Fuzzy Modeling Using LSTM Cells for Nonlinear Systems

Francisco Vega
Departamento de Control Automatico
CINVESTAV-IPN
Mexico City, Mexico

Wen Yu
Departamento de Control Automatico
CINVESTAV-IPN
Mexico City, Mexico
yuw@ctrl.cinvestav.mx

***Abstract*—The data driven black-box and gray-box models, like the neural networks and fuzzy systems, have some disadvantages, such as the high and uncertain dimensions and complex learning process. In this paper to affront these disadvantages, we use the Takagi-Sugeno fuzzy model and LSTM cells to propose a new fuzzy-network model. This novel model takes the advantages of the interpretability of the fuzzy system and the good approximation ability of the LSTM. We also propose a fast and stable learning algorithm for this model. Comparisons with others similar black-box and grey-box models are made, in order to observe the advantages of the proposal.**

***Index Terms*—LSTM, fuzzy neural networks, nonlinear system identification.**

I. Introduction

The model of a system is the representation of the structure (properties) of the system. The choice in which a model is developed depends on what is expected to be represented in it. Obtaining models can be done in different ways, such as through physical laws (mathematical modeling); it is the most common form, but this type of technique needs knowing exactly the environment in which the system operates, as well as making the biggest amount of theoretical considerations as possible. Another way to obtain models is to include measurements of the aspects of interest, together with some equations that describe the system behaviour, achieving high robustness and adaptability generating a gray-box model. Neural networks (NNs) and fuzzy systems are very common to use as black-box models (gray-box models without equations) and gray-box models, respectively. The use of NNs and fuzzy systems can generate models with the aforementioned characteristics, either for system modeling or adaptive control.

Among the simplest ways to adjust the parameters of NNs are supervised learning algorithms, highlighting the back propagation (BP) algorithm. The BP is one of the most popular algorithms to train NNs because of its simplicity [1]. Recurrent NNs (RNNs) are the most used in automatic control, because they can generate relatively fast system models [3]. Variations of the BP algorithm have been developed to be able to adjust the parameters of RNNs efficiently, stand out the back propagation through time algorithm (BPTT) [2]. By analysing the stability of RNNs, this networks can deal with the problem of noise and disturbances [4]. This network represents that conventional RNNs can be changed into more complex structures in order to obtain better results as the case may be.

Fuzzy systems use fuzzy rules of the IF-THEN type to model systems. There are two main types of fuzzy systems, Mamdani fuzzy systems and Takagi-Sugeno (TS) fuzzy systems, with several comparisons between them were made [13]. Fuzzy systems represents experts knowledge, but they can be constructed in such a way that they emulate an expert through learning processes (like a NN) resulting in an ANFIS (adaptive network based fuzzy inference system) [14]. The ANFIS systems are based on a TS fuzzy system and transform fuzzy systems into something similar to NNs. If the consequences (THEN parts) of a TS fuzzy system are taken as nonlinear functions, it is possible to obtain better results in the general performance [15], [16]. The inclusion of NNs of different types in ANFIS systems was introduced and discussed in many works, such as [17]–[19]. More recent works on this topic propose RNNs to estimate the consequences in fuzzy systems, for example the wavelet network (WN) is used [20]. In [22] different types of fuzzy systems are applied, which are structured with RNNs and conventional representations.

Recently a deep learning model, named LSTM (long-short term memory), has been developed [5]–[7], [21]. It has a recurrent structure and is based on information management through gates, these gates measure the suitability of the data they receive as input data, the stored data by the LSTM and the data generated by the LSTM as result. LSTM networks overcome many disadvantages of RNNs and they converge relatively faster [8]–[11]. Some training algorithms specifically for LSTM networks have been proposed, to further improve their performance [12]. But as a disadvantage, the internal structure of a LSTM is more complex than the conventional RNNs. The use of deep LSTM networks is still under development, as well as their use with other intelligent systems like the fuzzy systems [23]–[25].

In order to create a network that reacts faster and with a better approximation, specially for applications in real time related to the identification and control of systems, in this paper the LSTM network is employed inside the structure of a TS fuzzy system. The novel model is established by the fuzzy system and benefited by the LSTM network estimation. A learning process for this fuzzy-network is also proposed, it performs in a short period of time and it is feasible,

978-1-7281-6932-3/20/$31.00 ©2020 IEEE

computationally speaking. The stability of the proposed model taking into account the training algorithm is proved.

To show the advantages of the novel fuzzy LSTM network, comparisons between the proposal and other intelligent algorithms are made by using the Mackey-Glass time series and a nonlinear benchmark system. These comparatives are made to show the difference in the performance between the algorithms, the proposal offers fast convergence and it can achieve easily the assigned task. The task is focussed in the generation of a model for the systems that was mentioned and, based on the results, in a future the proposed fuzzy-neural network can be used for real world applications.

II. Fuzzy modeling using LSTM cells

A system can be represented as a nonlinear function in discrete time as follows:

$$y\left(k\right)=\varphi\left[U_r\left(k\right)\right] \tag{1}$$

where $\varphi\left(\cdot\right)$ is an unknown nonlinear difference equation, also the state vector $U_r\left(k\right)$ is defined as:

$$\begin{array}{c} U_r\left(k\right)=\left[y\left(k-1\right),\cdots,y\left(k-n_y\right),u\left(k\right),\cdots\right. \\ \left.\cdots,u\left(k-n_u\right)\right]^T=\left[u_{r_1}\cdots u_{r_m}\right]^T \end{array} \tag{2}$$

with $u\left(k\right)$ and $y\left(k\right)$ as the input and the output signals for the system, n_y indicates the number of the delayed output signal, n_u indicates the number of the delayed input signal, and m indicates the number of elements u_{r_m} in $U_r\left(k\right)$.

The representation shown in (1) and (2) is known as a NARMA model. To model that system, we use fuzzy IF-THEN rules similar to a conventional TS fuzzy system, then for the p-th rule it has:

$$\begin{array}{c} R_p:\text{IF } u_{r_1}\left(k\right) \text{ IS } A_{1p} \text{ \& } u_{r_2}\left(k\right) \text{ IS } A_{2p} \text{ \& } \cdots \\ \cdots \text{ \& } u_{r_m}\left(k\right) \text{ IS } A_{jp}, \text{ THEN } h_p\left(k\right)=\varrho_p\left(k\right) \end{array} \tag{3}$$

where $h_p\left(k\right)$ is an estimation to the function $\varrho_p(k)$ that represents the consequent part of each fuzzy rule. The sets A_{jp}, with $j=1\ldots\kappa$, are the fuzzy sets for the fuzzification (using κ fuzzy sets) of each u_{r_m} in (2).

