Quasi Optimization of Fuzzy Neural Networks

Rita Lovassy[1,2] , László T. Kóczy[1,3] and László Gál[1,4]

[1] Inst. of Informatics, Electrical and Mechanical Engineering, Faculty of Engineering Sciences, Széchenyi István University Győr, Hungary
[2] Inst. of Microelectronics and Technology, Kandó Kálmán Faculty of Electrical Engineering, Budapest Tech, Budapest, Hungary
[3] Dept. of Telecommunication and Media Informatics, Budapest University of Technology and Economics, Hungary
[4] Dept. of Technology, Informatics and Economy University of West Hungary
Email: lovassy.rita@kvk.bmf.hu, koczy@sze.hu, gallaci@ttmk.nyme.hu

Abstract: The fuzzy flip-flop based multilayer perceptron, named Fuzzy Neural Network, FNN is proposed for function approximation. In recent years much effort has been made for the development of a special kind of bacterial memetic algorithm for optimization and training of the fuzzy neural network parameters. In this approach the FNN parameters have been encoded in a chromosome and participate in the bacterial mutation cycle. The quasi optimized FNN's performance based on various fuzzy flip-flop types has been examined with a series of multidimensional input functions.

Keywords: fuzzy flip-flops, fuzzy neural network, bacterial memetic algorithm

1 Introduction

Artificial neural networks and fuzzy logic models can approximate any multivariate nonlinear function [9]. The aim of this paper is essentially to show how fuzzy models, neural networks, and bacterial algorithms can be usefully merged and deployed to solve in the first step parameter optimization and the function approximation problems. The combination of these three techniques tries to minimize their weakness and to profit their advantages. The main idea is using the property of learning from the data of the network using a parallel structure which can be implemented in hardware.

In analogy with the binary flip-flops which are the basic units in every synchronous sequential digital circuit, the fuzzy flip-flops can be reckoned as the basic functional blocks for developing dynamical fuzzy systems. The fuzzy flip-flops used as neurons are interlinking to form a complex fuzzy neural network. How these neurons are organized is itself a highly complex problem. For a given

problem the best network is selected from the network performance and complexity points of view. This study has been focused more or less on the neuro-computations direction. A Fuzzy Flip-Flop Neural Network (FNN) as a novel implementation possibility of multilayer perceptron neural networks is investigated and its optimization and learning algorithm is proposed.

The paper is organized as follows. After the Introduction, in Section 2 the behavior of fuzzy J-K and D flip-flops based on two parametrical fuzzy operations are given. The construction of a single input-single output neuron unit from arbitrary fuzzy J-K flip-flop where $\overline{Q}$ is fed back to K and (the old) Q is fixed is proposed. In Section 3 the concept of the fuzzy flip-flop based multilayer perceptron, named Fuzzy Neural Network, FNN is introduced. The neuron element of FNN may be any F^3 with more or less sigmoidal transfer characteristics. The FNN can approximate any multidimensional input functions. The Levenberg-Marquardt method (LM) [12] and a special kind of bacterial memetic algorithm [13], the Bacterial Memetic Algorithm with Modified Operator Execution Order (BMAM) [4] is proposed for FNN parameter optimization and network training. The near optimal values of Q are given which are quasi independent from the input function complexity and FNN neuron number for a fix fuzzy operation parameter value. Finally, in Section 4 the quasi optimized FNN's performance based on various fuzzy flip-flop types has been examined with a series of multidimensional input functions.

