Proceedings of the V Brazilian Conference on Neural Networks - V Congresso Brasileiro de Redes Neurais
pp. 355–360, April 2–5, 2001 - Rio de Janeiro - RJ - Brazil

A Neuro-Fuzzy Network for Generating Fuzzy Rule-based Nonlinear Dynamic Systems

Antonio Luiz S. Ferreira[1], Edson Nascimento[2]
[1]DEMAT/CCET/UFMA
[2]DEEE/CCET/UFMA
E-mails: aluiz@demat.ufma.br, edson@dee.ufma.br

Abstract

A fuzzy neural network (FNN) architecture for generating fuzzy rule database from sparse knowledge about the system dynamic is proposed and its performance in solving the truck backer-upper nonlinear control problem is analyzed. Since the FNN maps fuzzy input to fuzzy outputs vectors deriving a knowledge kernel for a given specific problem, considering the actual experience of experts in the domain, it may becomes an important tool for handling fuzzy mapping in many practical applications. Further improvements to allow fast convergence of the FNN learning phase, controlling the learning and momentum parameters through an adaptive fuzzy logic controller (FLC), is described. Simulations results are included to confirm that this approach may be of great practical interest since the actual results are very encouraging.

1. Introduction

Fuzzy systems have been applied in control problems for a long time, mainly for complex systems whose analytic formula is unknown and cannot be easily identified. Practical situations show that only a limited number of input-output pairs of vectors together with knowledge about the system dynamic are known *a priori* [1] [2]. Often, in these cases, practical control strategies have been directly implemented only relying on skilled system operators. The main task here is to design a neuro-fuzzy controller able to approximate the operator actions in some optimal manner and make it available as a screening tool for helping them in the control center.

Several attempts for solving nonlinear problems based on either neural networks [3] or fuzzy approaches [4] have been reported presenting very encouraging results. Nevertheless, it seems that neuro-fuzzy approaches whose inherit the generalization ability from standard neural networks and the robustness in handling input-output mapping of linguistic variables from fuzzy systems are natural candidates [5]. Recently, new classes of adaptive neural fuzzy networks for fuzzy modeling and control of dynamic systems have been presented. They have presented superior performance from the design point of view and accuracy compared to alternative approaches reported in the literature when applied to well-known benchmark problems [6] [7].

Since linguistic modeling of complex irregular systems constitutes the heart of many control and decision making systems, an approach to determine linguistic-fuzzy rules is timely needed. In fact, generally speaking, the operator's experience should be expressed as "IF-THEN" fuzzy knowledge rules that can be used in fuzzy-associative-memory (FAM) look-up-table approaches [8]. However, FAM banks are difficult to achieve for both main reasons: firstly, acquiring and expressing operator experiences as fuzzy knowledge rules it is not a simple task to achieve; secondly, often the final bank is sparse preventing practical use. Indeed, experts usually are able to express their knowledge about the system behavior mainly concerning to extreme points of the state variable ranges but out of stressing operation, they finding difficult to provide reliable information how the system performs.

In this paper, a fuzzy neural network (FNN) architecture for generating fuzzy rule based from sparse expert knowledge about the system dynamic is proposed and its performance in solving the truck backer-upper nonlinear control problem is analyzed. Even in situation where the FNN approaches are implemented with small constant learning parameters there is no guarantee they properly converge [9]. They have been adapted using heuristics to improve fast convergence or a desired level of performance [10]. In this paper, a fuzzy logic controller (FLC) for adapting the FNN learning and momentum parameters during the learning phase is also proposed.

The rest of the paper is organized as follows. In Section 2, the h level set (α-cuts) definition is brief reviewed and the neuro-fuzzy network architecture is described. Section 3 presents the FNN learning algorithm. Section 4 describes the fuzzy logic controller (FLC) for adapting both the FNN learning and momentum parameters during the learning phase. Section 5 illustrates some experimental results of the FNN using the truck backer-upper problem and compare its performance with related approaches [8]. Finally, concluding remarks are given in Section 6.

