) in (20), we have

$$\Delta V_{1m}(k) = -\frac{1}{2} \cdot \left[\eta_{1m} . P_1^T . \left(x_i^m - W_{ij}^m\right)\right]^2 \cdot \left[\frac{2e_m}{\eta_{1m} . P_1^T . \left(x_i^m - W_{ij}^m\right)} - 1\right] \tag{23}$$

The negative definiteness of $\Delta V_{ml}(k) < 0$ as long as :

$$\eta_{1m} < \frac{2e_m}{P_1^T . \left(x_i^m - W_{ij}^m\right)} \tag{24}$$

The change identification error with respect to the output layer is computed as :

$$\Delta e_{2m}(k) = \left(\frac{\partial e_m}{\partial W_{jl}^m}\right)^T \cdot \Delta W_{jl}^m = -\eta_{2m} \cdot e_m(k) . P_2^2 \tag{25}$$

Where $P_2 = \frac{\partial y_m}{\partial W_{jl}^m} = \left(y_m \cdot (1 - y_m)\right) \cdot f\left(\sum_{j \in V^m} W_{ij}^m . x_i^m\right)$ (26)

Using (25) in (20), we have

$$\Delta V_{2m}(k) = -\eta_{2m} \cdot e_m^2(k) \cdot (P_2)^2 \left[1 - \frac{1}{2}\eta_{2m} \cdot (P_2)^2\right] \tag{27}$$

The negative definiteness of $\Delta V_{m2}(k) < 0$ can be ensured as long as :

$$1 - \frac{1}{2}\eta_{2m} \cdot \left(P_2\right)^2 > 0 \tag{28}$$

Then $\eta_{2m} < \frac{2}{\left(P_2\right)^2}$ (29)

This concludes the proof of Theorem 3.1.

Theoreme 3.2 :

Let η_{1c} and η_{2c} are the learning rates for the tuning parameters of NC. Then the convergence is guaranted if the learning rates are chosen as :

$$\eta_{1c} = \min\left(\frac{2\tau_1.e_c}{\left(J\cdot Q_{ij}\right)^T\cdot\left(x_i^c - W_{ij}^c\right)}, \eta_{c1,max}\right) \text{ and } \eta_{2c} = \min\left(\frac{2\tau_2}{\left(J\cdot Q_{jl}\right)^2}, \eta_{2c,max}\right) \tag{30}$$

Where τ_s and $\eta_{sm,max}$ are two predefined positive constants satisfying $0 < \tau_s < 1$ and $\eta_{sm,max} = \max(\eta_{sm})$, respectively, $s = 1, 2$.

Proof of theorem 3.2 :

Define the Lyapunov function canditate :

$$V_c(k) = E_c(k) = \frac{1}{2}e_c^2(k) = \frac{1}{2}\left(y_d(k) - y_m(k)\right)^2 \tag{31}$$

Using the similar methods, we can obtain

$$\Delta V_c(k) = \Delta e_c(k)\left[e_c(k) + \frac{1}{2}\Delta e_c(k)\right] \tag{32}$$

The change tracking error with respect to the hidden layer is computed as :

$$\Delta e_{1c}(k) = \left(\frac{\partial e_c}{\partial W_{ij}^c}\right)^T.\Delta W_{ij}^c = -\left(J.Q_{ij}\right)^T.\eta_{1c}.\left(x_i^c - W_{ij}^c\right) \qquad \forall i, j \in V^c \tag{33}$$

$$Q_{ij} = \frac{\partial u}{\partial W_{ij}^c} = \left(u\cdot(1-u)\right)\cdot W_{jl}^c\cdot\left(f'\left(\sum_{j\in V^c} W_{ij}^c.x_i^c\right)\right)\cdot x_i^c \tag{34}$$

Using (33) in (32), we have

$$\Delta V_{1c}(k) = -\frac{1}{2}\cdot\left[\left(J\cdot Q_{ij}\right)^T\cdot\eta_{1c}.\left(x_i^c - W_{ij}^c\right)\right]^2.\left[1 - \frac{2.e_c}{\left(J\cdot Q_{ij}\right)^T\cdot\eta_{1c}\left(x_i^c - W_{ij}^c\right)}\right] \tag{35}$$

The negative definiteness of $\Delta V_{c1}(k) < 0$ as long as :

$$\left[1 - \frac{2.e_c}{\left(J.Q_{ij}\right)^T.\eta_{1c}\left(x_i^c - W_{ij}^c\right)}\right] > 0 \tag{36}$$

Then $\eta_{1c} < \dfrac{2.e_c}{\left(J.Q_{ij}\right)^T.\left(x_i^c - W_{ij}^c\right)}$ (37)

The change tracking error with respect to the output layer is computed as :

$$\Delta e_{2c}(k) = \left(\frac{\partial e_c}{\partial W_{jl}^c}\right)^T\cdot\Delta W_{jl}^c = -\eta_{2c}\cdot e_c\cdot\left(J\cdot Q_{jl}\right)^2 \tag{38}$$

$$Q_{jl} = \frac{\partial u(k)}{\partial W_{jl}^c(k-1)} = u(k)\cdot(1-u(k))\cdot f\left(\sum_{j\in V^c} W_{ij}^c(k).x_i^c\right) \tag{39}$$

Using (38) in (32), we have

$$\Delta V(k) = -\eta \cdot e \cdot (J \cdot Q) \cdot \left[1 - \frac{1}{2} \cdot \eta \cdot (J \cdot Q)\right] \tag{40}$$

The negative definiteness of $\Delta V_{c2}(k) < 0$ as long as :

$$1 - \frac{1}{2} \cdot \eta_{2c} \cdot \left(J \cdot Q_{jl}\right)^2 > 0 \tag{41}$$

Then $$\eta_{2c} < \frac{2}{\left(J \cdot Q_{jl}\right)^2} \tag{42}$$

The proof of theorem 3.2 is completed.

