# Fuzzy Neural Networks as "Good" Function Approximators

**Rita Lovassy[1], László T. Kóczy[2,3], Imre J. Rudas[4], László Gál[2,5]**

[1] Inst. of Microelectronics and Technology, Kandó Kálmán Faculty of Electrical Engineering, Óbuda University
Tavaszmező u. 15-17, 1084, Budapest, Hungary
E-mail: lovassy.rita@kvk.uni-obuda.hu

[2] Dept. of Informatics
Faculty of Engineering Sciences, Széchenyi István University
Egyetem tér 1, 9026, Győr, Hungary
E-mail: koczy@sze.hu

[3] Dept. of Telecommunications and Media Informatics
Budapest University of Technology and Economics
Magyar tudósok krt. 2. 1117, Budapest, Hungary

[4] Inst. of Intelligent Engineering Systems, John von Neumann Faculty of Informatics, Óbuda University
Bécsi út 96/b, 1034 Budapest, Hungary
E-mail: rudas@uni-obuda.hu

[5] Dept. of Technology, Informatics and Economy, University of West Hungary
Károlyi G. tér 4., 9700 Szombathely, Hungary
E-mail: gallaci@ttmk.nyme.hu

***Abstract:*** *The paper discusses the generalization capability of two hidden layer neural networks based on various fuzzy operators introduced earlier by the authors as Fuzzy Flip-Flop based Neural Networks (FNNs), in comparison with standard networks tansig function based, MATLAB Neural Network Toolbox in the frame of a simple function approximation problem. Various fuzzy neurons, one of them based on new fuzzy intersection and union pair, and two other selected well known fuzzy operators (Łukasiewicz and Dombi operators) combined with standard negation had been proposed as suitable for the construction of novel FNNs. The experimental results show that these FNNs provide rather good generalization performance, with far better mathematical stability than the standardneural networks and are more suitable to avoid overfitting in the case of test data containing noisy items in the form of outliers.*

***Keywords****: multilayer perceptrons based on fuzzy flip-flops, function approximation, fuzzy neural network generalization capability*

# 1 Introduction

Artificial neural networks and fuzzy logic systems share common features and techniques in the context of approximate reasoning. The main idea is using the high flexibility of neural networks produced by learning, in order to tune the membership functions used in fuzzy control. The approximate reasoning capability, transparency and interpretability of fuzzy sets, the learning capabilities and the property of auto-adaptability of neural networks, furthermore the optimal structure approximation properties of evolutionary, especially bacterial algorithms were developed with the aim to deal with problems which were hard to solve using traditional techniques.

In the years 1990-92 papers by D. Dubois, M. Grabisch and H. Prade [5], B. Kosko [16], furthermore Wang and Mendel [23], [24] proved almost simultaneously that fuzzy systems are universal approximators. In 1997 E. P. Klement, L. T. Kóczy and B. Moser [14] argued that fuzzy systems can only be universal approximators in a rather restricted sense, because of the limits set by computational complexity. The authors also exemplified the main approaches to realize the idea of controlling real world processes by means of linguistic variables.

In the field of artificial neural networks (connectionist models, parallel distributed processing systems and neuromorphic systems) mathematical function approximation using input-output data pairs from a set of examples is the object of study in different applications such as applied mathematics, and computer science. The paper of Hornik, Stinchcombe and White [11], established that standard multilayer feedforward networks with a single hidden layer constitute a class of universal approximators. They gave a general investigation of the capabilities and properties of multilayer feedforward networks, without any suggestion to the number of hidden units needed to achieve a given accuracy of approximation. The function approximation capability of multilayered neural networks was studied in detail by Ciuca [2], Cybenko [3], Funahashi [6], Hecht-Nielsen [10] and Ito [13]. They proved that any continuous function can be approximated by a three-layered feedforward network with hidden sigmoid units and one linear output unit. The use of four-layered (that have two sigmoid unit layers) neural network as universal approximators of continuous functions have been investigated by Funahashi [6], Girosi and Poggio [9] and Hecht-Nielsen [10]. Kurkova [17] studied also multilayer feedforward networks with sigmoid activation function approximation capabilities, analyzing also their computational complexity issues. Blum and Li showed [1] that four-layered feedforward networks with two hidden layers of semi linear units and with a single linear output unit are universal approximators. In [12] Hornik generalized the set of activation functions. He concluded that the neural networks approximation capability is very strongly dependent on the multilayer feedforward architecture, and less on the choice of the activation function. Furthermore, the number of hidden units exponentially depends on the

dimension of the approximated function. Unknown function approximation or model approximation is needed in many areas, and has been widely investigated.