2 Definitions and Properties of Several Fuzzy Flip-Flops

The concept of fuzzy flip-flop was introduced in the middle of 1980's by Hirota (with his students) [6]. The Hirota Lab recognized the essential importance of the concept of a fuzzy extension of a sequential circuit and the notion of fuzzy memory. From this point of view they proposed alternatives for "fuzzifying" digital flip-flops. The starting elementary digital units were the binary J-K flip-flops. Their definitive equation was used both in the minimal disjunctive and conjunctive forms. As fuzzy connectives do not satisfy all Boolean axioms, the fuzzy equivalents of these equations resulted in two non-equivalent definitions, "reset and set type" fuzzy flip-flops (F^3), using the concepts of fuzzy negation, t-norm, and s-norm operations. In [7] Hirota et al. recognized that the reset and set equations cannot be easily used as elements of memory module, because of their asymmetrical nature. In their 1988's paper [13] Ozawa, Hirota and Kóczy proposed a unified form of the fuzzy J-K flip-flop characteristic equation, involving the reset and set characteristics, based on min-max and algebraic norms. A few years later, the hardware implementation of these fuzzy flip-flop circuits in discrete and continuous mode was presented by the Hirota Lab [14]. The fuzzy flip-flop was proposed as basic unit in fuzzy register circuit.

The fundamental equation of the binary J-K flip-flops is:

$$Q(t+1) = \overline{\overline{J}K + \overline{J}\overline{Q} + KQ} = \left(J + \overline{K}\right)\left(J + Q\right)\left(\overline{K} + \overline{Q}\right) \tag{1}$$

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

$$Q(t+1) = (J \vee \overline{K}) \wedge (J \vee Q) \wedge (\overline{K} \vee \overline{Q}) \tag{2}$$

The concept of a novel fuzzy D flip-flop type was defined in [10]. Connecting the inputs of the fuzzy J-K flip-flop in a particular way, namely, by applying an inverter in the connection of the input J to K, case of $K = 1 - J$, a fuzzy D flip-flop is obtained. Substituting $\overline{K} = J$ in equation (2) and let $D = J$, the fundamental equation of fuzzy D flip-flop is

$$Q(t+1) = (D \vee D) \wedge (D \vee Q) \wedge (D \vee \overline{Q}) \tag{3}$$

In our previous papers [10], [11] the unified approach based on a set of eleven norms, namely standard, drastic, Łukasiewicz, Yager, Dombi, Hamacher, including algebraic, Frank, Dubois-Prade, Schweizer-Sklar and Fodor, combined with the standard negation, was analyzed in order to investigate, whether and to what degree they present more or less sigmoidal (s-shaped) $J \to Q(t+1)$ characteristics in particular cases, when $K = 1\text{-}Q$ (unified fuzzy J-K flip-flop with feedback), $K = 1 - J$ (new fuzzy D flip-flop derived from the unified fuzzy J-K one) with fixed value of Q.

Table 1 summarizes the transfer characteristics shape for every combination of unified J-K and D type F^3s with all above mentioned fuzzy operation pairs, the parameters of the norms chosen for suitable values. The S notation refers to sigmoidal character, the NS to non-sigmoidal shape.

TABLE 1 VARIOUS FUZZY FLIP-FLOPS CHARACTERISTICS

Fuzzy operation	J-K (unified; K=1-Q)	D (K=1-J)
Standard (min-max)	NS	NS
Algebraic	NS	NS
Drastic	NS	NS
Łukasiewicz	**NS**	**S**
Yager	**S**	**S**
Dombi	**S**	**NS**
Hamacher	**S**	**S**
Frank	**S**	**S**
Dubois-Prade	**S**	**S**
Schweizer-Sklar	NS	NS
Fodor	NS	NS

We have conducted extensive investigations by simulations and found that the $J \to Q(t+1)$ transfer characteristics of fuzzy J-K flip-flops based on Yager, Dombi, Hamacher, Frank and Dubois-Prade norms, further the $D \to Q(t+1)$ characteristics of (new) fuzzy D flip-flops of Łukasiewicz, Yager, Hamacher, Frank and Dubois-Prade operations show quasi sigmoidal curvature, while all other F^3s investigated have non-sigmoidal behavior.

The Yager and Dombi class of operations [16], [2] are given by the expressions:

$$i_Y(x,y) = 1 - \min\left(1, \left((1-x)^p + (1-y)^p\right)^{1/p}\right); \tag{4}$$

$$u_Y(x,y) = \min\left(1, \left(x^p + y^p\right)^{1/p}\right) \text{ and}$$

$$i_D(x,y) = 1/\left(1 + \left((1/x-1)^p + (1/y-1)^p\right)^{1/p}\right); \tag{5}$$

$$u_D(x,y) = 1/\left(1 + \left((1/x-1)^{-p} + (1/y-1)^{-p}\right)^{-1/p}\right)$$

where, parameter p lies within the open interval $(0,\infty)$. For simplicity we denote the t-norm with i, and the t-conorm with u, where the subscripts refer to the initial of the name of the norm: e.g., in case of Yager norms: $i_Y(x,y) = x\ i_Y\ y$ and $u_Y(x,y) = x\ u_Y\ y$.