4. Second Stability Approach

In this section, theorems and relevant proofs are presented to verify the Lyabunov Exponentiel Stability(LES) convergence :

Theoreme 4.1 :

Let η_{1m} and η_{2m} are the learning rates for the tuning parameters of NM. Then the convergence is guaranted if the learning rates are chosen as :

$$\frac{\left(1-\sqrt{(1-\gamma_1)}\right)e_m}{P_1^T \cdot \left(x_i^m - W_{ij}^m(k)\right)} \le \eta_{1m} \le \frac{\left(1+\sqrt{(1-\gamma_1)}\right)e_m}{P_1^T \cdot \left(x_i^m - W_{ij}^m(k)\right)} \quad \text{and} \quad \frac{1-\sqrt{(1-\gamma_2)}}{{P_2}^2} \le \eta_{2m} \le \frac{1+\sqrt{(1-\gamma_2)}}{{P_2}^2} \tag{43}$$

Where $0 < \gamma_s < 1,\ s = 1, 2$

Proof of theorem 4.1 :

We considere the function Lyabunov and its differential satisfies the following inequality :

$$\Delta V_m(k) \le -\gamma_s \cdot V_m(k)\ ,\ s = 1, 2 \tag{44}$$

Then $V_m(k+1) + (\gamma_s - 1) \cdot V_m(k) \le 0$ (45)

Using(17), Equation (45) can be expressed as :

$$\frac{1}{2}e_m^2(k+1) + (\gamma_s - 1)\left(\frac{1}{2} \cdot e_m^2(k)\right) \le 0 \tag{46}$$

Then the (46) can be rewritten as

$$\Delta e_m^2(k) + 2 \cdot \Delta e_m(k) \cdot e_m(k) + \gamma_s \cdot e_m^2(k) \le 0 \tag{47}$$

The change identification error with respect to the hidden layer is computed as in Eq. (21).
Using (21) in (47), we obtain :

$$\eta_{1m}(k)^2 \cdot (P_1^T)^2 \cdot \left(x_i^m - W_{ij}^m(k)\right)^2 - 2 \cdot \eta_{1m}(k) \cdot P_1^T \cdot \left(x_i^m - W_{ij}^m(k)\right).e_m + \gamma_1 \cdot e_m^2(k) \le 0 \tag{48}$$

Then

$$\left[P_1^T \cdot \left(x_i^m - W_{ij}^m(k)\right)\right]^2 \cdot \left[(\eta_{1m})^2 - \frac{2 \cdot e_m(k)}{P_1^T \cdot \left(x_i^m - W_{ij}^m(k)\right)} \cdot \eta_{1m} + \frac{\gamma_1 \cdot e_m^2(k)}{\left(P_1^T \cdot \left(x_i^m - W_{ij}^m(k)\right)\right)^2}\right] \le 0 \tag{49}$$

Eq. (49) can be modified as follows :

$$\left[\eta_{1m}-\frac{e_m(k)}{P_1^T\cdot\left(x_i^m-W_{ij}^m(k)\right)}\right]^2-(1-\gamma_1).\left[\frac{e_m(k)}{P_1^T\cdot\left(x_i^m-W_{ij}^m(k)\right)}\right]^2\le 0 \tag{50}$$

The change identification error with respect to the output layer is computed as in Eq. (27).
Using (27) in (47), we have :

$$\left(-\eta_{2m}\cdot e_m(k)\cdot P_2{}^2\right)^2-2\cdot\eta_{2m}\cdot P_2{}^2e_m^2(k)+\gamma_2\cdot e_m^2(k)\le 0 \tag{51}$$

Eq. (51) can be modified as follows :

$$(\eta_{2m}.P_2^2-1)^2+\gamma_2\le 0 \tag{52}$$

The proof of theorem 4.1 is finished.

Theoreme 4.2 :

Let η_{1c} and η_{2c} are the learning rates for the tuning parameters of NC. Then the convergence is guaranted if the learning rates are chosen as :

$$\frac{\left(1-\sqrt{(1-\tau_s)}\right)e_c(k)}{\left(J\cdot Q_{ij}\right)^T\cdot\left(x_i^c-W_{ij}^c\right)}\le\eta_{1c}\le\frac{\left(1+\sqrt{(1-\tau_s)}\right)e_c(k)}{\left(J.Q_{ij}\right)^T\cdot\left(x_i^c-W_{ij}^c\right)}\ \text{and}\ \frac{1-\sqrt{1-\tau_s}}{Q_{jl}^2\cdot J^2}\le\eta_{2c}\le\frac{1+\sqrt{1-\tau_s}}{Q_{jl}^2\cdot J^2} \tag{53}$$

Where $0<\tau_s<1,\ s=1,2$

Proof of theorem 4.2 :
We considere the function Lyabunov and its differential satisfies the following inequality :

$$\Delta V_c(k)\le-\tau_s\cdot V_c(k) \tag{54}$$

Then $V_c(k+1)+(\tau_s-1)\cdot V_c(k)\le 0$ (55)

Using(31), Equation (55) can be expressed as :

$$\frac{1}{2}e_c^2(k+1)+(\tau_s-1)\left(\frac{1}{2}\cdot e_c^2(k)\right)\le 0 \tag{56}$$

Then the (56) can be rewritten as

$$\Delta e_c^2(k)+2\cdot\Delta e_c(k)\cdot e_c(k)+\tau_s\cdot e_c^2(k)\le 0 \tag{57}$$

The change identification error with respect to the hidden layer is computed as in Eq. (33).
Using (33) in (57), we obtain

$$\left(-\eta_{1c}\cdot\left(J.Q_{ij}\right)^T\cdot\left(x_i{}^c-W_{ij}^c\right)\right)^2-2\cdot\left(J\cdot Q_{ij}\right)^T\cdot\eta_{1c}(k).\left(x_i{}^c-W_{ij}^c\right)\cdot e_c(k)+\tau_1\cdot e_c^2(k)\le 0 \tag{58}$$

$$\left[\left(J\cdot Q_{ij}\right)^T\cdot\left(x_i{}^c-W_{ij}^c\right)\right]^2\cdot\left[(\eta_{1c})^2-\frac{2\cdot e_c(k)}{\left(J\cdot Q_{ij}\right)^T\cdot\left(x_i{}^c-W_{ij}^c\right)}\cdot\eta_{1c}+\frac{\tau_s\cdot e_c^2(k)}{\left(\left(J\cdot Q_{ij}\right)^T\cdot\left(x_i{}^c-W_{ij}^c\right)\right)^2}\right]\le 0 \tag{59}$$

