

Parameter Identification of Recurrent Fuzzy Systems with Fuzzy Finite-State Automata Representation

Carlos A. Gama, Alexandre G. Evsukoff, Philippe Weber and Nelson F. F. Ebecken, *Senior Member, IEEE*

Abstract— This paper presents the identification of non-linear dynamical systems by recurrent fuzzy system models. Two types of recurrent fuzzy systems (RFS) models are discussed, the Takagi-Sugeno-Kang (TSK) type and the linguistic or Mamdani type. Both models are equivalent and the latter model may be represented by a fuzzy finite-state automaton. An identification procedure is proposed based on a standard general purpose genetic algorithm. First, the TSK rule parameters are estimated and, in a second step, the TSK model is converted into an equivalent linguistic model. The parameter identification is evaluated in some benchmark problems for non-linear system identification described in literature. The results show that RFS models achieve good numerical performance while keeping the interpretability of the actual system dynamics.

Index Terms— Fuzzy finite-state automata, genetic algorithms, nonlinear systems, system identification, recurrent fuzzy systems.

I. INTRODUCTION

NONLINEAR system identification has been extensively studied during the last decades and several methods and applications have been developed [1]. Multilayer feed-forward neural networks is certainly the most usual method for the identification of non-linear dynamical systems [2]. Feed forward neural networks are generally used as one step-ahead prediction models, in which the previous system outputs are available and are used as network inputs to compute the model output at the next sampling time step. Multi-step ahead prediction is achieved by additional output units to the neural network.

For complex system forecasting, the use of long delayed inputs to provide enough memory to the model and multiple output units results in a very large network size. Moreover, feed-forward neural networks cannot be used for simulation, when actual system outputs are not available.

Recurrent topologies allow a more compact model for processing dynamic systems, with good results for long term forecasting and simulation [3][4]. In a recurrent neural network, the system memory is achieved by feeding back some (or all) of the network units through time delay units.

Fuzzy and neuro-fuzzy models for nonlinear dynamic system identification have been also studied, both for feed-forward [5] and recurrent topologies [6]-[14]. Several recurrent fuzzy and neuro-fuzzy models have been developed in many applications, such as process modeling [7]-[9], identification and control [10]-[13] and fault diagnosis [14]. Neuro-fuzzy models provide high accuracy and “human-like” interpretation of the model by a set of linguistic rules.

The encoding of recurrent neural networks as fuzzy finite-state automata (FFA) has been recently investigated [15]-[18]. Such relationship is useful, since the

representation capabilities and theoretical properties derived from the FFA can be used to analyze and understand the dynamics of the identified model. Adamy and Kempf [19] show that recurrent fuzzy systems can be directly encoded into a FFA and their dynamical behavior may be deduced from their rule base configuration [20]. Fernández and López [21] show that, under certain assumptions, recurrent fuzzy systems are asymptotically stable, independently of the initial state. These results, when combined with the universal approximation property of general fuzzy systems [22][23], yield to the perspective that recurrent fuzzy systems identification and FFA representation may provide an efficient, accurate and sound procedure for identification and analysis of nonlinear dynamic systems.

The identification algorithms for recurrent neural networks and recurrent fuzzy and neuro-fuzzy models generally rely on gradient-based methods. Nevertheless, gradient calculations for recurrent neural networks are complex and highly dependant on the network topology. Recently, genetic algorithms (GAs) have also been used for model identification [24]-[26]. A GA does not require derivative information and avoids the problem of local minima usually found in gradient based optimization. Thus, the use of GA makes the model identification more flexible, since several different model structures may be identified by the same procedure.

Two types of recurrent fuzzy systems (RFS) are discussed in this work. The first type is the Takagi-Sugeno-Kang (TSK) RFS, which has been introduced by Gorrini and Bersini [6]. The second type, called linguistic or Mamdani RFS, is an extension of the model proposed by Adamy and Kempf [19][20], by the introduction of rule base weights. Both models are equivalent and the latter can be represented by a FFA, in which linguistic terms, associated to the fuzzy sets, are related to the automaton

alphabet and fuzzy inference correspond to the automaton transition function.

A parameter estimation procedure is proposed for the RFS models discussed in this work. The identification algorithm is based on a standard general purpose GA and is composed of two steps. In the first step a GA is used to compute the TSK-type RFS output parameters [26]. In the second step, the result obtained in the first step is used to compute the rule base weights of an equivalent linguistic type RFS model.

This paper is organized as follows: next section presents the TSK and linguistic types RFS and the representation of the latter as a FFA; section three presents the identification algorithm; the results of the proposed methodology applied to system identification benchmark problems are presented in section four and the conclusions and future works are discussed in the last section.

II. RECURRENT FUZZY SYSTEMS

Consider the general state space representation of discrete time, non-linear, multi input multi output dynamic systems:

$$\mathbf{x}(t+1) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)), \quad (1)$$

$$\mathbf{y}(t) = \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t)) \quad (2)$$

where $\mathbf{x}(t)$ is the state variables vector, $\mathbf{u}(t)$ the input variables vector, \mathbf{f} and \mathbf{g} are respectively the transition and output functions and $\mathbf{y}(t)$ is the system output.

The state space representation is quite general and can represent several types of dynamic systems according to the ranges of input and state variables and the choice of the transition and the output functions. Continuous valued input and state variables are used to model linear and nonlinear systems according to the choice of functions \mathbf{f} and \mathbf{g} . Discrete valued input, state variables and discrete functions \mathbf{f} and \mathbf{g} are used to model discrete systems, represented by discrete finite-state automata. In a recurrent fuzzy system (RFS) model, input and output variables are continuous valued and functions \mathbf{f} and \mathbf{g} are nonlinear piecewise continuous functions. Linguistic interpretation of fuzzy sets allows recurrent fuzzy models to be analyzed as fuzzy finite-state automata (FFA).

This work is mainly concerned with the identification of a RFS to model the transition function \mathbf{f} . The output function is considered to be a linear function of states, such that:

$$\mathbf{y}(t) = \mathbf{K}\mathbf{x}(t). \quad (3)$$

Two types of recurrent fuzzy systems are discussed in this section: the TSK-type recurrent fuzzy system and the linguistic or Mamdani type recurrent fuzzy system. The two models are equivalent and a representation of the linguistic type RFS as a FFA is proposed.

A. TSK-type RFS model

The TSK-type RFS model is an extension of the original TSK (or Sugeno) model in which the variable in the conclusion of the rules is a state variable, which also appears in the premise, providing recursion. For sake of simplicity, the TSK-type model is introduced for the first order, single input model and then generalized for higher order, multiple input models.

The first order, single input, TSK-type RFS rules are written as:

$$\text{If } x(t) \text{ is } A_i \text{ and } u(t) \text{ is } B_j \text{ then } x(t+1) = \theta_r, \quad (4)$$

where $x(t)$ is the state variable and $u(t)$ is the input variable. The fuzzy sets A_i and B_j are defined, respectively, on the state and input variables domains and θ_r is the output parameter for each one of the $r = 1 \dots M$ rules.

A fuzzy set is intrinsically¹ related to a symbolic label, which is, in turn, associated to a meaningful linguistic concept. The fuzzy partitions of the state and input variables domains are thus respectively related to the label sets $\mathbf{A} = \{A_i, i = 1 \dots p\}$ and $\mathbf{B} = \{B_j, j = 1 \dots q\}$. The ordered symbolic labels in the labels sets will then be assigned to the state and the input alphabets of the FFA representation, as described in section II.C.

The fuzzy membership functions define the relationship between the numeric and the symbolic representation of a variable. In this work, fuzzy membership functions are considered to be continuous, convex, triangular-shaped and normalized, such that the following conditions hold:

$$\sum_i \mu_{A_i}(x) = 1, \forall x \quad (5)$$

$$\forall i, \exists x : \mu_{A_i}(x) = 1 \quad (6)$$

Normalized and triangular-shaped membership functions are completely defined by the vector $\mathbf{a} = (a_1, \dots, a_p)$, which locates the membership functions vertices, called the prototypes of the corresponding fuzzy sets. Consider, for instance the fuzzy partition shown in Fig. 1 for $p = 5$ fuzzy sets.

