Learning and Nonlinear Models – Revista da Sociedade Brasileira de Redes Neurais, Vol. 1, No. 1, pp. 1-9, 2003

NEURO-FUZZY MODELLING AND CONTROL OF NONLINEAR DYNAMIC SYSTEMS

G. Quadrelli [1], R. Tanscheit [2], M. M. Vellasco [2]
[1] Depto. de Eng. Elétrica, UCP, CP 90.944, 25685-070 Petrópolis, RJ
[2] Depto. de Eng. Elétrica, PUC-Rio, CP 38063, 22452-970 Rio de Janeiro,RJ
E-mails: giovane.quadrelli@ucp.br, [ricardo, marley]@ele.puc-rio.Br

Abstract – The main goal of this paper is to propose procedures for modelling and control of nonlinear systems by using neuro-fuzzy topologies. For the modelling of a nonlinear system, its input space is initially divided into a number of fuzzy operating regions, within which reduced order models represent the system's behaviour. The complete system modelling – the global model – is obtained through the conjunction of the local models by using a neuro-fuzzy network. A neuro-fuzzy adaptive network, based on a hybrid learning algorithm (self-organised learning and supervised learning) and called FALCON-H, is used in the control of a nonlinear plant modelled as described above.

Keywords – Neuro-fuzzy systems, nonlinear systems, modelling, control

1. INTRODUCTION

Supervision and control of advanced processes frequently require high accuracy in their modelling and representation. Several industrial processes exhibit nonlinear dynamics, which introduces additional complexity in modelling procedures. Neural networks can approximate a wide range of nonlinear functions and have been successfully used in the modelling of nonlinear dynamic systems. However, results may be difficult to interpret, since a neural network performs as a "black box". Fuzzy logic has been widely used in control, but definition of appropriate rules and of membership functions is hardly a straightforward task [1].

Neural networks extract information from historical data of the systems to be modelled or controlled, while fuzzy inference systems often make use of linguistic information obtained from specialists. They can be combined in order to take advantage of their individual characteristics, giving origin to neuro-fuzzy systems. These systems have the learning and optimisation abilities of neural networks, as well as the capability of incorporating knowledge – structured as if-then rules – of fuzzy systems. On the neural network side the system becomes more transparent, whereas on the fuzzy logic side a learning capability [2] is introduced.

Based on approaches for modelling [3] and control [4] of nonlinear plants, this work presents a neuro-fuzzy procedure for both modelling and control of a nonlinear dynamic plant. It is shown through an application that combined neuro-fuzzy modelling and control can produce good results.

2. NEURO-FUZZY MODELLING OF THE PLANT

In practice many nonlinear processes are approximated by reduced order models, in general linear ones. However, these models may be valid only within certain operating ranges. When operating conditions change, a different model or changes in the parameters may be required. An alternative approach considers several operating regions and the use of local, reduced order models for approximation in each of the regions. These local models may be of ARMAX (auto-regressive, moving average with exogenous input) form.

The definition of operating conditions is often inherently vague. In general it is difficult to define precisely the operating regions and there may be overlappings between them. Fuzzy sets provide appropriate means for the definition of such regions. Takagi and Sugeno [5] have proposed an approach

where the input space is divided into various fuzzy regions and a local linear model is used in each of them; the overall output is obtained through a defuzzification procedure.

In this work a neuro-fuzzy network is used for computing the local linear models within an integrated system that represents the whole, nonlinear, system [3]. This network combines the neural network's capability of learning through examples with the fuzzy system's ability to deal with imprecise information. The fuzzy model structure is mapped onto a neural network, whose structure is determined by the fuzzy rules. Previous knowledge of the process is used for the initial definition of operating regions, as well as for weight initialisation. Input and output data are then used for training the neuro-fuzzy network.

