# NEURAL CONTROLLER TUNING PROCEDURE USING GENETIC ALGORITHM

*S. Kajan, I. Sekaj*

Institute of Robotics and Cybernetics, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, Slovak Republic

**Abstract**

**Control of nonlinear systems is a challenging task. One of the ways of control of such systems is the use of neural networks as controllers. In this paper a methodology is proposed, where for neural controller design the genetic algorithm (GA) has been used. This method allows to find the optimal adjustment of the neural network weights so that high performance is obtained. The tuning procedure of the neural controller consists of four steps. The proposed control method is realized in Matlab/Simulink and demonstrated on control of a nonlinear dynamic system.**

## 1 Introduction

For control of some classes of nonlinear dynamical systems with advantage neural controllers (NC) are used. The neural network can be applied as a direct controller. It can emulate expert or another type of controller, it can be direct inverse controller, neuro-predictive controller or direct controller [2, 4, 8, 13]. This article deals with last named type which is optimized by genetic algorithms (GA) [1, 3, 6, 7, 9, 12] (Fig. 1).

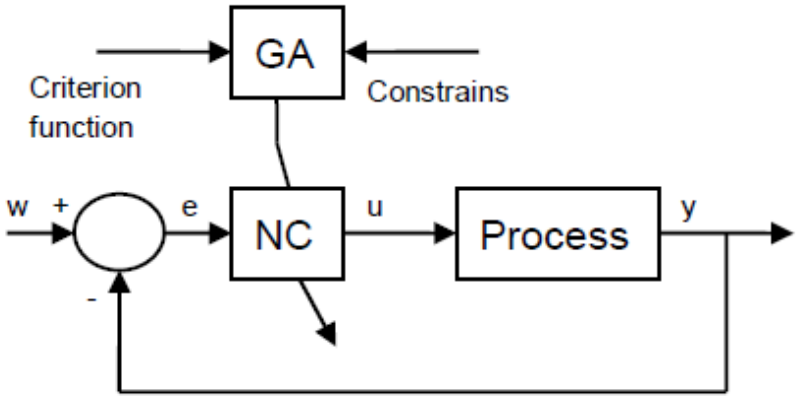


Figure 1: Block scheme of neural controller with tuning using genetic algorithm

## 2 Neural controller

Consider, the neural controller is represented by a multi-layer perceptron network (MLP) with a single hidden layer. This type of neural network is able to approximate any type of a continuous nonlinear function. The scheme of the proposed neural controller is shown in Fig. 2.

The inputs of the neural network are the control error *e(t)*, integral of control error , controlled output *y(t)* and the 1st derivative of the controlled output. If necessary, also higher order derivatives or even other input variables can be used. Output from the neural network is the control value *u(t)*. The neural network with such inputs and output represents a nonlinear positional PID controller, where its output is a non-linear function of its inputs. The output of the neural network is in form

$$u(t) = f\left(e(t), \int e(t)dt, \frac{dy(t)}{dt}, y(t)\right) \tag{1}$$

where $f$ is a nonlinear function.

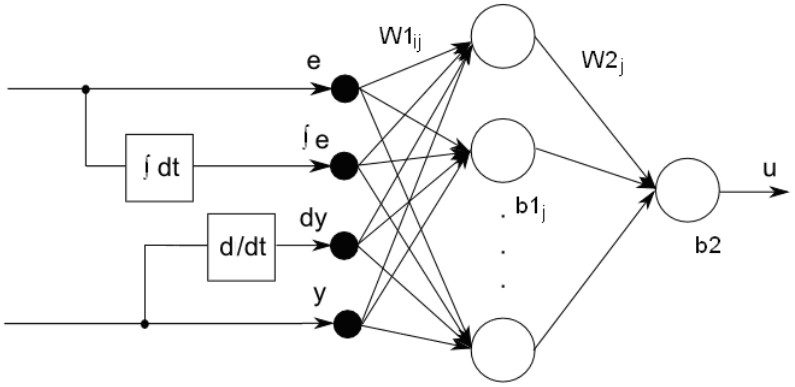


Figure 2: Scheme of the neural controller

In the hidden layer of the multilayer perceptron network (MLP) the hyperbolic tangent activation functions are used (tansig) in form

