OPTIMAL NEURAL NETWORK STRUCTURES FOR MODELING AND CONTROL DESIGN

Ladislav Körösi
Department of Automatic Control Systems
Faculty of Electrical Engineering and Information Technology, SUT
Ilkovičova 3, 812 19 Bratislava
Slovak Republic
korosi@kasr.elf.stuba.sk

Štefan Kozák
Department of Automatic Control Systems
Faculty of Electrical Engineering and Information Technology, SUT
Ilkovičova 3, 812 19 Bratislava
Slovak Republic
korosi@kasr.elf.stuba.sk

Abstract – **The proposed paper deals with modeling and control of continuous-time processes using artificial neural network with orthogonal activation functions, applicable for real-time control. Linear control laws are frequently used in control algorithms. They are applicable for small deviations about the operating point. If large deviations occur, the controlled process model and the control algorithm have to be currently updated. If such tuning is frequently required, then automatic tuning or self-tuning predictive control becomes inevitable. In such a case the process modeling and design of nonlinear self-tuning control algorithms can essentially improve the performance. A genetic algorithm has been used to find the optimal neural structure for on-line identification with the best learning algorithm. A moving prediction horizon in the control algorithm described with a 5-th order polynomial found using the genetic algorithm has been compared with a constant prediction horizon. The proposed algorithms were verified on practical control problem and have proved a good performance. The obtained simulation and verification results are presented in the paper.**

I. INTRODUCTION

Application of artificial neural networks (ANN) in modeling and control originates from an attempt to model the nervous system and its activity. Due to its universal approximation ability, a neural network can be used either as a model, or controller or another intelligent unit in the closed loop, and thus substitute conventional closed loop items. For modeling nonlinear dynamic systems, a recurrent multi-layer ANN is frequently used. In many cases, the neuron model is represented with a perceptron. Several research works and papers dealing with nonlinear process modeling using ANN show high approximation precision at the expense of computing time; therefore these techniques can't be used for real-time applications. Presented limitations (large number of iterations, long solution time, large number of training samples, etc.) can be eliminated by selecting other types of activation functions (AF), e.g. the sigmoidal AF (SAF) [1],[2]. Recently, orthogonal activation functions (OAF) have been used for modeling highly nonlinear processes with high precision and have been successfully applied in the design of linear and non-linear self-tuning controllers. Compared with conventional modeling and control techniques, the SAF based techniques verified on a quantity of examples have shown a considerable modeling speed up.

The proposed paper focuses on modeling and control of continuous-time processes using artificial neural networks with OAF using genetic algorithm to optimize the ANN structure, cost function weight parameters and moving prediction horizon.

II. NEURAL NETWORK STRUCTURE USING ORTHOGONAL ACTIVATION FUNCTIONS

A typical neural structure used to approximate a non-linear function is the three-layer structure in Fig.1 [3]. The input layer has m nodes with inputs $u=[u(1)\ u(2),...u(m)]$. The hidden layer consists of neurons with orthogonal (orthonormal) activation functions. It is assumed that the AF's for these neurons belong to the same class of orthogonal functions and no two neurons have the same order of AF in the input blocs. Each neuron has a different order activation function beginning from 0 to n. This ensures that no cross-correlation occurs among the neurons in the hidden layer. The input and output layers consist of linear neurons. The weights between the input and the hidden layers are fixed and depend on the OAF type. The output of the network with OAF is given by the linear combination of activation functions

$$y(u,w)=\sum_{n_1=0}^{N_1-1}\cdots\sum_{n_m=0}^{N_m-1} w_{n_1}..n_m\phi_{n_1}..n_m(u)=\Phi^T(u)W \qquad (1)$$

where $u=[u_1\ u_2...u_m]^T$ is a m-dimensional input vector, N_i is the number of neurons associated with the i-th input and W is the vector of weights between hidden and output layers.