2.1 Fuzzy modelling of nonlinear processes

The global operation of a nonlinear process is divided into several local regions. In each region R_i a linear, reduced order model in ARMAX form represents the process behaviour. This is not a restrictive condition; any other appropriate linear model may be used. In this work, for example, the MA part (random variable) is not considered Fuzzy sets are used for defining operating conditions, so that the dynamic model of a nonlinear process can be described by [3]:

R_i : IF operating condition *i* THEN $\quad y_i^p(t) = \sum_{j=1}^{no} a_{ij} y(t-j) + \sum_{j=1}^{ni} b_{ij} u(t-j) \qquad i = 1, 2, ..., nr$

where y: process output
u: process input
nr:number of fuzzy operating regions
ni: time lag in the input
no: time lag in the output
t: discrete time
a_{ij}, b_{ij}: model parameters
y_i^p : prediction of process output in the i-th operating region

The model output y^p is obtained through defuzzification where μ_i is the membership function for the i-th region/model:

$$y^p(t) = \sum_{i=1}^{nr} \mu_i y_i^p(t) \Big/ \sum_{i=1}^{nr} \mu_i \qquad (1)$$

Assume that x and y are the process variables used for defining operating regions and that they are assigned fuzzy sets *low*, *medium* and *high*. The i-th operating region can be defined, for example, as x is *high* AND y is *medium*. The membership function for this region can be built in various ways. Two possible approaches are:

$$\mu_i = \min(\mu_h(x), \mu_m(y)) \qquad (2)$$
$$\mu_i = \mu_h(x)\mu_m(y) \qquad (3)$$

In equations (2) and (3), μ_i is the membership function of the i-th region, $\mu_h(x)$ and $\mu_m(y)$ are the membership functions of x being *high* and for y being *medium*, respectively.

2.2 Representation of fuzzy models as neural networks

The fuzzy model described above can be represented by a special type of network topology shown in Figure 1 [3], where connections between neurons are hypothetical; they serve only as an illustration. The network consists of four layers, each of them representing one feature of a fuzzy inference system: fuzzification, rules, functions and defuzzification.

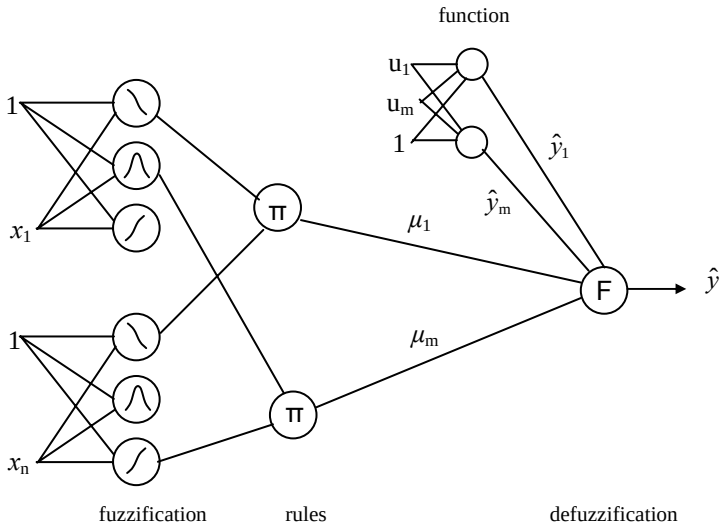

Figure 1: Neuro-fuzzy network

Inputs to the fuzzification layer are process variables used for defining fuzzy operating regions. Each neuron in this layer corresponds to a particular fuzzy set with the membership function being given by the neuron output. Three types of activation functions are used: sigmoidal, gaussian and complement sigmoidal.

Each neuron in the rules layer corresponds to an operating region of the process being modelled. Its inputs are the fuzzy actions, which determine the corresponding operating region. Its output is the membership function of the corresponding fuzzy operating region. Neurons in this layer implement the fuzzy intersection defined by equation (3). Previous knowledge of the number of operating regions and of how these are established is used for building the rules layer. There are no weights to be adjusted in this layer.

Neurons in the functions layer are linear and implement the reduced order models in the fuzzy operating regions. Each neuron corresponds to a particular region and its output is the sum of its inputs, which are process variables multiplied by their respective weights. A bias is included in each neuron to allow for situations in which there is a constant term in the local model. The weights are the parameters of the linear models in the operating regions.

The final output is computed in the defuzzification layer. The inputs to the defuzzification neuron are the membership functions of the fuzzy operating regions and the local models outputs in these regions. The activation function is given by equation (1) and there are no weights in this layer.