Our goal was to develop well-performing fuzzy neural networks with a satisfactory approximation of unknown datasets, without prior knowledge of the system properties. In our simulations we checked the generalization capability of the FNNs, by adding noise in the form of outlier points to the training data. In the next we will present how standard NNs do not recognize properly the original trend (function) represented by the data, and how they learn outlier data points similarly to the original data, resulting in overfitting. This means poor generalization, and lack of proper interpolation.

This paper is organized as follows. After the Introduction, in Section 2 the sigmoid function generators derived from fuzzy J-K and D flip-flops are briefly reviewed. The suggested neural network is four-layered with hidden nodes defined by fuzzy flip-flop neurons (as sigmoid function generators) and a linear output node. Comparison of various FNNs and the standard tansig neural network generalization capability is presented in Section 3. As a somewhat surprising result is proposed that the FNNs have a more robust behaviour than the tansig based simulator. The FNNs aim at the reduction of overfitting, leading to much better generalization abilities of the network. Finally, some concluding remarks are given.

# 2 Multilayer Perceptrons Based on Fuzzy Flip-Flops

## 2.1 Fuzzy J-K and D Flip-Flops ($F^3$s) Neurons

### 2.1.1 Fuzzy Flip-Flop Neurons Derived From Fuzzy J-K Flip-Flops

The unified formula of the fuzzy J-K flip-flop was expressed as follows [22]:

$$Q_{out} = (J \vee \overline{K}) \wedge (J \vee Q) \wedge (\overline{K} \vee \overline{Q}) \tag{1}$$

The output state $Q_{out}$ of a J-K flip-flop is characterized as a function of both the present state $Q(t)$ and the two present inputs $J(t)$ and $K(t)$. In the next, $J$, $K$ and $Q$ will be used instead of $J(t)$, $K(t)$ and $Q(t)$, respectively, as simplified notations. The over bar denotes the standard negation (e.g. $\overline{K} = 1 - K$); furthermore $\wedge$ and $\vee$ denote fuzzy operations (t-norm and t-conorm, labeled in the next as *i* and *u*). $J, K, Q, Q_{out} \in [0,1]$.

In our previous work [19] we proposed the construction of a neuron unit, a combinational sigmoid generator derived from arbitrary fuzzy J-K flip-flop where $\overline{Q}$ is fed back to *K* and (old) *Q* is fixed.

In this approach, the output of fuzzy J-K flip-flop neuron depends on the value of $Q_{\text{fix}}$ and input values of *J*. Substituting $\overline{K} = Q$ (1 - *K* = *Q*) in the unified formula of the fuzzy J-K flip-flop (eq. 1), for a fix *Q* value, the characteristic equation of fuzzy J-K flip-flop neuron is

$$Q_{out} = (J \; u \; Q_{\text{fix}}) \; i \; (J \; u \; Q_{\text{fix}}) \; i \; \left(Q_{\text{fix}} \; u \; \left(1 - Q_{\text{fix}}\right)\right) \tag{2}$$

### 2.1.2 Fuzzy Flip-Flop Neurons Derived From Fuzzy D Flip-Flops

In [19] we proposed the construction of fuzzy D flip-flop neuron which is a combinational sigmoid generator. This unit is derived from arbitrary fuzzy J-K flip-flop by connecting the inputs of fuzzy J-K flip-flop in a particular way, namely by applying an inverter in the connection of the input *J* to *K*. Starting from the fundamental equation of fuzzy J-K flip-flop and substituting $\overline{K} = J$ in equation (1) and letting *D* = *J* for a fix *Q* value, the characteristic equation of fuzzy D flip-flop neuron is

$$Q_{out} = (D \; u \; D) \; i \; (D \; u \; Q_{\text{fix}}) \; i \; \left(D \; u \; \left(1 - Q_{\text{fix}}\right)\right) \tag{3}$$

Among others, a new pair of conjunction and disjunction, the Trigonometric t-norm and t-conorm were introduced in [8]. These new fuzzy operations, furthermore the well known Dombi and Łukasiewicz norms combined with the standard negation were applied for forming fuzzy neurons according to 2.1.1 and 2.1.2. Table I shows the above mentioned two well known fuzzy operations, and the new triangular t-norm and t-conorm expressions.