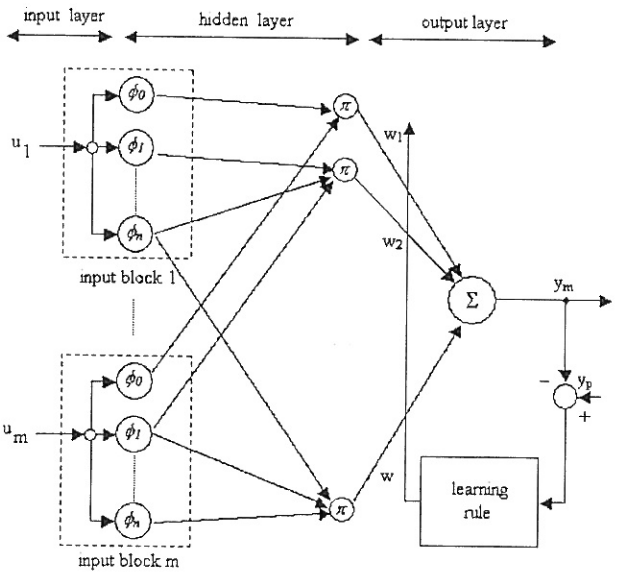

Fig.1 Block scheme of a three-layer ANN with OAF

The learning in ANN is performed by adapting the network weights so that the expected value of the mean squared error (MSE) between the network output and the training output is minimized. The recursive MSE learning algorithm is a popular training method for the OAF based ANN.
Minimizing the cost function

$$J_N = \frac{1}{2}e^2 \tag{2}$$

we obtain the final equations for the on-line recursive MSE algorithm

$$\varepsilon\left(k,\hat{W}(k-1)\right) = y_p(k) - \phi^T(k)\hat{W}(k-1) \tag{3}$$

$$K(k) = \frac{P(k-1)\phi(k)}{1+\phi^T(k)P(k-1)\phi(k)} \tag{4}$$

$$P(k) = P(k-1) - K(k)\phi^T(k)P(k-1) \tag{5}$$

$$\hat{W}(k) = \hat{W}(k-1) + K(k)\varepsilon\left(k,\hat{W}(k-1)\right) \tag{6}$$

where $\hat{W}(k)$ is a weight matrix, $K(k) = P(k)\phi(k)$.

The on-line recursive MSE algorithm modifications are:

1. Constant trace

$$A_m = 1, B_m = 1$$

$$\hat{y}(k) = \hat{y}(k-1) + K(k)\left(y(k) - \hat{y}(k)\right) \tag{7}$$

$$\bar{P}(k) = \frac{\left(P(k-1) + \frac{Z_1}{Z_2}\right)^{-1}}{\lambda} \tag{8}$$

$$P(k) = \frac{\alpha_{\max} - \alpha_{\min}}{tr\left(\bar{P}(k)\right)}\bar{P}(k) - \alpha_{\min} I \tag{9}$$