3 Multilayer Perceptrons Constructed of Fuzzy Flip-Flops (Fuzzy Neural Network, FNN)

In general, two trainable layer networks with sigmoid transfer functions in the hidden layer and linear transfer functions in the output layer are universal approximators [8]. The neuro-fuzzy system proposed here is based on two hidden layers constituted from fuzzy flip-flop neurons. The FNN is a supervised feedforward network, applied in order to approximate a real life application, a two dimensional trigonometric function and a six dimensional benchmark problem. In a two-stage approximation process some neurons in the first layer split the input space into regions, and other neurons in that layer learn the local features [5]. A neuron in the second layer combines the output of neurons in the first layer, learning the global features for that region.

3.1 Parameter Optimization in the Fuzzy Neural Network

A change of t-norms, Q and p parameter value pairs in the fuzzy flip-flops characteristic equations leads to the modification of the slope of the transfer function, which will affect the learning rate in the implementation of neural networks. In the next it will be show that the near-optimal values of Q are independent from the input function type and FNN neuron number for a fix fuzzy operation parameter value. For simplicity the value of the parameters were fixed to $p = 2$ and respectively to 5 in case of Yager and Dombi fuzzy operations. It is known that simple parametrical t-norms, furthermore simple fuzzy flip-flop characteristic equation are uncomplicated for tuning and hardware realization. Rudas et al. [15] proposed the hardware implementation of a generation of parametric families of fuzzy connectives together with min-max, Łukasiewicz and drastic t-norms and co-norms. In [17] Zavala et al. used FPGA technology to implement the above mentioned fuzzy t-norms into an 8 bit single circuit that allows operation selection.

In this section optimal FNN parameter searching methods are presented. In particular for a predefined network architecture a gradient descent optimization method and an evolutionary algorithm tune the values of the parameter Q for fixed fuzzy operation parameter values. The application of several model identification algorithms in order to achieve as good as possible approximation features of the FNN have been proposed. The Levenberg-Marquardt algorithm is applied to determine the optimal Q intervals in subsection 3.1.1. In subsection 3.1.2 the BMAM method is proposed to identify in the established interval the problem dependent quasi optimal values.

3.1.1 Levenberg-Marquardt Algorithm (LM)

A second-order gradient algorithm, the Levenberg-Marquardt method is applied to find the optimal Q intervals for every combination of fuzzy J-K and D flip-flops with Yager and Dombi fuzzy operations.

To train this kind of network with the usual Levenberg-Marquard technique the derivatives of the transfer functions have to be calculated. Then the FNN can be used and trained in the usual way. During network training, the weights and thresholds are first initialized to small, random values and the network was trained in order to minimize the network performance function according with Levenberg-Marquardt algorithm with 100 maximum numbers of epochs as more or less sufficient. This can be used for training any network as long as its inputs, weights, and transfer functions can be differentiated. During the simulations the input data patterns were used like this: the first 60 percent for training and the remaining 20-20 percents for validation and test results. In order to check the goodness of the training in the test phase, the Mean Squared Error (MSE) as a measure of the error made by the FNN was used.

The errors for all input patterns were propagated backwards, from the output layer towards the input layer. The corrections to the weights were selected to minimize the residual error between actual and desired outputs. This algorithm can be viewed as a generalized least squares technique applied to the multilayer perceptron. The chosen target activation function is the *tansig* (hyperbolic tangent sigmoid transfer function).

The lowest MSE value indicated the optimal Q value which might lead to good learning and approximation properties. The result obtained for the training and test sets give the 300 runs average approximation goodness value. During simulations the median MSE values have been compared, considering them as the most important indicators of trainability. The median is a robust estimate of the center of a data sample, since outliers have little effect on it.