The membership functions associate to each A_{jp} are described as follows:

$$\mu_{A_{jp},u_{r_m}}\left(k\right)=\exp\left(-\frac{\left(u_{r_m}\left(k\right)-\varsigma_{jp}\right)^2}{2\nu_{jp}}\right) \tag{4}$$

In this Gaussian function, the center is $\varsigma_{jp}\in\mathbb{R}$ and the width is $\nu_{jp}\in\mathbb{R}^+$. For the final estimation of a system, the contribution of each input element to the premise part (IF part) of a fuzzy rule in (3) is obtained by the T-norm, $z_p\left(k\right)=\prod_{j=1}^{\kappa}\mu_{A_{jp},u_{r_m}}\left(k\right)$ (assuming $j=m$).

A more general representation of the value of each element of (4) in each fuzzy set can be done in a vectorial way:

$$\zeta_j=\exp\left[\left(U_r\left(k\right)-\chi_j\right)^2\otimes\left(-\frac{1}{2}\Upsilon_j\right)\right] \tag{5}$$

with χ_j, $\Upsilon_j\in\mathbb{R}^m$ as the center and width vectors for $\zeta_j\in\mathbb{R}^m$, respectively. The vector ζ_j represents the value of each element of $u_{r_m}(k)$ in fuzzy set A_j, and $\otimes$ is the operator for the element to element product in vectors. This representation will be useful for the adjustment of the parameters of the fuzzy-network.

The consequent part (THEN part) of one fuzzy rule in (3) is represented by $h_p(k)$. The function $h_p(k)$ usually is defined as a linear combination of the inputs signals (2) of the system (1), but as was said in the introduction, better estimations are achieve with the use of nonlinear functions (with the input signals as arguments); this nonlinear functions can be easily obtained by a NNs, and one of the best to do this is a LSTM cell. However, when n_y and n_u in (2) are unknown, *i.e.*, we do not known how long the current status depends on their previous information, especially when the time series is long, the information between the relevant and place becomes smaller and smaller. So we need a model which can hands the "long-term dependencies ", and LSTM cells has this property.

The estimation of (1) is obtained by the defuzzification of the fuzzy system (3) with p rules:

$$\hat{y}\left(k\right)=\frac{\sum_{n=1}^{p}z_nh_n(k)}{\sum_{n=1}^{p}z_n}=\sum_{n=1}^{p}\bar{z}_nh_n\left(k\right) \tag{6}$$

where:

$$\bar{z}_p=z_p/\left(z_1+z_2+\cdots+z_n\right)$$

The premise can be represented in a vectorial way as $Z_F\in\mathbb{R}^p$, where all the elements of this new vector are the organized multiplications as was explained in (4). Also, each element of Z_F is normalized as in (6). For multiple estimations, the elements of Z_F can be organized in such a way that the premise parts repeats for every estimation, hence the consequent parts are the only ones that are different for several estimation in a same system.

The concept of the fuzzy system using the LSTM cells is shown in Fig. 1, and it is divided in 4 layers: in the first layer the inputs of the network are organized, in the second layer this inputs are fuzzificate, in the third layer the values of the IF and THEN parts are calculated, and in the fourth layer the estimation of the system is made according to (6).

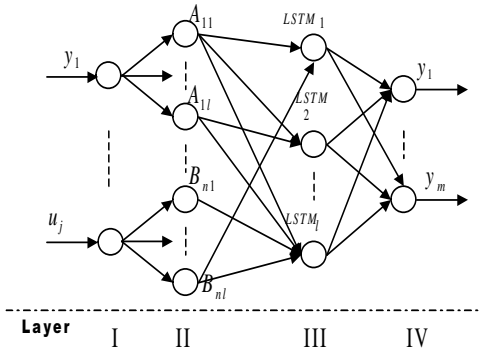

Fig. 1: Fuzzy model with LSTM cells

So, the LSTM cells in the consequent part is shown in Fig.2. The cells process data using the "gate" technique to let useful

information pass through its structure. This cell is capable of handling long-term and short-term data dependencies in more efficient way than a conventional RNN. The cells can work together, as a netwotwork and also can be organized as an array.

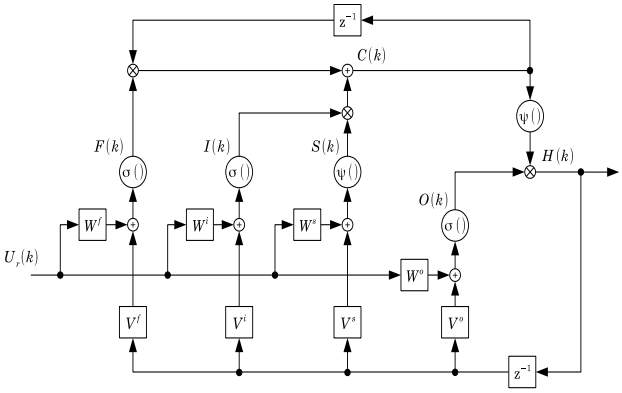

Fig. 2: LSTM cell for the consequent part.

The LSTM network has several stages, which are describe by:

$$F(k) = \sigma\left(W^f U_r(k) + V^f H(k-1)\right) \quad (7)$$

$$I(k) = \sigma\left(W^i U_r(k) + V^i H(k-1)\right) \quad (8)$$

$$S(k) = \psi\left(W^s U_r(k) + V^s H(k-1)\right) \quad (9)$$

$$C(k) = F(k) \otimes C(k-1) + I(k) \otimes S(k) \quad (10)$$

$$O(k) = \sigma\left(W^o U_r(k) + V^o H(k-1)\right) \quad (11)$$

$$H(k) = O(k) \otimes \psi\left(C(k)\right) \quad (12)$$

where: $F(k)$, $I(k)$, $S(k)$, $C(k)$, $O(k)$ and $H(k) \in \mathbb{R}^p$ are sections of the network, they are: the fitness of the internal state, the fitness of the internal input, the internal input, the internal state, the fitness of the output, and he output of the LSTM network, respectively. The synaptic weights are: W^f, W^i, W^s and $W^o \in \mathbb{R}^{p\times m}$; V^f, V^i, V^s and $V^o \in \mathbb{R}^{p\times p}$ as diagonal matrices or V^f, V^i, V^s and $V^o \in \mathbb{R}^p$ as vectors, according to the need. The functions $\sigma(\cdot)$ and $\psi(\cdot)$ are the sigmoid and hyperbolic tangent functions, respectively, $U_r(k) \in \mathbb{R}^m$ is the input in (2).

