Neuro-Fuzzy Approach for Nonlinear System Control

Daikh Fatima Zohra[1][0009-0006-2928-2410]

[1] University Mustapha Stambouli of Mascara, Algeria
fatima_daikh@yahoo.fr

Abstract. In this article, we address the exploitation of Artificial Intelligence properties in the field of control engineering. Our work focuses on the use of neuro-fuzzy networks, specifically ANFIS (Adaptive Neuro-Fuzzy Inference System) and STFIS, for the identification of models required to design control laws for a nonlinear dynamic system the inverted pendulum.
In the first part, the ANFIS model is used as a controller in multiple structures. The second section presents an application of the STFIS controller on the nonlinear system. The objective of this paper is to improve the performance of the ANFIS and STFIS models for a system subjected to a constant disturbance. The proposed approach is validated through simulations carried out in the MATLAB environment.

Keywords: Neuro-fuzzy, Self tuning fuzzy inference system STFIS, Adaptive Neuro-Fuzzy Inference System ANFIS, Nonlinear, Inverted pendulum.

1 Introduction

In the control of real-world dynamic systems, the availability of an accurate mathematical model is essential for the design and implementation of any control structure. However, obtaining such a model through analytical methods is often difficult or nearly impossible. Even when a model is available, it is frequently affected by uncertainties and modeling errors. As a result, the use of Artificial Intelligence (AI) techniques for identifying the mathematical model becomes an absolute necessity [1].

A significant body of research has explored the application of AI in the control of nonlinear systems.

Among AI-based approaches, neuro-fuzzy networks stand out due to their ability to combine the global reasoning and adaptability of fuzzy logic with the powerful learning and generalization capabilities of neural networks [2]. Various hybridizations of these two paradigms have led to the development of neuro-fuzzy systems, which are particularly well-suited for the control of complex and multivariable systems [3].

Several researchers have sought to leverage the strengths of neuro-fuzzy networks for controlling dynamic systems, especially in areas such as robotics and asynchronous motor control [4].

This work introduces a novel neuro-fuzzy technique applied to the control of nonlinear systems. We describe two neuro-fuzzy models: ANFIS (Adaptive Neuro-Fuzzy

Inference System) and STFIS (Self tuning fuzzy inference system). Then, we present the application of these models to nonlinear system control.

The first section is dedicated to the development of the inverse neuro-fuzzy model (ANFIS), followed by its implementation within a control structure based on Direct Inverse Control.

The second section presents the STFIS controller applied to a nonlinear system. The results are validated through MATLAB simulations.

Neuro-Fuzzy

Neuro-fuzzy is a term used to describe systems or methods that combine neural networks and fuzzy logic techniques [2]. These systems utilize the learning capabilities of neural networks along with the human-like reasoning style of fuzzy logic to address complex problems, particularly in areas involving uncertainty, imprecision, and incomplete information[4].

Adaptive Neuro-Fuzzy Inference System (ANFIS)

The ANFIS method is an optimization technique for Takagi-Sugeno-type fuzzy inference systems, proposed by Jang [5]. It is used to adjust the parameters of the system by combining the least squares method with the gradient descent algorithm. This approach is based on the use of multilayer networks. It is assumed that the fuzzy inference system has two inputs, x and y, and a single output f (see Fig. 1).

The fuzzy rule base is expressed in the Takagi-Sugeno "if-then" format as follows:

rule i: if x is Ai and y is Bi then

$$f_i = p_i{}^{x} + q_i{}^{y} + r_i$$

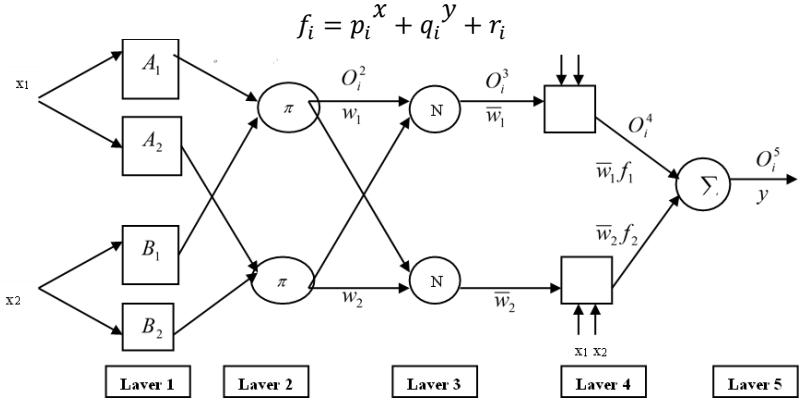

Fig. 1. Adaptive Neuro-Fuzzy Inference System (ANFIS)