3.1.2 Bacterial Memetic Algorithm with Modified Operator Execution Order Algorithm (BMAM)

In the Bacterial Memetic Algorithm with Modified Operator Execution Order Algorithm, the learning of a neural network is formulated as a weight optimization problem, usually using the mean square error as fitness evaluation scheme. It has been shown that evolutionary algorithms work efficient for solving nonlinear and constrained optimization problems. These methods do not use derivatives of the functions, such as the gradient-based training algorithms. Similar to biological recombination, these methods are based on the search for a large number (population) of solutions. The algorithm starts with several alternative solutions to the optimization problem, which are the population individuals. The solutions are coded as binary strings. The basic steps followed by the algorithm embrace the bacterial mutation operation and the LM method.

The FNN weights, biases and Q values have been encoded in a chromosome and participated in the bacterial mutation cycle. In this application, according to the network size, a population with different parameters was initialized. Therefore a procedure is working on changing the variables, testing the model obtained in this way and selecting the best models. During simulations 200 generations of 5 individuals with 5 clones were chosen to obtain the best fitting variable values, with the lowest performance. Then the same part or parts of the chromosome is choose and mutate randomly, except one single clone that remains unchanged during this mutation cycle. The LM method nested into evolutionary algorithm is applied for 5 times for each individual. The selection of the best clone is made and transfers its parts to the other clones. The part choosing-mutation-LM method-selection-transfer cycle is repeated until all the parts are mutated, improved and tested. The best individual is remaining in the population, and all other clones are deleted. This process is repeated until all the individuals have gone through the modified bacterial mutation. The Levenberg-Marquardt method is applied 7 times for each individual, executing several LM cycles during the bacterial mutation after each mutation step.

In this way the local search is done for every global search cycle. The quasi optimal values can be identified at the end of the BMAM training algorithm.

Using the BMAM algorithm, executing several LM cycles after each bacterial mutation step, there is no need to run a few hundred training cycles to find the lowest MSE value. After just one run the optimal parameter values could be identify. The optimal *Q* intervals obtained with the LM algorithm have been reduced in this case to a single quasi optimal value identified at the end of the training process. In this approach, with only one training cycle an acceptable model whose error does not exceed an acceptable level has been obtained.

4 Numerical Simulation and Results

The FNN architecture is predefined, depending on the input function complexity. The proposed network topologies approximate with sufficient accuracy the test functions. The FNN training algorithm applied to different architectures will have different performance. In this approach the choice of an optimal network design for a given problem is a guessing process. With a sufficient neuron number the network can be trained with a sufficient accuracy in sufficient training time. In particular, the application of LM and BMAM algorithms are proposed for training various FNNs with different structures in order to approximate a real-life application, a two dimensional trigonometric function and a benchmark problem which dates were selected from the input/output test points of a six dimensional non-polynomial function. The quasi optimization of FNN is demonstrated in the following case studies. The test functions are arranged in the order of complexity.

Real - Life Application: Approximation of a Nickel Metal Hydride Battery Cell Charging Characteristics

In this particular case, the FNN approximates a Nickel Metal Hydride (NiMH) Battery Cell charging characteristics [3], a on one-input real-life application.

The nickel-metal batteries can be repeatedly charged and discharged for about 500 cycles, in approx. 1 hour. The charge characteristics are affected by current, time and temperature. The battery voltage raises when the charging current is increased or when the temperature is low. The test function is a characteristic between the battery capacity input and the cell voltage. The battery type was GP 3.6V, 300mAH, 3x1.2V NiMH, tested during 1.5 hour with 300mA and 25˚C.