The fuzzification of input and state variables yields the fuzzification (row) vectors:

$$\boldsymbol{\mu}_{\mathbf{A}}(x(t)) = (\mu_{A_1}(x(t)), \dots, \mu_{A_p}(x(t))) \text{ and} \quad (7)$$

$$\boldsymbol{\mu}_{\mathbf{B}}(u(t)) = (\mu_{B_1}(u(t)), \dots, \mu_{B_q}(u(t))). \quad (8)$$

The RFS model output is the state variable $x(t+1)$, which is computed as in standard TSK models as:

¹ In a strictly formal notation, a fuzzy set should be noted differently from the label to which it is related. However, as it is generally the case in the fuzzy systems literature, the same notation is used to refer to the fuzzy set and to its label, the difference being clear in the context.

$$x(t+1) = \mathbf{w}(t)\boldsymbol{\theta} \quad (9)$$

where $\boldsymbol{\theta} = (\theta_1, \dots, \theta_M)$ is the output parameters vector and $\mathbf{w}(t)$ is the vector whose components are the activation values of the rules premises.

The components of the vector $\mathbf{w}(t)$ are computed by the conjunction of the fuzzy sets in the premise. In this work, the product operator is used as t-norm such that the vector $\mathbf{w}(t)$ can be computed by the Kronecker tensor product as:

$$\mathbf{w}(t) = \boldsymbol{\mu}_A(x(t)) \otimes \boldsymbol{\mu}_B(u(t)). \quad (10)$$

All the combinations of fuzzy sets are considered in the rules premises, such that the model covers the entire domain. The number of rules M is defined by the number of fuzzy sets in the state and input variables fuzzy partitions; considering p fuzzy sets in the state variable fuzzy partition and q fuzzy sets in the input variables fuzzy partition, the total number of rules in the first order single input TSK-type RFS is $M = p \cdot q$.

More complex dynamics may be achieved by higher order RFS models. In this case, n state variables are considered in the vector $\mathbf{x}(t) = (x_0(t), \dots, x_{n-1}(t))$. In order to reduce the number of parameters, at each simulation time step, higher order state variables are updated as delayed copies of the state variable $x_0(t)$ such that:

$$x_k(t) = x_0(t-k), \quad k = 1 \dots n-1. \quad (11)$$

Adopting this representation, all the state variables are considered in the rule premises, but only the value of the state variable $x_0(t)$ is computed in the output of the RFS, such that the rules are written as:

$$\text{If } \mathbf{x}(t) \text{ is } C_i \text{ and } u(t) \text{ is } B_j \text{ then } x_0(t+1) = \theta_r. \quad (12)$$

The fuzzy set C_i in rule (12) is a multidimensional fuzzy set computed as the Cartesian product of fuzzy sets in the state variables domain. All combinations must be considered in such a way that the entire state variables domain is covered. The symbolic label set $\mathbf{C} = \{C_i, i = 1 \dots p^n\}$ is thus the Cartesian product of the elements of the state variable the label set \mathbf{A} , such that:

$$\mathbf{C} = \mathbf{A} \times \dots \times \mathbf{A} = \mathbf{A}^n. \quad (13)$$

As the state variables are delayed copies of the state variable $x_0(t)$, the same fuzzy partition can be used to compute all the state variables membership vectors, avoiding additional parameters to be set. The membership vector $\boldsymbol{\mu}_C(\mathbf{x}(t))$ is computed as the Kronecker tensor product of the corresponding components on each state variable domain, as:

$$\boldsymbol{\mu}_C(\mathbf{x}(t)) = \boldsymbol{\mu}_A(x_{n-1}(t)) \otimes \dots \otimes \boldsymbol{\mu}_A(x_0(t)) \quad (14)$$

where $\boldsymbol{\mu}_A(x_k(t))$ is the fuzzification vector of the state variable $x_k(t)$, computed as (7).

The same reasoning can be adopted to extend the single input TSK-type RFS models to multiple input RFS models, of which the rules are written as:

$$\text{If } \mathbf{x}(t) \text{ is } C_i \text{ and } \mathbf{u}(t) \text{ is } B_j \text{ then } x_0(t+1) = \theta_r \quad (15)$$

where $\mathbf{u}(t) = (u_1(t), \dots, u_m(t))$ is the input variables vector.

The input membership vector $\boldsymbol{\mu}_B(\mathbf{u}(t))$ can be computed as the combination of fuzzy sets in the input variables domain, as in the case of the state variables, but it will generally result in a large number of rules. It is preferable to compute the input membership vector directly over the multidimensional domain by fuzzy clustering methods [5]. This approach becomes useful as the number of input variables increases, since the number of rules, and consequently the number of output parameters, grows exponentially with the number of variables

In order to simplify the notations, the number of fuzzy sets in the input variable fuzzy partition (single or multi-dimensional) is considered to be q . The total number of rules in the multi-variable TSK-type RFS model is thus $M = p^n \cdot q$, where n is the model order and p the number of fuzzy sets in the states variable fuzzy partition.

The fuzzy model output $x_0(t+1)$ is then computed as:

$$x_0(t+1) = \mathbf{w}(t)\boldsymbol{\theta}, \quad (16)$$

where $\boldsymbol{\theta} = (\theta_1, \dots, \theta_M)$ is the output parameter vector and $\mathbf{w}(t)$ is the rules activation vector, which is computed as:

$$\mathbf{w}(t) = \boldsymbol{\mu}_C(\mathbf{x}(t)) \otimes \boldsymbol{\mu}_B(\mathbf{u}(t)) \quad (17)$$

Comparing (9) and (10) with (16) and (17), it can be observed that the extension from the first order single input model to higher order multiple input model is straightforward.

The TSK-type RFS can be extended to more complex models by using polynomial functions in rules outputs [24]. This work focuses only the models with constant values in the rules output, which can be equivalent to the linguistic RFS model as presented as follows.

B. Linguistic recurrent fuzzy model

In linguistic recurrent fuzzy systems, the output of each rule is a fuzzy expression on the state variable $x_0(t)$, such that the rules are written as:

$$\text{If } \mathbf{x}(t) \text{ is } C_i \text{ and } \mathbf{u}(t) \text{ is } B_j \text{ then } x_0(t+1) \text{ is } A_k \quad (18)$$

where C_i and B_j are multi-dimensional fuzzy sets like before, and A_k is a fuzzy set defined on the state variables domain.

The rule base can be represented by the fuzzy relation

matrix Φ , of which the components $\phi_{rk} \in \{0,1\}$ represent the relationship between the premise and the output expressions. There is one rule $r=1\dots M$ for each combination (C_i, B_j) of the terms in the premise, in such a way that $\phi_{rk} = 1$ if the output of the rule r is the fuzzy set A_k and $\phi_{rk} = 0$, otherwise. The matrix Φ has, thus, M lines and p columns.

The output of the linguistic recurrent fuzzy system is computed as an ordinary fuzzy system, choosing the sum-product inference and singleton defuzzification [19]-[21]. The result of sum-product inference is computed as:

$$\mathbf{v}(t) = \mathbf{w}(t) \cdot \Phi, \quad (19)$$

where $\mathbf{w}(t)$ is computed as above.

The state variable value at the next simulation time step is computed as:

$$x_0(t+1) = \mathbf{v}(t) \cdot \mathbf{a}^T = \mathbf{w}(t) \cdot \Phi \cdot \mathbf{a}^T, \quad (20)$$

where the vector $\mathbf{a} = (a_1, \dots, a_p)$ is the state variable prototype (row) vector used in singleton defuzzification operation, where the superscript T indicates the transpose.