2.3 Training of the neuro-fuzzy network

The neuro-fuzzy network can be trained through various methods; in this work backpropagation is used. The network weights, given by equations (4) and (5) below, are adjusted by an algorithm that minimises the sum of squared errors, in accordance with equation (6).

$$\Delta\omega(k+1) = \alpha\Delta\omega(k) - \eta\delta \tag{4}$$

$$\omega(k+1) = \omega(k) + \Delta\omega(k+1) \tag{5}$$

$$J = \sum_{i=1}^{N} (y_{\mathrm{i}}^{\mathrm{p}} - y_{\mathrm{i}})^2 \tag{6}$$

In equations (4) and (5) $\omega(k)$ e $\Delta\omega(k)$ are the weight and weight adaptation at step k, respectively; α is the momentum coefficient, η is the learning rate and δ is the gradient of the sum of squared error with respect to the weight ω. In equation (6) N is the number of data points, y^p is the network prediction and y is the target value. As with other gradient-based methods, training ends when the error falls below a pre-specified value.

3. THE FALCON-H CONTROLLER

The FALCON-H neuro-fuzzy controller (Fuzzy Adaptive Learning CONtrol network) has been proposed by Lin and Lee to study Hybrid learning paradigms, which involve structure and parameters learning capability [4]. The FALCON-H controller is a feedforward multi-layer network that implements all basic modules of a fuzzy controller in a connectionist structure by using the learning ability of artificial neural networks.

In this connectionist structure, input and output nodes represent state variables and the output control variable respectively. Nodes in the hidden layers represent functional nodes, such as membership functions and fuzzy rules.

3.1 FALCON-H structure

The FALCON-H structure, depicted in Figure 2 [4], has a total of five layers. Each layer in the feedforward topology can be associated with a module of a fuzzy control system. The input nodes in layer 1 are related to linguistic variables. In this work, these variables are associated with error and change in error. Layer 2 consists of the linguistic terms of each input variable and is associated with the fuzzification module of a fuzzy control system. Layers 3 and 4 on their turn are related to the antecedents and consequents of the fuzzy rules, computing the T-norm (AND connective) and the linguistic terms of the output variable respectively. Finally, the defuzzification process is performed in layer 5, generating the final output variables. However, during the training process, layer 5 can work in both directions, as will be explained in the next sections.

To formally describe the functionality of each layer, let's assume that an integration function is associated to the inputs to each node. This function provides the *net* input, as described in equation (7):

$$net_i = f(u_1^{(k)}, u_2^{(k)}, ..., u_p^{(k)}, \omega_1^{(k)}, \omega_2^{(k)}, ..., \omega_p^{(k)}) \quad (7)$$

where u_i are the inputs to node i, ω_i are the associated link weights and the superscript k indicates the layer number.
The output of node i (o_i) is calculated by the activation function a:

$$o_i^k = a(net_i) = a(f) \quad (8)$$

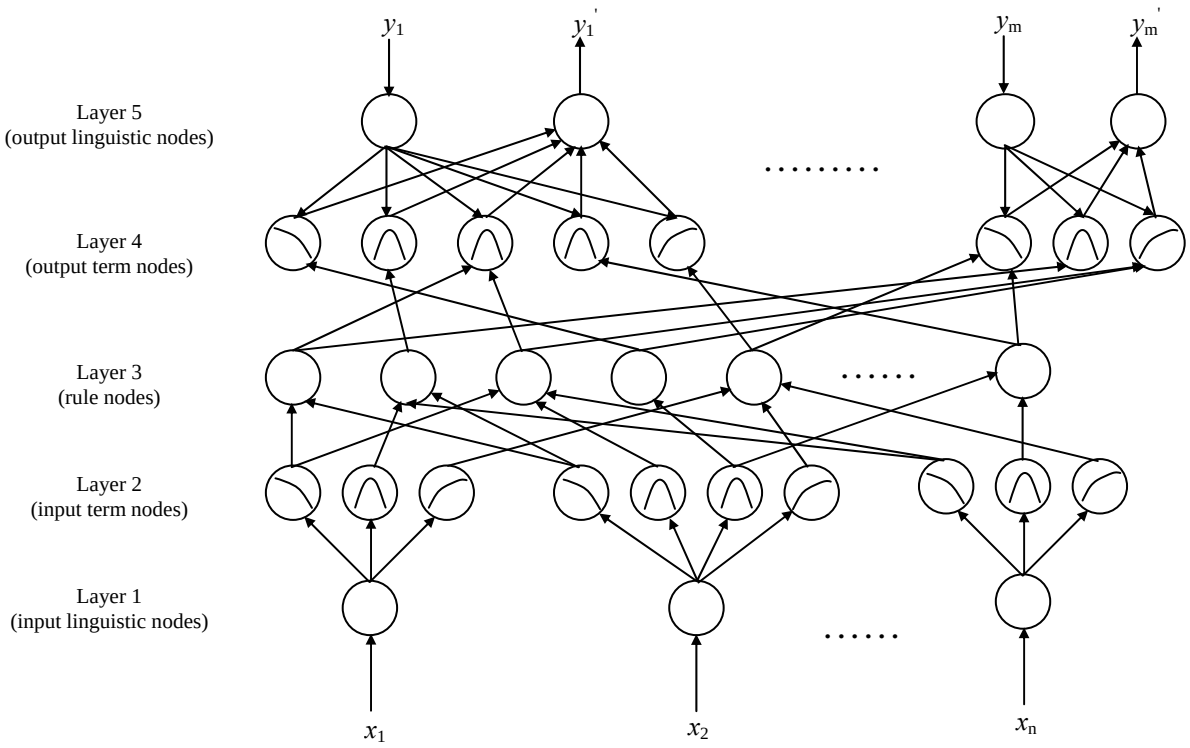

Figure 2: FALCON-H structure

3.1.1 FALCON-H layers

This section describes the functionality of nodes in each of the five layers of FALCON-H.

Layer 1 $\Rightarrow$ Input linguistic variables: directly transmit the input values to layer 2.

$$f = u_i^{(1)} \text{ and } a = f \tag{9}$$

The link weights of each node in layer 1 are equal to 1 ($\omega_i^{(1)} = 1$).

Layer 2 $\Rightarrow$ Fuzzification layer: each node represents a membership function of an input variable. The membership function is usually represented by a bell-shaped function (gaussian):

$$f = M_{xi}^{j}(m_{ij}, \sigma_{ij}) = -(u_i^{(2)} - m_{ij})^2 / \sigma_{ij}^2 \tag{10}$$

where $a = e^f$

m_{ij} = centre (mean) of the bell-shaped function of the *j*th linguistic term of the *i*th input variable [4]

σ_{ij} = width (variance) of the bell-shaped function of the *j*th linguistic term of the *i*th input variable [4]

The weight of each link between layer 1 and 2 $\omega_{ij}^{(2)}$ can be interpreted as m_{ij}.

Layer 3 $\Rightarrow$ Rule nodes: the links in this layer represents the rules antecedent, that is, the AND fuzzy operation:

$$f = \min(u_1^{(3)}, u_2^{(3)}, ..., u_p^{(3)}) \tag{11}$$

where: $a = f$ and $\omega_i^{(3)} = 1$

Layer 4 $\Rightarrow$ Membership functions of the output variables. In this specific layer, the nodes operate in two different modes:

- *Down-up Mode*: The nodes perform the fuzzy OR operation to integrate fired rules with the same consequent [4]:

$$f = \sum_{i} u_i^{(4)} \tag{12}$$

where: $a = \min(1, f)$ and $\omega_i^{(4)} = 1$

- *Up-down Mode*: the nodes in this mode work exactly the same as those in layer 2, with only one single membership function for output variable.

Layer 5 $\Rightarrow$ Output Variables – this layer has two different types of nodes:

- *Up-down Nodes*: used to train the network structure, in the phase 1 of the learning process:

$$f = y_i \text{ and } a = f \tag{13}$$

- *Down-up Nodes*: Defuzzifier nodes – when the centre of area is used, equation (14) is applied:

$$f = \sum \omega_{ij}^{(5)} u_{ij}^{(5)} = \sum (m_{ij}\sigma_{ij}) u_{ij}^{(5)} \tag{14}$$

where: $a = f \Big/ \sum_{j} \sigma_{ij} u_{ij}^{(5)}$ and $\omega_{ij}^{(5)} = m_{ij}\sigma_{ij}$

3.2 Hybrid Learning Algorithm

The FALCON-H training process is performed in two distinct phases: an *unsupervised* algorithm for structure learning; and a *supervised* algorithm (Back Propagation) for parameter tuning. This hybrid learning algorithm gives better results than a purely supervised (back-propagation) one because of the a priori classification of training data through an overlapping receptive field before executing the supervised learning phase [4]. The unsupervised learning phase is used to learn the rules structure and the initial format of the membership functions. The second phase, based on supervised learning, is executed to tune the inputs and outputs membership functions.

3.2.1 Unsupervised Learning

In this initial phase, the network operates in a *two-sided* form, that is, the nodes and links in layer 4 are in the *up-down* transmission mode. In this configuration, the input and output training data can be feed into the FALCON-H controller from both sides.