The network consists of five layers, each performing a specific function (see Fig. 1):

- Layer1: fuzzification

Each node i in this layer is a square node that represents a membership function.

$$O_i^1 = U_{Ai}(x) \tag{1}$$

With x: the input of node i, Ai: the linguistic label associated with the function node. In other words, it is the membership function of Ai, and it specifies the degree of membership with which x satisfies it. The function is chosen in the form of:

- bell-shaped form

$$U_{Ai}(x) = \frac{1}{1+\left[\left(\frac{x-c_i}{\alpha_i}\right)^2\right]^b} \tag{2}$$

Or a Gaussian function

$$U_{Ai}(x) = \exp\left[-\left(\frac{x-c_i}{\alpha_i}\right)^2\right] \tag{3}$$

Or $\{\alpha_i, b_i, c_i\}$ These are parameters that refer to the premise parameters. Their values change according to different representations of the membership function.

- Layer 2: Rule Strength

Each node multiplies the incoming signals:

$$w_i = U_{Ai}(x) \times U_{Bi}(x) \tag{4}$$

- Layer 3: Normalization

Each node computes the normalized firing strength:

$$\overline{w}_i = \frac{w_i}{w_1+w_2}, i = 1,2(x) \tag{5}$$

- Layer 4 – Consequent Parameters

Each node computes the output of the rule based on the first-order Sugeno function:

$$O_i^4 = \overline{w_i} f_i = \overline{w_i}(p_i x + q_i y + r_i) \tag{6}$$

- Layer 5 – Output

A single node that sums all incoming signals to produce the final output:

$$O_i^5 = \sum_i \overline{w_i}\, f_i \tag{7}$$

Inverse model.

Although the inverse model of a system plays an important role in control theory, deriving its analytical form is quite laborious. Several system modeling methods have been presented in the literature [6]. A dynamic system can be described by equation (8), which relates its inputs to its outputs:

$$y(k+1) = f(y(k),...,y(k-n+1),u(k),...,u(k-m+1)) \tag{8}$$

Where the system output *y(k+1)* depends on the previous n output values and the past m input values. In general, the inverse model of this system can be expressed in the following form (9):

$$u(k) = f^{-1}(y(k+1), y(k)..., y(k-n+1), u(k-1), ..., u(k-m)) \quad (9)$$

Neuro-fuzzy networks can be used to develop the inverse model of the system [7]; however, representing the dynamic aspect of the system remains a challenge. Applying delays to the input layer of this type of network may offer a solution to address this shortcoming.

The corresponding network is:

$$\hat{u}(k) = g(x(k), w) \quad (10)$$

The function *g* will be approximated by an ANFIS by adjusting its weights *w*. The network has an input vector *x(k)* composed of the output *k+1* and past values of outputs and inputs, with the output being the signal û, which will be used to control the system.

The identification of the inverse model begins with the determination of the input vector, namely the number of output and input delays, which is related to the system's order.

ANFIS control

As the name suggests, the inverse neuro-fuzzy model, placed in front of the system, is used as a controller to operate the system in open-loop mode (see Fig. 2).

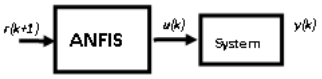

Fig. 2. Direct control by inverse model.

The value of *y(k+1)* in equation (9) is replaced by the desired output *r(k+1)*. The network is fed with the delayed values of *u(k)*and *y(k)* [87]. If the ANFIS model is an exact inverse of the system, it drives the output to follow the reference signal.

