# AN HYBRID ALGORITHM FOR GENERATING FUZZY RULES TO NONLINEAR DYNAMIC SYSTEMS

Antonio L. S. Ferreira, Edson Nascimento

*aluiz@demat.ufma.br, edson@dee.ufma.br*
*DEMAT/CCET/UFMA, DEEE/CCET/UFMA*
*Federal University of Maranhão*
*Av. dos Portugueses, s/n - Campus do Bacanga - 65080-040*
*São Luís - MA - Brazil*

**Abstract**— In designing controllers for complex and ill defined nonlinear dynamical systems there are needs not sufficiently addressed by conventional control theory. These are mainly related to the problem of environmental uncertainty and often call for human-like decision making requiring the use of heuristic reasoning, learning from past experience and a set of input-output crisp data describing the system. In general, only one of the two information has been used in the design phase: either a set of crisp input-output data or knowledge acquired from experts. This research work deals with the development of a new approach for implementing combining both information. The simulation results show that the proposal effectively solve the backing up a truck problem from several initial conditions presenting a good robustness. The controller design and the simulation are further presented and discussed.

**Keywords**— Neural network, Fuzzy systems, Hybrid systems, Fuzzy rules

## 1 Introduction

In a complex control system design problem, neither a precise analytic description, nor a mathematical model of the system is well known. Practical applications have showed that only a set of crisp input-output pairs in addition to expert knowledge about the system dynamic are known *a priori*. In general, only one of the two information has been used in the design task: either a set of crisp input-output data (Ljung, 1983) or knowledge acquired from experts (Adachi, 1996), (Joo, 1995).

Although the system may be successfully controlled by a human operator, when his experience is expressed as "IF-THEN" fuzzy rules in the design process, information could be lost and the final Fuzzy Associative Memory (FAM) bank become sparse, preventing practical use.

Therefore, only a set of crisp input-output data is not enough for guaranteing a good control system performance, because it cannot cover all the situations the control system may face. Usually, the training set of input-output crisp data is related with "good trajectories", i.e, a set of input-output pairs whose inputs had led to successful outputs.

Each of the two kinds of information alone is often insufficient. In this paper, an hybrid approach that takes in account both a set of numerical information from a historical database and expert knowledge about the system behavior is proposed, based on two design phases.

In Ferreira and Nascimento (2001) a Fuzzy Neural Network (FNN) architecture for generating fuzzy rule database from system dynamic sparse knowledge was proposed for the first design phase. Starting from few known rules, the FNN was able to generalize them, filling out the initial sparse FAM bank, namely $\Re^{'}$.

In this paper, for the second design phase, a FAM bank, $\Re^{''}$, is generated from a set of input-output crisp data, using Wang's algorithm (Wang and Mendel, 1992). Finally, both banks, $\Re^{'}$ and $\Re^{''}$, are combined yielding in another one, $\Re_0$ and it's performance in solving the truck backer-upper nonlinear control problem is analyzed.

The rest of the paper is organized as follows. In Section 2, the $h$ level set ($\alpha$-cuts) definition is brief reviewed and the neuro-fuzzy network architecture is described. Section 3 presents the FNN learning algorithm. Section 4 describes the Wang's algorithm. Section 5 shows the hybrid algorithm proposed and Section 6 illustrates experimental results, using the truck backer-upper problem and compare its performance to related approaches (Kosko, 1992). Finally, concluding remarks are presented in Section 7.

## 2 The Neuro-fuzzy architecture

A neuro-fuzzy network for handling fuzzy information may be designed according to Ishibuchi and Tanaka (1995) approach considering the following procedures:

a) fuzzy numbers are propagated through a multi-layer feedforward neural network;

b) a single unit is used for dealing with a fuzzy number;

c) the extension principle (Zadeh, 1975) defines the input-output relation for each unit and the actual fuzzy output calculations are performed using interval arithmetic for all $h$ level sets ($\alpha$-cuts);

d) the standard BP (Back Propagation) learning algorithm (Rumelhart and McClelland, 1986)

must be extended to accomplish with fuzzy weights and biases requirements. Inputs and outputs are also fuzzy vectors.