## 2.2 Neural Networks Based on Fuzzy Flip-Flops

We have proposed an FNN network as a four-layered Multilayer Perceptron (MLP) with hidden nodes defined by fuzzy flip-flop neurons (as sigmoid function generators) and a linear output node. This network type presented rather good function approximation properties. In our approach the weighted input values were connected to inputs *J* and *K* of the new fuzzy flip-flop neuron based on a pair of t-norm and t-conorm, having quasi sigmoid transfer characteristics. The output signal is then computed as the weighted sum of the input signals transformed by the transfer function.

TABLE I.
SELECTED T-NORMS AND T-CONORMS

| Fuzzy Operation | t-norm $i(x, y)$ |
|---|---|
| Dombi (D) [4] | $\left(1+\left(\left(\frac{1}{x}-1\right)^{\alpha}+\left(\frac{1}{y}-1\right)^{\alpha}\right)^{1/\alpha}\right)^{-1}$ |
| Trigonometric (Trig) [8] | $\frac{2}{\pi}\cdot\arcsin\left(\sin\left(x\frac{\pi}{2}\right)\cdot\sin\left(y\frac{\pi}{2}\right)\right)$ |
| Łukasiewicz (L) [15] | $\max(0, x+y-1)$ |
| Fuzzy Operation | t-conorm $u(x, y)$ |
| Dombi (D) | $\left(1+\left(\left(\frac{1}{x}-1\right)^{-\alpha}+\left(\frac{1}{y}-1\right)^{-\alpha}\right)^{-1/\alpha}\right)^{-1}$ |
| Trigonometric (Trig) | $\frac{2}{\pi}\cdot\arccos\left(\cos\left(x\frac{\pi}{2}\right)\cdot\cos\left(y\frac{\pi}{2}\right)\right)$ |
| Łukasiewicz (L) | $\min(1, x+y)$ |

The deployment of the Levenberg-Marquardt algorithm (LM) [21] and of the Bacterial Memetic Algorithm with Modified Operator Execution Order Algorithm (BMAM) [7] was proposed and applied for FNNs variables optimization and training [18] as follows.

During the execution of the algorithm each individual is selected one by one, and 20 generations of 5 individuals with 5 clones are chosen to obtain the best fitting variable values, with the lowest performance. The same part or parts are selected randomly from the clones and mutated. The LM method nested into the evolutionary algorithm is applied for 5 times for each clone. Several tests have shown that it is enough to run 3 to 5 of LM iterations per mutation to improve the performance of the whole algorithm. The best clone is selected and transferred with its all parts to the other clones. Choosing-mutation-LM-selection-transfer cycle is repeated until all the parts are mutated, improved and tested. The best individual is remaining in the population, all other clones are deleted. This procedure is repeated until all the individuals are taking part in the modified bacterial mutation. As a second main step, LM is applied 7 times for each individual executing several LM cycles during the bacterial mutation after each mutation step. In the applied algorithm the gene transfer operation is completely excluded.

# 3 Results on the Generalization Capability of FNNs Compared to Standard Simulated Neural Networks

The following two dimensional test function was investigated among others:

$$f_2(x_1,x_2)=\left(\cos\left(\arctan\left(\frac{x_1}{x_2}\right)\right)+1\right)\cdot e^{-\frac{r^2}{50}}\cdot\sin^5\left(\frac{r}{10}\right)$$

$$r=\sqrt{x_1^{\,2}+x_2^{\,2}},\ x_1,x_2\in[-20,20] \tag{4}$$

where arctan is the four-quadrant inverse tangent function. The two-input function is represented by 1600 input/output samples. Several numerically distant outlier points were introduced in our data set, in order to investigate the robustness of the approach.