The change identification error with respect to the output layer is computed as in Eq. (38).
Using (38) in (57), we have :

$$\eta_{2c}^2\cdot e_c^2(k)\cdot Q_{jl}^4\cdot J^4-2\cdot e_c^2(k)\cdot Q_{jl}^2\cdot J^2\cdot\eta_{2c}+\tau_2\cdot e_c^2(k)\le 0 \tag{60}$$

Eq. (61) can be modified as follows :

$$(\eta_{2c}\cdot Q_{jl}^2\cdot J^2-1)^2+\tau_2\le 0 \tag{61}$$

The proof of theorem 4.2 is finished.

5. Simulation Results

This section presents two illustrative examples to validate the performance and stability of the proposed indirect adaptive neural network based on the hybrid learning algorithm.

5.1. Example 1

The nonlinear system under consideration is defined as follows [14].

$$y(k+1) = \frac{y(k)}{1+y^2(k)} + A(k).u^3(k) \tag{62}$$

where

$$A(k) = a_0.(1+\sin(\frac{\pi.k}{100})) \tag{63}$$

For the purpose of trajectory tracking, the desired output is defined as a smooth multi-step signal. The architectures of the NM and NC each comprise two input neurons and a single output neuron, with initial synaptic weights randomly set within the interval $[-2,2]$.

The parameters are chosen as: $V^m = V^c = 5,\ \ a_0 = 0.5\ ,\ \lambda_1 = 0.9,\ \ \lambda_2 = 0.8,\ \ \tau_1 = 0.8,\ \ \tau_2 = 0.9$

$\eta_{1m,max} = 0.8,\ \ \eta_{2m,max} = 0.6,\ \ \eta_{1c,max} = 0.6,\ \ \eta_{2c,max} = 0.7$

Figure 4 (a) and Figure 4 (b) show the performance of the controller in example 1 by LAS_BP method and LAS_Hybrid method, respectively.

Figure 5 and Figure 6 show the adaptive learning rates with theorem 3.1 and theorem 3.2 in example 1, respectively.

Figure 7 (a) and Figure 7 (b) show the performance of the controller in example 1 by LES_BP method and LES_Hybrid method, respectively.

Figure 8 and Figure 9 show the adaptive learning rates with theorem 4.1 and theorem 4.2 in example 1, respectively.

5.2. Example 2

The continuous stirred tank reactor (CSTR) under study can be modeled by a polynomial ARMA representation as given in [30].

$$y(k+1) = \theta_0 + \theta_1 u(k) + \theta_2 y(k) + \theta_3 u^3(k) + \theta_4 y(k-1)u(k-1)u(k) \tag{64}$$

The parameters are given by:

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

In this study, y denotes the output of the process , while u represents the dilution rate around the operating point. The control objective is to regulate y through appropriate manipulating of u , and the system is excited by randomly varying input within the prescribed amplitude range $[0,1]$.

In this work, both the NC and NM are implemented with the same architecture, consisting of four input neurons and a single output neuron.

The parameters are chosen as: $V^m = V^c = 5,\ \ a_0 = 0.5\ ,\ \lambda_1 = 0.9,\ \ \lambda_2 = 0.8,\ \ \tau_1 = 0.8,\ \ \tau_2 = 0.9$

$\eta_{1m,max} = 0.8,\ \ \eta_{2m,max} = 0.6,\ \ \eta_{1c,max} = 0.6,\ \ \eta_{2c,max} = 0.5$

Figure 10 (a) and Figure 10 (b) show the performance of the controller in example 2 by LES_BP method and LES_Hybrid method, respectively.

Figure 11 and Figure 12 show the adaptive learning rates with theorem 4.1 and theorem 4.2 in example 2, respectively.

Figure 13 (a) and Figure 13 (b) show the performance of the controller in example 2 by LAS_BP and LES_Hybrid algorithm, respectively.

Figure 14 and Figure 15 show the adaptive learning rates with theorem 4.1 and theorem 4.2 in example 2, respectively.

5.3. Discussion and results

The performance of the proposed hybrid learning-based indirect adaptive control system was evaluated through two sets of simulation examples. As illustrated in Figures 4(b)–7(b) and 10(b)–13(b), the system demonstrates satisfactory trajectory tracking, effectively following the desired reference signals. These results confirm the ability of the hybrid algorithm to maintain stability while achieving accurate tracking. Table 1 provides a comparison of error criteria and convergence rates across various stable learning approaches, highlighting that the hybrid algorithm delivers the best performance in terms of both tracking accuracy and convergence efficiency compared to the conventional backpropagation method. This improvement can be attributed to the integration of the Kohonen self-organizing rule with gradient descent, which enhances convergence speed and overall control robustness. Overall, the simulation results validate the effectiveness of the proposed control approach. The hybrid learning mechanism not only ensures the stability of the closed-loop system but also optimizes tracking accuracy and response speed, outperforming classical adaptive neural control methods.