From (6), the output of the fuzzy system is

$$\hat{y}(k) = Z_F H(k) \quad (13)$$

where $H(k) = [h_1(k) \cdots h_p(k)]^T$ corresponds to the THEN parts, $Z_F \in \mathbb{R}^n$ is the elements of the IF parts. The number of LSTM cells, as well as the number of fuzzy rules, are defined as $p = \kappa^m$ for the case of 1 estimation, for several estimations it has $p = l(\kappa^m)$ where l is the number of estimations (thus $\hat{y} \in \mathbb{R}^l$), as was described for (6).

According to function approximation theories of fuzzy systems [27], the identified nonlinear process (1) can be represented as:

$$Y(k) = Z_F(W^*) H(W^*) + \mu(k) \quad (14)$$

where W^* is the unknown weights which can minimize the unmodeled dynamic $\mu(k)$. The identification error:

$$e(k) = \hat{y}(k) - y(k) \quad (15)$$

can be represented by (13) and (14)

$$e(k) = Z_F\left(\tilde{W}\right) H\left(\tilde{W}\right) + \mu(k) \quad (16)$$

where

$$Z_F\left(\tilde{W}\right) H\left(\tilde{W}\right) = Z_F(W^*) H(W^*) - Z_F H(k)$$

$\widetilde{W}(k) = W(k) - W^*$. In this paper we are only interested in open-loop identification, we assume that the plant (1) is bounded-input and bounded-output stable, *i.e.*, $y(k)$ and $U_r(k)$ in (1) are bounded. By the bound of the membership function (5), $\mu(k)$ in (14) is bounded.

III. Training of the fuzzy system

Once the structure of the fuzzy system has already been defined, it is necessary to design a training algorithm to adjust its parameters or weights. In this paper, a variation of the BPTT algorithm is chosen to train the fuzzy system. We apply a narrow "window" to apply the BPTT. This window only considers the values generated by the fuzzy LSTM network in the current iteration and its immediate past iteration. In this training method the values generated by the fuzzy LSTM network in the oldest iterations are forgotten, also this can be easily applied for online training. The training algorithm is defined by:

$$W(k+1) = W(k) + \eta_W \Delta W(k) + \alpha_W \Delta W(k-1) \quad (17)$$

where W is any synaptic weight array of the fuzzy LSTM, ΔW is the weight adjustment, $\eta_W \in (0,1]$ is the learning rate, $\alpha_W \in (0,1]$ is the momentum term for the training algorithm, and $\eta_W > \alpha_W$.

In (17), η_W determines the amount that increases or decreases each weight, while α_W helps to stabilize the modification by considering the past weight adjustment. The modelling error between the desired value and the fuzzy model is defined as:

$$\begin{aligned} \xi(k) &= \tfrac{1}{2} e^T(k)\, e(k) \\ E(k) &= \tfrac{1}{N} \textstyle\sum_{k=1}^{N} \xi(k) \end{aligned} \quad (18)$$

where $e(k)$ is the modeling error between the fuzzy model $\hat{y}(k)$ and the unknown plant $y(k)$, $\xi(k)$ is the instant error energy, $E(k)$ is the total energy during the whole processes, and N is the total number of iterations.

The modeling objective of the fuzzy system is $\min_{W(k)} \xi(k)$. The adjustment of each element of ΔW is defined as follows:

$$\Delta w_{ij}(k) = \frac{\partial \xi(k)}{\partial w_{ij}(k)} \quad (19)$$

where $\xi(k)$ is defined in (18).

The modification (19) can be obtained by the application of the the chain rule, diagrammatic rules, and the signal flow of the network and it can be organized into an array like in

(17). By the considerations made before, the adjustment of the parameters of the fuzzy LSTM network described in (4)-(13) can be easy to obtain. To illustrate this fact, for example, if we consider $m = 1$, $l = 1$ and $\kappa > 1$ the gradient (19) for each element of W^i in the consequent part is:

$$\Delta w_p^i = \frac{\partial \xi(k)}{\partial e(k)} \cdot \frac{\partial e(k)}{\partial \hat{y}(k)} \cdot \frac{\partial \hat{y}(k)}{\partial h_p(k)} \cdot \frac{\partial h_p(k)}{\partial \varepsilon_1} \cdot \frac{\partial \varepsilon_1}{\partial c_p(k)} \cdot \frac{\partial c_p}{\partial i_p(k)} \cdot \frac{\partial i_p(k)}{\partial w_p^i(k)}$$

with $\varepsilon_1 = \psi(c_p(k))$. Then, the adjustment for the matrix W^i is:

$$\Delta W^i(k) = (\dot{\sigma}(W^i U_r(k) + V^i H(k-1)) \otimes D_i) U_r(k)$$
$$D_i = S(k) \otimes \dot{\psi}(C(k)) \otimes Z_F^T e(k) \otimes O(k)$$

A similar calculation is made for the adjustment of W^f, W^s, W^o, V^f, V^i, V^s and V^o. In other hand, for the premise part, for example, the adjustment of χ_j in the membership functions of (5) are:

$$\Delta \chi_j = \frac{\partial \xi(k)}{\partial e(k)} \cdot \frac{\partial e(k)}{\partial \hat{y}(k)} \cdot \frac{\partial \hat{y}(k)}{\partial z_{Fj}} \cdot \frac{\partial z_{Fj}}{\partial \zeta_j(k)} \cdot \frac{\partial \zeta_j(k)}{\partial \chi_j} \quad (20)$$

and in a vectorial form:

$$\Delta \chi_j = (U_r(k) - \chi_j) \otimes \Upsilon_j \otimes D_\chi \otimes e(k) H(k)$$
$$D_\chi = \exp\left[(U_r(k) - \chi_j)^2 \otimes \left(-\tfrac{1}{2}\Upsilon_j\right)\right]$$

Also, something similar for Υ_j is done to compute its adjustment. As it was said before, in this paper we are only are interested in open-loop identification, we assume that the plant (1) is bounded-input and bounded-output stable, *i.e.*, $y(k)$ and $U_r(k)$ in (1) are bounded. The following theorem gives a stable gradient descent training algorithm for the fuzzy neural model.