During simulations we generate three set of data for each test function. The first belongs to the original input data set (*OrigY*), witch is without any noise. Next, we add noise to the original data set in the form of outliers, labeled as *NoisyY*, which we use for NNs training. We generate the third output data set by simulating the trained neural networks (*SimY*). In order to evaluate the networks' generalization capability, we will use the following three measures:

**Training MSE** (T MSE): The goodness of fit of the estimation (*SimY*) (mean squared errors with the noisy training samples, *NoisyY*);

$$\text{T MSE}=\frac{1}{N}\sum_{i=1}^{N}\left(SimY(i)-NoisyY(i)\right)^2 \tag{5}$$

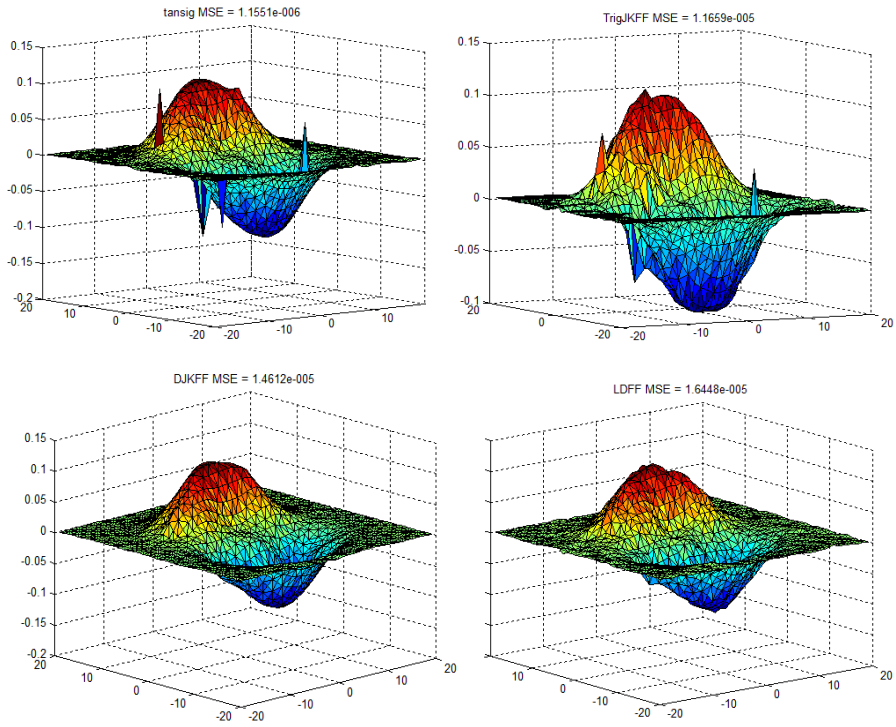
**Outliers MSE** (O MSE): The mean squared errors between the estimated data (*SimY*), and the original data set without outliers (*OrigY*);

$$\text{O MSE}=\frac{1}{N}\sum_{i=1}^{N}\left(SimY(i)-OrigY(i)\right)^2 \tag{6}$$

**Outliers Maximum Absolute Error** (O MAE): The maximum absolute error between the estimated data (*SimY*) and the original data set without noise (*OrigY*);

$$\text{O MAE}=\text{Max}\left|\overline{SimY}-\overline{OrigY}\right| \tag{7}$$

For the above two dimensional test function (eq. 4) with noise in the form of outliers the following observations were made based on the 10 runs average approximation goodness values (see Figure 1, and Table II.).




(a) tansig based NN, poor generalization
(b) Trigonometric type NN; poor generalization
(c) Dombi type FNN, good function approximation properties
(d) Łukasiewicz type FNN; good generalization

Figure 1. Some examples for good and bad generalization and robustness, 20-20 hidden nodes

The NNs function approximation goodness (T MSE value) is in the same order of magnitude when the order of the model is increased from 4 to 12. That means that the same number of $F^3$ neurons should be applied in order to achieve similar training properties with the tansig based NNs. For more than 8-8 neurons in the hidden layers, the standard NNs' O MSE and O MAE values are huge in comparison with the corresponding FNNs simulation results, which are less dependent on the network complexity. Using 20-20 hidden neurons, the tansig based and Trigonometric type networks have significant deviations from the target function which illustrate Figure 1(a, b). The functions surface indicate obvious overfitting. The Dombi type FNN perform good, followed by Łukasiewicz type FNN see Figure 1 (c, d), as the best generalizing network.