The linguistic recurrent fuzzy model described above is equivalent to the one investigated by Adamy and Kempf [19]. In this work, a more flexible model is adopted associating a confidence factor to each rule, represented by the weight $\phi_{rk} \in [0,1]$. In this case, the rules are written as:

If $\mathbf{x}(t)$ is C_i and $\mathbf{u}(t)$ is B_j then

$$x_0(t+1) \text{ is } (A_1 / \phi_{r1}, \dots, A_p / \phi_{rp}). \quad (21)$$

The output of the weighted linguistic recurrent fuzzy model is also computed by (20).

Comparing the (16) and (20), the TSK-type RFS and the linguistic type RFS are equivalent when fuzzy rules weights are related to the output parameters as:

$$\boldsymbol{\theta} = \Phi \cdot \mathbf{a}^T. \quad (22)$$

The linguistic RFS model described above is derived from the numerical values of the state and input variables. A dual symbolic representation of the linguistic RFS allows it to be seen as a fuzzy finite-state automaton, as described next.

C. FFA representation

At each simulation time, the state variable value $x_0(t)$ yields to the symbolic state S_t , which is a fuzzy subset of the state variable label set \mathbf{A} . The membership values of the symbolic state S_t to each label $A_i \in \mathbf{A}$ are defined by the fuzzification vector $\boldsymbol{\mu}_{\mathbf{A}}(x_0(t))$ as:

$$\mu_{S_t}(A_i) = \mu_{A_i}(x_0(t)), \quad i = 1, \dots, p. \quad (23)$$

For higher order models, the multidimensional state variable $\mathbf{x}(t)$ is described by the symbolic state Q_t , which is a fuzzy subset of the symbolic label set \mathbf{C} , of which the membership values are defined as:

$$\mu_{Q_t}(C_i) = \mu_{C_i}(\mathbf{x}(t)), \quad i = 1, \dots, p^n. \quad (24)$$

Analogously, the symbolic input U_t describes the input variable value $\mathbf{u}(t)$ as a fuzzy subset of the input variable label set \mathbf{B} , of which the membership values are defined by fuzzification vector $\boldsymbol{\mu}_{\mathbf{B}}(\mathbf{u}(t))$ as:

$$\mu_{U_t}(B_j) = \mu_{B_j}(\mathbf{u}(t)), \quad i = 1, \dots, q. \quad (25)$$

The rule base weights matrix Φ is a fuzzy relation defined as a fuzzy subset of $(\mathbf{C} \times \mathbf{B}) \times \mathbf{A}$, such that a membership value $\phi_{rk} \in [0,1]$ is assigned to each combination of labels $((C_i, B_j), A_k)$ in the premise and conclusion of rule (21).

The fuzzy (symbolic) inference computes the next simulation time symbolic state S_{t+1} based on rule base weights matrix as:

$$S_{t+1} = (Q_t \times U_t) \circ \Phi. \quad (26)$$

where “ \circ ” stands for the sum-product composition operator and “ \times ” is the fuzzy Cartesian product, which is computed by the product t-norm operator.

If the rule base weights are defined such that conditions (5) and (6) hold for the inference result $\mathbf{v}(t)$ in (19), then $\mathbf{v}(t) = \boldsymbol{\mu}_{\mathbf{A}}(x_0(t+1))$ and it can be fed back directly into the RFS, without defuzzification. In this case, the symbolic states computed by symbolic inference (26) are coherent with (19) such that:

$$\mu_{S_{t+1}}(A_i) = \mu_{A_i}(x_0(t+1)), \quad i = 1, \dots, p. \quad (27)$$

Considering the update rule (11), the multivariate symbolic state Q_{t+1} may be computed directly by the fuzzy Cartesian product of the symbolic states $S_{t-i}, i = 0, \dots, n-1$ as:

$$Q_{t+1} = S_{t-n} \times \dots \times S_{t+1}. \quad (28)$$

Equations (26) and (28) define the function $\delta: (\mathbf{C} \times \mathbf{B}) \times \mathbf{C} \rightarrow [0,1]$, based on the rule base weights matrix Φ , which can be regarded as the transition function of a fuzzy finite-state automaton.

A fuzzy finite-state automaton is usually defined as a six-tuple $F = \langle Q, \Sigma, \delta, Q_0, Z, \omega \rangle$ [27][28], where Q is a finite set of symbolic states; Σ is a finite set of input symbols; $\delta: Q \times \Sigma \times Q \rightarrow [0,1]$ is the fuzzy finite-state automaton transition function that computes the next state membership values based upon the current state and the input; $Q_0 \in Q$

the initial state; Z is a finite set of output symbols and $\omega: Q \rightarrow Z$ is the output function that maps the states into the output set.

The dual symbolic derivation of the linguistic RFS model can be seen as a special type of fuzzy finite-state automaton, of which both the states and the inputs are fuzzy subsets of the corresponding numeric variables' label sets. The output set and the output function are not considered in this fuzzy automaton representation, since the RFS models only the state space transition function (1). Therefore, the fuzzy automaton corresponding to the linguistic RFS is the four-tuple $F = \langle \mathbf{C}, \mathbf{B}, \delta, Q_0 \rangle$, where:

- the set of symbolic states is the set of symbolic labels $\mathbf{C} = \{C_i, i = 1 \dots p^n\}$, of which the symbolic state Q_t is a fuzzy subset;
- the input alphabet is the input variables label set $\mathbf{B} = \{B_j, j = 1 \dots q\}$, of which the symbolic input U_t is a fuzzy subset;
- the transition function $\delta: (\mathbf{C} \times \mathbf{B}) \times \mathbf{C} \rightarrow [0,1]$ is computed by (26) and (28), and
- the initial fuzzy state Q_0 is a fuzzy subset of the set of the symbolic states \mathbf{C} , whose membership values are computed as the fuzzification of the state variables at initial simulation time $\mathbf{x}(t=0)$, such that $\mu_{Q_0}(C_i) = \mu_{C_i}(\mathbf{x}(0))$.

The RFS model defined by non-weighted fuzzy rules as (18) is a special case where the rule base weights $\varphi_{rk} \in \{0,1\}$. Moreover, when input and states variables values are constraint to the prototypes of the corresponding fuzzy sets, the non-weighted linguistic RFS model behaves like the deterministic finite-state automaton with no uncertainty in the activation of the states. In this case, only one symbolic state of the fuzzy automaton is activated at each time and the membership values are binary.

Adamy and Kempf have investigated the non-weighted recurrent fuzzy models and formally derive several dynamical properties from structural configuration of the rule base and membership functions [19][20]. They showed that linguistic recurrent fuzzy models are nonexpansive and have at least one equilibrium point for any input vector $\mathbf{u}(t)$ [20]. Moreover, for a certain rule structure, called by the authors "rule-continuous" [19], and if a regular (equidistant) prototype vector is used for defuzzification, the resulting linguistic recurrent fuzzy model is stable in the sense of Liapunov and converges to a stable output, for any initial state variable value [20].

Fernández and López [21] presented a similar fuzzy recurrent system, called by the authors Fuzzy Feedback System (FFS). In their system, non-weighted rules are used to compute the fuzzy membership values that are directly fed back into the inputs, without a defuzzification step as in the approach by Adamy and Kempf [19]. The authors have

used an inference methodology based on transition matrices to show that, for the sum-product inference operator, the FFS asymptotically reaches a steady state provided that the transition matrix is diagonalizable and the largest eigenvalue is real.

Although related to slightly different models, the results presented by Kempf and Adamy [20] and Fernández and López [21] show that RFS models described by non-weighted rules as (18) can be used to adjust stable, oscillatory or even chaotic systems. Theoretical results to characterize the RFS model described by weighted rules as (21) are still an open issue.

The FFA representation allows a better interpretation of the dynamics of the linguistic RFS model, as illustrated by the following example.

D. Discussion

Consider, for instance, a linguistic RFS model, represented by the following set of rules:

If $x(t)$ is A_1 and $u(t)$ is B_1 then

$x(t+1)$ is $(A_1 / \phi_{11}, A_2 / \phi_{12})$,

If $x(t)$ is A_1 and $u(t)$ is B_2 then $x(t+1)$ is $(A_1 / \phi_{21}, A_2 / \phi_{22})$,

If $x(t)$ is A_2 and $u(t)$ is B_1 then $x(t+1)$ is $(A_1 / \phi_{31}, A_2 / \phi_{32})$,

If $x(t)$ is A_2 and $u(t)$ is B_2 then $x(t+1)$ is $(A_1 / \phi_{41}, A_2 / \phi_{42})$.