Theorem 1: If the learning rates in the training algorithm (17) satisfy

$$\eta_{W_q}(k) = \frac{\eta}{1 + \|\mathrm{v}(\Delta W_q(k))\|^2}$$
$$\alpha_{W_q}(k) = \frac{\alpha}{1 + \|\mathrm{v}(\Delta W_q(k-1))\|^2} \quad (21)$$

where $1 \geq \eta > 0$ and $\eta \geq \alpha > 0$, W_q represents the weights arrays W^f, W^i, W^s, W^o, V^f, V^i, V^s, V^o, $\chi_1, \ldots, \chi_\kappa$, $\Upsilon_1, \ldots, \Upsilon_\kappa$, then the normalized identification error,

$$e_N(k) = \sum_{q=1}^{\rho} \left[\frac{\eta e(k)}{1 + \max_k \|\mathrm{v}(\Delta W_q(k))\|^2} + \frac{\alpha e(k)}{1 + + \max_k \|\mathrm{v}(\Delta W_q(k-1))\|^2} \right]$$

with $\rho = 8 + 2\kappa$, satisfies the following average performance

$$\limsup_{T \to \infty} \frac{1}{T} \sum_{k=1}^{T} e_N^2(k) \leq (\eta + \alpha) \bar{\mu} \quad (22)$$

where $\bar{\mu} = \max_k \left[\mu^2(k)\right]$, the unmodeled dynamic $\mu(k)$ is defined in (16).

Proof 1: To find the required stability conditions, the next Lyapunov function is given:

$$L(k) = \sum_{q=1}^{\rho} L_q(k)$$
$$L_q(k) = \mathrm{tr}\left\{\tilde{W}_q^T(k) \tilde{W}_q(k)\right\} \quad (23)$$

where the functions $L_q(k)$ are associate with W^f, W^i, W^s, W^o, V^f, V^i, V^s, V^o, $\chi_1, \ldots, \chi_\kappa$, $\Upsilon_1, \ldots, \Upsilon_\kappa$, respectively. $\tilde{W}_q(k) = W_q^* - W_i(k)$, "tr" is the denomination for the trace of a matrix.

Each element in (23) works in an independent way, and every element is defined in a similar manner. Here we only show how to prove $L_1(k)$

$$L_1(k) = \mathrm{tr}\left\{\tilde{W}^{fT}(k) \tilde{W}^f(k)\right\}$$

where $\tilde{W}^f(k) = W^{f*} - W^f(k)$, W^{f*} is the unknown optimal value of W^f. Using the trace properties: $\mathrm{tr}(A^T B) = \mathrm{tr}(AB^T) = \mathrm{tr}(B^T A) = \mathrm{tr}(BA^T)$ for any $A,\ B \in \mathbb{R}^{m \times n}$, also considering (17),

$$\begin{aligned} \Delta L_1(k) &= L_1(k+1) - L_1(k) \\ &= \mathrm{tr}\left\{\tilde{W}^{fT}(k+1) \tilde{W}^f(k+1)\right\} \\ &\quad -\mathrm{tr}\left\{\tilde{W}^{fT}(k) \tilde{W}^f(k)\right\} \\ &= L_f + L_\eta \end{aligned}$$

where by the training algorithm $W^f(k+1) = W^f(k) + \eta_{W^f} \Delta W^f(k) + \alpha_{W^f} \Delta W^f(k-1)$,

$$\begin{aligned} L_f = &-2\eta_{W^f} \mathrm{v}\left(W^{f*}\right)^T \mathrm{v}\left(\Delta W^f(k)\right) \\ &-2\alpha_{W^f} \mathrm{v}\left(W^{f*}\right)^T \mathrm{v}\left(\Delta W^f(k-1)\right) \\ &+2\eta_{W^f} \mathrm{v}\left(W^f(k)\right)^T \mathrm{v}\left(\Delta W^f(k)\right) \\ &+2\alpha_{W^f} \mathrm{v}\left(W^f(k)\right)^T \mathrm{v}\left(\Delta W^f(k-1)\right) \\ &+2\eta_{W^f}\alpha_{W^f} \mathrm{v}\left(\Delta W^f(k)\right)^T \mathrm{v}\left(\Delta W^f(k-1)\right) \\ &+\eta_{W^f}^2 \mathrm{v}\left(\Delta W^f(k)\right)^T \mathrm{v}\left(\Delta W^f(k)\right) \\ &+\alpha_{W^f}^2 \mathrm{v}\left(\Delta W^f(k-1)\right)^T \mathrm{v}\left(\Delta W^f(k-1)\right) \end{aligned}$$

and

$$\begin{aligned} L_\eta = &-\eta_{W^f} \left\|\mathrm{v}\left(W^{f*}\right)\right\|^2 - \alpha_{W^f} \left\|\mathrm{v}\left(W^{f*}\right)\right\|^2 \\ &+\eta_{W^f} \left\|\mathrm{v}\left(W^f(k)\right)\right\|^2 + \alpha_{W^f} \left\|\mathrm{v}\left(W^f(k)\right)\right\|^2 \\ &+\eta_{W^f}\alpha_{W^f} \left\|\mathrm{v}\left(\Delta W^f(k)\right)\right\|^2 \\ &+\eta_{W^f}\alpha_{W^f} \left\|\mathrm{v}\left(\Delta W^f(k-1)\right)\right\|^2 \\ &+\eta_{W^f}^2 \left\|\mathrm{v}\left(\Delta W^f(k)\right)\right\|^2 \\ &+\alpha_{W^f}^2 \left\|\mathrm{v}\left(\Delta W^f(k-1)\right)\right\|^2 \end{aligned} \quad (24)$$

here "v" is an operator that organize the elements of a matrix into a vector. If the properties

$$X^T X + Y^T Y \geq 2X^T Y, \quad X^T X = \|X\|^2$$

with $\forall X,\ Y \in \mathbb{R}^n$ are considered, then (24) becomes

$$L_\eta = -(\eta_{W^f} + \alpha_{W^f})\gamma$$

where

$$\begin{aligned} \gamma = &\left\|\mathrm{v}\left(W^{f*}\right)\right\|^2 - \left\|\mathrm{v}\left(W^f(k)\right)\right\|^2 \\ &-\eta_{W^f} \left\|\mathrm{v}\left(\Delta W^f(k)\right)\right\|^2 - \alpha_{W^f} \left\|\mathrm{v}\left(\Delta W^f(k-1)\right)\right\|^2 \end{aligned}$$