### *2.1 Triangular fuzzy numbers and h level sets*

Symmetric triangular fuzzy numbers may be used for representing fuzzy weights, biases, input and output variables. Let $X = (x^L, x^C, x^U)$ denotes a triangular fuzzy number as shown in Fig.1. The membership function X may be defined as

$$\mu_X(x) = \begin{cases} 0, \text{for } x \le x^L \text{ or } x > x^U \\ \frac{x - x^L}{x^C - x^L}, \text{for } x^L < x \le x^C \\ \frac{x^U - x}{x^U - x^C}, \text{for } x^C < x \le x^U \end{cases} \quad (1)$$

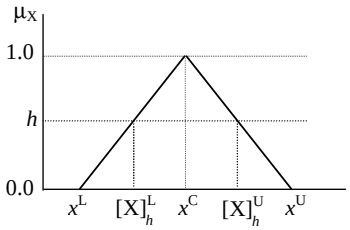



Figure 1. Representation of the fuzzy number X

The $h$ level set of fuzzy numbers may be defined as:

$$[X]_h = \{x : \mu_X(x) \ge h,\ x \in \mathrm{R}\} \quad (2)$$

where $\mu_X(x)$ is the membership function of $X$ and R is the set of all real numbers. Each $h$-level set defines a closed interval on the support of $X$, denoted by $[X]_h = \left[[X]_h^L, [X]_h^U\right]$ and it can be calculated as

$$[X]_h^L = x^L \cdot (1 - h/2) + x^U \cdot h/2 \quad (3)$$

$$[X]_h^U = x^L \cdot h/2 + x^U \cdot (1 - h/2) \quad (4)$$

### *2.2 Architecture*

A three-layer feedforward neural network with $n_I$ input units, $n_H$ hidden units and $n_O$ output units may be fuzzified according to the following equations, for each $h$-level set:

Input units:

$$[O_{pi}]_h = [X_{pi}]_h \quad (5)$$

Hidden units:

$$[O_{pj}]_h = f([Net_{pj}])_h \quad (6)$$

$$[Net_{pj}]_h = \sum_{i=1}^{n_I} [W_{ji}]_h [O_{pi}]_h + [\Theta_j]_h \quad (7)$$

Output units:

$$[O_{pk}]_h = f([Net_{pk}])_h \quad (8)$$

$$[Net_{pk}]_h = \sum_{i=1}^{n_H} [W_{kj}]_h [O_{pj}]_h + [\Theta_k]_h \quad (9)$$

where $W_{ji}$ and $W_{kj}$ are the fuzzy weights between the input-hidden layers and hidden-output layers, respectively. $f(.)$ is the activation function $f(x) = 1/(1 + e^{-x})$.

## 3 The Neuro-fuzzy Learning Algorithm

Let $\boldsymbol{T_p} = (T_{p1}, T_{p2}, \ldots, T_{pn_0})$ be the $n_0$-dimensional fuzzy target vector corresponding to the fuzzy input vector $\boldsymbol{X_p}$. A cost function for h-level sets of the fuzzy output $O_{pk}$ from the $k$th output units and the corresponding fuzzy target $T_{pk}$ may be calculated as follows:

$$e_{pkh} = e_{pkh}^L + e_{pkh}^U \quad (10)$$

where

$$e_{pkh}^L = h\frac{([T_{pk}]_h^L - [O_{pk}]_h^L)^2}{2} \quad (11)$$

$$e_{pkh}^U = h\frac{([T_{pk}]_h^U - [O_{pk}]_h^U)^2}{2} \quad (12)$$

The cost function for the h-level sets of the fuzzy output vector $\boldsymbol{O_p}$ and the fuzzy target vector $\boldsymbol{T_p}$ are defined as

$$e_{ph} = \sum_{k=1}^{n_0} e_{pkh} \quad (13)$$

The input-output pair $(\boldsymbol{X_p}, \boldsymbol{T_p})$ has the cost function:

$$e_p = \sum_h e_{ph} \quad (14)$$

A backpropagation learning algorithm may be derived using the cost function $e_{ph}$ without distorting the fuzzy weight shapes if their h-level sets are independently updated (Ishibuchi and Tanaka, 1995).