(29)

The model output $y(t) = x(t)$ such that only the transition function is modeled. The fuzzy partition $\mathbf{B} = \{B_1, B_2\}$ for the input variable and the fuzzy partition $\mathbf{A} = \{A_1, A_2\}$ for the state variables are defined on their respective domains. The fuzzy relation matrix in Table I defines the set of rules (29).

This fuzzy model can be represented as a fuzzy finite-state automaton $F = \langle \mathbf{A}, \mathbf{B}, \delta, S_0 \rangle$, since $\mathbf{C} = \mathbf{A}$ for first order models (cf. (13)), such that the initial state $Q_0 = S_0$. The transition rule base weights are defined by and the initial state $S_0 = A_1$. The fuzzy finite-state automaton is represented by its state graph, as shown in Fig. 2.

With all other parameters fixed, the dynamic behavior of the model is a function only on the rule base weights. A qualitative discussion of possible RFS model behavior is made for four model parameters configurations presented in Table II.

Consider the linguistic terms {"low", "high"} to represent the states $\mathbf{A} = \{A_1, A_2\}$ and the terms {"small", "big"} to represent the input alphabet $\mathbf{B} = \{B_1, B_2\}$. The model 1 is a non weighted model. According to the rule base matrix Φ_1 , when the model state is "low", the state in the next time step will be "high", no matter the input variable value. Conversely, when the state is "high", the next state will be "low", independently of the input variable. If the initial

state $x(0) \neq 0$, this rule base yields to an oscillatory behavior, independently of the input variable.

The second model is a weighted model defined by the rule matrix Φ_2 that produces a different behavior depending on the state variable value. When the state is "low", it will remain "low" in the next step if the input is "small" or it will change to "high" if the input is "big". When the state is "high" and the input is "small", the state "high" will still be activated in the next step, at a rate defined by the weight $\phi_{32}^2 = 0.7$, but the "low" state is also activated by the weight $\phi_{31}^2 = 0.3$. These weights produce a damping effect and while the input value remains "small", the state variable value (and the output) will gradually stabilize on the "low" state prototype value. When the state is "high" and the input is "big", the model activates the state "low" at $\phi_{41}^2 = 0.8$, but also activates the state "high" at $\phi_{42}^2 = 0.2$. While the input remains "big", these weights produce a damped oscillation effect since 80% of the "low" state activation will return to the state "high" in the next simulation time step, due to the second rule and the rule weight $\phi_{21}^2 = 1.0$.

In the third model, defined by the rule matrix Φ_3 , when the model state is "low" and the input is "small", the state will remain "low". As the input increases to "big", the model gradually activates the state "high", according to the weight $\phi_{22}^3 = 0.7$. The state will remain "high" as long as the input value is "big". As the input decreases to "small", the model gradually activates the state "low", according to the weight $\phi_{31}^3 = 0.2$. This rule base configuration produces a model that behaves like a linear first order dynamic system, but with different time responses according to the direction of the input change.

The fourth model is a non weighted model defined by the rule base matrix Φ_4 , in which the state variable just follows the input. The outputs for each one of the four models, according to a sequence of step inputs are shown in Fig. 3.

In this simple example, the rule base weights were chosen to illustrate their effect on the RFS model according to the location of the weights. More complex non-linear dynamics can be obtained by a recurrent fuzzy system with more complex structure. For an unknown system, if a data set is available, a recurrent fuzzy model can be identified by the algorithm presented in next section.

III. IDENTIFICATION ALGORITHM

The identification of complex processes generally follows a three steps methodology:

1. Structure identification that is generally sub-divided in three steps:
 - a. Choice of model *type*, *i.e.* linear, non-linear polynomial, fuzzy, neural, etc.
 - b. Choice of model *size*, by selecting input and output variables, system order and delay in the case of dynamic systems.
 - c. Choice of parameters set for the model, according to model's type and size.
2. Parameters estimation from a data set that is representative of a system behavior called *training set*;
3. Model validation, considering modeling objectives: prediction, simulation, diagnosis, etc.

For a fuzzy system model, the structure identification is the determination of the model type and the number and location of fuzzy sets in the domain of each variable [29]. The estimation of the model parameters is generally associated with the rule conclusions. The model validation must check the model precision, but also certifies that the model is readable by domain experts.

The structure identification step is generally the most difficult one and has a strong influence on the model performance and flexibility [5]. This work is mainly concerned with the interpretability of the identified model, such that the model parameters are estimated for a given model structure, which is supposed to be meaningful for domain experts. It is thus assumed that the number of fuzzy sets associated to each variable is given and the corresponding prototypes are known.

The output parameters of the TSK-type RFS model are estimated from a training set T composed by N samples of input-output pairs² $(\mathbf{u}(t), y(t))$, by minimizing the mean squared error (MSE) between the real output $y(t)$ and its estimation by the model output $\hat{y}(t)$ as:

$$J = \frac{1}{N} \sum_{t=1..N} (y(t) - \hat{y}(t))^2. \quad (30)$$

In a non-recurrent fuzzy system, the minimization of the criterion (30) is done by solving a linear system of equations, since output parameters occur linearly in the output estimate $\hat{y}(t)$. Whereas in a recurrent fuzzy system, the values of the state variables depend on their values in the past time sample, as shown by (9) and (10), and the output estimate becomes non-linear with respect to the output parameters.

The most common approach for the parameter estimation in recurrent fuzzy (or neuro-fuzzy) systems is to use a gradient based algorithm [6], [8], [10], [14]. Recently, the use of genetic algorithms (GA) for the identification of recurrent fuzzy systems has been proposed with good

² The presentation will be focused on the multiple input single output (MISO) systems. The extension while multiple output (MIMO) systems is straightforward considering a MIMO system as a set of MISO systems.

results [24], [25], [26].

The proposed identification algorithm is composed of two steps. First, a standard GA is used to compute the TSK-type RFS output parameters [26] and, in the second step, this result is used to compute the rule base weights of an equivalent linguistic RFS model defined by (22).

A. Genetic algorithms

The problem solving procedure of genetic algorithms is inspired in natural evolution. Candidate solutions for a given problem are encoded into an individual. An initial population of randomly generated individuals is evolved towards the solution. At each iteration (called generation), the individuals are rated by their fitness to the solution and a selection mechanism assures that only the best individuals are kept for the next generation. The next generation individuals are computed by recombination operators such as mutation and crossover. Randomly driven recombination operators allow the individuals to explore the entire solution space, avoiding local minima. Although many variants exist, a typical GA structure, which has been used in this work, is sketched in Fig. 4.

Genetic algorithms are simple, flexible, robust and have been widely used in a number of applications of fuzzy systems [30]. GAs can overcome the local minima problems found in most of gradient-based optimization algorithms due to its stochastic nature [31]. Moreover, many implementations of GAs are available on the internet.

When applying a GA to a problem, two main issues must be considered: the codification of the variables into individuals and the fitness function. These issues are discussed in the following subsection for parameters estimation of the TSK-type RFS.

B. TSK-type RFS parameter identification

The fitness function is defined by the MSE criterion (30), where the output estimate $\hat{y}(t)$ is computed by a linear function of the state variables as:

$$\hat{y}(t) = \mathbf{K}\mathbf{x}(t), \quad (31)$$

where $\mathbf{x}(t) = (x_0(t), \dots, x_{n-1}(t))$ is the vector of n state variables and $\mathbf{K} = [\kappa_1 \dots \kappa_n]$ is the model output parameters vector, which must also be computed by the identification algorithm.