First, the mean and width of each membership function is determined by a self-organising algorithm, similarly to statistical clustering techniques. This is used to specify the domain of all membership functions, covering only those regions of the input-output space where data are present [4]. The Kohonen algorithm is applied in this case to establish the mean m_i of the ith membership function of x, where x represents any input $(x_1,..., x_n)$ or output $(y_1,.., y_m)$ variable:

$$\| x(t) - m_{closest}(t) \| = \min_{1 \le i \le k} \{ \| x(t) - m_i(t) \| \} \tag{15}$$

$$m_{closest}(t+1) = m_{closest}(t) + \alpha(t)[x(t) - m_{closest}(t)] \tag{16}$$

$$m_i(t+1) = m_i(t) \text{ for } m_i \neq m_{closest} \tag{17}$$

where $\alpha(t)$ is a monotonically decreasing learning rate.

Once the centres of the membership functions have been determined, their widths can be computed through the N-nearest-neighbour algorithm. Since the second phase will optimally adjust the parameters of the membership functions, this calculation can be simplified by the use of the following equation:

$$\sigma_i = | m_i - m_{closest} | / r \tag{18}$$

where an initial value for the intersection parameter r is set by the user.

Once the initial membership functions formats have been established, the process of rule creation begins. The basic idea is to determine the correct consequent link of each rule node (layer 4). Again a competitive learning process is used; more details can be found in [4]. After that, a rule combination is

used to reduce the number of rules. The criteria for combining a set of rule nodes are: those that have exactly the same consequents; some antecendents are common to all rule nodes; and the union of other antecedents encompasses the whole linguistic terms of some input variables.

3.2.2 Supervised Learning

In the second phase of the hybrid learning algorithm, after the whole network structure has been established, the neuro-fuzzy structure processes data in a *feedforward* mode – nodes and links in layers 4 and 5 are in the *down-up* transmission mode. The gradient descent algorithm is used (Back Propagation) to adjust the parameters of the input and output membership functions. The complete computation of the derivatives for determination of the learning rule for each layer can be found in [4].

4. CASE STUDY

The proposed neuro-fuzzy system (for modelling and control) has been applied to a dynamic nonlinear process which relates the percentage of carbonic gas (%CO_2) to gas flow rate (feet^3/min) in a specific plant [6], as shown in Figure 3.

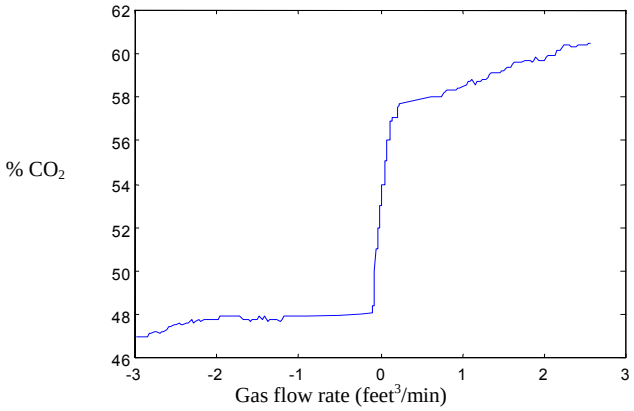

Figure 3: Relation between %CO_2 and the gas flow rate.

4.1 Neuro-fuzzy modelling

The first neuro-fuzzy model was developed to model the relation between %CO_2 and the gas flow rate. The process was divided into three operating regions: low %CO_2, medium %CO_2 and large %CO_2. Two hundred points were used for training and another 200 points for testing the neuro-fuzzy model. The training parameters used in the learning process were: learning rate (η) = 0.1, momentum (α) = 0.2; and maximum gradient error = 10^{-3}. The following fuzzy model was obtained:

R1: If %CO_2 is low: $y(t) = 0.9996\,y(t-1) - 0.0170\,u(t-1)$

R2: If %CO_2 is medium: $y(t) = 1.0015\,y(t-1) - 0.1813\,u(t-1)$

R3: If %CO_2 is high: $y(t) = 1.001\,y(t-1) - 0.0143\,u(t-1)$

The performance of the neuro-fuzzy plant for the testing data set is illustrated in Figure 4. The Mean Absolute Percentage Error (MAPE = 1.33%) was used as a performance measure.