The triangular fuzzy weights $W_{kj}$, $W_{ji}$ and triangular fuzzy biases $\Theta_k$, $\Theta_j$ are denoted by:

$$W_{kj} = (w_{kj}^L, w_{kj}^C, w_{kj}^U),\ \ W_{ji} = (w_{ji}^L, w_{ji}^C, w_{ji}^U) \quad (15)$$

$$\Theta_k = (\theta_k^L, \theta_k^C, \theta_k^U),\ \ \Theta_j = (\theta_j^L, \theta_j^C, \theta_j^U) \quad (16)$$

The fuzzy weight $W_{kj} = (w_{kj}^L, w_{kj}^C, w_{kj}^U)$ between the $j$th hidden unit and the $k$th output unit may be updated by using the cost function $e_{ph}$ as follows:

$$\Delta w_{kj}^L(t) = -\eta \frac{\partial e_{ph}}{\partial w_{kj}^L} + \alpha \Delta w_{kj}^L(t-1) \quad (17)$$

$$\Delta w_{kj}^U(t) = -\eta \frac{\partial e_{ph}}{\partial w_{kj}^U} + \alpha \Delta w_{kj}^U(t-1) \quad (18)$$

where $\eta$ and $\alpha$ are learning and momentum parameters, respectively, and $t$ represents the number of epochs.

The derivatives in equations (17) and (18) are calculated as follows:

$$\frac{\partial e_{ph}}{\partial w_{kj}^L} = \frac{\partial e_{ph}}{\partial [W_{kj}]_h^L}\frac{\partial [W_{kj}]_h^L}{\partial w_{kj}^L} + \frac{\partial e_{ph}}{\partial [W_{kj}]_h^U}\frac{\partial [W_{kj}]_h^U}{\partial w_{kj}^L} \quad (19)$$

$$\frac{\partial e_{ph}}{\partial w_{kj}^U} = \frac{\partial e_{ph}}{\partial [W_{kj}]_h^L}\frac{\partial [W_{kj}]_h^L}{\partial w_{kj}^U} + \frac{\partial e_{ph}}{\partial [W_{kj}]_h^U}\frac{\partial [W_{kj}]_h^U}{\partial w_{kj}^U} \quad (20)$$

Therefore, due to (3) and (4), the equations (19) and (20) are rewritten as:

$$\frac{\partial e_{ph}}{\partial w_{kj}^L} = \frac{\partial e_{ph}}{\partial [W_{kj}]_h^L}\left(1 - \frac{h}{2}\right) + \frac{\partial e_{ph}}{\partial [W_{kj}]_h^U}\frac{h}{2} \quad (21)$$

$$\frac{\partial e_{ph}}{\partial w_{kj}^U} = \frac{\partial e_{ph}}{\partial [W_{kj}]_h^L}\frac{h}{2} + \frac{\partial e_{ph}}{\partial [W_{kj}]_h^U}\left(1 - \frac{h}{2}\right) \quad (22)$$

These equations show how the error signals are back propagated through the neuro-fuzzy network. Similarly, the fuzzy weight $W_{kj} = (w_{kj}^L, w_{kj}^C, w_{kj}^U)$ is updated according to the following rules:

$$w_{kj}^L(t+1) = w_{kj}^L(t) + \Delta w_{kj}^L(t) \quad (23)$$

$$w_{kj}^U(t+1) = w_{kj}^U(t) + \Delta w_{kj}^U(t) \quad (24)$$

$$w_{kj}^C(t+1) = \frac{w_{kj}^L(t+1) + w_{kj}^U(t+1)}{2} \quad (25)$$

After adjusting $W_{kj}$ through equations (23)-(25), the range limits have to be checked. In case where the lower limit becomes larger than the upper limit, the following heuristics should be used:

$$w_{kj}^L(t+1) = \min\{w_{kj}^L(t+1), w_{kj}^U(t+1)\} \quad (26)$$

$$w_{kj}^U(t+1) = \max\{w_{kj}^L(t+1), w_{kj}^U(t+1)\} \quad (27)$$

The fuzzy weight $W_{ji}$ and the fuzzy biases $\Theta_k, \Theta_j$ are updated in the same way as the fuzzy weight $W_{kj}$.