Two - Input Trigonometric Function

We used the next two dimensional polynomial input function as test function

$$y = e^{-\frac{r^2}{100}} \cdot \cos\left(\frac{r}{2}\right); \text{ where } r = \sqrt{x_1^2 + x_2^2},\ x_1,\ x_2 \in [-20, 20] \tag{6}$$

Six Dimensional Non-Polynomial Function

This widely used target function expression is given by the following expression [1]:

$$y = x_1 + x_2^{0.5} + x_3 x_4 + 2e^{2(x_5 - x_6)} \tag{7}$$

4.1 Case Study 1: Optimization of *Q* with LM Algorithm

Figure 1 shows the fluctuation of the *Q* values for fuzzy J-K and D flip-flops based on Yager norms. The real-life application (denoted with 1D) is approximated with a 1-2-2-1 FNN size, described by a set of 543 input/output data sets selected equidistantly from a set of 2715 test points. A 1-20-20-1 feedforward neural network structure based on F^3s was proposed to approximate the two input trigonometric function (denoted with 2D), represented by 1600 input/output data sets. To approximate the six dimensional problem we studied a 1-10-5-1 FNN (6D-1) furthemore a 1-10-10-1 FNN (6D-2) sizes given by 200 datas.

The number of neurons was chosen after experimenting with different size hidden layers. Smaller neuron numbers in the hidden layer result in worse approximation properties, while increasing the neuron number results in better performance, but longer simulation time.

Evaluating the simulation results, low MSE values appears in the same domains which fact leads to assess optimal *Q intervals*, depending less from the input function type and FNN neuron number for a fix fuzzy operation parameter value. In case of J-K FNN based on Yager norms the *Q* optimum interval is $[0;\ 0.3]$, in case of D type FNN based on the same norms $Q \in [0.1;\ 0.3]$ (see Figure 1). The LM training algorithm is very sensitive to the parameter's initial position in the search space. An inconvenient generated random parameter set leads in a hardly trainable neural network with bad performance, because the LM method is a local searcher.

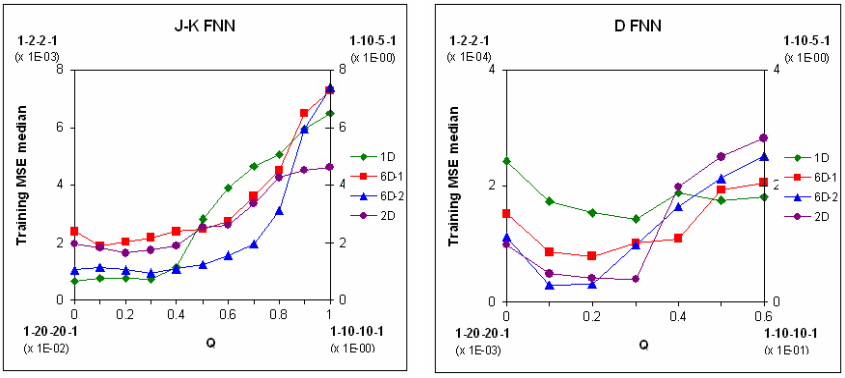

Figure 1
Training MSE median values for J-K FNN and D FNN based on Yager norms

4.2 Case Study 2: Optimization of *Q* with BMAM Algorithm

Using the BMAM algorithm, executing several LM cycles after each bacterial mutation step, there is no need to run 300 training cycles to find the lowest MSE value. After just one run the optimal parameter values could be identified. The quasi optimal *Q* values for various FNNs identified by the BMAM algorithm are listed in Table 2. The optimal *Q* intervals obtained with the LM algorithm have been reduced in this case to a single quasi optimal value identified at the end of the training process. In this approach, with only one training cycle an acceptable model whose error does not exceed an acceptable level has been obtained.

4.3 Comparison of Different FNNs with Respect to the Training Algorithm

Figure 2 (real-life application) presents the graphs of the simulations in case of fuzzy J-K and D flip-flops trained with BMAM algorithms based on Yager and Dombi norms, using quasi optimized values.