$$\varphi(a) = \frac{2}{1+e^{-2a}} - 1 \tag{2}$$

In the output layer linear activation function has been used. The optimized parameters are the weights between input and hidden layer $W1_{ij}$, weights between hidden and output layer $W2_j$, biases in the hidden layer $b1_j$ and bias in output layer $b2$ [2, 4]. For the initial setting of the neural controller parameters the Levenberg-Marquardt method [4, 5] is possible to use, where the data from a designed PID controller [6, 7], as training data can be used. Before network training, simulated data are normalized.

## 3 Genetic algorithm

A general scheme of the used GA can be described by following steps (Fig.3) [10, 11, 12]:

1. Initialization of the population of chromosomes (set of randomly generated chromosomes).

2. Evaluation of the cost function (fitness) for all chromosomes.

3. Selection of parent chromosomes.

4. Crossover and mutation of the parents → children.

5. Completion of the new population from the new children and selected members of the old population. Jump to the step 2.

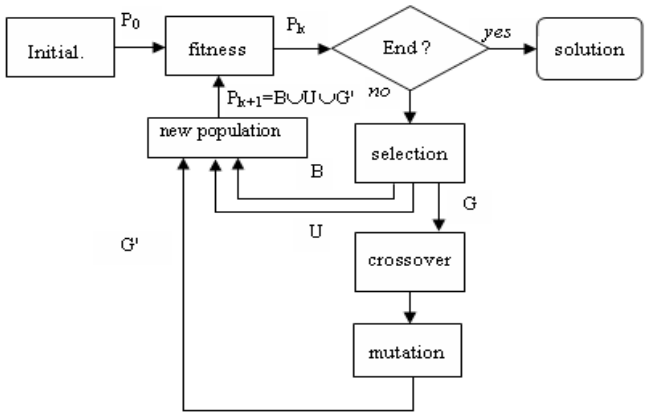


Figure 3: The Block scheme of genetic algorithm

The chromosome contains the set of neural network parameters - weights and biases. The optimized fitness function can use performance index in form (3), (4) or other [11, 12] where $N$

represents the number of simulation samples, $e$ is the control error, $y$ is the controlled output, $w$ is the reference value and $\alpha$ is weight constant.

$$J = \sum_{i=1}^{N} |e_i| \quad = \quad \sum_{i=1}^{N} |w_i - y_i| \tag{3}$$

$$J = \sum_{i=1}^{N} |e_i| + \alpha \sum_{i=1}^{N} \left| \frac{dy_i}{dt} \right| \tag{4}$$

After the initialization of the population, fitness of each chromosome of the population is evaluated. Fitness contains closed loop simulation with the model of the nonlinear system and the neural controller and the performance index evaluation. The design procedure is based on the genetic algorithm (Fig.4).

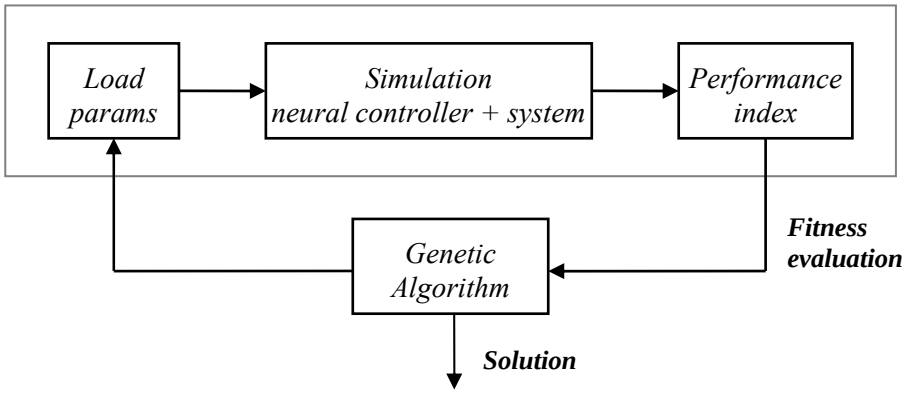


Figure 4: The Block scheme of the GA-based neural controller design

## 4 Experimental results

The nonlinear dynamic system described by differential equation (5) has been used