## 4 The Wang's Algorithm

This algorithm was proposed by Wang and Mendel (1992) which consists of the following five steps. Consider a set of desired input-output data pairs given by:

$$\{(x_1^{(1)}, x_2^{(1)}, y^{(1)}), ((x_1^{(2)}, x_2^{(2)}, y^{(2)}), \ldots\} \quad (28)$$

The task is to generate a set of fuzzy rules from (28) and use them to determine a mapping $f : (x_1, x_2) \rightarrow y$.

*Step 1 - Divide the input and output spaces into fuzzy regions*

Assume that the domain interval of $x_1$, $x_2$ and $y$ are $[x_1^-, x_1^+]$, $[x_2^-, x_2^+]$ and $[y^-, y^+]$, respectively. Divide each domain interval into $2N + 1$ regions ($N$ can be different for different variables, and the lengths of these regions can be equal or unequal), denoted by SN (Small $N$), ..., S1 (Small 1), CE (Center), B1 (Big 1), ..., BN (Big $N$), and assign each regions a fuzzy membership function.

*Step 2 - Generate Fuzzy Rules from a given data pairs*

Here, it is necessary to fuzzify $x_1^{(i)}$, $x_2^{(i)}$ and $y^{(i)}$ for all fuzzy membership functions. For example, consider that $x_1^{(1)}$ has degree 0.8 in $B1$, degree 0.2 in $B2$, and zero degree in all other sets. Similarly, $x_2^{(2)}$ has degree 1.0 in $CE$, and zero degrees in all others sets and $y^{(1)}$ has degree 0.9 in $CE$, 0.3 in $B1$. Assign $x_1^{(i)}$, $x_2^{(i)}$ and $y^{(i)}$ to the fuzzy set with maximum degree. Therefore, $x_1^{(1)}$ belongs to $B1$, $x_2^{(2)}$ to $CE$ and $y^{(1)}$ to $CE$. Finally, one rule from one pair of desired input-output data is obtained, e.g.,

Rule 1 from $(x_1^{(1)}, x_2^{(1)}, y^{(1)})$ is:

$$\begin{aligned}(x_1^{(1)}, x_2^{(1)}, y^{(1)}) \Rightarrow\ & x_1^{(1)}\,(0.8 \text{ in } B1, \max)\\ & x_2^{(1)}\,(1.0 \text{ in } CE, \max)\\ & y^{(1)}\,(0.9 \text{ in } CE, \max)\end{aligned}$$

IF ($x_1$ is $B1$) and ($x_2$ is $CE$) THEN ($y$ is CE);

*Step 3 - Assign a degree of confidence to each rule*

This step will prevent conflicting rules, i.e., ones that have the same IF part (antecedents) but different THEN part (consequents). One way for solving this problem is to assign a degree to each rule generated from data pairs, and accept only the rule from a conflict group that has maximum degree. In this way, not only is the conflict problem resolved, but also the number of rules is greatly reduced. For example, in order to assign a degree to rule 1 , the following product may be used:

$$\begin{aligned}D(Rule1) &= m_{B1}(x_1)\, m_{S1}(x_2)\, m_{CE}(y)\\ &= 0.8 \times 0.7 \times 0.9 = 0.504\end{aligned} \quad (29)$$