2. The Neuro-Fuzzy Proposal

A multi-layer feedforward neural network that can handle fuzzy information may be designed according to Ishibuchi et al [11] proposal considering the following procedures:

a) fuzzy numbers are propagated through the neural network;

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

c) the extension principle [12] defines the input-output relation for each unit and the actual fuzzy output calculations are performed using interval arithmetic for all h level sets (α-cuts);

d) the standard BP (Back Propagation) learning algorithm [13] must be extended to accomplish with fuzzy weights and biases needs. 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, inputs and outputs variables. Let X = (x^L, x^C, x^U) denotes a triangular fuzzy number as shown in Fig.1. The membership function of X may be defined as

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

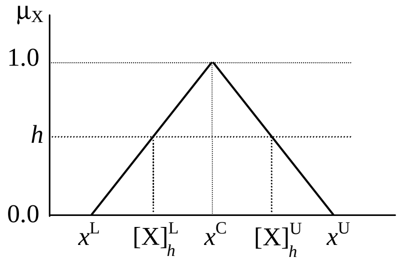


Figure 1: Representation of the fuzzy number X

The level sets of fuzzy numbers may be defined as

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

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

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

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

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:

$$[O_{pi}]_h = [X_{pi}]_h \tag{5}$$

Hidden units:

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

$$[Net_{pj}]_h = \sum_{i=1}^{n_i} [W_{ji}]_h \, [O_{pi}]_h + [\Theta_j]_h \tag{7}$$

Output units:

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

$$[Net_{pk}]_h = \sum_{i=1}^{n_i} [W_{kj}]_h \, [O_{pj}]_h + [\Theta_k]_h \tag{9}$$

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 $f(x) = 1/(1+e^{-x})$.

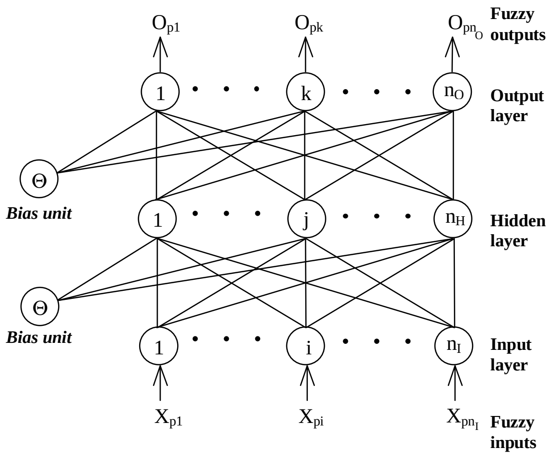


Figure 2: The Fuzzy Neural Network Architecture

3. FNN 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 kth output units and the corresponding fuzzy target T_{pk} may be calculated as follows:

$$e_{pkh} = e_{pkh}^L + e_{pkh}^U \tag{10}$$

where

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

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

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

$$e_{ph} = \sum_{k=1}^{n_O} e_{pkh} \tag{13}$$

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

$$e_p = \sum_h e_{ph} \tag{14}$$

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[11].

The triangular fuzzy weights W_{kj}, W_{ji} and triangular fuzzy biases Θ_k, Θ_j may be denoted by:

$$W_{kj} = (w^L_{kj}, w^C_{kj}, w^U_{kj}), \;\; W_{ji} = (w^L_{ji}, w^C_{ji}, w^U_{ji}) \quad (15)$$

$$\Theta_k = (\theta^L_k, \theta^C_k, \theta^U_k), \;\; \Theta_j = (\theta^L_j, \theta^C_j, \theta^U_j) \quad (16)$$

The fuzzy weight $W_{kj} = (w^L_{kj}, w^C_{kj}, w^U_{kj})$ between the jth hidden unit and the kth output unit can be updated by using the cost function e_{ph} as follows:

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

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

where η and α are learning and momentum parameters, respectively, and t represents the number of epochs.

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

$$\frac{\partial e_{ph}}{\partial w^L_{kj}} = \frac{\partial e_{ph}}{\partial [W_{kj}]^L_h} \frac{\partial [W_{kj}]^L_h}{\partial w^L_{kj}} + \frac{\partial e_{ph}}{\partial [W_{kj}]^U_h} \frac{\partial [W_{kj}]^U_h}{\partial w^L_{kj}} \quad (19)$$

$$\frac{\partial e_{ph}}{\partial w^U_{kj}} = \frac{\partial e_{ph}}{\partial [W_{kj}]^L_h} \frac{\partial [W_{kj}]^L_h}{\partial w^U_{kj}} + \frac{\partial e_{ph}}{\partial [W_{kj}]^U_h} \frac{\partial [W_{kj}]^U_h}{\partial w^U_{kj}} \quad (20)$$