If we use (21),

$$\begin{aligned}\left\|\mathrm{v}\left(W^{f*}\right)\right\|^2 \geq \left\|\mathrm{v}\left(W^f(k)\right)\right\|^2 + \eta_{W^f}\left\|\mathrm{v}\left(\Delta W^f(k)\right)\right\|^2 \\ +\alpha_{W^f}\left\|\mathrm{v}\left(\Delta W^f(k-1)\right)\right\|^2\end{aligned}$$

so

$$\begin{aligned}\Delta L_1(k) &= L_f + L_\eta \\ &\leq -\pi_{W^f} e^2(k) + \lambda_{W^f}\mu^2(k)\end{aligned} \tag{25}$$

where π_{W^f} and λ_{W^f} are

$$\begin{aligned}\pi_{W^f} &= \frac{\eta}{1+\left\|\mathrm{v}\left(\Delta W^f(k)\right)\right\|^2} + \frac{\alpha}{1+\left\|\mathrm{v}\left(\Delta W^f(k-1)\right)\right\|^2} \\ \lambda_{W^f} &= (\eta+\alpha)\end{aligned}$$

because

$$n\min\left[\left(\tilde{W}^f\right)^2\right] \leq L_1 \leq n\max\left[\left(\tilde{W}^f\right)^2\right]$$

where $n\min\left(\left(\tilde{W}^f\right)^2\right)$ and $n\max\left(\left(\tilde{W}^f\right)^2\right)$ are $\mathcal{K}_\infty$-functions, $\pi_{W^f}e^2(k)$ is a $\mathcal{K}_\infty$-function, $\lambda_{W^f}\mu^2(k)$ is a $\mathcal{K}$-function.

So, L_1 admits a ISS-Lyapunov function, the dynamic of the identification error is input-to-state stable. Because L_1 is the function of $e(k)$ and $\mu(k)$. The "INPUT" corresponds to the second term of (25), *i.e.*, the modeling error $\mu(k)$. The "STATE" corresponds to the first term of (25), *i.e.*, the identification error $e(k)$. Because the "INPUT" $\mu(k)$ is bounded and the dynamic is ISS, the "STATE" $e(k)$ is bounded.

Continuing, (25) can be rewritten as

$$\begin{aligned}\Delta L_1 \leq \frac{\eta e^2(k)}{1+\max\limits_k\|\mathrm{v}(\Delta W^f(k-1))\|^2} \\ +\frac{\alpha e^2(k)}{+\max\limits_k\|\mathrm{v}(\Delta W^f(k))\|^2} + (\eta+\alpha)\bar{\mu}\end{aligned} \tag{26}$$

Summarizing (26) from 1 up to T, and by using $L_T > 0$ and considering L_1 as a constant, we obtain

$$L_1(T) - L_1(1) \leq -\sum_{k=1}^{T}\left\|e_N(k)\right\|^2 + T(\eta+\alpha)\bar{\mu}$$

so

$$\begin{aligned}\textstyle\sum_{k=1}^{T}\left\|e_N(k)\right\|^2 &\leq L_1(1) - L_1(T) + T(\eta+\alpha)\bar{\mu} \\ &\leq L_1(1) + T(\eta+\alpha)\bar{\mu}\end{aligned}$$

then (22) is established. □

IV. Comparisons

In this section, we talk about the performance of the proposed fuzzy LSTM network, several simulations using theoretical examples were made to determinate if this algorithm is useful for real world applications. Between the simulations, we choose two examples that stand out and these were worked in an online fashion. This chosen simulations consist in compare the results offered by our method (our fuzzy system with LSTM cells,"fuzzy LSTM") with the results offered by others establish intelligent algorithms. The other algorithms are: a RNN together with a Kalman Filter (KFRNN) [4]; a deep LSTM networks (LSTM) [9]; a zero order ANFIS system (ANFIS 0) [19]; a first order ANFIS system (ANFIS 1) [18]; a fuzzy wavelet network (fuzzy WN) [20]; and a stable fuzzy-neural network similar to the KFRNN (fuzzy KFRNN) [29].

A. Mackey-Glass time series

The first example consist on a model generation for the Mackey-Glass (MG) time-delay system, also known as MG time series:

$$\dot{x}(t) = \frac{0.2x(t-\tau)}{1+x^{10}(t-\tau)} - 0.1x(t) \tag{27}$$

with $x(0) = 1.2$, $\tau = 17$, and $(t) = 0$ for $t < 0$.

This time series is chaotic with no clearly defined period. The series does not converge or diverge, and the trajectory is highly sensitive to initial conditions. So, (27) was solved for $1,200$s, samples of the time series were taken with a sampling period $T = 1$s, creating the vector $y(k)$ with $k = 1,\ldots,1201$. We use the values of $y(k)$ to define $U_r(k) = [y(k-3), y(-20)]^T$, that was used to made the estimation $\hat{y}(k)$. We employed the first 601 iterations to train the intelligent algorithms, meanwhile the rest data were used for testing these algorithms.

We established $p = 9$ fuzzy rules for the fuzzy systems ($m = 2$, $\kappa = 3$, $l = 1$), the dimensions of the NNs were defined from several tests with different sizes and choosing the smallest NNs that offers a good performance. The comparison results are shown in the Table I. Here the modeling error $E(k)$ at the end of each phase is defined like in (18) and it represents the performance of the algorithms, a low value indicates a better performance. This table shows that all algorithms have similar performances in average, but our algorithm has little advantages than the others.

The Fig. 3 gives the modeling process of the "LSTM", the "fuzzy WN" and the "fuzzy LSTM". We can see that only our method is able to generated an acceptable model for the MG time series. Also, we only show three algorithms, because the performance of the "KFRNN" was very similar to the "LSTM", and the others fuzzy systems performance were similar with the "fuzzy WN". This example is important because we can watch the capabilities of the algorithms to generated models when we do not have access to the immediate past information of a process and when the data to construct a model are not close between them, for which the proposal overcomes the others algorithms.