*Step 4 - Create a Fuzzy Rule Base*

*Step 5 - Determine a Mapping based on Fuzzy Rule Base*

Given the inputs ($x_1$, $x_2$), the following defuzzification strategy is used in order to determine the output $y$:

a) combine the antecedents of the $i$th fuzzy rule, using product operation to determine the degree, $m_{O^i}^i$, of the output corresponding to ($x_1$, $x_2$), i.e.,

$$m_{O^i}^i = m_{I_1^i}(x_1)\, m_{I_2^i}(x_2) \quad (30)$$

where $O^i$ denotes the input region of Rule $i$, and $I_j^i$ denotes the input fuzzy set of Rule $i$

for the $j$th component e.g., Rule 1 gives

$$m_{CE}^{1} = m_{B1}(x_1)\, m_{S1}(x_2) \qquad (31)$$

b) Use the following centroid defuzzification formula to determine the output

$$y = \frac{\sum_{i=1}^{K} m_{O^i}^{i}\, \overline{y}^{i}}{\sum_{i=1}^{K} m_{O^i}^{i}} \qquad (32)$$

where $\overline{y}^{i}$ denotes the center value of region $O^i$ and $K$ is the number of fuzzy rules in the FAM bank.

## 5 The Hybrid Identification Algorithm

In this section, the steps that show how to use both precedent tools which were described, are related as follow:

1. Define the fuzzy input-output vectors ($\boldsymbol{X_p}$, $\boldsymbol{T_p}$) available for FNN training;
2. Obtain a FAM bank, $\Re^{'}$, performing the FNN algorithm;
3. Obtain a FAM bank, $\Re^{''}$, performing Wang and Mendel's algorithm, using the data from the process;
4. Combine the FAM banks from steps 2. and 3. yielding the FAM bank, $\Re_0$

## 6 Results

The performance of the proposed hybrid approach is analyzed in solving the backing up a trucker problem, originally proposed by Nguyen and Widrow (1993). This problem consists in successfully driving a truck to the final position $(50, 100)$ on the plan $[0, 100] \times [0, 100]$, modeled by the following equations:

$$x_{k+1} = x_k + r\, \cos(\phi_{k+1}) \qquad (33)$$

$$y_{k+1} = y_k + r\, \sin(\phi_{k+1}) \qquad (34)$$

$$\phi_{k+1} = \phi_k + \theta_k \qquad (35)$$

where $(x, y)$ gives the truck position; $\phi$ is the angle between the car and the horizontal reference, $\theta$ is the control angle and $r = vt$ ($v = 1.0$ m/s is the truck speed and $t = 1.0$s the sampling time). As in Kosko (1992), the input variables are $\phi$ and $x$ and the output variable is $\theta$. In Fig. 2 the fuzzy sets defined to these variables, similarly to Kosko, are shown.

### 6.1 Application of the hybrid algorithm

*Step 1 - Define the fuzzy input-output vectors ($\boldsymbol{X_p}$, $\boldsymbol{T_p}$) available for FNN training;*

A set of known rules (scenario) was considered as shown in Table 1. It corresponds to the expert

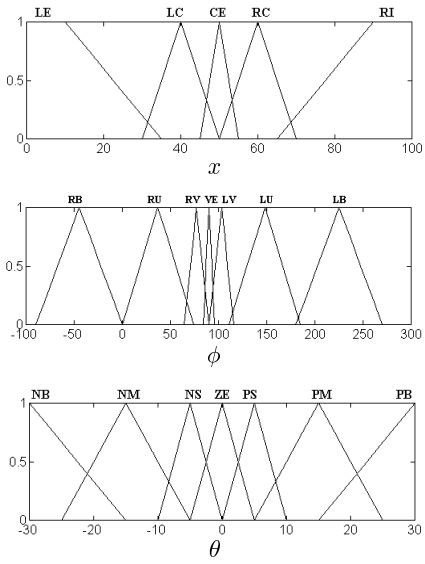



Figure 2. Membership functions for fuzzy variables in the backing up a truck problem

knowledge about the system behavior. Each fuzzy input-output pair is a IF-THEN rule. Consider, for example, the last cell in the first row. It corresponds to $(\boldsymbol{X_p}, \boldsymbol{T_p}) = ((\text{RB}, \text{RI}), \text{PB})$ or IF ($\phi$ IS *RB*) AND ($x$ IS *RI*) THEN ($\theta$ IS *PB*)