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

$$\frac{\partial e_{ph}}{\partial w^L_{kj}} = \frac{\partial e_{ph}}{\partial [W_{kj}]^L_h} \left(1 - \frac{h}{2}\right) + \frac{\partial e_{ph}}{\partial [W_{kj}]^U_h} \frac{h}{2} \quad (21)$$

$$\frac{\partial e_{ph}}{\partial w^U_{kj}} = \frac{\partial e_{ph}}{\partial [W_{kj}]^L_h} \frac{h}{2} + \frac{\partial e_{ph}}{\partial [W_{kj}]^U_h} \left(1 - \frac{h}{2}\right) \quad (22)$$

These equations show how the error signals are back propagated through neural network. Similarly, the fuzzy weight $W_{kj} = (w^L_{kj}, w^C_{kj}, w^U_{kj})$ is updated according to the following rules:

$$w^L_{kj}(t+1) = w^L_{kj}(t) + \Delta w^L_{kj}(t) \quad (23)$$

$$w^U_{kj}(t+1) = w^U_{kj}(t) + \Delta w^U_{kj}(t) \quad (24)$$

$$w^C_{kj}(t+1) = \frac{w^L_{kj}(t+1) + w^U_{kj}(t+1)}{2} \quad (25)$$

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:

$$w^L_{kj}(t+1) = \min\{w^L_{kj}(t+1), w^U_{kj}(t+1)\} \quad (26)$$

$$w^U_{kj}(t+1) = \max\{w^L_{kj}(t+1), w^U_{kj}(t+1)\} \quad (27)$$

The fuzzy weight W_{ji} and the fuzzy biases Θ_k, Θ_j are updated in the same way as the fuzzy weight W_{kj}.

4. Convergence control

In spite of the learning and momentum parameters play an important role to improve the convergence of the learning phase, in practical situation they have been obtained by trial-error attempts, left as small constant values or updated by using heuristics [10]. In these cases, there is no guarantee the learning process will converge[9].

In this section, to provide a desired level of performance, a fuzzy logic controller (FLC) for adapting the FNN learning and momentum parameters during the learning phase is proposed. The input signals used in the fuzzy controller are both the error $e(t-1)$ and change-of-error, $ce(t-1)$, that can be given as (Fig.3):

$$e(t) = \sum_{p=1}^{m} e_p \quad (28)$$

$$ce(t) = e(t) - e(t-1) \quad (29)$$

where t is the number of epochs, m is the total number of input-output pairs $(\boldsymbol{X_p}, \boldsymbol{T_p})$ and e_p is the cost function defined by (14).

The output of the fuzzy controller is the learning parameter η. The momentum parameter α is defined as follows:

$$\alpha = 1 - \eta \quad (30)$$

The equation (30) above shows that at the beginning of the learning process when η is large, α is small. This guides the net fast towards a minimum point. however, nearby a minimum, it is desirable having η small and α large.

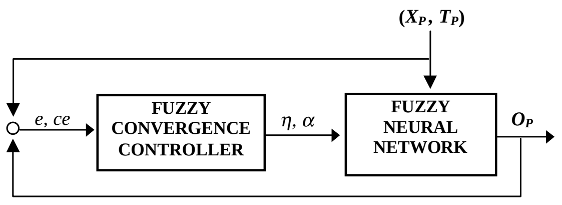


Figure 3: Convergence control system

The membership functions and the FAM table used in the fuzzy convergence controller are shown in Fig. 4 and Table 1, respectively. Each position in this table defines one rule, for example:

"IF e is *medium* and ce is *zero* THEN η is *medium*"

which corresponds to the central cell in Table 1.

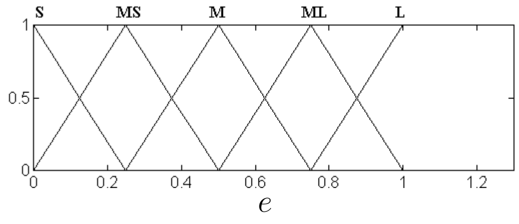
5. Simulations and results

5.1. Backing up a truck problem

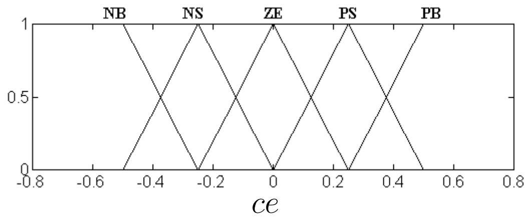
The performance of the proposed FNN approach is analyzed in solving the backing up a trucker problem, originally proposed by Nguyen & Widrow [14]. This