Table 1. Comparative results of learning methods

Examples	Lyabunov Analysis	Stable Learning Methods	$\sum_{k=1}^{199} E_c$	CPU time (s)
Example 1	Asymptotic Stability	LAS_BP	0.110	3.815
		LAS_Hybrid	0.070	2.251
	Exponential Stability	LES_BP	0.118	3.509
		LES_Hybrid	0.060	2.917
Example 2	Asymptotic Stability	LAS_BP	0.091	3.735
		LAS_Hybrid	0.060	2.109
	Exponential Stability	LES_BP	0.190	4.090
		LES_Hybrid	0.042	3.257

6. CONCLUSION

This paper investigates the stability of an indirect adaptive neural control scheme applied to nonlinear systems. The proposed control framework utilizes a multilayer neural network with online learning based on a hybrid algorithm for both system identification and control tasks. Furthermore, a stability analysis of the online hybrid learning algorithm is carried out using the Lyapunov synthesis method. As a result, sufficient conditions for asymptotic and exponential stability are derived by appropriately adjusting the adaptive learning rates of both the Neural Model and Neural Controller, ensuring the stability of the closed-loop adaptive system. Finally, two simulation studies are presented, along with comparative results, demonstrating effective tracking of the desired signals and confirming the stability of the closed-loop system through proper selection of learning rates for the neural networks.

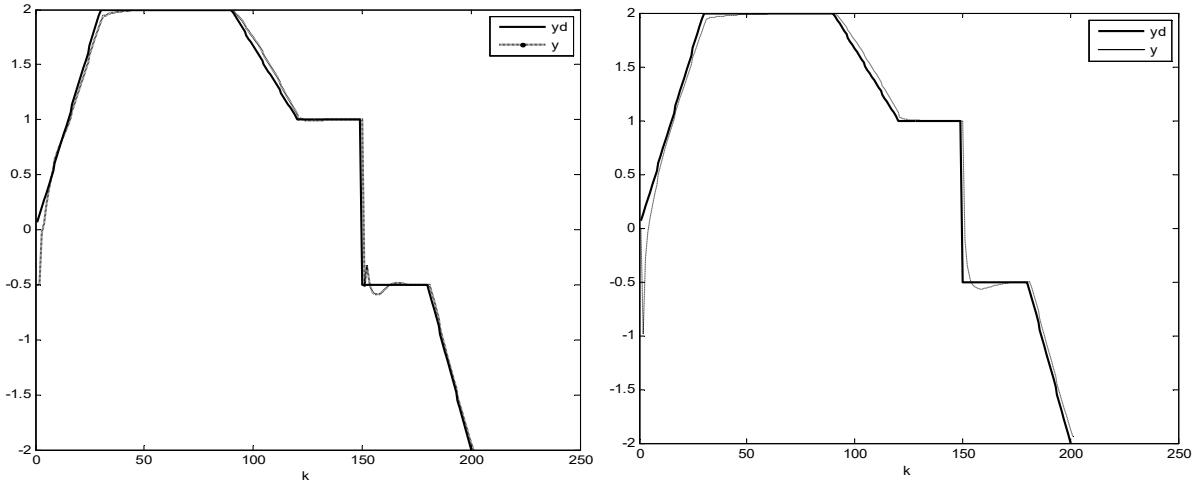


Figure 4. Evolution of y_d and y in example 1: (a) LAS_BP method and (b) LAS_Hybrid method.