Application of the ANFIS Control for Inverted Pendulum

Controlling an inverted pendulum involves applying control inputs to stabilize the pendulum in the vertical position. Various control strategies can be used to achieve this goal. below, we will apply the backstepping control for the stability of the inverted pendulum.

The dynamic equation for an inverted pendulum can be described in the form of a second-order ordinary differential equation [9]:

$$\begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = \dfrac{(m_c + m)g \sin x_1 - mlx_2^2 \cos x_1 \sin x_1}{l\left(\dfrac{4}{3(m_c + m)} - m\cos^2 x_1\right)} \\ \quad + \dfrac{\cos x_1}{l\left(\dfrac{4}{3(m_c + m)} - m\cos^2 x_1\right)} u(t) + d(t) \end{cases} \tag{11}$$

$$\begin{cases} \dot{x}_3 = x_4 \\ \dot{x}_4 = \dfrac{4/3mlx_2^2 \sin x_1 + mg \cos x_1 \sin x_1}{4/3(m_c+m) - m\cos^2 x_1)} \\ + \dfrac{4}{3.(4/3(m_c+m) - m\cos^2 x_1)} u(t) + d(t) \end{cases}$$

$x_1$is Angular position, $x_2$is Angular velocity, x_3 is Cart position and x_4 is Cart velocity, g = $9.8m/s^2$, mc is the mass of the cart and $mc = 1kg$, m is the mass of the pendulum and $m = 0.1kg$, $l = 0.5m$" is the length of the pendulum from the centre of mass, u is control input, $d(t)$ is a noise. This inverted pendulum dynamic model is constructed using MATLAB Simulink software.

first, we developed the inverse model of the system, and then used it in a control structure with and without noise.

To generate the training and validation data, system (11) was excited by an input signal *u(k)*, which is a random sequence with amplitude uniformly distributed in the interval [−5,5], in order to provide a rich input.

Our ANFIS model takes as inputs {*y(k), y(k−1), y(k−2)*}, and uses two bell-shaped membership functions (2).

The error between the two signals, the system input *u(k)* and the output of the inverse ANFIS *û (k)* is shown in Fig. 3, along with a cost value of 10^{-3} .

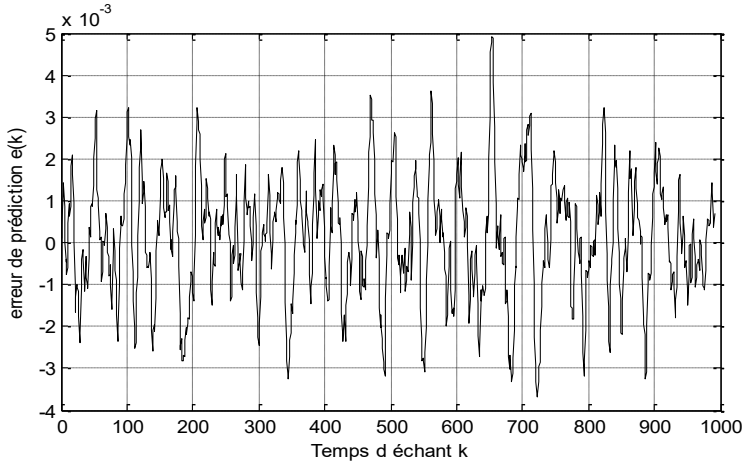

Fig. 3. Prediction error of the inverse model.

For the application of the inverse model in a direct control scheme, we tested the model using a random reference signal y_d

The error between the reference signal y_d and the output of the system controlled by the inverse model is shown in Fig. 4. We observe that the error is minimal.

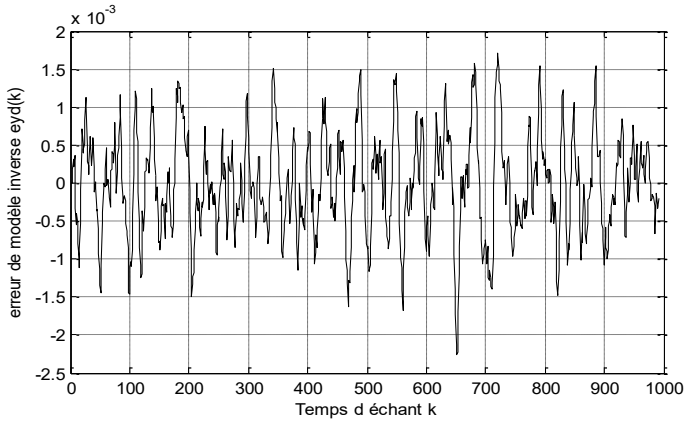

Fig. 4. Tracking error in direct control based on the inverse model (reference signal y_d).