$$y'' + 0.7y' + 0.2y + 0.3y^3 - u = 0 \tag{5}$$

Both systems have nonlinear behavior and their dynamics changes according the operating point (position of output $y$). Time responses of the nonlinear system for various step sizes are depicted in Fig. 5.

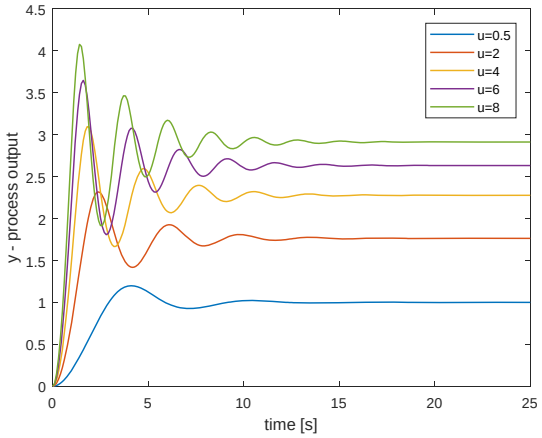


Figure 5: Step responses of nonlinear system

The tuning procedure of the NC consists of four steps. In the first step, optimal parameters of the PID controller are found using GA. Without loss of generality let us consider now a PID controller with feedforward structure, described by the equation (6), where $P$, $I$, $D$ are controller parameters [6, 8].

$$u(t) = P.e(t) + I.\int e(t)dt - D.\frac{dy(t)}{dt} \tag{6}$$

The searched PID controller parameters are $P \in R^+$, $\boldsymbol{I} \in R^+$, $D \in R^+$. The chromosome representation in this case can be in form ***ch***$=\{P, I, D\}$. For this case consider the cost function (4), where the parameters α is set to α =0.5 [8]. Evolution of fitness function for the PID controller design is shown in Fig. 6. The controller parameters found are P=16.7810, I=5.8651, D=8.6679.

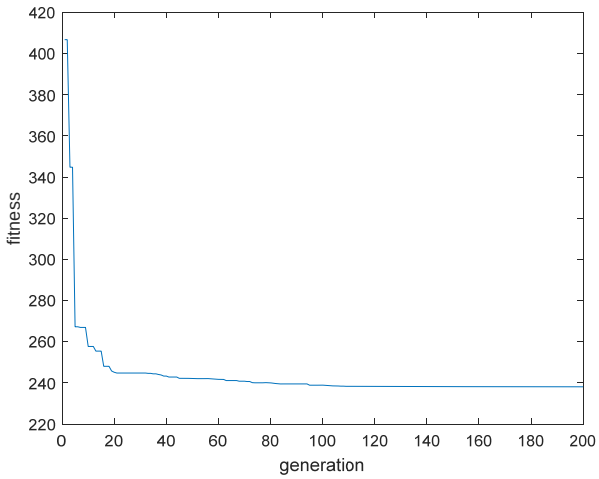


Figure 6: Evolution of fitness function for PID controller design

In the second step, for the PID controller input/output data are generated, which will be used for training of the neural network. The input and output data for training of the neural controller are shown in Fig. 7. Before network training, simulated data are normalized. The input data is multiplied by normative constants $K_{ni}$ =[0.5855 0.6299 0.3648 0.3237]. The controller output is multiplied by normative constant $K_{no}$ =21.8184.