$$Z_1 = \left(P(k-1)\psi(k)\psi^T(k)P(k-1)\right) \tag{10}$$

$$Z_2 = 1 + \psi^T(k)P(k-1)\psi(k) \tag{11}$$

2. Exponential forgetting

$$K(k) = P(k-1)\psi(k)\left(Z_3\right)^{-1} \tag{12}$$

$$\hat{y}(k) = \hat{y}(k-1) + K(k)\left(y(k) - \hat{y}(k)\right) \tag{13}$$

$$\bar{P}(k) = \frac{P(k-1) - K(k)\psi^T(k)P(k-1)}{\lambda} \tag{14}$$

$$Z_3 = \lambda I + \psi^T(k)P(k-1)\psi(k) \tag{15}$$

3. Exponential forgetting and resetting algorithm

$$K(k) = \alpha P(k-1)\psi(k)\left(Z_4\right)^{-1} \tag{16}$$

$$\hat{y}(k) = \hat{y}(k-1) + K(k)\left(y(k) - \hat{y}(k)\right) \tag{17}$$

$$\bar{P}(k) = \frac{1}{\lambda}P(k-1) - Z_5 + Z_6 \tag{18}$$

$$Z_4 = 1 + \psi^T(k)P(k-1)\psi(k) \tag{19}$$

$$Z_5 = K(k)\psi^T(k)P(k-1) \tag{20}$$

$$Z_6 = \beta I - \delta P^2(k-1) \tag{21}$$

where P is a covariance matrix, w are the weights, $\psi(k) = \frac{dy(k)}{dw}$, $\alpha_{\min}, \alpha_{\max}$ are normalization intervals, λ is vector of eigenvalues, I is eye matrix.

The on-line method is described in more detail in [4], [5], [6] and [7].

III. CONTROLLER DESIGN

The control of nonlinear processes using the ANN is extraordinarily interesting from a practical point of view, leading to many new studies and results on control problems such as robust stability and performance improvement. As ANN modeling adapts to parametric and structural variations of the system, artificial ANN with OAF provides an efficient means for robustness maximization when modeling non-linear systems.

A nonlinear self-tuning predictive controller (Fig.2) consists from two blocks [6], [7]. Its first block is the feed-forward OAF based ANN; the ANN is trained on-line. The output from the network is the predicted system output. The second (optimization) block computes the system input, which ensures equality between system and network outputs.
The control variable can be determined as follows

$$u(k) = u(k-1) - \alpha\frac{\partial J(k)}{\partial u(k)}, \tag{22}$$

where $J(k)$ is the cost function in the form

$$J(w,e) = \frac{1}{2}\left\{[r(k+1) - y_m(k+1)]^2 + \gamma[\Delta u(k)]^2\right\}, \tag{23}$$

where r is a reference signal, y_m is the ANN model output, Δu is the control signal and γ is the weighting factor.
Applying the Quasi-Newton method, the control variable is

$$u(k) = u(k-1) - \left(\frac{\partial^2 J}{\partial u^2(k-1)}\right)^{-1}\cdot\frac{\partial J}{\partial u(k-1)} \tag{24}$$

The robust version of the proposed control algorithm can be expressed in the form

$$u(k) = u(k-1) - \left(\max\left(abs\left(\frac{\partial^2 J}{\partial u^2(k-1)}\right), h\right)\right)^{-1}\frac{\partial J}{\partial u(k-1)}, \tag{25}$$

where $h>0$ is a small constant.

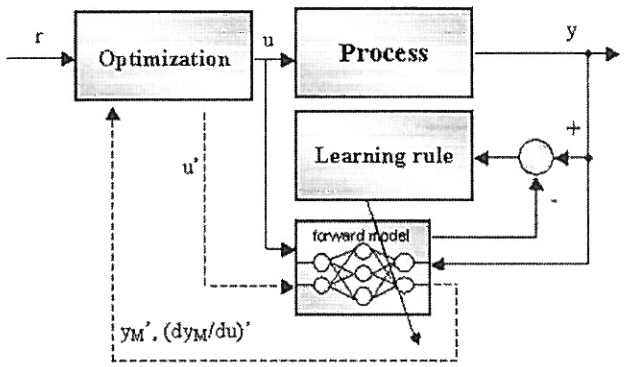

Fig.2 Nonlinear self-tuning predictive controller

The control algorithm contains these basic four steps:

1. initialization (process input-output sampling and normalization, $u(k)=u(k-1)$)
2. prediction of y_M and calculation of derivations
3. computation the optimal control u (recursive algorithm using equation 25)
4. denormalization

The above algorithm is described in more detail in [4], [5], [6] and [7].