TABLE I: Modeling errors of MG time series estimation ($\times 10^{-2}$)

System	Training	Testing
KFRNN	3.62	3.26
LSTM	1.09	1.53
ANFIS 0	1.70	1.16
ANFIS 1	0.98	0.82
Fuzzy KFRNN	3.47	2.15
Fuzzy WN	1.07	1.35
Fuzzy LSTM	1.41	1.03

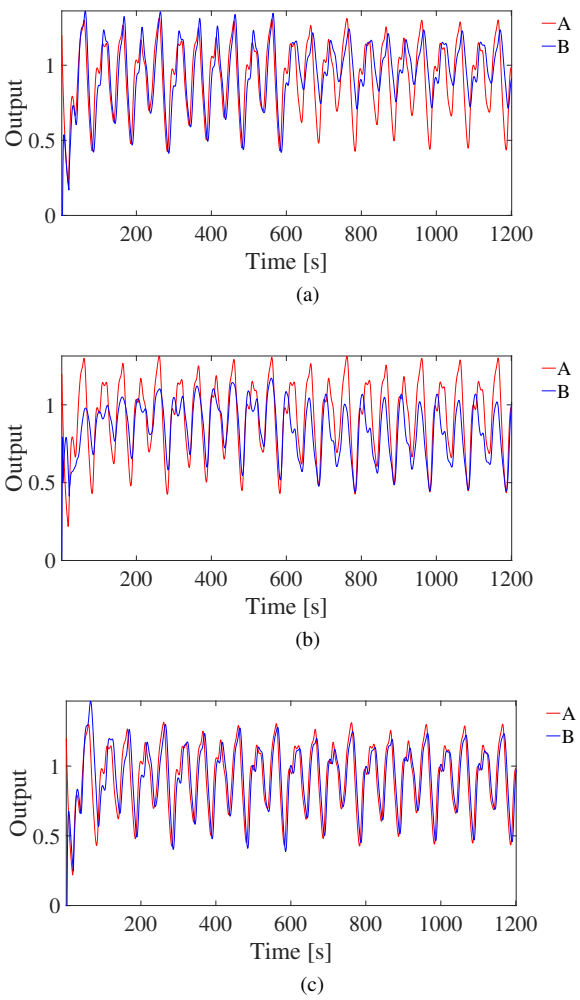

Fig. 3: Modeling of MG time series. The subfigures: (a) LSTM, (b) fuzzy WN, and (c) fuzzy LSTM. "A" is the time series response and "B" is the network response.

B. Nonlinear system

We selected the benchmark problem proposed in [28] and [26] as the second example, this problem corresponds to a MIMO (multi-input-multi-output) nonlinear system in discrete time. As in the first example, a model generation for this system is required.

So, the system is defined as:

$$\begin{aligned} y_1(k+1) &= \frac{0.5y_1(k)}{1+y_2^2(k)+u_1(k)} \\ y_2(k+1) &= \frac{0.5y_1(k)\,y_2(k)}{1+y_2^2(k)+u_2(k)} \\ y(k) &= [y_1(k), y_2(k)]^T \end{aligned} \tag{28}$$

We used different input signals for the training and the testing of (28). The training signals were:

$$\begin{aligned} u_1(k) &= 24.7\sin\left(\tfrac{2\pi kT}{10}\right) + 0.5\cos(2\pi kT) \\ u_2(k) &= 24.5\sin\left(\tfrac{\pi kT}{10}\right) + 0.5\sin(\pi kT) \end{aligned} \tag{29}$$

and the testing signals were:

$$u_1(k) = \begin{cases} \text{If } 0 < kT \leq 50, \\ u_1 = 3.3\sin\left(\tfrac{2\pi kT}{10}\right) + 0.1\cos(2\pi kT) \\ \text{If } 50 + n_{u_1} < kT \leq 55 + n_{u_1}, \\ u_1 = 3.5 \\ \text{If } 55 + n_{u_1} < kT \leq 60 + n_{u_1}, \\ u_1 = -3.5 \\ \text{If } 100 < kT, \\ u_1 = 3.6\cos\left(\tfrac{2\pi kT}{10}\right) \end{cases}$$

$$u_2(k) = \begin{cases} \text{If } 0 < kT \leq 50, \\ u_2 = 3.5\sin\left(\tfrac{\pi kT}{10}\right) + 0.3\sin(\pi kT) \\ \text{If } 50 + n_{u_2} < kT \leq 55 + n_{u_2}, \\ u_2 = -3.5 \\ \text{If } 55 + n_{u_2} < kT \leq 60 + n_{u_2}, \\ u_2 = 3.5 \\ \text{If } 100 < kT, \\ u_2 = 3.6\cos\left(\tfrac{2\pi kT}{10}\right) \end{cases} \tag{30}$$

with $n_{u_1} = n_{u_2} = 10,\ 20,\ 30,\ 40$.

Similar to the past example, the vector $y(k) = [y_1(k), y_2(k)]^T$ was constructed by taking samples of the system with a sample period $T = 0.01$s, the input vector was defined as $U_r(k) = [u_1(k), u_2(k)]$. To simulate perturbations, random values in $[-0.5, 0.5]$ were added to U_r and $y(k)$ in the training phase and random values in $[-0.2, 0.2]$ were added to U_r and $y(k)$ in the testing phase of the algorithms. While the. In this example, we used $p = 18$ fuzzy rules for the fuzzy systems ($m = 2$, $\kappa = 3$, $l = 2$), the dimensions of the NNs were defined from several tests with different sizes and choosing the smallest NNs that offers a good performance.

We simulated the system in the following way: we train the algorithms to learn the system (28) with (29) during 180s, obtaining $18,001$ iterations for the training process, a testing is made immediately after the training with the same input signal during 60s (6,001 iterations). Also, a testing with a different input from the training (30) was made during 180s, obtaining $18,001$ iterations for this process.

In the Table II are shown the modeling errors, according to (18), obtained for each intelligent algorithm at the end of the training and testing phases. In this table, the NNs seem to have a better performance than the fuzzy systems, in the sense that this algorithms converges fast and offers a lower modeling error. Only our proposal has a similar (even slightly better) performance that the NNs.

TABLE II: Modeling errors of the nonlinear system estimation ($\times 10^{-2}$)

Model	Training	Testing	
		After training	Other input
KFRNN	52.56	57.95	17.95
LSTM	48.21	58.53	16.94
ANFIS 0	303.21	315.14	127.88
ANFIS 1	102.80	104.38	59.86
Fuzzy KFRNN	182.69	223.30	22.93
Fuzzy WN	1,382.32	1,070.40	48.20
Fuzzy LSTM	43.69	42.06	8.04

The Fig. 4 and Fig.5 give the modeling processes of the "fuzzy WN" and the "fuzzy LSTM". We show the "fuzzy WN" and the "fuzzy LSTM" because for this example all the other fuzzy systems had a similar performance that the "fuzzy WN", and the other neural model ahad a similar performance the "fuzzy LSTM". So, our proposal can generate an acceptable model for nonlinear systems with fast convergence, like a NN but offering a more complete approach, a gray box model instead of a black box model.