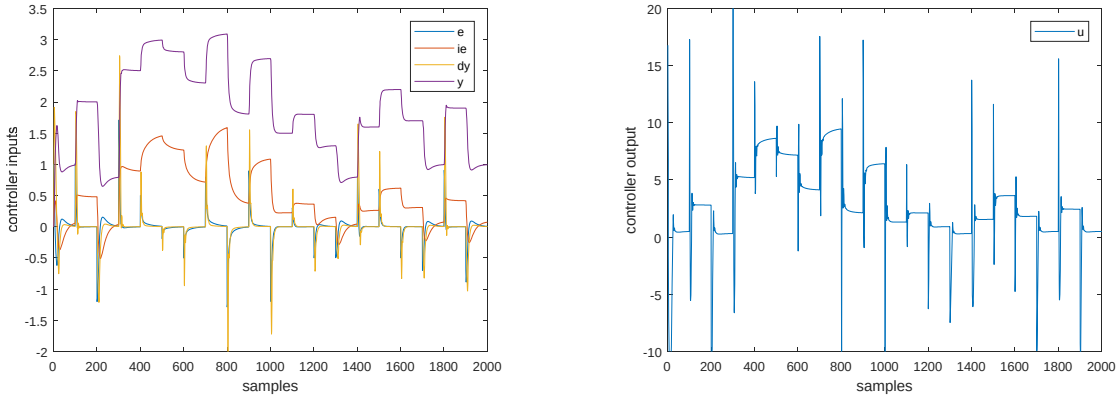


Figure 7: Time responses of input and output data of the PID controller for training of the neural controller

In the third step, the neural network is trained using the simulation data (Fig. 7). In case of the closed loop controller the neural controller with the MLP network with a single hidden layer is used [4]. The hidden layer contains 8 neurons. For the neural network training Levenberg-Marquardt method was used [4, 5] (Fig. 8). This NC weight setting is used as the default. Matrix of weights between input and hidden layer *W1*, matrix of weights between hidden and output layer *W2*, vector of biases in the hidden layer *b1* and bias in output layer *b2* were set as follows:

W1 =[0.2554, -0.1669, -1.0417, 3.8285; 1.6325, 0.2490, -0.7421, -3.5117; …

1.4691, 0.0101, -1.1729, 0.4020; 0.3057, 0.0997, -0.2534, -0.0001; …

0.7716, 0.5449, 0.9774, -3.3838; -0.8069, -0.2594, 0.6687, 0.0015; …

-1.4713, 1.6146, -1.6262, 0.3814; 1.2346, -2.0383, -1.7549, 0.4290]

W2 =[0.0003 0.0004 0.0161 3.0823 -0.0001 -0.3773 0.0000 -0.0002]

b1 =[0.0184; -0.4602; -2.1811; -0.2966; 1.4699; -0.9991; -2.2751; 3.1222]; b2 = 0.6174

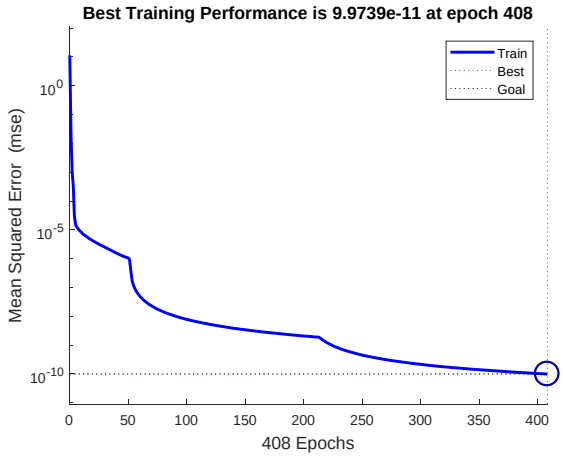


Figure 8: Training process of neural network (NC)

In the last step, the NC weights are optimized using GA. The same cost function (4) was chosen as for the PID controller design. Evolution of fitness function for NC controller design is shown in Fig. 9.