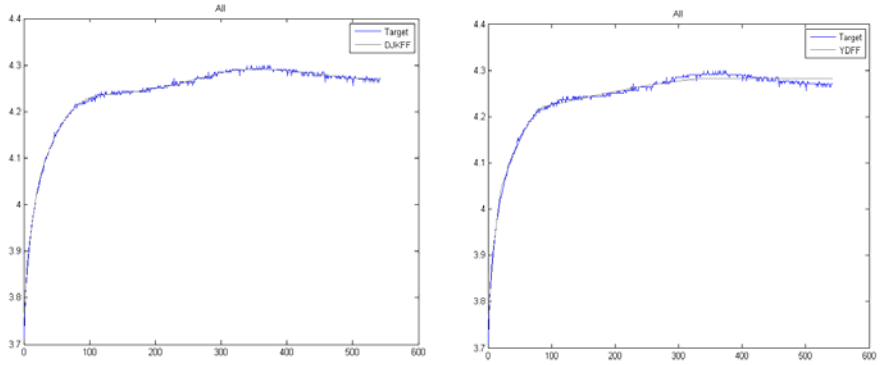

Figure 2
Simulation results of J-K FNN based on Dombi norms and D FNN based on Yager norms

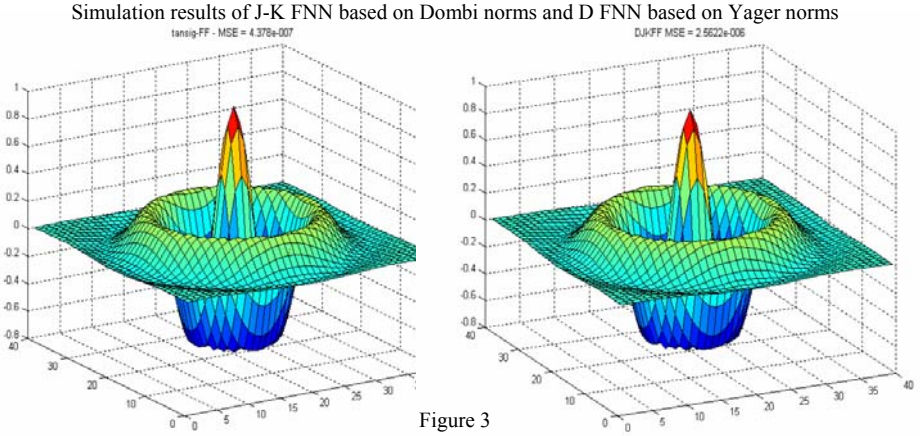

Figure 3
Simulation results of traditional NN and J-K FNN based on Dombi norms

TABLE 2 TRAINING MSE MEDIAN AND QUASI OPTIMAL *Q* VALUES

Test Func.	*tansig* NN	J-K FNN				D FNN			
		Yager type		Dombi type		Yager type		Dombi type	
		Q	MSE	*Q*	MSE	*Q*	MSE	*Q*	MSE
1D 1-2-2-1	1.34 e-05	0.06	7.82 e-05	0.27	1.96 e-05	0.26	2.12 e-05	0.21	3.27 e-04
2D 1-20-20-1	4.38 e-07	0.13	1.41 e-05	0.21	2.56 e-06	0.24	5.77 e-06	0.22	8.41 e-04
6D 1-10-10-1	3.84 e-07	0.30	3.21 e-04	0.25	1.31 e-06	0.31	1.14 e-04	0.19	4.45 e-02

Figure 3 compares the approximation goodness of the traditional neural network and J-K FNN based on Dombi norms in order to approximate the two dimensional test function. Table 2 summarizes the near optimal *Q* values for fixed fuzzy operation parameter values furthermore the average approximation goodness made by BMAM algorithm, indicating the median MSEs.

The BMAM algorithm is efficient in searching the global minimum, but this method still search in a subset of all possible parameters. The number of generations and the population size are fixed, chosen corresponding to the difficulty of the problem to be solved. There is no guarantee that a FNN will converge to the local optimum after those generations. The mentioned training algorithms produce different training results. Comparing the corresponding MSE median values (Figure 1 and Table 2) we concluded that the bacterial memetic algorithm combined with the LM method produces better function approximation performance in every case than the LM alone. For example, the Yager type D FNN, 2D function approximation case, LM may reduce the FNN error to 1.51 e-04 (Figure 1) through training, but the BMAM could reduce the error to 5.77 e-06 (Table 2) due to its global search capability. Several test function have been used to test that the values of the parameter *Q* are less dependent (with values in a narrow interval) from the input functions and network size for a fixed fuzzy operation parameter value.