Table 1. The known rules for FNN training

| | | $x$ | | | | |
|---|---|---|---|---|---|---|
| | | LE | LC | CE | RC | RI |
| $\phi$ | RB | | | | | PB |
| | RU | | | | | |
| | RV | | | | | |
| | VE | | | ZE | | PM |
| | LV | | | | | |
| | LU | | | | | |
| | LB | NB | | NM | | NS |

*Step 2 - Obtain a FAM bank, $\Re^{'}$, performing the FNN algorithm;*

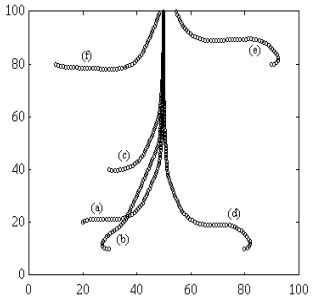
In all simulations, the net structure is composed by $n_I = 2$ input units, $n_H = 6$ hidden units and $n_O = 1$ output unit. The $h$ level set is defined as $\{0.2,\ 0.4,\ 0.6,\ 0.8,\ 1.0\}$. Fuzzy weights and biases are initialized in the closed interval $[-1, 1]$ and the condition for halting the training is the number of training epochs (1000 iterations). The FAM bank $\Re^{'}$ generating from that scenario is showed in Table 2

*Step 3 - Obtain a FAM bank, $\Re^{''}$, performing Wang's algorithm, using the data from the process;*

In this phase, "good trajectories" are necessary in order to generate $\Re^{''}$. Fig.3 shows trajectories, from different initial states that were used in the Wang's algorithm. Table 3 shows the fuzzy rules generated from those trajectories.

Table 2. FAM bank $\Re^{'}$ obtained from FNN

| | | $x$ | | | | |
|---|---|---|---|---|---|---|
| | | LE | LC | CE | RC | RI |
| | RB | PS | PM | PM | PM | PB |
| | RU | NS | PS | PS | PM | PM |
| | RV | NM | NS | ZE | PS | PM |
| $\phi$ | VE | NM | NS | NS | ZE | PM |
| | LV | NM | NS | NS | ZE | PM |
| | LU | NM | NM | NM | NS | PS |
| | LB | NB | NM | NM | NM | ZE |




| Initial States $(x_0, y_0, \phi_0)$ | |
|---|---|
| (a) | (20, 20, 30°) |
| (b) | (30, 10, 220°) |
| (c) | (30, 40, −20°) |
| (d) | (80, 10, −20°) |
| (e) | (90, 80, −30°) |
| (f) | (10, 80, −30°) |

Figure 3. Trajectories used to generate $\Re^{''}$

Table 3. FAM bank $\Re^{''}$ obtained from Wang's algorithm

| | | $x$ | | | | |
|---|---|---|---|---|---|---|
| | | LE | LC | CE | RC | RI |
| | RB | PS | PM | | | PB |
| | RU | NS | PS | | | PB |
| | RV | NM | NS | PS | | PB |
| $\phi$ | VE | | | ZE | | PM |
| | LV | NB | | NS | PS | PM |
| | LU | NB | | | NS | PS |
| | LB | NB | | | | NS |

Notice that bank $\Re^{''}$ is sparse, i.e, there are empty cells since the available trajectories cannot cover all possible points in the state variable space.

*Step 3 - Combine the FAM banks from steps 2 and 3 yielding the FAM bank,* $\Re_0$

Those empty cells in Table 3 can be filled using expert knowledge. As an alternative, consider the following definition:

Given $(r_{ij}^{'})$, $(r_{ij}^{''})$ and $(r_{ij})$ cells from the FAM banks $\Re^{'}$, $\Re^{''}$ and $\Re_0$, respectively, a cell from $\Re_0$ is defined as

$$(r_{ij}) = \begin{cases} r_{ij}^{'}, \text{ if } r_{ij}^{''} = 0 \text{ or } r_{ij}^{'} = r_{ij}^{''} \\ \\ r_{ij}^{''}, \text{ if } r_{ij}^{''} \neq 0 \text{ and } r_{ij}^{'} \neq r_{ij}^{''} \end{cases} \quad (36)$$