TABLE II.
COMPARISON OF THE MSE VALUES FOR TWO-DIMENSIONAL TEST FUNCTION

| MSE | NN size and type | | | |
|---|---|---|---|---|
| | 2D (1-4-4-1) | | | |
| | *tansig* | DJKFNN | TrigJKFNN | LDFNN |
| T MSE | $1.9 \cdot 10^{-5}$ | $2.69 \cdot 10^{-5}$ | $2.87 \cdot 10^{-5}$ | $5.19 \cdot 10^{-5}$ |
| O MSE | $3.23 \cdot 10^{-6}$ | $1.09 \cdot 10^{-5}$ | $1.32 \cdot 10^{-5}$ | $3.7 \cdot 10^{-5}$ |
| O MAE | 0.0095 | 0.0198 | 0.0187 | 0.0297 |
| | 2D (1-8-8-1) | | | |
| | *tansig* | DJKFNN | TrigJKFNN | LDFNN |
| T MSE | $1.55 \cdot 10^{-5}$ | $1.98 \cdot 10^{-5}$ | $1.83 \cdot 10^{-5}$ | $2.57 \cdot 10^{-5}$ |
| O MSE | $2.84 \cdot 10^{-6}$ | $5.37 \cdot 10^{-6}$ | $4.16 \cdot 10^{-6}$ | $1.22 \cdot 10^{-5}$ |
| O MAE | 0.0217 | 0.0143 | 0.0125 | 0.0191 |
| | 2D (1-12-12-1) | | | |
| | *tansig* | DJKFNN | TrigJKFNN | LDFNN |
| T MSE | $1.09 \cdot 10^{-5}$ | $1.62 \cdot 10^{-5}$ | $1.54 \cdot 10^{-5}$ | $2.26 \cdot 10^{-5}$ |
| O MSE | $7.39 \cdot 10^{-6}$ | $3.88 \cdot 10^{-5}$ | $4.1 \cdot 10^{-6}$ | $9.4 \cdot 10^{-6}$ |
| O MAE | *0.0762* | 0.0162 | 0.0222 | 0.0183 |
| | 2D (1-16-16-1) | | | |
| | *tansig* | DJKFNN | TrigJKFNN | LDFNN |
| T MSE | $6.18 \cdot 10^{-6}$ | $1.37 \cdot 10^{-5}$ | $1.38 \cdot 10^{-5}$ | $1.84 \cdot 10^{-5}$ |
| O MSE | $1.17 \cdot 10^{-5}$ | $7.08 \cdot 10^{-6}$ | $4.53 \cdot 10^{-6}$ | $7.3 \cdot 10^{-6}$ |
| O MAE | *0.0916* | 0.0424 | 0.0287 | 0.0257 |
| | 2D (1-20-20-1) | | | |
| | *tansig* | DJKFNN | TrigJKFNN | LDFNN |
| T MSE | $2.67 \cdot 10^{-6}$ | $1.22 \cdot 10^{-5}$ | $1.16 \cdot 10^{-5}$ | $1.69 \cdot 10^{-5}$ |
| O MSE | $1.53 \cdot 10^{-5}$ | $6.95 \cdot 10^{-6}$ | $6.42 \cdot 10^{-5}$ | $7.01 \cdot 10^{-6}$ |
| O MAE | *0.0934* | 0.0512 | 0.0521 | 0.0246 |

Comparing the simulation results, we concluded, that the Łukasiewicz type fuzzy D flip-flop neurons based NNs performed best, with good generalization capability, keeping the O MAE values very close to each other (values between 0.0183 and 0.0257), followed by the Dombi FNNs: 0.0143-0.0512 and Trignometric one: 0.0125-0.0521, all of them are less dependent on the hidden layers neuron numbers. Finally, the tansig based NNs' O MAE value fluctuation is from 0.0095 until 0.0934. For more than 12-12 neurons, the maximum deviation of the outliers is nearly equal to the 0.1 value.

## Conclusions

In this paper we proposed a function approximator for a general class of multi-input and one-output systems. Our simulation experiments have shown that some of the best FNNs clearly outperform standard tansig based NNs from the generalization capability point of view, case of noisy data sets. Further advantage of these structures is their easy hardware implementability [20]. Thus, we propose that in the future FNN circuits be integrated and used as standard elements. Such circuits could be rather universal concerning the area of applicability, and cheap as

well because of their independence on the actual problem. In the future we will further investigate the use of such NNs for more complex data sets and models.

**Acknowledgement**

This work is partially supported by the project TAMOP 421B, a Széchenyi István University Main Research Direction Grant, further National Scientific Research Fund Grant OTKA K75711, and Óbuda University Grants.