Self-Tuning Fuzzy Inference System (STFIS)

The Self-Tuning Fuzzy Inference System (STFIS) described here follows a structure similar to a zero-order Takagi-Sugeno fuzzy inference system. Let's break down the system's architecture based on the provided description[4]:

1. Input Layer (Layer 1): This layer receives the inputs to the system. Inputs could represent variables or features relevant to the problem being addressed.

2. Fuzzification Layer (Layer 2): The second layer calculates the degrees of membership of the inputs to their respective fuzzy subsets. Each input is associated with fuzzy sets defined by membership functions. The outputs of this layer represent the degree to which the inputs belong to each fuzzy set.

The network weights between the input layer and this layer correspond to the parameters defining the membership functions. These weights are adjusted during the training process to capture the input-output relationships.

3. Rule Layer (Layer 3): The third layer calculates the truth values of the fuzzy rules based on the membership degrees obtained in the fuzzification layer. The weights between the fuzzification layer and this layer define the chosen AND operator for combining the antecedents of the fuzzy rules [4].

Each fuzzy rule typically consists of an antecedent (conditions) and a consequent (output).

4. Output Layer (Layer 4): The fourth layer represents the output layer of the system. The weights between the rule layer and this layer correspond to the conclusion parts of the rules. These weights determine the contribution of each rule to the overall output.

The output of this layer is computed based on the activations received from the rule layer, representing the system's final output.

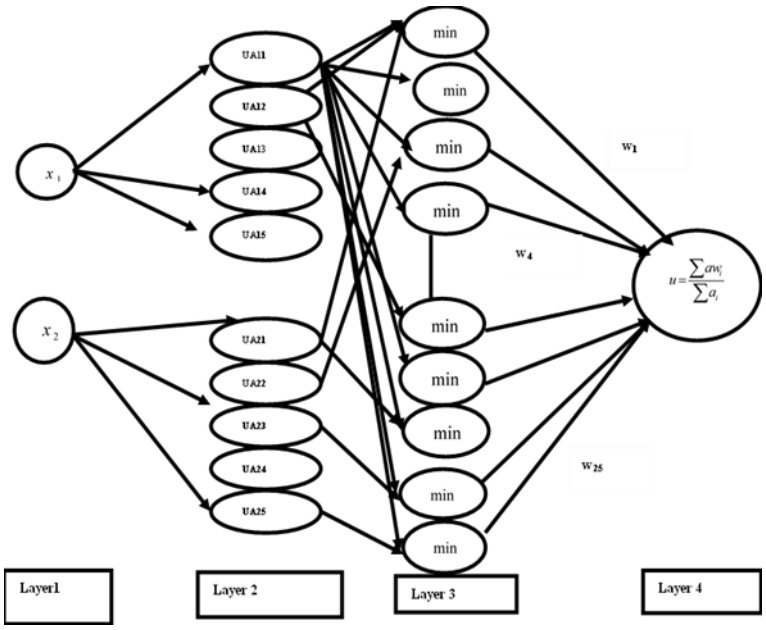

Fig. 5. Self-Tuning Fuzzy Inference System.

Architecture and learning Algorithm Architecture

Several algorithms can be used to optimize the adjustable parameters of the network. In our application, we will use the backpropagation algorithm to adjust the weights of the last layer of the network[10]. The general principle of this method can be summarized as follows: at each iteration, we modify the weights of the output layer in the opposite direction of the cost function gradient. We reiterate this process until the weights of the output layer have converged, meaning that the difference between the network output and the desired output becomes acceptable.

The optimization process in a Self-Tuning Fuzzy Inference System (STFIS) is indeed conducted online; meaning it dynamically adjusts and updates its parameters as new data is at hand. The primary goal of this optimization process is to reduce a cost function, which usually consists of two key components: Quadratic Error Term, Parameter Regression Term [11].