The equation (36) means that if a cell is similar in both FAM banks ($\Re^{'}$ and $\Re^{''}$) or if there is an empty cell in $\Re^{''}$, it is assumed that the cell in $\Re^{'}$ is correct. Otherwise, if a cell is conflicting (there is not an empty cell in $\Re^{''}$ and the rule has the same antecedents, but different consequents in both banks), then it is assumed that cell from $\Re^{''}$ is correct. In the other words, numeric information prevail over linguistic one (from expert knowledge). Table 4 shows the resulting FAM bank derived from equation (36).

Table 4. The FAM bank $\Re_0$ obtained from eq. ( 36)

| | | $x$ | | | | |
|---|---|---|---|---|---|---|
| | | LE | LC | CE | RC | RI |
| | RB | PS | PM | PM | PM | PB |
| | RU | NS | PS | PS | PM | **PB** |
| | RV | NM | NS | **PS** | PS | **PB** |
| $\phi$ | VE | NM | NS | **ZE** | ZE | PM |
| | LV | **NB** | NS | NS | **PS** | PM |
| | LU | **NB** | NM | NM | NS | PS |
| | LB | NM | NM | NM | NM | **NS** |

After all FAM banks have been obtained from linguistic information ($\Re^{'}$), numerical information ($\Re^{''}$), and both of them ($\Re_0$), they were used for solving the truck backer-upper problem and the final result compared to those obtained by Kosko (1992). In Table 5, the final states of the variables $(x, y, \phi)$ for two initial conditions $(75, 15, -40°)$ and $(40, 10, 200°)$ are shown. The results for banks $\Re^{'}$, $\Re_0$ and Kosko's FAM bank are shown in Fig. 4.

Table 5. FNN performance results

| SETS | FINAL STATES | | | | | |
|---|---|---|---|---|---|---|
| | $(75, 15, -40°)$ | | | $(40, 10, 200°)$ | | |
| | $x$ | $y$ | $\phi$ | $x$ | $y$ | $\phi$ |
| $\Re^{'}$ | 54.50 | 99.95 | 88.56 | 54.76 | 99.17 | 89.34 |
| $\Re_0$ | 50.04 | 99.96 | 90.06 | 49.95 | 99.87 | 89.87 |
| Kosko | 50.00 | 99.96 | 90.00 | 49.99 | 99.86 | 89.99 |

The obtained results indicate that the FAM bank $\Re_0$ presents a performance similar to Kosko's and better than $\Re^{'}$. Due to it's sparsity, $\Re^{''}$ could not conducts the truck for the "docking zone" (final state), because those initial conditions did not belong to the set of "good trajectories". Although this, some cells from $\Re^{''}$ were useful to correct wrong cells from $\Re^{'}$. See the boldface cells in Table 4 and compare them with those in Table 2. Possible similarity degree between $\Re_0$ and Kosko's FAM bank may be better analyzed using the concept of "control surface" (Driankov, 1993). Hence, consider that the state variables may be related by some function $\varphi : (\phi, x) \rightarrow \theta$. The surface defined by $\varphi$ obtained from banks $\Re_0$ and Kosko's FAM bank are shown in Fig. 5.

## 7 Conclusion

An hybrid approach for generating fuzzy rule database from both numerical and linguistic information usually available about nonlinear dynamic system was proposed. A fuzzy neural net-

work (FNN) that can handle linguistic information (fuzzy input and output vectors) was used for generating a FAM bank, from sparse expert knowledge. The Wang's algorithm generate other FAM bank from crisp input-output data and, finally, another bank is obtained from combining them. The proposed algorithm was analyzed in solving the truck backer-upper nonlinear control problem and the results compared with Kosko (1992). Simulation results and comparison of results show that the proposed algorithm presents good trajectories, driving the truck to the desired final state.