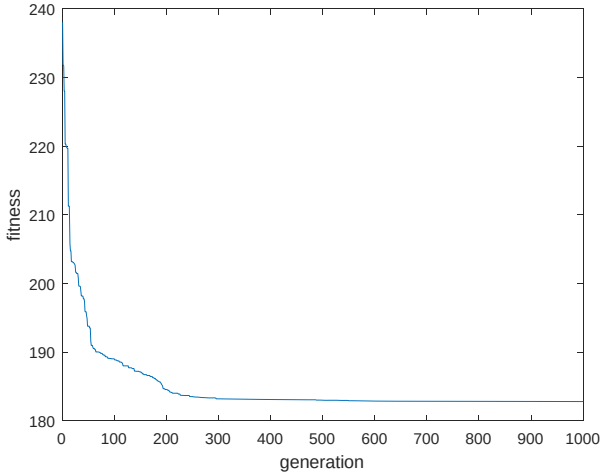


Figure 9: Evolution of fitness function for NC controller design

In Table 1 the performance indices are computed in case of training and testing experiments. Sum of absolute control error (SAE) according equation (3), average values of overshoot and settling time are compared. Time responses of the controlled variable and control variable for nonlinear system are depicted in Figure 10. Both experiments are for test data, which are different than training data used for the controller design. Finally in Figures 11 and 12 time responses of NC are compared with linear PID controllers, which are also optimized by genetic algorithm. The conventional PID controller is not able to reach perfect control performance because of nonlinear character of the controlled system.

Table 1: PERFORMANCE INDICES OF PID AND NC CONTROLLERS

| *Controller* | *Train Data* | | | *Test Data* | | |
|---|---|---|---|---|---|---|
| | *SAE* | *Overshoot [%]* | *Settling time [s]* | *SAE* | *Overshoot [%]* | *Settling time [s]* |
| PID | 143.85 | 7.88 | 3.61 | 139.37 | 7.75 | 3.37 |
| NC | 100.61 | 0.95 | 1.35 | 97.74 | 1.80 | 1.53 |

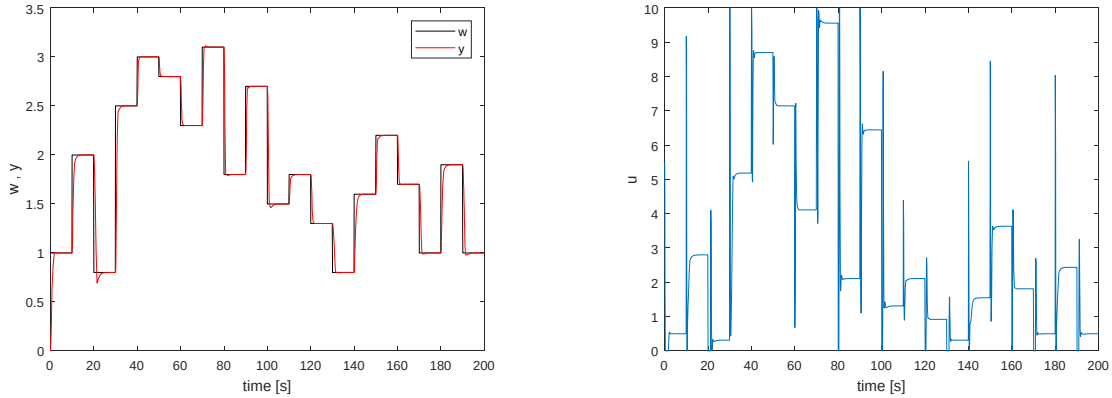


Figure 10: Time responses of the controlled value y and control value u for training data