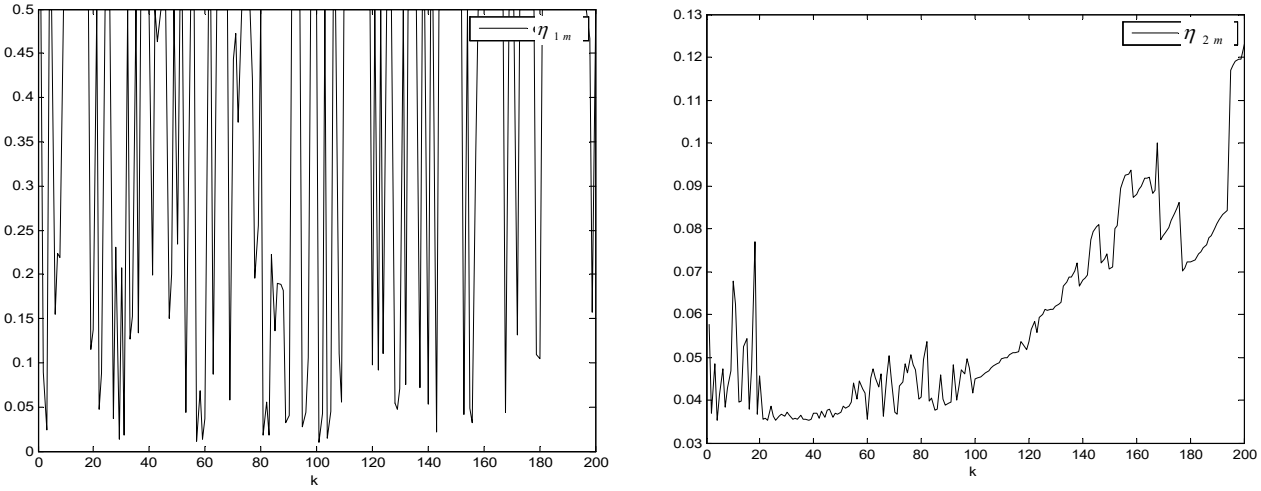


Figure 5. Trajectories of η_{1m} and η_{2m} with theorem 3.1 in example 1

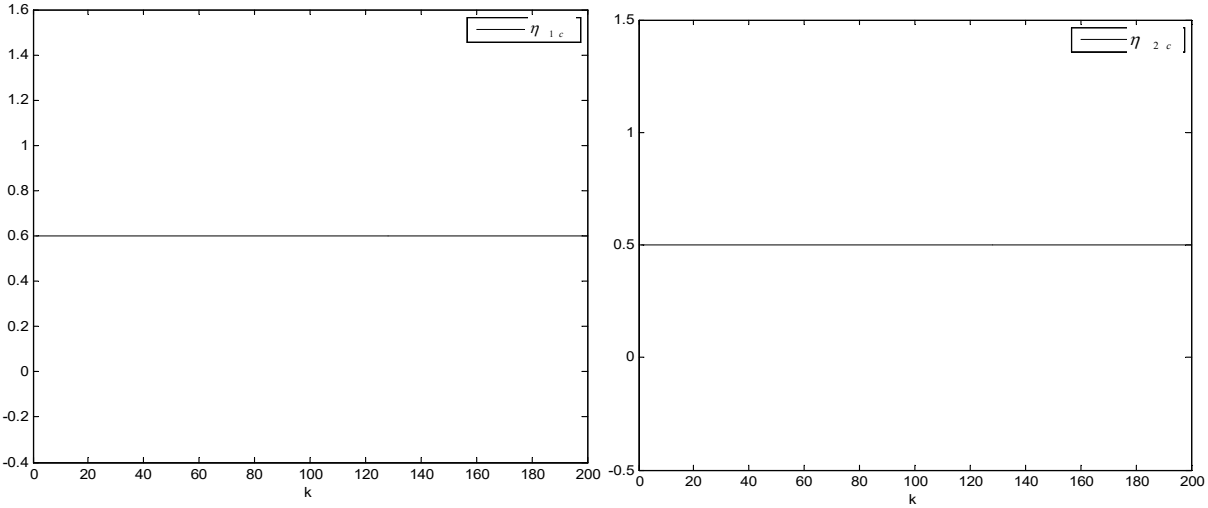
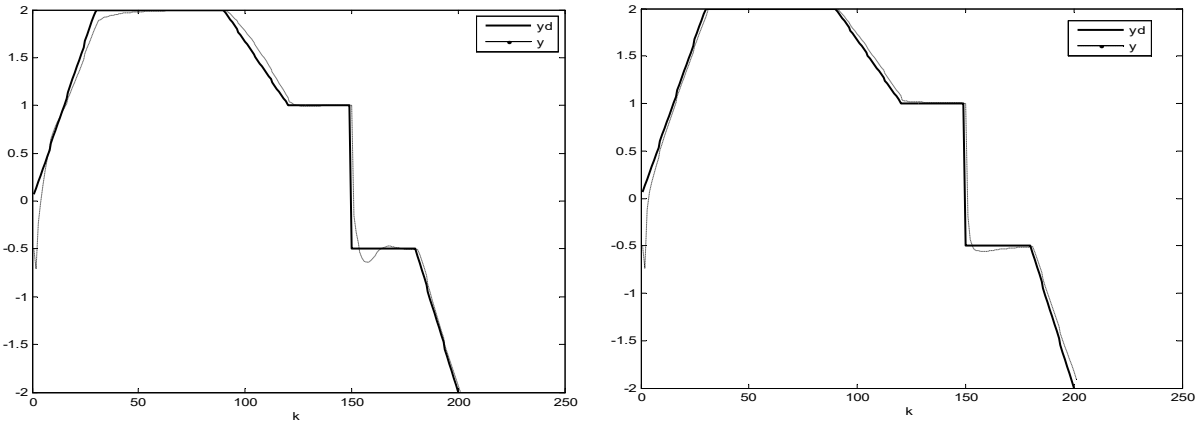


Figure 6. Trajectories of η_{1c} and η_{2c} with theorem 3.2 in example 1



(a) (b)

Figure 7. Evolution of y_d and y_d in example 1: (a) LES_BP method and (b) LES_Hybrid method.