Each individual is coded as a vector, in which the components are the parameters to be estimated: the rules output parameters $\boldsymbol{\theta} = (\theta_1, \dots, \theta_M)$ and the model output parameters $\mathbf{K} = [\kappa_1 \dots \kappa_n]$. The codification imposes bounds to the solution, such that the rules output parameters $\boldsymbol{\theta} \in [-1, 1]^M$ and model output parameters $\mathbf{K} \in [-1, 1]^n$. The codification bounds must be respected by the output variables, such that the output values in the training set must be scaled to the $[-1, 1]$ interval. The minimum and the maximum values of the actual system output are saved to

convert the model output into the actual system scale. This codification ensures bounded input-bounded output model stability.

In this work, a public domain genetic algorithm source code called GENESIS, version 5.0 [32], was used. The program is a standard genetic algorithm implementation where the user must provide an evaluation function for computing the individuals' fitness and specify a set of algorithm's parameters. In the benchmark problems discussed in section IV, the parameters were set to: population size of 20 individuals, 60% of crossover rate and 1% of mutation rate and maximum number of iterations of 10,000. The selection procedure used was the default one provided by GENESIS, which implements the standard roulette procedure [31].

C. Equivalent rule base weights

The parameters $\boldsymbol{\theta}^*$ computed for the TSK-type RFS are used to compute an equivalent linguistic RFS rule base weights by (22), which lead to an underdetermined linear system of equations. The solution is therefore not unique but could be computed in the least squares sense through standard numerical techniques [33]. Nevertheless, the numerical solution may not be interpreted as rule base weights since they may lie outside the $[0, 1]$ interval.

The equivalent fuzzy rule base weights can be determined in a much more simple, direct and efficient way based on the complementary property of the fuzzification and defuzzification methods used in linguistic type RFS.

A fuzzification method *Fuz* is said to be complementary to a defuzzification method *Def* when, for any variable ξ , the following condition holds [34]:

$$Def(Fuz(\xi)) = \xi. \quad (32)$$

Rondeau et al. [34] showed that the fuzzification computed by the normalized and triangular-shaped fuzzy membership functions is complementary to the singleton defuzzification, such that condition (32) can be written as:

$$\boldsymbol{\mu}_A(\boldsymbol{\theta}^*) \cdot \mathbf{a}^T = \boldsymbol{\theta}^*. \quad (33)$$

From (22) and (33), the rule base matrix $\boldsymbol{\Phi}^*$ corresponding to the rules output parameter vector $\boldsymbol{\theta}^*$ is computed by its fuzzification as:

$$\boldsymbol{\Phi}^* = \boldsymbol{\mu}_A(\boldsymbol{\theta}^*) = \begin{bmatrix} \mu_{A_1}(\theta_1^*) & \dots & \mu_{A_p}(\theta_1^*) \\ \vdots & \ddots & \vdots \\ \mu_{A_1}(\theta_M^*) & \dots & \mu_{A_1}(\theta_M^*) \end{bmatrix}. \quad (34)$$

Additionally, the lines of the matrix $\boldsymbol{\Phi}^*$ satisfy conditions (5) and (6), such that they are also meaningful as transition membership values in the FFA representation.

The same reasoning can also be applied to the state variable $x_0(t)$ such that the (20) can be rewritten by

condition (32) as:

$$\boldsymbol{\mu}_A(x_0(t+1))\mathbf{a}^T = x_0(t+1) \quad (35)$$

and, from equation (20), the inference result can be written as:

$$\mathbf{v}(t) = \mathbf{w}(t) \cdot \boldsymbol{\Phi}^* = \boldsymbol{\mu}_A(x_0(t+1)). \quad (36)$$

The result of the sum-product operator inference can thus be directly fed back to compute the membership vector $\boldsymbol{\mu}_C(\mathbf{x}(t+1))$ in the next simulation time step (cf. (28)), without defuzzification.

The overall identification procedure can be summarized in the diagram shown in Fig. 5. A set of input/output data, collected from the actual system, is used for the model identification. In the first step of the identification procedure, the rules parameters of the TSK-type RFS are computed by a standard GA. Next, the equivalent rule base weights of the linguistic type RFS, are computed by (34). The set of weighted if-then fuzzy rules and the corresponding fuzzy automaton, issued from the model identification, can be used to understand and analyze the RFS model.

IV. RESULTS

The identification procedure presented above was tested with two benchmarks for non-linear system identification problems found in the literature. The first example is the classic Box and Jenkins data set [35]. The second example is the non-linear dynamic system first presented by Narendra and Parthasarathy [2], but also tested in many other works [6], [8], [11], [24].

The results are discussed from two different points of view. In the first example, the focus is in the fuzzy automaton representation and the resulting model interpretability, while in the second example the numeric performance is compared to other results presented in the literature.

A. Example 1

This example is the well-known Box & Jenkins data set, where the system to be modeled is a gas furnace. The original data set is composed of 296 pairs. This data set has been used to evaluate several system identification methods, but the data have not always been used in the same way. In this work, the model was built using delayed values of the system input $u(t-3)$ to compute an estimation $\hat{y}(t)$. Each training sample is, thus, expressed in the form $(u(t-3), y(t))$.

In order to discuss FFA representation, a simple recurrent fuzzy system structure was chosen, using $n=2$ state variables with $p=2$ fuzzy sets for their fuzzification and $q=3$ fuzzy sets for the fuzzification of the input variable. Fuzzy sets are defined equally spaced in the domain of the corresponding variable. This structure results in

$M = 2^2 \cdot 3 = 12$ rules and the total number of parameters to be identified in the first step of the identification algorithm is 14, considering the two parameters of the output matrix (cf. (31)).

The model output against the training data set points is shown in Fig. 6. It can be seen that this simple fuzzy recurrent model is able to simulate the process, except in the end of simulation.

The output parameters $\boldsymbol{\theta}^*$ computed in the first step of the identification procedure and equivalent rule base weights $\boldsymbol{\Phi}^*$ are shown in Table III for each combination of terms in the rules premise. The state graph for the FFA representation of the linguistic RFS model is shown in Fig. 7 where the transition membership values ϕ_{rk} refers to the rule weights presented in Table III for the row r and column k of the rule matrix $\boldsymbol{\Phi}^*$.

The multidimensional fuzzy sets C_i are combinations of fuzzy sets A_k defined on the state variables domain such that a rule as (15) can also be written as:

“if $x_0(t-1)$ is A_i and $x_0(t)$ is A_j and $u(t-3)$ is B_k , then $x_0(t+1)$ is $(A_1 / \phi_{r1}, A_2 / \phi_{r2})$ ”

The rules written in this way allow a better understanding of the temporal relationship among the input variable and the state variables.

In the state graph representation shown in Fig. 7, only the transitions that are consistent with the state variables update rule (11) are shown. There are some interesting patterns that should be mentioned: the stable states are those who have a transition to themselves; there are transitions that leave the stable states, transitions that reach the stable states and oscillatory transitions. Observing which kind of inputs activate those transitions, some conclusions can be drawn about system dynamics.

It can be seen, for instance, that when the system is in the “low” stable state $C_1 = (A_1, A_1)$, the system remains in the “low” state when the input is B_3 (“big”) while it leaves stability to the state A_2 (“high”) when the input is B_1 (“small”). The opposite behavior occurs when the state is in the “high” stable state $C_4 = (A_2, A_2)$. When the input is B_2 (“medium”), the system remains on its last state. Moreover, it can be seen that the system does never have an oscillatory behavior if the input remains constant.

The model performance depends on the model structure and a more complex model can adjust a more complex behavior. For instance, a RFS model structure with $n=2$ state variables, $p=4$ fuzzy sets for their fuzzification and $q=5$ fuzzy sets for the fuzzification of the input variable was able to reduce the model error in the end of simulation as shown in Fig. 8. Nevertheless, this structure results in $M = 4^2 \cdot 5 = 80$ rules and 82 model parameters.

For a complex system, model interpretability and

numerical performance are generally not achieved simultaneously. Moreover, a large number of additional rules are necessary for a small decrease in the model error.