IV. GENETIC ALGORITHMS

A genetic algorithm is an optimization technique that relies on parallels with nature. It can tackle a variety of optimization techniques provided that they can be parameterized in such a way that a solution to the problem provides measure of how accurate the solution found by the algorithm is. This measure we define as fitness. Using genetic algorithms for solving optimizations problems can be briefly described as follows. A population of possible solutions to an optimization problem is obtained in the vector form, the so-called chromosomes. Each vector consists of genes from the range given by its lower and upper limits. After calculating the fitness function for vectors a new population is a created. The new population comprises:

1. the best string or strings (with the smallest fitness - minimizations) to ensure the convergence
2. crossover and mutation operations be applied to the chosen strings
3. addition of new random strings

It is expected that the objective criterion values for different strings will gradually improve over generations, approaching the optimal values.

V. SIMULATION EXAMPLES

For testing the ANN modeling and control, the bioprocess (bioreactor) presented in [4] has been considered (Fig.3).

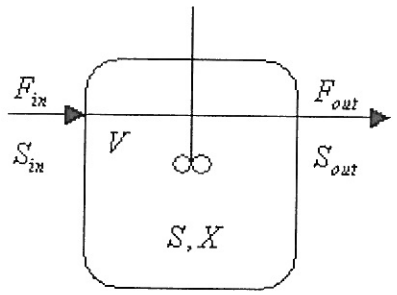

Fig.3 Bioreactor

The process is described by the following equations:

$$\frac{dX}{dt} = \mu(S)X - DX \tag{26}$$

$$\frac{dS}{dt} = -k_1\mu(S)X - k_2 v(S)X - DS + DS_{in} \tag{27}$$

$$\frac{dP}{dt} = v(S)X - DP \tag{28}$$

The matrix representation:

$$\frac{d}{dt}\begin{bmatrix} X \\ S \\ P \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -k_1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} SX & 0 \\ 0 & SX \end{bmatrix}\begin{bmatrix} \alpha \\ \beta \end{bmatrix} - D\begin{bmatrix} X \\ S \\ P \end{bmatrix} + \begin{bmatrix} 0 \\ DS_{in} \\ 0 \end{bmatrix}, \tag{29}$$

where X – cell mass concentration [g/l], P – product concentration [g/l], S – substrate concentration [g/l] with the following specified parameters:

$k_1 = 27.3, k_2 = 0, D = 0.3, S_{in} = 5, X(0) = 0.3, S(0) = 0.1,$

$\alpha = 0.2, \beta = 0.1.$

A. Neural network modeling

Process identification can be defined as finding such input-output relations and parameters, which describe the process behavior with a specified precision. For nonlinear process modeling and control, the black-box approach is used, that means observing the input-output data without any knowledge about the structure, order, mathematical description, etc.

To generate the optimal ANN structure a genetic algorithm has been used because finding the optimal ANN structure (including learning method, etc.) is a optimization problem with several parameters. The population consists of 100 strings. Each string contains 6 genes:

1. Number of delayed process inputs (nu) : <1,6>
2. Number of delayed process outputs (ny) : <1,6>
3. OAF order : <0,5>
4. OAF type: <Hermit, Laguerre, Legendre, Chebychev>
5. Learning method (rule): <constant trace, exponential forgetting, exponential forgetting and resetting algorithm >
6. Learning parameter : <0.05,1>

The algorithm uses only combination of the mutation and selection operations because each string contains different gene types. After 2000 simulations (20 populations) the sum squared error (SSE – fitness function) settled at 398.25 for training data and 147.36 for testing data with the following string

1. Number of delayed inputs : 4
2. Number of delayed outputs : 4
3. OAF order : 2
4. OAF type: Hermit
5. Learning method : exponential forgetting
6. Learning parameter : 0.98 (constant exp. forgetting factor)

For simulation, 2000 training and 2000 testing samples have been used for the sampling time $T=0.1h$. Several 2nd order OAF generated with the genetic algorithm are compared in the Table1. For each OAF the first and second rows show the SSE for training data and for testing data, respectively. The OAF based ANN has been compared with the perceptron with 8 input neurons and 24 hidden neurons (Tab.2). Graphical simulation results are shown in Figures 4 and 5.