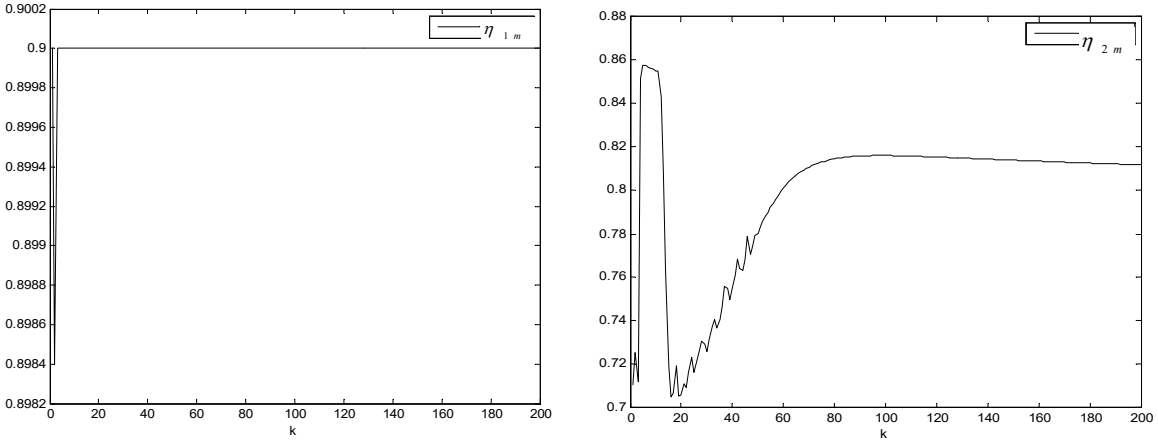


Figure 8. Trajectories of η_{1m} and η_{2m} with theorem 4.1 in example 1

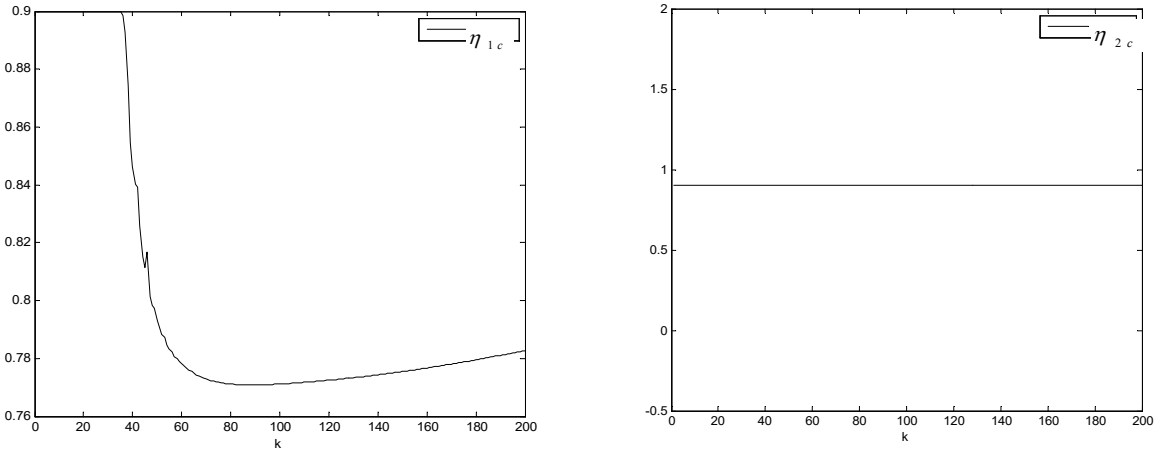
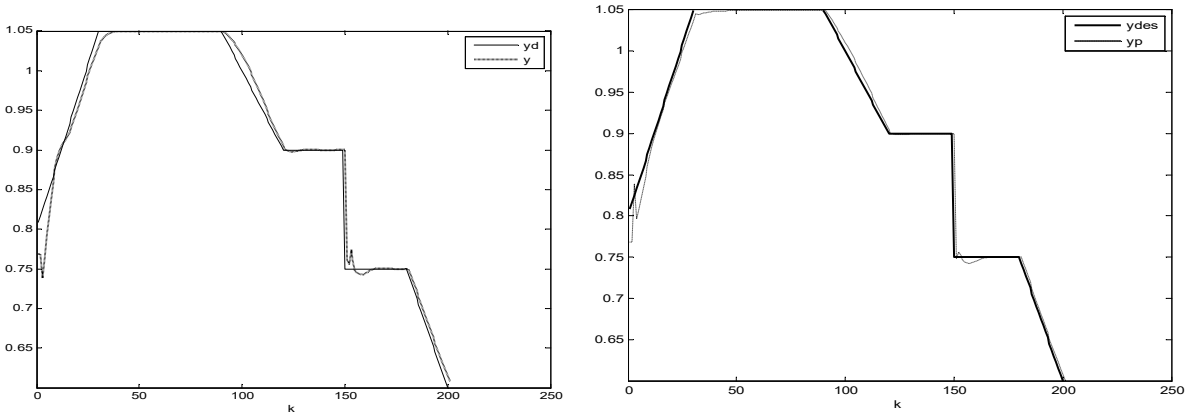


Figure 9.Trajectories of η_{1c} and η_{1c} with theorem 4.2 in example 1



(a) (b)

Figure 10. Evolution of y_d and y_d in example 2: (a) LAS_BP method and (b) LAS_Hybrid method.