Table 1: Fuzzy convergence controller rules

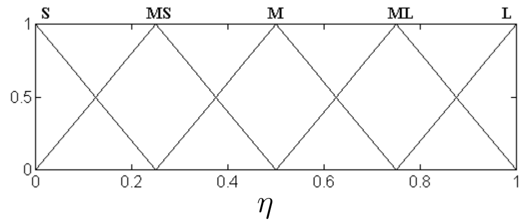
e \ ce	NB	NS	ZE	PS	PB
S	MS	MS	S	S	S
MS	M	M	MS	S	S
M	ML	ML	M	MS	S
ML	L	ML	ML	M	MS
L	L	L	ML	ML	M



(S: *small*, MS: *medium small*, M: *medium*, ML: *medium large*, L: *large*)



(NB: *negative big*, NS: *negative small*, ZE: *zero*, PS: *positive small*, PB: *positive big*)



(S: *small*, MS: *medium small*, M: *medium*, ML: *medium large*, L: *large*)

Figure 4: Membership functions for the fuzzy convergence controller

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:

$$x_{k+1} = x_k + r\,\cos(\phi_{k+1}) \quad (31)$$
$$y_{k+1} = y_k + r\,\sin(\phi_{k+1}) \quad (32)$$
$$\phi_{k+1} = \phi_k + \theta_k \quad (33)$$

where (x, y) gives the truck position; ϕ is the angle between the car and the horizontal reference, θ 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 [8], the input variables are the angle ϕ and the position x. The output variable is the angle θ. Fig. 6 shows the fuzzy sets defined to these variables, similarly to Kosko's .

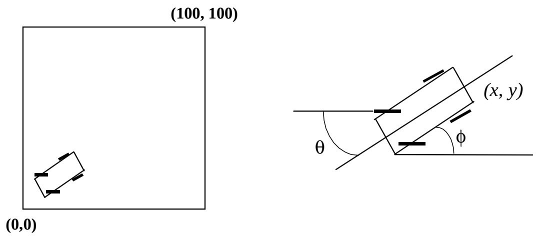
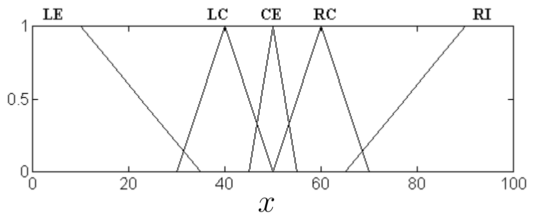
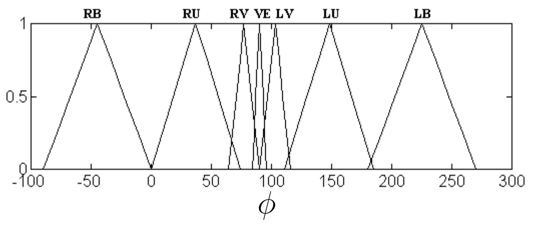


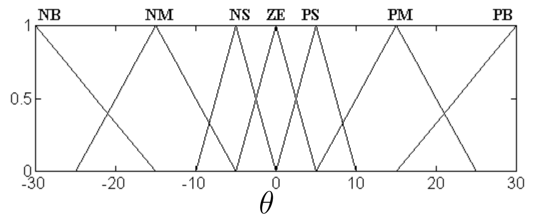
Figure 5: Backing up a trucker problem



(LE: *left*, LC: *left center*, CE: *center*, RC: *right center*, RI: *right*)



(RB: *right below*, RU: *right upper*, RV: *right vertical*, VE: *vertical*, LV: *left vertical*, LU: *left upper*, LB: *left below*)



(NB: *negative big*, NM: *negative medium*, NS: *negative small*, ZE: *zero*, PS: *positive small*, PM: *positive medium*, PB: *positive big*)

Figure 6: Membership functions for fuzzy variables in the backing up a truck problem

5.2. Results

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).

In order to verify the generalization ability of the FNN, three sets of known rules (scenarios) were considered as shown in Table 2. The superscripts a, b or c indicate the scenario at which a specific rule belongs to. Hence, scenario (a) has nine known rules, (b) has six and (c) five. The goal is training the FNN in order to generate a complete table (FAM bank) for each scenario.