## References


Adachi, G. (1996). An automatic design method of fuzzy controllers based on linguistic specifications and fuzzy model of controlled object, *AMC'96-MIE* **4**(5): 144–149.

Driankov, D. (1993). *An introduction to fuzzy control*, Spring-Verlag.

Ferreira, A. L. S. and Nascimento, E. (2001). A neuro-fuzzy network for generating fuzzy rule-based nonlinear dynamic system, *Proceedings on V Brazilian Conference on Neural Networks*, pp. 355–360.

Ishibuchi, H. and Tanaka, H. (1995). A learning algorithm of fuzzy neural networks with triangular fuzzy weights, *Fuzzy Sets and Systems* **71**(2): 277–293.

Joo, Y. H. (1995). Linguistic model identification for fuzzy system, *Eletronics Letters* **31**(4): 330–331.

Kosko, B. (1992). *Neural Networks and Fuzzy Systems: a dynamical approach to machine intelligence*, Prentice Hall.

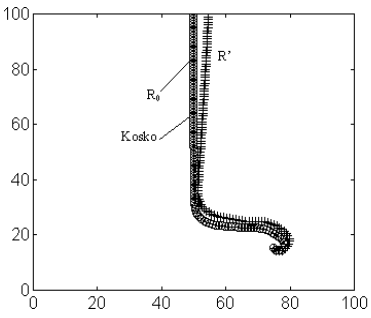
Ljung, L. (1983). *System Identification - theory for the user*, Prentice Hall.

Nguyen, D. and Widrow, B. (1993). The truck baker-upper: An example of self-learning in neural networks, *Proceedings of the I International Joint Conference on Neural Networks*, pp. 357–363.

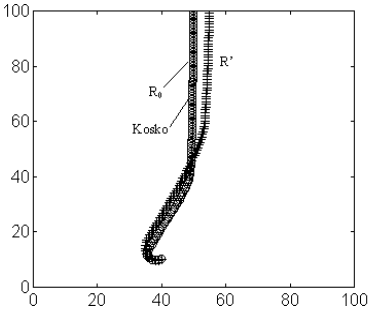
Rumelhart, D. and McClelland, J. (1986). *Parallel Distributed Processing*, MIT Press.

Wang, L.-X. and Mendel, J. (1992). Generating fuzzy rules by learning from examples, *IEEE Trans. on System, Man, and Cybernetics* **22**(6): 1414–1427.

Zadeh, L. A. (1975). The concept of a linguistic variable and its application to aproximate reasoning, *Inf. Sciences* **8**(2): 199–249.

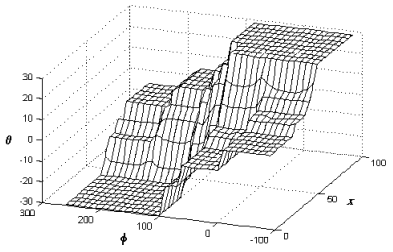




(a) - Trajectories with initial position $(75, 15, -40^\circ)$

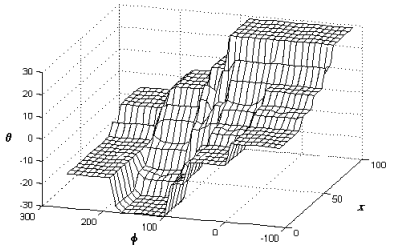



(b) - Trajectories with initial position $(40, 10, 200^\circ)$

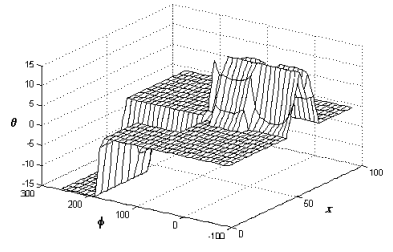
Figure 4. Trajectories for FAM banks $\Re'$, $\Re_0$ and Kosko



(a) - Control Surface to Kosko's FAM bank;



(b) - Control Surface to $\Re_0$ FAM bank;



(c) - Difference between the surfaces (a) and (b)

Figure 5. Control surfaces examples