Tab.1 Comparison of SSE for 2nd order OAF

	nu + ny	*nu + ny*	*nu + ny*	*nu + ny*
OAF	**1+1**	**2+2**	**3+3**	**4+4**
Chebychev	*397.2*	*398.5*	*401.63*	*398.41*
	159.02	*154.11*	*151.08*	*147.4*
Hermit	*397.2*	*398.5*	*401.61*	*398.25*
	159.02	*154.11*	*151.08*	*147.36*
Laguerre	*398.91*	*400.49*	*403.9*	*401.06*
	158.93	*154.31*	*151.57*	*148.16*
Legendre	*397.2*	*398.5*	*401.63*	*398.41*
	159.02	*154.11*	*151.08*	*147.4*

Tab.2 Comparison of SSE results for perceptron with different number of input and hidden neurons

	off-line		**on-line**	
input/hidden neurons	**sse1**	**sse2**	**sse1**	**sse2**
2 / 6	*5656.77*	*2381.84*	*5141.63*	*2098.76*
4 / 12	*3823.17*	*1486.96*	*2002.84*	*783.19*
6 / 15	*881.20*	*340.52*	*753.84*	*295.73*
8 / 24	*420.91*	*188.13*	*341.39*	*131.20*

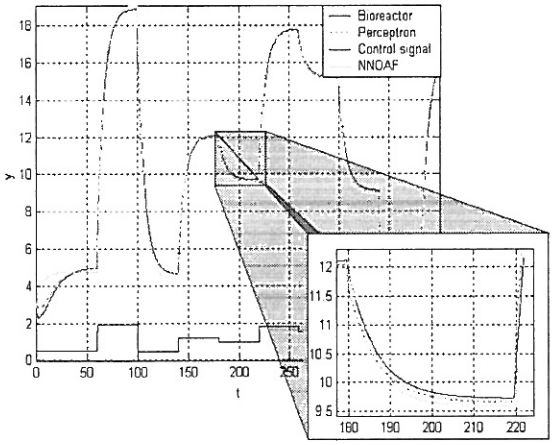

Fig.4 Time responses of the process and the neural model outputs with Hermit OAF and SAF (training data)

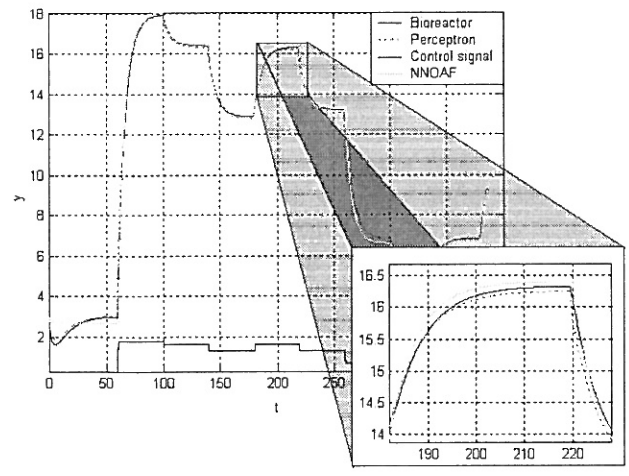

Fig.5 Time responses of the process and the neural model outputs with Hermit OAF and SAF (testing data)

B. Neural network control

For process control using the adaptive ANN model, the previous optimal structure has been used. Because the control quality depends on the prediction horizon and penalization factors the maximal prediction horizon (32) has been found. Simulation result is in Figure 6.