After all FAM banks have been obtained for the scenarios a, b and c they were used for solving the truck backer-upper problem and the final result compared with those obtained by Kosko[8]. In Table 3, the final states of the variables (x, y, ϕ) for two initial positions $(20, 20, 30°)$ and $(30, 10, 220°)$ are shown. Fig. 7 shows the results for all scenarios.

Table 2: Sets of known rules for testing the FNN

		LE	LC	CE	RC	RI
		x				
ϕ	RB	PS^{a}		PM^{a,c}		PB^{a,b}
	RU					
	RV					
	VE	NM^{a,c}		ZE^{a,b,c}		PM^{a,b,c}
	LV					
	LU					
	LB	NB^{a,b}		NM^{a,b,c}		NS^{a,b,c}

Table 3: FNN performance results

SETS	FINAL STATES					
	$(20, 20, 30°)$			$(30, 10, 220°)$		
	x	y	ϕ	x	y	ϕ
Kosko	50.00	99.21	90.00	50.00	99.66	89.94
a	45.34	99.87	90.17	45.49	99.46	90.32
b	54.77	99.34	89.36	54.63	99.02	88.96
c	49.09	99.60	88.75	49.02	99.82	88.65

The obtained results indicate scenario c presents a performance similar to Kosko's. This similarity degree may be better analyzed using the concept of linguistic trajectory. Hence, consider that the state variables x_1 e x_2 could have n_1 and n_2 linguistic values, respectively. In this situation, $(n_1 \times n_2)$ corresponds to the maximum number of rules in the fuzzy knowledge database. By definition, the crisp point (x_1, x_2) in the state space belongs to the subspace associated with rule j, if:

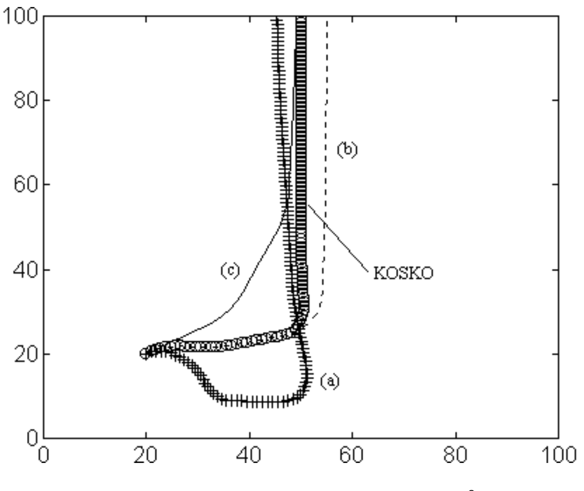
$$\forall j \neq k \, \forall (x_1, x_2) : \mu_{R_j}(x_1, x_2) \geq \mu_{R_k}(x_1, x_2) \quad (34)$$

where, $\mu_{R_j}(x_1, x_2)$ and $\mu_{R_k}(x_1, x_2)$ are fuzzy relation degrees associated to the j-th and k-th rules, respectively.

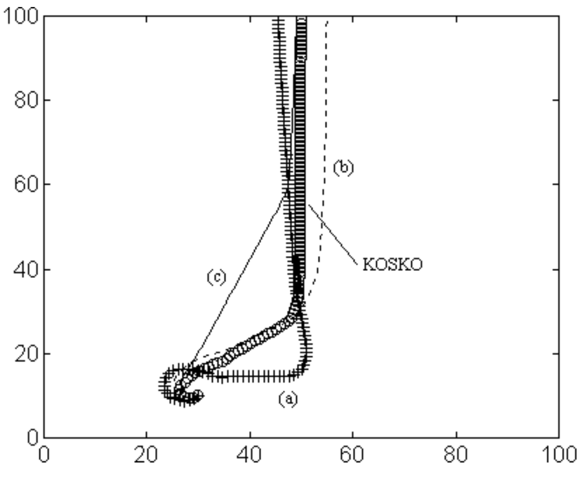
The state variables trajectories can be mapped in the fuzzy rules subspaces and the rules sequence been fired is called "linguistic trajectory". Fig. 8 shows an example of the linguistic trajectories obtained from scenario c and the Kosko's FAM bank, both starting from the same initial position $(30, 10, 220°)$. Notice that the FNN was able to determine a bank that conducts the truck for the "docking zone" (final state), spending less energy than Kosko's bank.