B. Example 2

In this example, the actual system output is governed by the following difference equation:

$$y(t+1) = \frac{y(t)y(t-1)y(t-2)u(t-1)(y(t-2)-1)+u(t)}{1+y^2(t-1)+y^2(t-2)} \quad (37)$$

The training data set was computed from an input $u(t)$ generated as follows: 400 samples of an independent and identically distributed uniform sequence over the $[-1,1]$ interval and 400 samples of a sinusoidal signal $u(t) = 1.05\sin(\pi t/45)$. The testing data set was generated considering 1000 samples of the following signal:

$$u(t) = \begin{cases} \sin(\pi t/25) & t < 250 \\ 1 & 250 \leq t < 500 \\ -1 & 500 \leq t < 750 \\ 0.3\sin(\pi t/25) & 750 \leq t < 1000 \\ +0.1\sin(\pi t/32) & \\ +0.6\sin(\pi t/10) & \end{cases} \quad (38)$$

In order to find the best model structure, an exhaustive search was performed. Several structures were evaluated by combining different values for the model structure parameters n (range: 1 to 3), p and q (range: 2 to 11). A model structure was evaluated if the number of rules $M < 90$. The evaluation of a total of 112 model structures has taken about 4 hours in a Pentium IV 3.0 GHz computer.

The best RFS model structure found used $n=2$ state variables with $p=3$ fuzzy sets for their fuzzification and $q=5$ fuzzy sets for the fuzzification of the input variable.

This structure results in $M = 3^{2.5} = 45$ rules and, consequently, 47 parameters must be identified, considering the two output parameters.

The plot of the model output against the actual system output is shown in Fig. 9. It can be observed that the model can track adequately the actual systems.

Results obtained with the proposed method (RFS-TSK) are compared to the ones reported in the literature in Table IV. The present model is better than those reported in [11] and [24], but is worse than those reported in [8] and [10]. The RFNN model [10] employs a very large number of parameters. The DFNN [8] proposes an interesting identification algorithm that could also be applied to the current model. Nevertheless, most of these methods use Gaussian membership functions, of which the parameters are optimized by gradient-based algorithms, yielding to fuzzy sets with no linguistic meaning.

The same RFS-TSK model was presented before [26], using $n=2$ state variables with $p=3$ fuzzy sets for their fuzzification and $q=3$ fuzzy sets for the fuzzification of the input variable, resulting in a slightly worst performance but with a more compact model.

The linguistic RFS model for this example can be represented as a fuzzy automaton with $3^2 = 9$ symbolic states, of which the state graph is shown in Fig. 10. It is obviously hard to understand such a state graph but it is worst to read 45 recurrent rules. Nevertheless, the same patterns discussed previous example can be found in this topology: the stable states can be identified by the nodes with transitions to themselves; the transitions that make the system leave the stable states, the oscillatory transitions and the transitions that reach the stable states. Identifying which kind of inputs activate those transitions could help the domain experts to get a better understanding of the system dynamics as a whole, in complement to the localized view provided by the recurrent rules.

V. CONCLUSIONS

This work has discussed two types of equivalent RFS for non-linear dynamical systems modeling: the TSK type and the linguistic type. A standard genetic algorithm is used for the parameters estimation of a TSK-type RFS model and the solution is used to compute the equivalent rule base weights of the linguistic type. The linguistic type RFS can be represented as a fuzzy finite-state automaton.

The main contribution of this work is the conversion from the TSK-type to the linguistic type RFS and its corresponding fuzzy automaton representation, which allow the domain experts to analyze and interpret the identified model. Furthermore this method can be a powerful tool for modeling systems with hybrid (discrete and continuous) behavior.

The GA has shown to be an efficient and flexible tool for model identification. As a gradient-free optimization technique, GA is almost independent of the model structure and its stochastic nature avoids local minima. The results have shown that the proposed strategy has good performance when compared to other methods described in literature. A standard GA using default parameters was used to obtain the results, showing the robustness of the model with respect to the identification algorithm. This is very important for real world applications, where practitioners are more concerned with the model structure than with mathematical details of the identification algorithm.

In the present work the structure of the fuzzy model was fixed off-line and was not optimized. Several methods are reported in the literature in which a GA is used to identify the fuzzy systems structure. An optimized structure can certainly achieve a better numerical performance. This is one of the directions for future work in this research project.

ACKNOWLEDGMENT

The authors are also grateful for the important comments and suggestions of reviewers, who greatly helped to improve the paper quality.

REFERENCES

- [1] G. B. Giannakis and E. Serpedin. "A bibliography on nonlinear system identification", *Signal Processing*, vol. 81, pp. 533-580, 2001.
- [2] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks", *IEEE Trans. on Neural Networks*, vol. 1, no. 1, pp. 4-26, 1990.
- [3] R. J. Williams and D. Zisper, "A learning algorithm for continually running fully recurrent neural networks" *Neural Computation* vol. 1, pp. 270-280, 1989.
- [4] J. L. Elman, "Finding structure in time", *Cognitive Science*, vol. 14, pp. 179-211, 1990.
- [5] R. Babuška and H. Verbruggen. "Neuro-fuzzy methods for nonlinear system identification" *Annual Reviews in Control* vol. 27, pp. 73-85, 2003.
- [6] V. Gorrini and H. Bersini, "Recurrent fuzzy systems", *Proc. of the IEEE World Congress on Computational Intelligence*, 26-29 June 1994 pp. 193 - 198.
- [7] J. Zhang and A.J. Morris, "Recurrent neuro-fuzzy networks for nonlinear process modeling", *IEEE Trans. on Neural Networks*, vol. 10, no. 2, pp. 313-326, 1999.
- [8] P.A. Mastorocostas and J.B. Theocharis, "A recurrent fuzzy-neural model for dynamic system identification", *IEEE Trans. on Systems, Man and Cybernetics, Part B*, vol. 32, no. 2, pp. 176-190, 2002.
- [9] Y. Gao and M. J. Er, "NARMAX time series model prediction: feedforward and recurrent fuzzy neural network approaches", *Fuzzy Sets and Systems*, vol. 150, no. 2, pp. 331-350, 2005.
- [10] C.-H. Lee and C.-C. Teng, "Identification and control of dynamic systems using recurrent fuzzy neural networks", *IEEE Trans. on Fuzzy Systems*, vol. 8, no. 4, pp. 349-366, 2000.
- [11] C.-F. Juang and C.-T. Lin, "A recurrent self-organizing neural fuzzy inference network", *IEEE Trans. Neural Networks*, vol. 10, no. 4, pp. 828-845, 1999.
- [12] C.-M. Lin and C.-F. Hsu, "Supervisory recurrent fuzzy neural network control of wing rock for slender delta wings", *IEEE Trans. on Fuzzy Systems*, vol. 12, no. 5, pp. 733-742, 2004.
- [13] J. Zang, "Modeling and Optimal Control of Batch Processes Using Recurrent Neuro-Fuzzy Networks", *IEEE Trans. on Fuzzy Systems*, vol. 13, no. 4, pp. 417-427, 2005.
- [14] A. G. Evsukoff and S. Gentil, "Recurrent neuro-fuzzy system for fault detection and isolation in nuclear reactors", *Advanced Engineering Informatics*, vol. 19, no. pp. 55-66, 2005.
- [15] C. W. Omlin, K. K. Thornber and C. L. Giles, "Fuzzy finite-state automata can be deterministically encoded into recurrent neural networks", *IEEE Trans. on Fuzzy Systems*, vol. 6, no. 1, pp. 76-89, 1998.
- [16] C. L. Giles, C. W. Omlin, and K. K. Thornber, "Equivalence in knowledge representation: automata, recurrent neural networks and dynamical fuzzy systems", *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1623-1640, 1999.
- [17] A. Blanco, M. Delgado and M. C. Pegalajar, "Fuzzy automaton induction using neural network". *Int. Journal of Approximate Reasoning* 27, pp. 1-26, 2001.
- [18] I. Gabrijel and A. Dobnikar, "On-line identification and reconstruction of finite automata with generalized recurrent neural networks", *Neural Networks* 16, pp. 101-120, 2003.
- [19] J. Adamy and R. Kempf, "Regularity and chaos in recurrent fuzzy systems", *Fuzzy Sets and Systems*, vol. 140, pp. 259-284, 2003.
- [20] R. Kempf and J. Adamy, "Equilibria of recurrent fuzzy systems", *Fuzzy Sets and Systems*, vol. 140, pp. 231-257, 2003.
- [21] S. A.-Fernández and C. A.-López, "Fuzzy feedback system analysis using transition matrices". *Fuzzy Sets and Systems*, vol. 157, pp. 516-543, 2006.
- [22] X.-J. Zeng and M. G. Singh, "Approximation theory of fuzzy systems - SISO case", *IEEE Trans. on Fuzzy Systems*, vol. 2, no. 2, pp. 162-176, 1994.
- [23] X.-J. Zeng and M. G. Singh, "Approximation theory of fuzzy systems - MIMO case", *IEEE Trans. on Fuzzy Systems*, vol. 3, no. 2, pp. 219-235, 1995.
- [24] C.-F. Juang, "A TSK-type recurrent fuzzy network for dynamic systems processing by neural network and genetic algorithm", *IEEE Trans. on Fuzzy Systems*, vol. 10, no. 2, pp. 155-170, 2002.
- [25] H. Surmann and M. Maniadakis, "Learning feed-forward and recurrent fuzzy systems: a genetic approach", *Journal of Systems Architecture* vol. 47, pp. 649-662, 2001.
- [26] A. G. Evsukoff and N. F. F. Ebecken, "Identification of Recurrent Fuzzy Systems with Genetic Algorithms", *Proc. of the IEEE International Conference on Fuzzy Systems*, 25-29 July 2004, pp.1703 - 1708.
- [27] J. N. Moderson, D. S. Malik, *Fuzzy Automata and Languages, Theory and Applications*, Chapman and Hall/CRC, London/Boca Raton, FL, 2002.
- [28] M. Doostfateme, S. C. Kremer, "New directions in fuzzy automata", *International Journal of Approximate Reasoning*, vol. 38, pp. 175-214, 2005.
- [29] A. G. Evsukoff; A. C. S. Branco and S. Galichet. "Structure identification and parameter optimization for non-linear fuzzy modeling". *Fuzzy Sets and Systems*, vol. 132, pp 173-188, 2002.
- [30] O. Cordon, F. Gomide, F. Herrera, F. Hoffmann, and L. Magdalena, "Ten years of genetic fuzzy systems: current framework and new trends", *Fuzzy Sets and Systems*, vol. 141, pp. 5-31, 2003.
- [31] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Pub. Co. 1989.
- [32] J. J. Grefenstette, "A User's Guide to GENESIS Version 5.0", Navy Center for Applied Research in AI, Naval Research Laboratory 1990.
- [33] G. H. Golub and C. F. Van Loan. *Matrix Computations*, 3rd Edition, The John Hopkins Press, London, 1996.
- [34] L. Rondeau, R. Ruelas, L. Levrat and M. Lamotte, "A defuzzification method respecting the fuzzification", *Fuzzy Sets and Systems*, vol. 86, pp 311-320, 1997.
- [35] R. M. Tong. "The evaluation of fuzzy models derived from experimental data". *Fuzzy Sets and Systems*, vol. 4, pp 1-12, 1980.
- [36] C. W. Xu and Y. Z. Lu. "Fuzzy model identification and self learning for dynamic systems". *IEEE Trans. on Systems Man and Cybernetics*, vol. 17 no.4, pp. 683-689, 1987.