As shown in above figures and tables, the proposal model offers very good modelling results for the time series and the nonlinear system. Also it has better robustness and adaptability. It has been shown that our method has better testing results for multi-step prediction, or when some recent data are not available.

V. Conclusions

In this paper, a novel fuzzy-neural network is proposed, this based on the LSTM networks. Also, it can be interpreted as a more complete LSTM network, because the data with which the network is fed are subjected to a better analysis due to the characteristics of the fuzzy systems and LSTM networks. We design a fast training method for this fuzzy LSTM network, the stability of the proposed training method is also given. We use two examples to compare our model with the other intelligent algorithms, the results show that the new model is faster and has better performance that the other algorithms for nonlinear system identification. With this, our proposal can be use for real world applications in the future.

References

[1] R. Chaudhary, H. Patel, and M. Scholar, A survey on backpropagation algorithm for neural networks,Int. J. Technol. Res. Eng, vol.2, no. 7, 2015.

[2] H. Jaeger, Tutorial on training recurrent neural networks, covering BPPT, RTRL, EKF and the echo state network approach. GMD-Forschungszentrum Informationstechnik Bonn, 2002, vol. 5.

[3] Z. C. Lipton, J. Berkowitz, and C. Elkan, A critical review of recurrent neural networks for sequence learning , arXiv preprint arXiv:1506.00019, 2015.

[4] Wen Yu, Nonlinear system identification using discrete-time recurrent neural networks with stable learning algorithms, Information Sciences, Vol.158, No.1, 131-147, 2004.

[5] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling , arXiv preprint arXiv:1412.3555, 2014.

[6] H. Salehinejad, S. Sankar, J. Barfett, E. Colak, and S. Valaee, Recent advances in recurrent neural networks , arXiv preprint arXiv:1801.01078, 2017.

[7] S. Hochreiter and J. Schmidhuber, Long short-term memory , Neural computation, vol. 9, no. 8, pp. 1735-1780, 1997.

[8] O. Ogunmolu, X. Gu, S. Jiang, and N. Gans, Nonlinear systems identification using deep dynamic neural networks , arXiv preprint arXiv:1610.01439, 2016.

[9] Y. Wang, A new concept using lstm neural networks for dynamic system identification , in 2017 American Control Conference (ACC). IEEE, 2017, pp. 5324 -5329.

[10] F. Nicola, Y. Fujimoto, and R. Oboe, A lstm neural network applied to mobile robots path planning , in 2018 IEEE 16th International Conference on Industrial Informatics (INDIN). IEEE, 2018, pp.349 - 354.

[11] Y. Liu, Y. Zhou, and X. Li, Attitude estimation of unmanned aerial vehicle based on lstm neural network , in 2018 International Joint Conference on Neural Networks (IJCNN). IEEE, 2018, pp. 1-6.

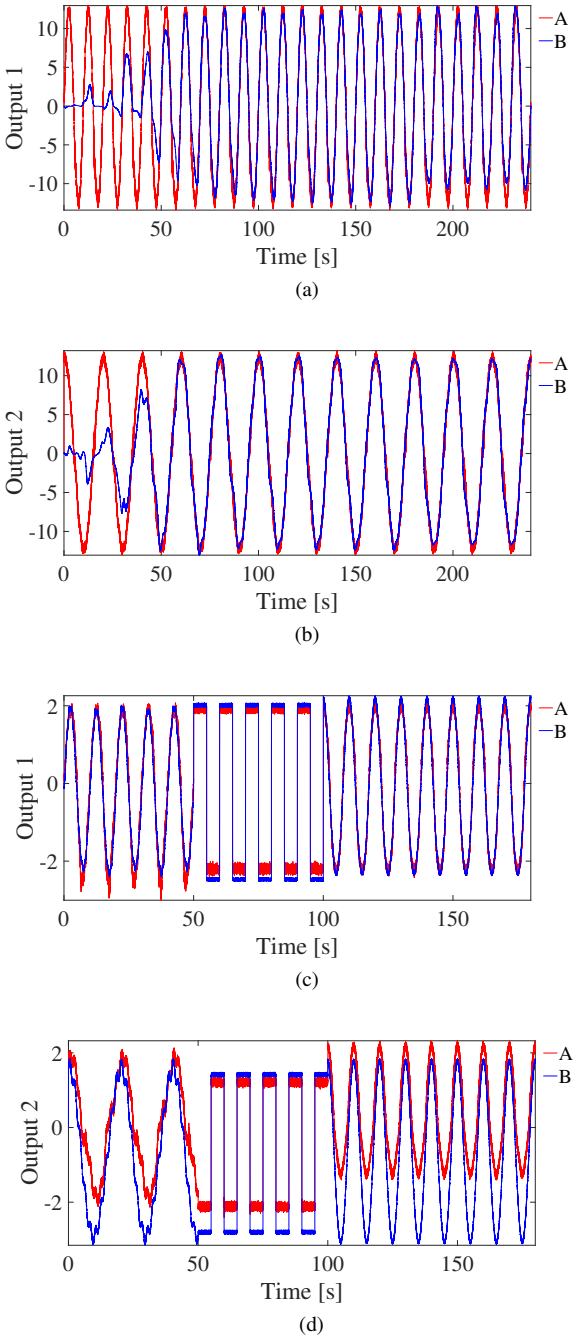

Fig. 4: Nonlinear system modeling with "fuzzy WN". The subfigures: (a) $y_1(k)$ in the training, (b) $y_2(k)$ in the training, (c) $y_1(k)$ in the testing, (d) $y_2(k)$ in the testing. "A" is the system response and "B" is the model response.