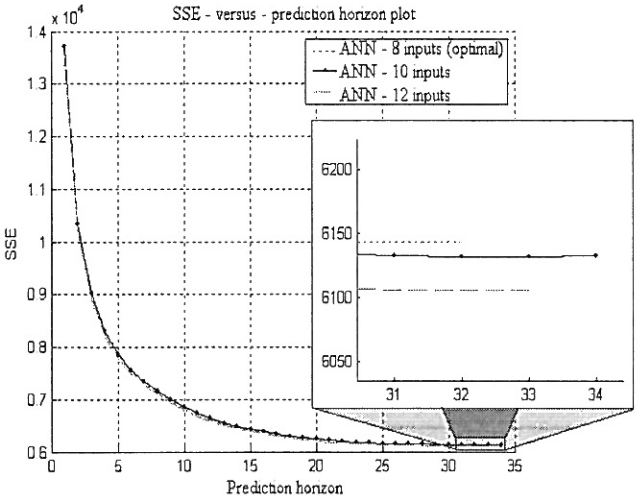

Fig.6 SSE - versus - prediction horizon plot

From de figure is clear that raising the number of inputs (other than optimal number) doesn't cause expressive control improvement.

The next step is choosing the penalization factors α and β. A genetic algorithm has been used to find these parameters for the prediction horizon 15. The population consists of 100 strings. Each string contains 2 genes:

1. α : <0.05, 2>
2. β : <0.05 2>

In the algorithm, the mutation, crossover and selection operations have been combined. After 1000 simulations (10 populations) the sum squared error (SSE – fitness function) settled with the penalization factors α_u=0.95 and β_r=2. The simulation results for different prediction horizons are in Fig. 7.

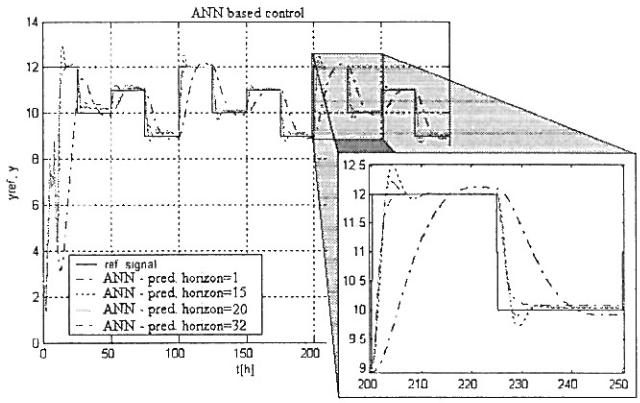

Fig.7 ANN based control using Chebychev OAF for prediction horizons 1,15,20 and 32

A rising prediction horizon speeds up the control process at the beginning (with an overshoot) and then slows it down. With a smaller prediction horizon the control is very slow. There are several possibilities how to improve the control, e.g. using an off-line trained ANN, leaving more time to adapting the ANN to the reference variable, using a moving prediction horizon, etc. In our case a moving prediction horizon has been applied.

To construct the optimal prediction horizon path a genetic algorithm has been used. The population consists of 150 strings. Each string contains 2 genes:

1. X – point
2. Y – point

The optimal time path consists from two sections defined by the points <0, max. prediction horizon>, <X,Y> and <7,1>. The point X represents the time elapsed after the reference change and Y represents the prediction horizon in time X. The obtained graphical results (for X=4 and Y=26) are in Figures 8 and 9.

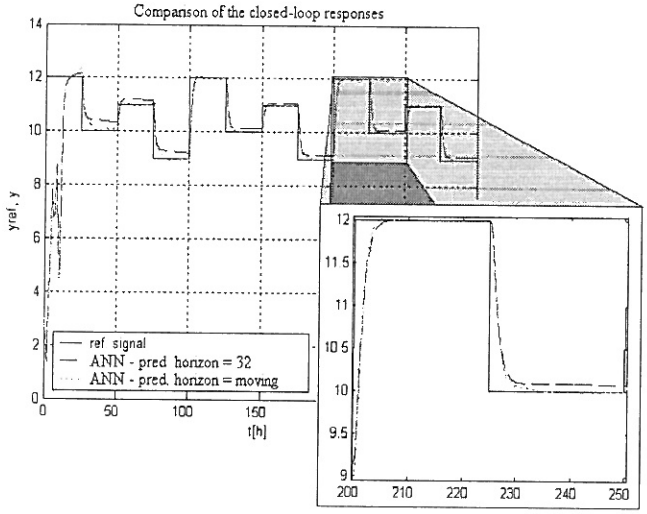

Fig.8 Comparison of the closed-loop responses using the ANN with Hermit OAF for the prediction horizon 32, and the moving prediction horizon