The combination of these two components with in the cost function, the optimization process aims to continuously refine the STFIS model to better capture the underlying relationships in the data and make more accurate predictions or decisions. This dynamic adaptation is essential for ensuring the effectiveness and adaptability of the STFIS in real-world applications [11].

$$J = E + \lambda \sum w_i^2 \tag{12}$$

$$E = \frac{1}{2}\varepsilon^2 \tag{13}$$

λ : is a constant that controls the growth parameters.

ε : error term.

w_i: layer weight.

By focusing solely on the optimization of the output conclusions, we obtain:

$$\Delta w_{1j}^4(k) = -\eta\delta_1^4\alpha_j^3 + b\Delta w_{1j}^4(k-1) - \alpha_j^3\beta w\ {}^4_{1j}(k-1)/\sum\alpha_j^3 \tag{14}$$

$$\delta_1^4 = y_e - y_d/\sum_j \alpha_j^3 \tag{15}$$

Where:

y_e: actual output value. y_d : desired output.

$\beta = 2\lambda\eta$ (regression coefficient).

α_i is the truth value of the premise part of the triggered rule.

b: Moment: This parameter ranges between 0 and 1; $b\Delta w_{ij}^n(k-1)$ helps avoid local minima by incorporating weight variations.

k : the iterations.

w_{ij}^n : Weight between then neural i[th] of layer and j[th] the neural of layer n-1.

η : Optimization gain.

For the control of the inverted pendulum, we have used the architecture known as the" mini -JEAN" as illustrated in the Fig.6. This architecture doesn't require an emulator net-work. It uses only one network as a controller, the learning of which is done directly by the back propagation of the output error[11].

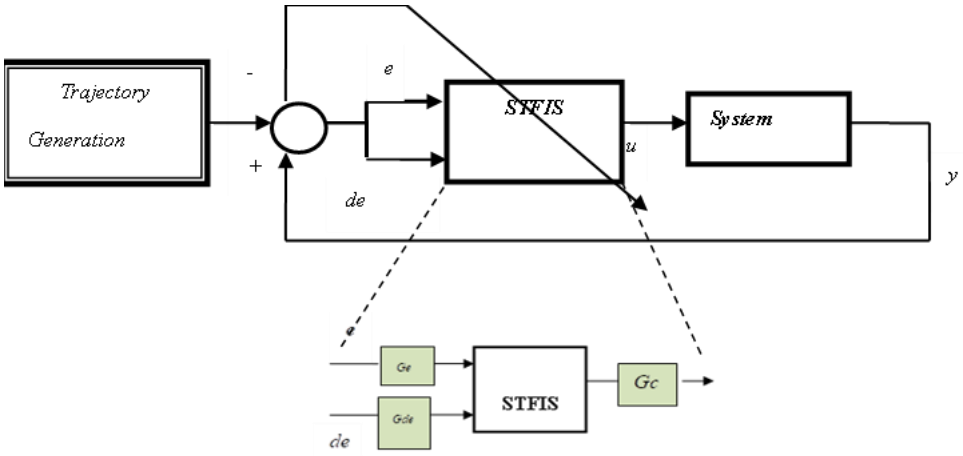

Fig. 6. control architecture mini-JEAN.

Application of the STFIS Control for Inverted Pendulum

In this work, we first apply the Self-Tuning Fuzzy Inference System (STFIS) to the inverted pendulum problem, and then integrate it into the proposed control law, as illustrated in Fig.6.

A fuzzy controller, based on online optimization of a zero-order Takagi-Sugeno inference system, is successfully implemented. It is used to minimize a cost function consisting of a quadratic error term. The controller architecture used is the well-known mini-JEAN structure, see Fig.6.