**References**

[1] E. K. Blum and L. K. Li.: Approximation theory and feedfordward neural networks, Neural Networks, 4(4), 1991, pp. 511-515

[2] I. Ciuca and J. A. Ware: Layered Neural Networks as Universal Approximators, Fuzzy Days Dortmund, Germany, 1997, pp. 411-415

[3] G. Cybenko: Approximation by superposition of sigmoidal functions, Math. Contr. Signals. Syst. 2, 1989, pp. 303-314

[4] J. Dombi: A general class of fuzzy operators, the De Morgan class of fuzzy operators and fuzziness measures induced by fuzzy operators, Fuzzy Sets and Systems, 8, 1982, pp. 149-163

[5] D. Dubois, M. Grabisch and H. Prade: Gradual rules and the approximation of functions, Proc. of the 2nd International Fuzzy Systems Association Congress Iizuka, Japan, 1992, pp. 629-632

[6] K. I. Funahashi: On the approximate realization of continuous mapping by neural networks, Neural Networks, 2, 1989, pp. 183-192

[7] L. Gál, J. Botzheim and L. T. Kóczy: Improvements to the Bacterial Memetic Algorithm used for Fuzzy Rule Base Extraction, Proc. of CIMSA 2008, Computational Intelligence for Measurement Systems and Applications, Istanbul, Turkey, 2008, pp. 38-43

[8] L. Gál, R. Lovassy and L. T. Kóczy: Function Approximation Performance of Fuzzy Neural Networks Based on Frequently Used Fuzzy Operations and a Pair of New Trigonometric Norms, Proc. of WCCI 2010, IEEE World Congress on Computational Intelligence, Barcelona, Spain, 2010, pp. 1514-1521

[9] F. Girosi and T. Poggio: Representation properties of network: Kolmogorov's theorem is irrelevant, Neural Computation 1, 1989, pp. 456-469

[10] R. Hecht-Nielsen: Theory of the backpropagation neural network, Proc. of the Neural Networks, IJCNN, International Joint Conference on Neural Networks, 1989, pp. 593-605

[11] K. Hornik, M. Stinchcombe and H. White: Multilayer Feedforward Networks are Universal Approximators, Neural Networks, 2, 1989, pp. 359-366

[12] K. Hornik: Approximation capabilities of multilayer feedfordward networks, Neural Networks, 4, 1991, pp. 251-257

[13] Y. Ito: Approximation Capability of Layered Neural Networks with Sigmoid Units on Two Layers, Neural Computation, 6(6), 1994, pp. 1233-1243

[14] E. P. Klement, L. T. Kóczy and B. Moser: Are fuzzy systems universal approximators? International Journal of General Systems 28 (2–3), 1999, pp. 259 – 282

[15] E. P. Klement, R. Mesiar and E. Pap: Triangular Norms, Series: Trends in Logic, 8, 2000

[16] B. Kosko: Fuzzy Systems are Universal Approximators, Proc. of the IEEE International Conference on Fuzzy Systems, San Diego, CA, 1992, pp.1153-1162

[17] V. Kurkova: Kolmogorov's theorem and multilayer neural networks Neural Networks, 5, 1992, pp. 501-506

[18] R. Lovassy, L. T. Kóczy and L. Gál: Parameter Optimization in Fuzzy Flip-Flop Based Neural Networks, Proc. of INISTA 2009, International Symposium on Innovations in Intelligent Systems and Applications, Trabzon, Turkey, 2009, pp. 205-209

[19] R. Lovassy, L. T. Kóczy and L. Gál: Function Approximation Performance of Fuzzy Neural Networks, special issue of Acta Polytechnica Hungarica, Vol. 7(4), 2010, pp. 25-38

[20] R. Lovassy, L. T. Kóczy, I. J. Rudas and L. Gál: Hardware Implementable Neural Networks Based on Fuzzy Operators, World Conference on Soft Computing, WCONSC 2011, San Francisco, USA, Paper 175

[21] D. Marquardt: An Algorithm for Least-Squares Estimation of Nonlinear Parameters, SIAM J. Appl. Math. 11, 1963, pp. 431-441

[22] K. Ozawa, K. Hirota, L. T. Kóczy and K. Omori: Algebraic fuzzy flip-flop circuits, Fuzzy Sets and Systems 39/2, Holland, 1991, pp. 215-226

[23] L. X. Wang: Fuzzy systems are universal approximators, Proc. of the IEEE International Conference on Fuzzy Systems, San Diego, CA, 1992, pp. 1163-1169

[24] L. X. Wang and J. M. Mendel: Fuzzy basis functions, universal approximations and orthogonal least-squares learning, IEEE Transactions on Neural Nets, 3, 1992, pp. 807-814