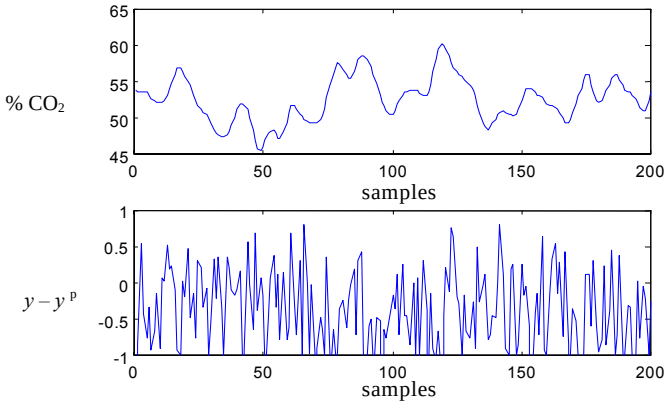

Figure 4: Performance of the neuro-fuzzy plant.

4.2 Neuro-fuzzy control of the plant

The FALCON-H neuro-fuzzy controller was used to control the neuro-fuzzy plant in a direct fashion, giving origin to the result depicted in Figure 5. The setpoint is a step varying from 60% CO_2 to 50% CO_2. The final FALCON-H structure is the following:

Layer 1: two input nodes — error (e) and change in error (Δe);
Layer 2: seven nodes with gaussian membership functions for each input variable from layer 1;
Layer 3: 49 rule nodes (AND);
Layer 4: seven nodes with gaussian membership functions;
Layer 5: one output node (control action).

In order to generate the control action (manipulated input) for training, a procedure suggested in [3] has been employed. In this procedure, given the setpoint and a desirable output behaviour , a quadratic objective function is minimised, giving the manipulated input as a result. As in the neuro-fuzzy plant, 200 points were used for training and testing, with learning rate η=0.15, intersection parameter r = 2 and gradient error = 10^{-2}.

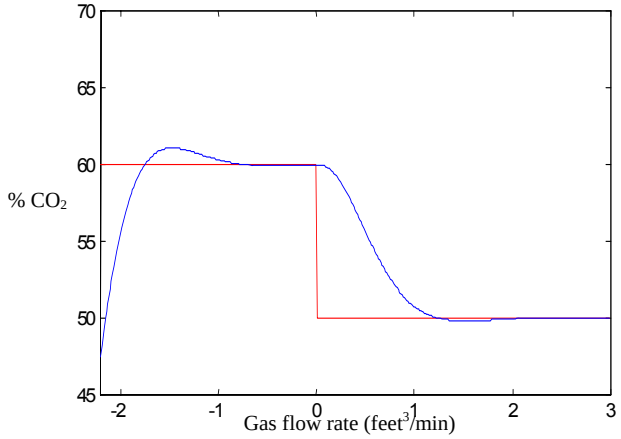

Figure 5: Performance of the FALCON Neuro-Fuzzy controller

5. CONCLUSION

This paper presented a new approach that combines neuro-fuzzy structures for both modelling and control of nonlinear systems. The neuro-fuzzy approach combines the advantages of fuzzy inference systems and artificial neural networks, integrating interpretability and learning capabilities. In addition, neuro-fuzzy systems create the fuzzy rule base automatically, eliminating the difficulty of having an expert to express the rule base.

The results obtained demonstrate that the neuro-fuzzy approach is effective for both modelling and controlling the nonlinear dynamic process and that the combined approach can be a good alternative when a mathematical model of the plant is not available.

6. REFERENCES

[1] J. R. Jang and C. T. Sun. Neuro-fuzzy modeling and control. *Proc. IEEE*, 83(3): 378-406, 1995.

[2] M. Figueiredo and F. Gomide. Design of fuzzy systems using neurofuzzy networks. *IEEE Trans. on Neural Networks*, 10(4): 815-827,1999.

[3] J. Zhang and A. J. Morris. Fuzzy neural networks for nonlinear systems modelling. Proc. *IEEE - Control Theory Applications*, 142(6): 551-561, 1995.

[4] C. T. Lin and C. S G. Lee. *Neural fuzzy systems*. Prentice Hall, 1996.

[5] T. Takagi and M. Sugeno. Fuzzy identification of systems and its aplication to modelling and control. *IEEE Transactions on Systems, Man and Cybernetics,* 15, (1): 116-132, 1985.

[6] G. E. P. Box, G. M. Jenkins, G. C. Reinsel. *Time Series Analysis*. Prentice Hall, 1994.