Regarding the STFIS network parameters, five membership functions of the sigmoid and Gaussian types were employed (see Fig.7). These membership functions are normalized and distributed into five subsets across the full range of displacement. The associated linguistic labels are defined as follows:

- NB: Negative Big
- NS: Negative Small
- Z: Approximately Zero
- PS: Positive Small
- B: Positive Big

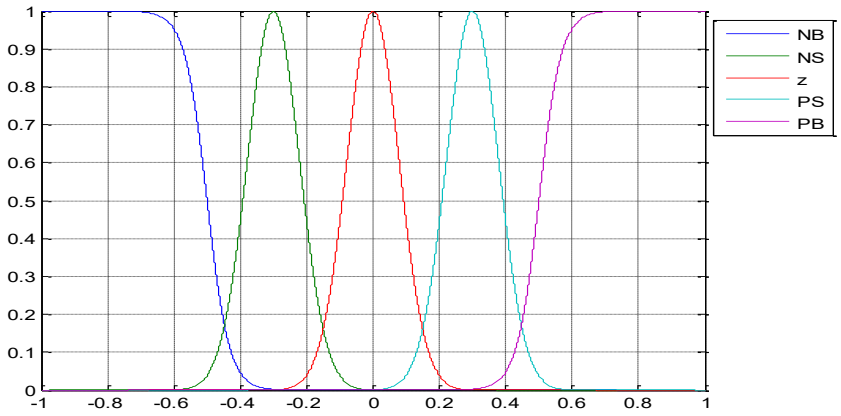

Fig. 7. Memberships function.

In order to validate our approach, we will present the parameters of the robust command in the following table:

Table 1. parameters of the controller STFIS.

The parameters of *STFIS*	η		0.3
	β		0.00006
	b		0.9
	STFIS	Ge	0.1 ,0.1,10
		$G\Delta e$	
		Gc	

Fig.8. illustrates the response of the STFIS controller applied to the inverted pendulum, following a specified desired angular trajectory. It can be observed that the actual trajectory of the pendulum closely matches the desired trajectory, even in the presence of external disturbances.

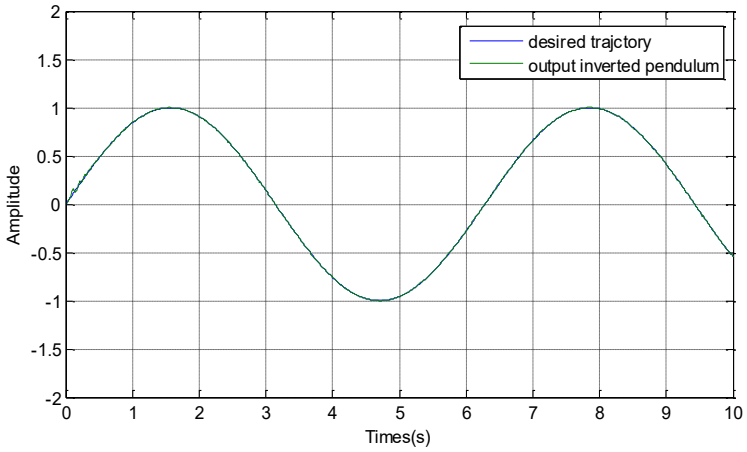

Fig. 8. Tracking trajectories for inverted pendulum.

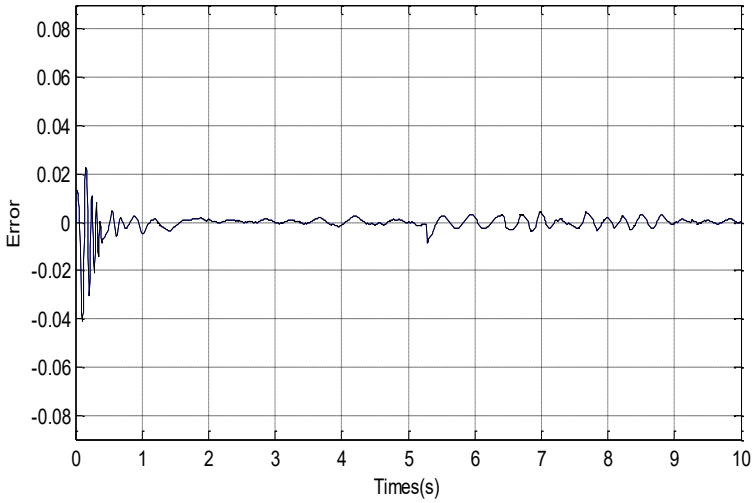

Fig. 9. Tracking errors for inverted pendulum

Based on this inference, it can be concluded that the STFIS training has been successfully achieved, and the tracking error of the inverted pendulum is nearly zero, as shown in Fig.6.