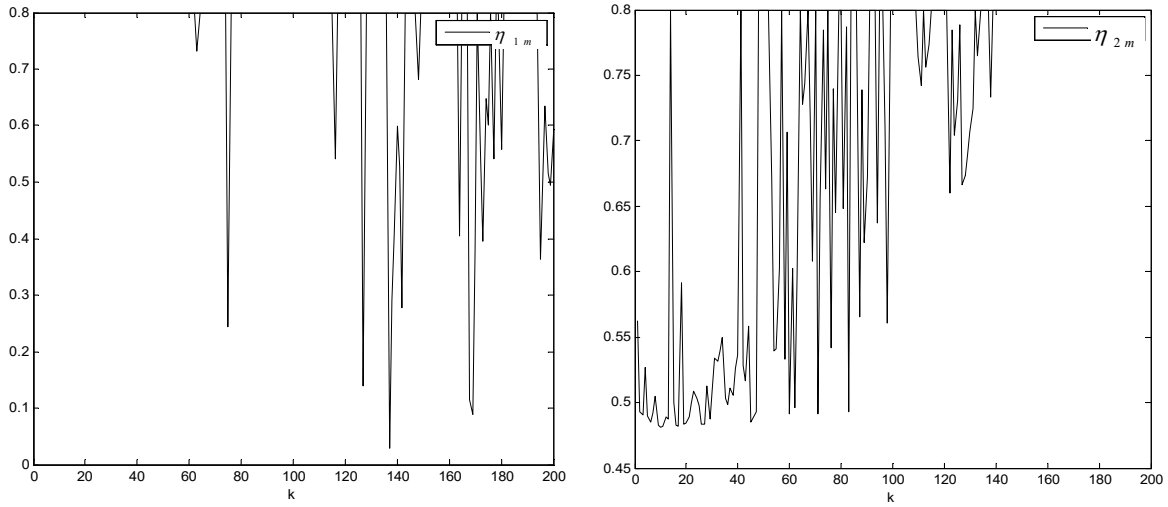


Figure 11. Trajectories of η_{1m} and η_{2m} with theorem 3.1 in example 2

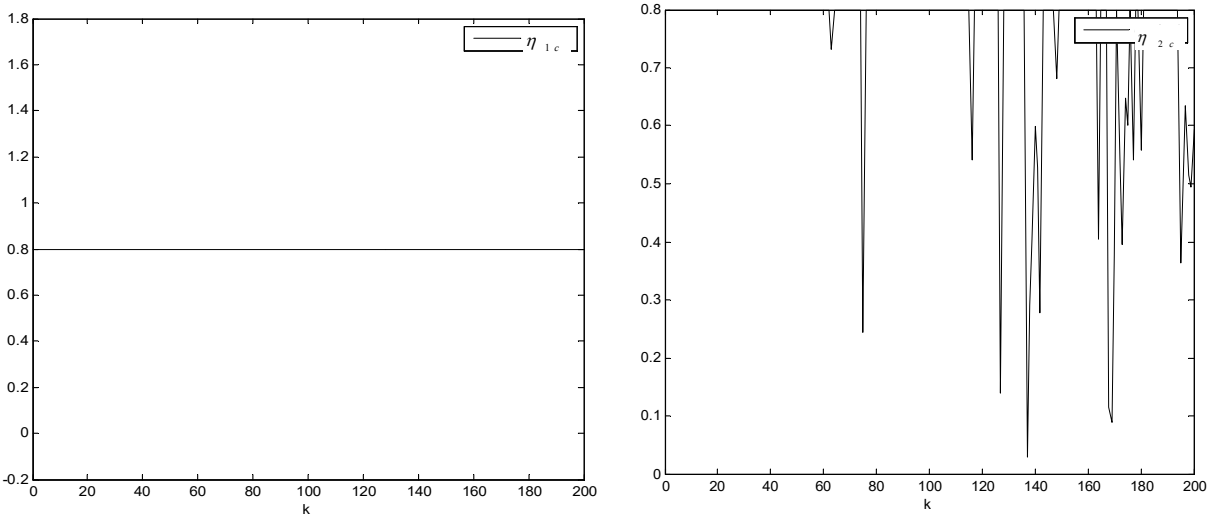
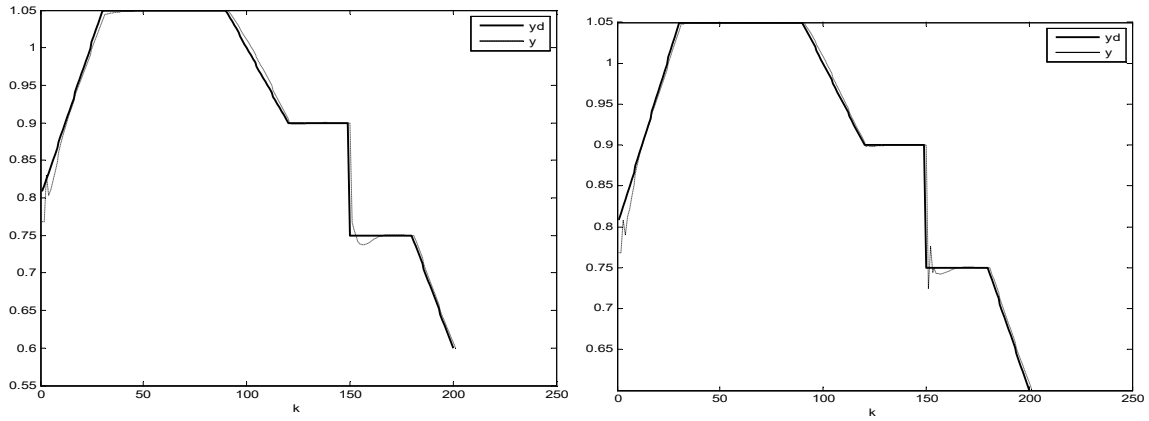


Figure 12.Trajectories of η_{1c} and η_{1c} with theorem 3.2 in example 2



(a) (b)

Figure 13. Evolution of y_d and y_d in example 2: (a) LES_BP method and (b) LES_ Hybrid method.