Conclusions

The benefits arising from the combination between ANNs, fuzzy systems and bacterial memetic algorithms have been investigated in this paper. A hybrid algorithm is proposed combining the mentioned methods for FNN parameter optimization and training. The BMAM method achieves better function

approximation results than the non-evolutionary LM algorithm. The best training algorithm is always problem dependent. Future scope for research lies in the investigation of a simultaneous optimization of the FNN learning parameters and network architecture.

Acknowledgement

This paper was supported by the Széchenyi István University Main Research Direction Grant 2009, National Scientific Research Fund Grant OTKA K75711, further by Budapest Tech Grants.

References

[1] J. Botzheim, B. Hámori, L. T. Kóczy, A. E. Ruano: Bacterial algorithm applied for fuzzy rule extraction, IPMU 2002, Annecy, France, pp.1021-1026

[2] 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

[3] R. Fan, "NiMH Battery Charger Reference Design", Designer Refernce Manual, 2003

[4] L. Gál, J. Botzheim, L. T. Kóczy: Improvements to the Bacterial Memetic Algorithm used for Fuzzy Rule Base Extraction, in Proc. of CIMSA 2009, Computational Intelligence for Measurement Systems and Applications, Istanbul, Turkey, 2008, pp. 38-43

[5] S. Haykin: Neural Networks, A Comprehensive Foundation, second edition, Prentice-Hall, 1999

[6] K. Hirota, K. Ozawa: Concept of fuzzy flip-flop, Preprints of 2nd IFSA Congress, Tokyo, 1987, pp. 556-559

[7] K. Hirota, K. Ozawa: Fuzzy flip-flop as a basis of fuzzy memory modules, In: M. M. Gupta et al., eds., Fuzzy Computing. Theory, Hardware and Applications, North Holland, Amsterdam, 1988, pp. 173-183

[8] K. M. Hornik, M. Stinchcombe, H. White: Multilayer feedforward networks are universal approximators, Neural Networks, Vol. 2(5), 1989, pp. 359-366

[9] V. Kecman: Learning and Soft Computing: Support Vector Machines, Neural Networks, and Fuzzy Logic Models, The MIT Press, Cambridge, MA, 2001

[10] R. Lovassy, L. T. Kóczy, L. Gál: Applicability of Fuzzy Flip-Flops in the Implementation of Neural Networks; Proc. of CINTI 2008, 9th International Symposium of Hungarian Researchers on Computational Intelligence, Budapest, Hungary, pp. 333-344

[11] R. Lovassy, L. T. Kóczy, L. Gál: Function Approximation Capability of a Novel Fuzzy Flip-Flop Based Neural Network, Proc. of IJCNN 2009, International Joint Conference on Neural Networks, Atlanta, USA, pp. 1900-1907

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

[13] N. E. Nawa, T. Furuhashi: Fuzzy Systems Parameters Discovery by Bacterial Evolutionary Algorithms, IEEE Tr. Fuzzy Systems 7(1999), pp. 608-616

[14] K. Ozawa, K. Hirota, L. T. Kóczy: Fuzzy flip-flop, In: M. J. Patyra, D. M. Mlynek, eds., Fuzzy Logic. Implementation and Applications, Wiley, Chichester, 1996, pp. 197-236

[15] I. J. Rudas, I. Z. Batyrshin, A. H. Zavala, O. C. Nieto, L. Horváth, L. V. Vargas: Generators of Fuzzy Operations for Hardware Implementation of Fuzzy Systems, Proc. of MICAI 2008, Advances in Artificial Intelligence, 7th Mexican International Conference on Artificial Intelligence, Mexico, October 27-31, 2008, pp.710-719

[16] R. R. Yager: On a general class of fuzzy connectives, Fuzzy Sets and Systems, 4, pp.235-242

[17] A. H. Zavala, O. C. Nieto, I. Batyrshin, L. V. Vargas: VLSI Implementation of a Module for Realization of Basic t-norms on Fuzzy Hardware, Proc. of FUZZ-IEEE 2009, IEEE Conference on Fuzzy Systems, Jeju Island, Korea, August 20-24, 2009, pp. 655-659