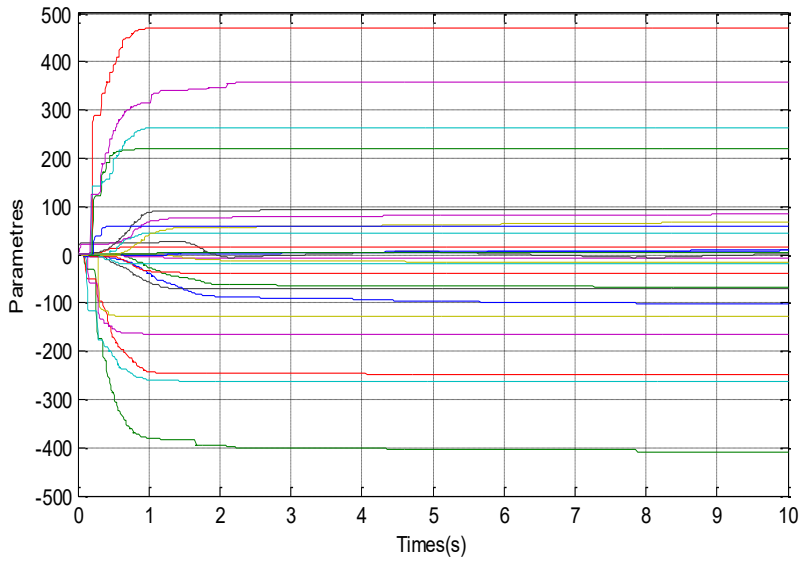

Fig. 10. The parameters (weights)

Fig. 9 illustrates the convergence of the parameters (weights) in the final layer of the STFIS network.

The STFIS controller demonstrates highly effective tracking performance and exhibits strong adaptability to varying dynamic conditions.

References

1. Davila, J.: Exact Tracking Using Backstepping Control Design and High-Order Sliding Modes. doi:10.1109/TAC.2013.2246894(2013).
2. Chen Hung, L. and Yuan Chung, L.: Decoupled sliding-mode with Fuzzy-neural network controller for nonlinear systems International Journal of Approximate Reasoning, 46 74–97(2006).
3. Krstic, M. Kanellakopoulos, I. Kokotovic, P.V.: Nonlinear and Adaptive Control Design. Wiley, New York (1995).
4. Nauck, D. and Kruse R.: What are Neuro Fuzzy Classifiers? Seventh International Fuzzy Systems Association World Congress IFSA'97, Vol. IV, pp. 228-233, Academie de Prague (1997).
5. M. A. Denaï , F. Palis, A. Zeghbib: Modeling and control of non-linear Systems using soft computing techniques. Applied Soft Computing (1997).
6. L. Ljung: System Identification. Theory for the User, Prentice Hall, (1987).
7. L. Yan and C.J. Li:Robot Learning Control Based on Recurrent Neural Network Inverse Model. Journal. of Robotic Systems, Vol. 14, pp.199-212, (1997).
8. M. Salem, D. E. Chaouch, M. F. Khelfi: Commande neuronale inverse des systèmes non linéaires .4th International Conference on Computer Integrated Manufacturing CIP, Setif, Algérie (2007).
9. Wang, W.: Adaptive Fuzzy Sliding Mode Control for Inverted Pendulum. Proceedings of the Second Symposium International Computer Science and Computational Technology (ISCSCT '09) Huangshan, P. R. China, 26, pp. 231-23428(2009).
10. Takagi, T Sugeno.: Fuzzy identification of systems and its applications to modeling and control. IEEE Transactions on systems Man and Cybernetics, vol 15, no1, pp 116-132(1985).
11. Maaref, H. Barret, C.: Progressive Optimization of a Fuzzy Inference System. IFSA-NAFIPS'2001, Vancouver, pp.665-679 (2001).