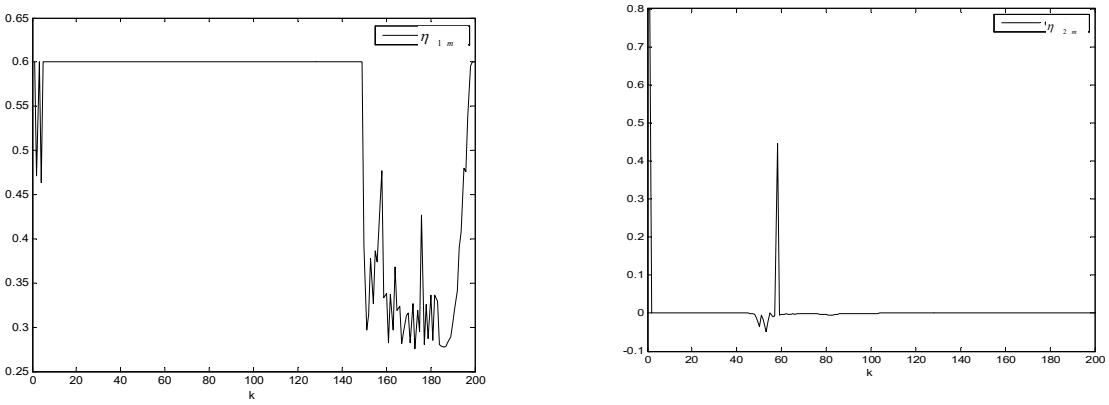


Figure 14.Trajectories of η_{1m} and η_{2m} with theorem 4.1 in example 2

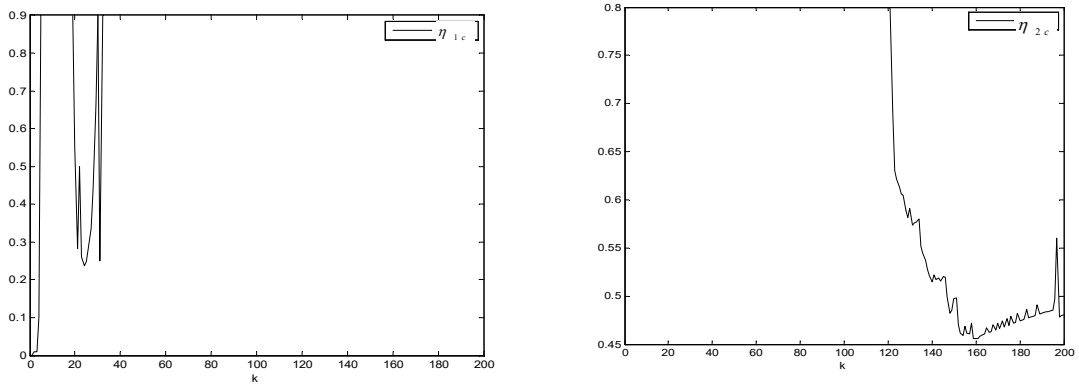


Figure 15.Trajectories of η_{1c} and η_{1c} with theorem 4.2 in example 2

7. References

[1]	Guo Y., Hill D J., Wang Y. Nonlinear Decentralized Control of Large Scale Power Systems. *Automatica* ,36(9),(2000), 1275-1289.
[2]	Sadati N., Ghadami R. Adaptive Multi-Model Sliding Mode Control of Robotic Manipulators Using Soft Computing. *Neurocomputing* ,(71),(2008),2702-2710.
[3]	Yao N., Tomizuka M. Adaptive Robust Control of MIMO Non linear System in Semi -Strict Feedback Forms . *Automatica* ,37(9),(2001),1305-1321.
[4]	Zhou J., Wen C., Yang G. Adaptivation backstepping stabilization of nonlinear uncertain systems with quantized input signal. *IEEE Trans. Autom.Control* ,59(2),(2014),460-464.
[5]	Cui G, Jiao T, Wei Y, Song G, Chu Y. Adaptive Neural Control of Stochastic Nonlinear System with Multiple Time Varying Delays and Input Saturation. *Neural Comput & Applic* ,(27),(2014),779-791.
[6]	Chen M., Ge S S., How B V E. Robust Adaptive Neural Network Control for a Class of Uncertain MIMO Nonlinear Systems With Input Nonlinearities. *IEEE. Trans. On Neural Networks* ,21(5),(2010),796-812.
[7]	Chen Z, Li Z, Philip C.L.(2017) Adaptive Neural Control of Uncertain MIMO Nonlinear Systems With State and Input Constraints. *IEEE. Trans. On Neural Networks* ,28(3),(2017), 1318-1330.
[8]	Damak F., Ben Nasr M. et Chtourou M. Indirect adaptive neural control using a hybrid learning algorithm. *ICIC Express Letters, Part B: Applications* , 9(11) ,(2018),1125-1132.
[9]	Guoping P.L, Visakan K, Stephen A. B.Variable Neural Networks for Adaptive Control of Nonlinear System. *IEEE Trans. On Systems, Man and Cybernetic* ,29(1),(1999), 34-43.
[10]	Jeon G.J, Lee I. Neural network indirect adaptive control with fast learning algorithm. *Neurocomputing,* (13),(1996),185-199.
[11]	Liu Y J, Li J, Tong S, Philip Chen C L.Neural Network Control- Based Adaptive Learning Design for Nonlinear Systems with Full-State Constraints. *IEEE. Trans. on Neural Networks* ,27(7),(2016), 1562-1571.
[12]	Mekki H, Chtourou M, Derbel N. Variable Structure Neural Networks for Adaptive Control of nonlinear Systems using the Stochastic Approximation. *Simulation Modelling Practice and Theory* ,(14), (2006),1000-1009.
[13]	Ben Nasr M., Chtourou M. Neural network control of nonlinear dynamic systems using hybrid algorithm. *Applied Soft Computing* , (24),(2014),423–431.
[14]	Narendra K S., Parthasarathy K. Identification and Control of Dynamical Systems Using Neural.Networks. *IEEE Trans. On Neural Networks* ,(1),(1990), 4-27.
[15]	Rong H J., Zhao G S. Direct Adaptive Neural Control of Nonlinear Systems with Exteme Learning Machine. *Neural Comput & Applic* ,(22),(2013), 577-586.
[16]	Tong S C., Li Y M., Zhang H G. Adaptive Neural Network Decentralized Backstepping Output-Feedback Control for Nonlinear Large Scale Systems with Time Delays. *IEEE. Trans. On Neural Networks* ,(7),(2011),1073- 1086.
[17]	Tong Z, Feng-Li F.Neural Network -Based Adaptive Output Control for MIMO Non-Affine Systems. *Neural Comput & Applic* ,(21),(2012),145-151.
[18]	Widrow B, Walach E. Adaptive Inverse Control . *Englewood Cliffa, NJ:Prentice-Hall,(* 1996).
[19]	Wang Z, Zhang Y, Fang H. Neural Adaptive Control for a Class of nonlinear System With Unknown deadzone. *Neural Comput & Applic* ,(17),(2008),339-345.
[20]	Behera L., Kumar S., Patnaik A. On Adaptive Learning rate that guarantees convergence in feed forward Networks. *IEEE Trans. Neural Networks* ,17(5) ,(2006),1116- 1125.
[21]	Gao M M., Jin X Z., Ding L G. Robust adaptive backstepping INTSM control for robotic manipulators based on ELM. *Neural Comput & Applic* ,(34),(2022),5029-5039.
[22]	Kumar T., Jeyasseli S. Neighborhood based modified backpropagation algorithm using adaptive learning parameters for training Feedforward neural networks. *Neurocomputing* (72),(2009),3915-3921.

[23]	Lee C H., Teng C C.Identification and Control of Dynamic Systems Using Recurrent Fuzzy Neural Networks. *IEEE Trans. Neural Networks* ,8(4),(2000),349-366.
[24]	Man Z., Wu H R., Liu S., Yu X. A New Adaptive Backpropagation Algorithm Based on Lyapunov Stability Theory for Neural Networks. *IEEE Trans. Neural Networks* ,17(6) ,(2006),1580- 1591.
[25]	Park J W., Harley R G., Venayagamoorthy G K. Indirect Adaptive Control for Synchronous Generator: Comparison of MLP/RBF Neural Networks Approach With Lyapunov Stability Analysis. *IEEE Trans. Neural Networks* ,5(2) ,(2004),460- 464.
[26]	Qiu J., Ma M., Wang T., Gao H. Gradient Descent-Based Adaptive Learning Control for Autonomous Underwater Vehicles With Unknown Uncertainties. *IEEE Trans. Neural Networks and learning Systems*,32(12) ,(2021),5266-5273.
[27]	Wang D, Bao P. Enhancing the Estimation of .Plant Jacobian for Adaptive neural inverse control. *Neurocomputing* ,(34),(2000),99–115.
[28]	Rumelhart D.E, Hinton G.E, Williams R.J. Learning Internal Representations by Error Propagation. *Parallel Distributed Processing : Explorations in the Microstructures of Cognition. MIT. Press*, (1) (1986),318-362.
[29]	Kohonen T. he Self- Organizing Map. Proceedings of the IEEE, 78(9),(1990),1464-1480.
[30]	Hernandez E. and Arkun Y. Stability of nonlinear polynomial ARMA models and their inverse. *International Journal of Control*, (63),(1996),885-906.