Carlos A. Gama is currently a Ph.D. student at the Engineering Graduated Center COPPE/UFRJ. His research focuses the development of efficient models for dynamical systems.

Alexandre G. Evsukoff received the M.S. degree in Mechanical Engineering in 1990 from *Universidade Federal do Rio de Janeiro*, Rio de Janeiro, Brazil, and the Ph.D. degree in Automatic Control in 1998 from *Institut National Polytechnique de Grenoble*, Grenoble, France. He is currently assistant Professor of Computational Systems at the Engineering Graduated Center COPPE/UFRJ.

His research interests is on the application of soft computing tools for data mining, decision support and the modeling of complex systems. He has published about 30 papers in journal and conference proceedings.

Philippe Weber received the M.S. degree in Automatic control and Signal processing in 1995 from the University Henri Poincaré, Nancy, France, and the Ph.D. degree in 1999 from *Institut National Polytechnique de Grenoble*, Grenoble, France. He is assistant Professor at the Nancy University since 2000, and member of the Research Centre for Automatic Control (CRAN) associated with the National Research Center of Science CNRS (UMR 7039).

He focuses his interest on modeling problem in maintenance, prognosis, decision-making processes and dynamic reliability. He develops fault tolerant control systems including reliability analysis. Since 2000 his research interest is focused in modeling methods based on Bayesian Networks.

Nelson F. F. Ebecken (SM'05) received the M.S. and Ph.D. degrees in Civil Engineering in 1973 and 1977 respectively from *Universidade Federal do Rio de Janeiro, Rio de Janeiro*, Brazil. He is currently Professor of Computational Systems at COPPE/UFRJ, the Engineering Graduated Center of Federal University of Rio de Janeiro.

His research focuses on basic methodologies for modeling and extracting knowledge from data and their application across different disciplines. He develops and integrates ideas and computational tools from statistics and information theory with artificial intelligence paradigms. He has published more than 200 papers in journals and conference proceedings. In 1998, he started the International Conference on Data Mining (Rio de Janeiro, Brazil). In 2005 he was awarded as a member of the Brazilian Academy of Sciences.

TABLE I
FUZZY RELATION MATRIX

$x(t)$	$u(t)$	A_1	A_2
A_1	B_1	ϕ_{11}	ϕ_{12}
A_1	B_2	ϕ_{21}	ϕ_{22}
A_2	B_1	ϕ_{31}	ϕ_{32}
A_2	B_2	ϕ_{41}	ϕ_{42}

TABLE II
FOUR MODELS FOR DISCUSSION

Φ_1	Φ_2	Φ_3	Φ_3
$\begin{bmatrix} 0.0 & 1.0 \\ 0.0 & 1.0 \\ 1.0 & 0.0 \\ 1.0 & 0.0 \end{bmatrix}$	$\begin{bmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \\ 0.3 & 0.7 \\ 0.8 & 0.2 \end{bmatrix}$	$\begin{bmatrix} 1.0 & 0.0 \\ 0.3 & 0.7 \\ 0.2 & 0.8 \\ 0.0 & 1.0 \end{bmatrix}$	$\begin{bmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \\ 1.0 & 0.0 \\ 0.0 & 1.0 \end{bmatrix}$

TABLE III
FUZZY RELATION MATRIX AND OUTPUT PARAMETERS

<i>rule</i>	<i>Rules' premise</i>		Φ^*		θ^*
	$\mathbf{x}(t)$	$u(t-3)$	A_1	A_2	
1	$C_1 = (A_1, A_1)$	B_1	0.2483	0.7517	0.5034
2	$C_1 = (A_1, A_1)$	B_2	0.6363	0.3637	-0.2727
3	$C_1 = (A_1, A_1)$	B_3	0.8827	0.1173	-0.7654
4	$C_2 = (A_1, A_2)$	B_1	0.4585	0.5415	0.0831
5	$C_2 = (A_1, A_2)$	B_2	0.2747	0.7253	0.4506
6	$C_2 = (A_1, A_2)$	B_3	0.8231	0.1770	-0.6461
7	$C_3 = (A_2, A_1)$	B_1	0.1760	0.8240	0.6481
8	$C_3 = (A_2, A_1)$	B_2	0.7498	0.2502	-0.4995
9	$C_3 = (A_2, A_1)$	B_3	0.9941	0.0059	-0.9883
10	$C_4 = (A_2, A_2)$	B_1	0.0420	0.9580	0.9159
11	$C_4 = (A_2, A_2)$	B_2	0.2874	0.7126	0.4252
12	$C_4 = (A_2, A_2)$	B_3	0.6285	0.3715	-0.2571

TABLE IV:
COMPARATIVE ANALYSIS

Model	MSE	Parameters
TRFN-S [24]	0.0346	33
RSONFIN [11]	0.0441	30
RFNN [10]	0.0013	112
DFNN [8]	0.0025	39
RFS-TSK [26]	0.0082	27
RFS-TSK	0.0036	47

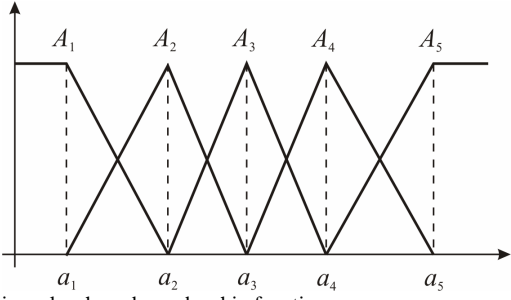


Fig. 1: Triangular shaped membership functions.