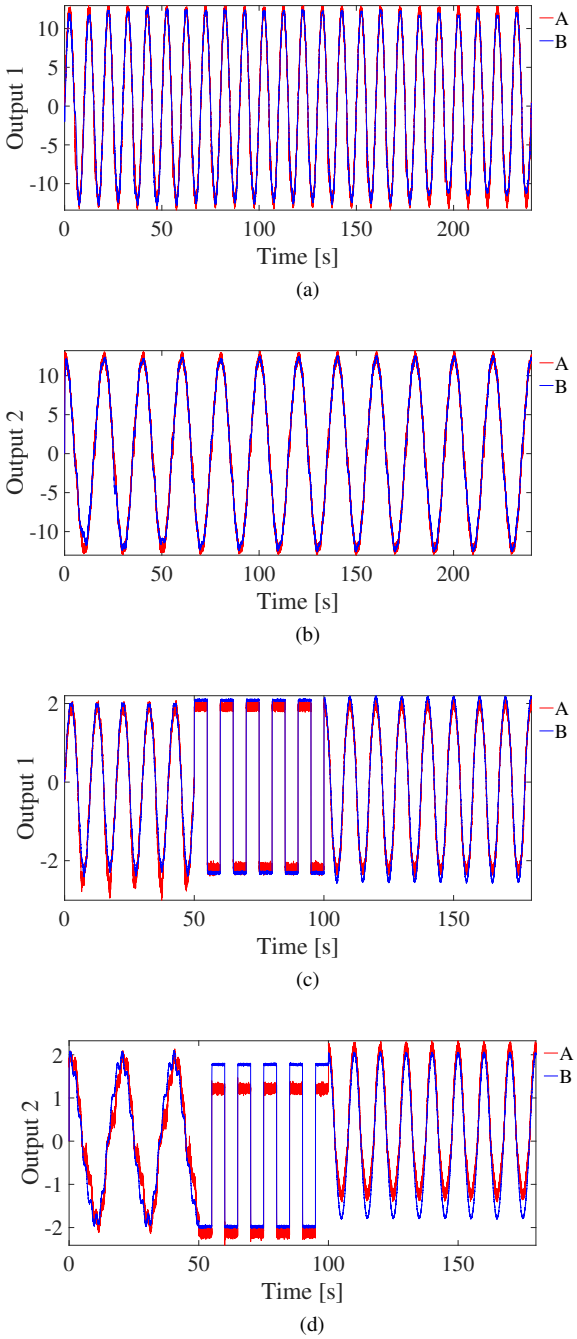

Fig. 5: Nonlinear system modeling with "fuzzy LSTM". The subfigures: (a) $y_1(k)$ in the training, (b) $y_2(k)$ in the training, (c) $y_1(k)$ in the testing, (d) $y_2(k)$ in the testing. "A" is the system response and "B" is the model response.

[12] T. Ergen and S. S. Kozat, Effcient online learning algorithms based on lstm neural networks , IEEE transactions on neural networks and learning systems, vol. 29, no. 8, pp. 3772-3783, 2017.

[13] M. Blej and M. Azizi, Comparison of mamdani-type and sugeno-type fuzzy inference systems for fuzzy real time scheduling , International Journal of Applied Engineering Research, vol. 11, no. 22, pp. 11071 -11075, 2016.

[14] J.-S. Jang, Anfis: adaptive-network-based fuzzy inference system , IEEE transactions on systems, man, and cybernetics, vol. 23, no. 3, pp. 665 -685, 1993.

[15] J. Dong, Y. Wang, and G.-H. Yang, Output feedback fuzzy controller design with local nonlinear feedback laws for discrete-time nonlinear systems , IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 40, no. 6, pp. 144-41459, 2010.

[16] J. Kabzi ski and J. Kacerka, Tsk fuzzy modeling with nonlinear consequences , in IFIP International Conference on Artificial Intelligence Applications and Innovations. Springer, 2014, pp. 498 -507.

[17] J. M. BenÃ-tez, J. L. Castro, and I. Requena, Are artificial neural networks black boxes?, IEEE Transactions on neural networks, vol. 8, no. 5, pp. 1156 -1164, 1997.

[18] R. Babu ka and H. Verbruggen, Neuro-fuzzy methods for nonlinear system identification , Annual reviews in control, vol. 27, no. 1, pp. 73 -85, 2003.

[19] Y. Jin and B. Sendhoff, Extracting interpretable fuzzy rules from rbf networks , Neural Processing Letters, vol. 17, no. 2, pp. 149 -164, 2003.

[20] S. Ganjefar and M. Tofighi, Single-hidden-layer fuzzy recurrent wavelet neural network: Applications to function approximation and system identification , Information Sciences, vol. 294, pp. 269 -285, 2015.

[21] Wen Yu, Jose de Jesus Rubio, Recurrent neural networks training with stable bounding ellipsoid algorithm, IEEE Transactions on Neural Networks, Vol.20, No.6, 983-991,2009

[22] K. Shihabudheen and G. Pillai, Recent advances in neuro-fuzzy system: A survey , Knowledge Based Systems, vol. 152, pp. 136 -162, 2018.

[23] A. I. Aviles, S. M. Alsaleh, E. Montseny, P. Sobrevilla, and A. Casals, A deep-neuro-fuzzy approach for estimating the interaction forces in robotic surgery , in 2016 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE). IEEE, 2016, pp. 1113 -1119.

[24] H. Sang, C. Yang, F. Liu, J. Yun, and G. Jin, A fuzzy neural network sliding mode controller for vibration suppression in robotically assisted minimally invasive surgery , The International Journal of Medical Robotics and Computer Assisted Surgery, vol. 12, no. 4, pp. 670-679, 2016.

[25] H. M. Sri, P. Rao, P. K. Kammardi, S. S. Shekar, S. Kathavate, and K. Gowranga, A smart adaptive lstm technique for electrical load forecasting at source , in 2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT). IEEE, 2017, pp. 1717 - 1721.

[26] P. Sastry, G. Santharam, and K. Unnikrishnan, Memory neuron networks for identification and control of dynamical systems , IEEE transactions on neural networks, vol. 5, no. 2, pp. 306-319, 1994.

[27] L.X.Wang, *Adaptive Fuzzy Systems and Control*, Englewood Cliffs NJ: Prentice-Hall, 1994.

[28] K.S.Narendra and S.Mukhopadhyay, Adaptive Control Using Neural Networks and Approximate Models, *IEEE Trans. Neural Networks*, Vol.8, No.3, 475-485, 1997.

[29] Wen Yu, Xiaoou Li, Fuzzy identification using fuzzy neural networks with stable learning algorithms, IEEE Transactions on Fuzzy Systems, Vol.12, No.3, 411-420, 2004.