6. Conclusion

A fuzzy neural network (FNN) architecture for generating fuzzy rule database from sparse expert knowledge and handling fuzzy information (fuzzy input and

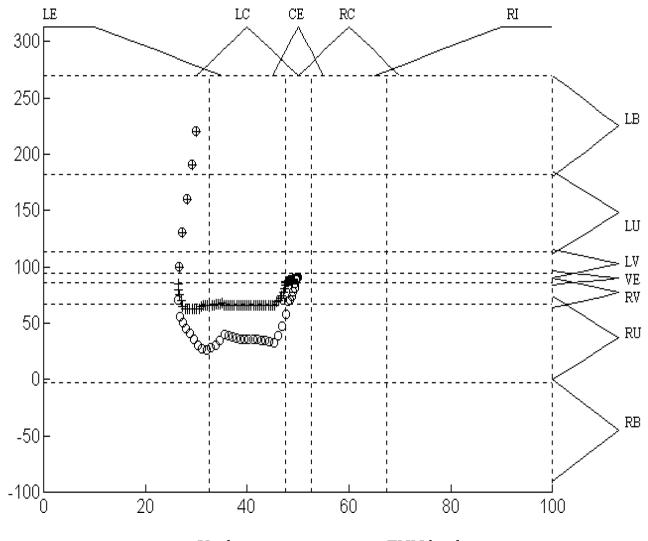


(a) - Trajectories with initial position $(20, 20, 30°)$



(b) - Trajectories with initial position $(30, 10, 220°)$

Figure 7: Trajectories for scenarios a, b and c (Table2)



ooooo Kosko +++++ FNN bank

Figure 8: Linguistic trajectories in the plane of the state variables $\phi \times x$

output vectors) was proposed. In order to improve its convergence during the learning phase, a fuzzy convergence controller for automatic adaptation of the learning and momentum parameters was described. The FNN performance was analyzed in solving the truck backer-upper nonlinear control problem and the results compared to Kosko's [8]. Simulation results and comparison of analysis show that the proposed FNN presents good trajectories, driving the truck to the desired final state, consuming less energy than Kosko's. A self-organizing FNN is under investigation for solving practical real-time MIMO problems taking into account sparse information acquired from multiple experts. Further improvements will be soon reported.

References

[1] E. Mandani. Applications of fuzzy algorithms for simple dynamic plant. *Neural Networks*, 121(5):1585–1588, 1974.

[2] D. Driankov. *An introduction to fuzzy control*. Spring-Verlag, New York, 1993.

[3] A. A. da Silva, P. Nascimento, G. Lambert-Torres, and L. B. da Silva. An alternative approach for adaptive real-time control using a nonparametric neural network. In *Proceedings on Industry Applications Conference*, pages 1788–1794, 1995.

[4] R. Gudwin, F. Gomide, and M. A. Neto. A neural fuzzy approach for fuzzy system design. In *Proceedings on IEEE Internacional Conference on Fuzzy Systems*, pages 481–486, 1998.

[5] H. Ishibuchi and H. Tanaka. Neural networks that learning from fuzzy if-then rules. *IEEE Trans. on Fuzzy Systems*, 1(2):85–97, 1993.

[6] M. Figueiredo and F. Gomide. A neural fuzzy approach for fuzzy system design. In *Proceedings on Internacional Conference on Neural Networks*, pages 420–425, 1997.

[7] W. A. Farag, V. Quintana, and G. Lambert-Torres. A genetic-based neuro-fuzzy approach for modeling and control of dynamical systems. *IEEE Trans. on Neural Networks*, 5(9):756–767, 1998.

[8] B. Kosko. *Neural Networks and Fuzzy Systems: a dynamical approach to machine intelligence*. Prentice Hall, N.J, USA, 1992.

[9] C.-M. Kuan and K. Hornik. Convergence of learning algorithms with constant learning rates. *IEEE Trans. on Neural Networks*, 2(5):484–489, 1991.

[10] P. Arabshahi. Fuzzy parameter adaptation in optimization: some neural net training examples. *IEEE Trans. on Computacional Science and Engineering*, 3(1):57–65, 1996.

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

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

[13] D. Rumelhart and J. McClelland. *Parallel Distributed Processing*. MIT Press, Cambridge, MA, 1986.

[14] D. Nguyen and B. Widrow. The truck baker-upper: An example of self-leraning in neural networks. In *Proceedings of the I International Joint Conference on Neural Networks*, pages 357–363, 1989.