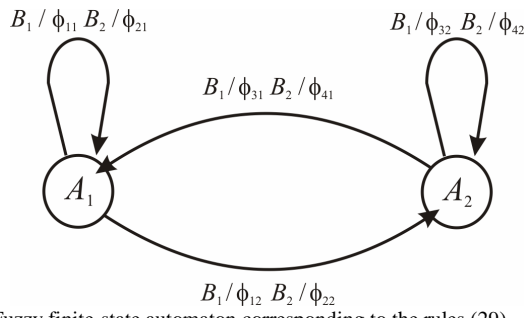


Fig. 2: Fuzzy finite-state automaton corresponding to the rules (29).

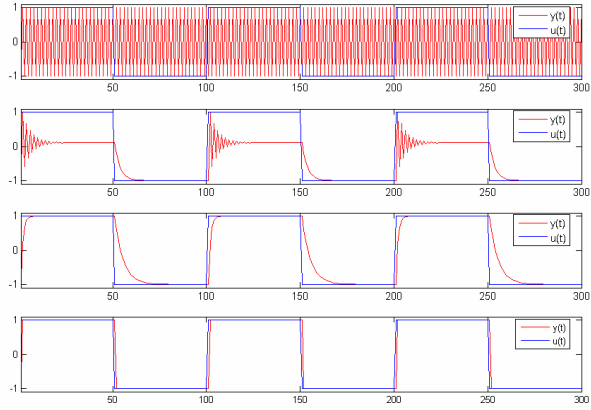


Fig. 3: Example of four configuration of the recurrent fuzzy model.

Procedure GA

```
1) begin
2)  t = 0;
3)  initialize P(t);
4)  evaluate structures in P(t);
5)  while (not termination)
6)  begin
7)    t = t + 1;
8)    select P(t) from P(t-1);
9)    recombine structures in P(t);
10)   evaluate structures in P(t);
11)  end
12) end.
```

Fig. 4: A Typical GA structure.

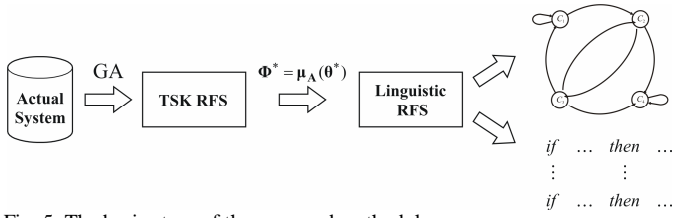


Fig. 5: The basic steps of the proposed methodology.

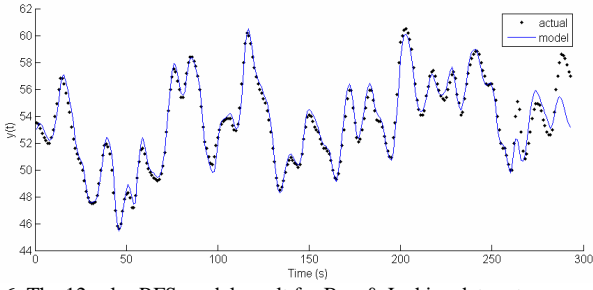


Fig. 6: The 12 rules RFS model result for Box & Jenkins data set.

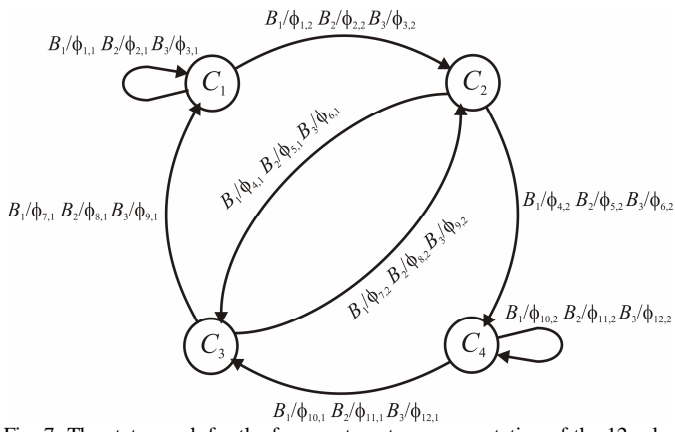


Fig. 7: The state graph for the fuzzy automaton representation of the 12 rules RFS model.

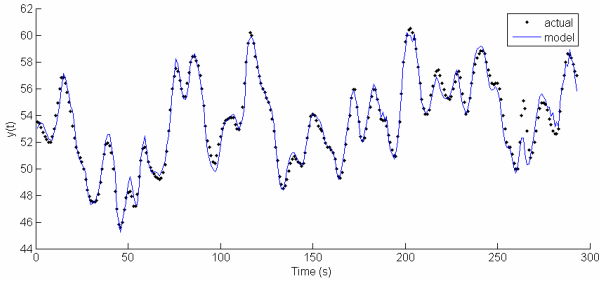


Fig. 8: The 80 rules RFS model result for Box & Jenkins data set.

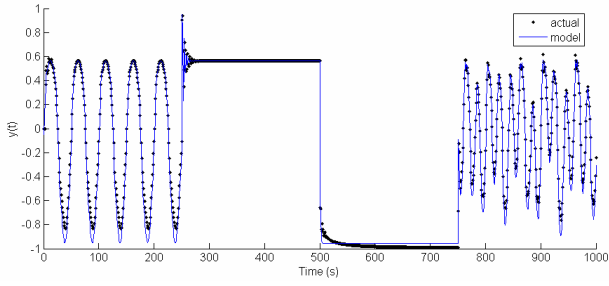


Fig. 9: RFS model result for testing data in example 2.

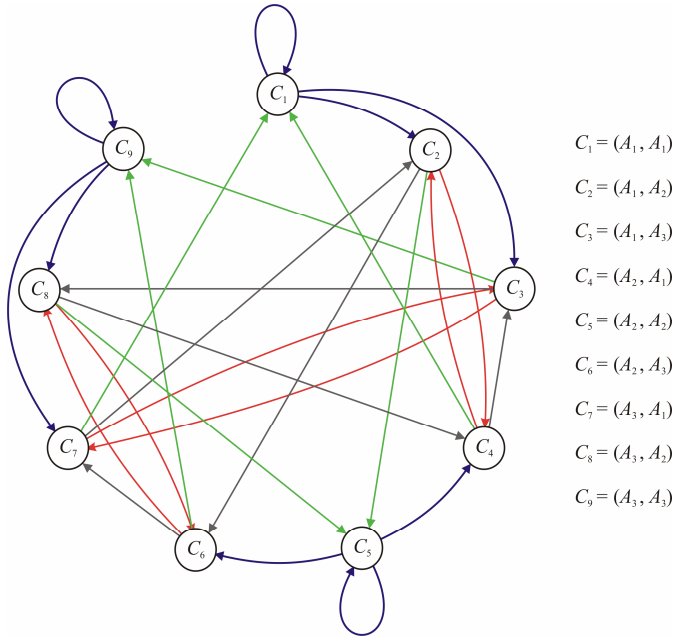


Fig. 10: The fuzzy automaton representation for RFS model in example 2.