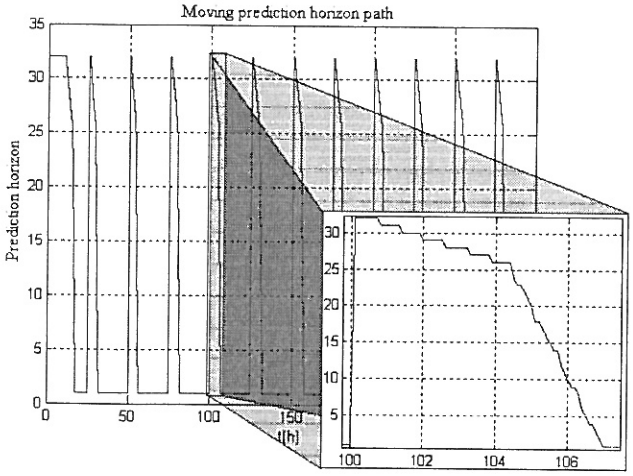

Fig.9 Specification of the time path of the moving prediction horizon

Using the moving prediction horizon (Fig.9) brought about performance improvement in terms of SSE and settling time (Fig.8). This control algorithm is less computationally demanding because of a smaller prediction horizon.

VI. CONCLUSION

The presented paper deals with improving the conventional ANN modeling and control methods using orthogonal activation functions (OAF). The proposed approach to modeling and control has been demonstrated for a real plant model of a bioreactor. Simulation results confirm the advantages of the designed solution in terms of precision, speed and quality. Theoretical, numerical and graphical results prove the effectiveness of this approach applied in modeling and control of highly nonlinear practical problems.

VII. ACKNOWLEDGMENT

This paper was partially supported by the Slovak Scientific Grant Agency VEGA, Grant No. 1/0115/03.

VIII. REFERENCES

[1] M. Oravec, J. Polec, S. Marchevský, *Neurónové siete pre číslicové spracovanie signálov.* Vydavateľstvo FABER, Bratislava, 1998, 196p.

[2] M. Šnorek, M. Jiřina,*Neuronové sítě a neuropočítače.* Vydavateľstvo ČVUT, Praha, 1995, 124p.

[3] Ch. Zhu, D. Shukla, F.W. Paul, Orthogonal Function for System Identification and Control, Control and Dynamic System, Edited by Cornelius T. Leondes , *Academic Press,* 1998, p.1-72

[4] L. Körösi, *Metódy identifikácie a riadenia nelineárnych procesov pomocou ortogonálnych a wavelet funkcií,* FEI STU Bratislava, KASR p. 1-70., 2002

[5] P. Linder , *Modelovanie a riadenie dynamických systémov pomocou neurónových sietí s ortogonálnymi aktivačnými funkciami.* FEI STU Bratislava, KASR, 2002, p. 1-55

[6] Š. Kozák, L. Körösi, Využitie umelých neurónových sietí v praxi a v priemysle, *Conf. SSKI,* Trebišov, 2002

[7] Š. Kozák, L. Körösi, A novel type of selt-tuning neural controller, *IFAC Conference CAO 2003,* Visegrád, Hungary, 2003

[8] G. P. Liu, Neural-learning control of nonlinear systems using variable neural networks, IFAC, *15th Triennial World Congress,* Barcelona, Spain, 2002

[9] Bundzel M., *Neurónové siete s adaptívnou topológiou,* Katedra kybernetiky a umelej inteligencie, Disertačná práca, Košice, 1999