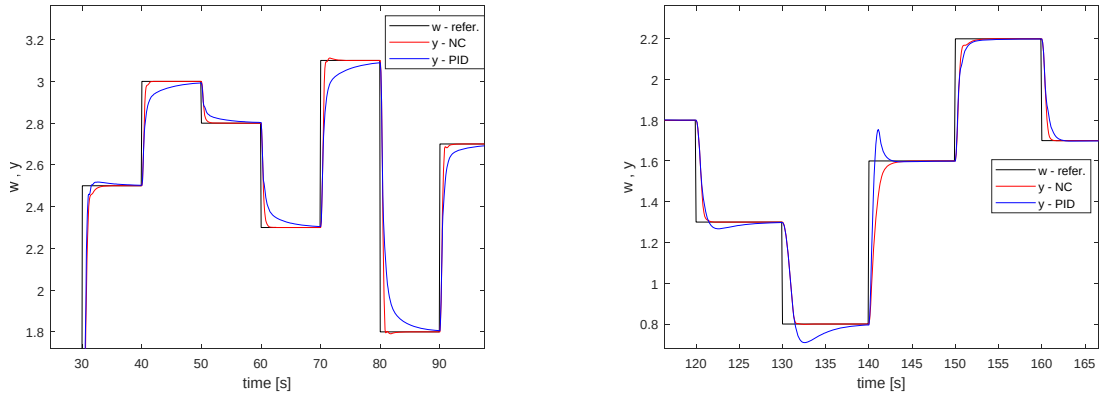


Figure 11: Time responses of the controlled variable y under neural (NC) and PID controllers for training data

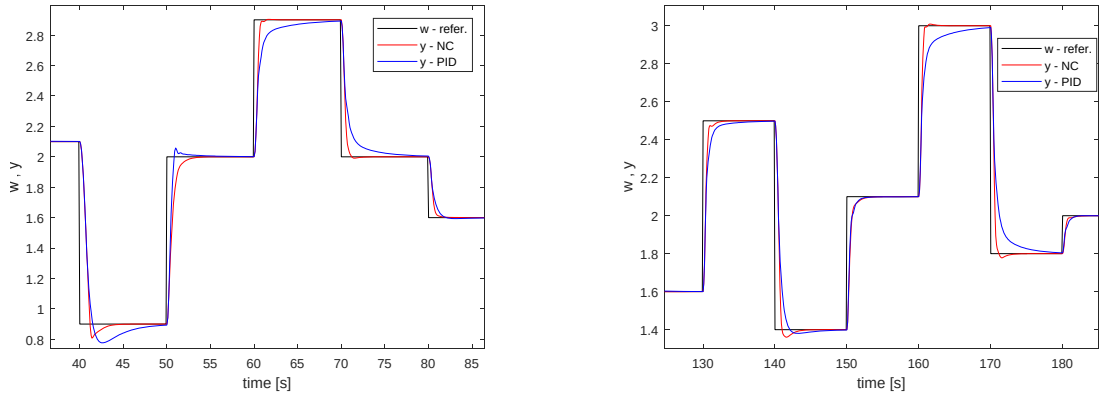


Figure 12: Time responses of the controlled variable y under neural (NC) and PID controllers for testing data

## 5 Conclusion

Genetic algorithm is an efficient means for neural controller design. Neural controllers are able to provide high performance in control of nonlinear systems. But the neural network parameterization is a complex and time consuming optimization problem because of large number of designed parameters in comparison to simple controllers as PID or similar ones. On the other hand the neural network based controllers can provide nonlinear behavior and they can be used for control of (highly) nonlinear systems, systems with limitations, saturations, MIMO-systems etc.

## Acknowledgement

The work has been supported by the grant agency VEGA no. 1/0475/16. This support is very gratefully acknowledged.

## References

[1] Z. Dideková, S. Kajan. *Neural Control of Non-Linear Processes Designed by Genetic Algorithms*, In: ELITECH `09 : 11th Conference of Doctoral Students. Bratislava, STU in Bratislava FEI, 2009
[2] A. Jadlovská. *Modelling and control of dynamic system using neural networks* (Edition of science documents, FEI TU Košice, 2003, in slovak).
[3] I. Sekaj. *Genetic Algorithm Based Controller Design*, In: 2nd IFAC conference Control System Design'03, Bratislava, 2003.
[4] M. Beale, M. Hagan, H .Demuth. *Neural Network Toolbox, User's Guide*, 2017
[5] M. T. Hagan, M. B. Menhaj. *Training Feedforward Networks with the Marquardt Algorithm*, Submitted to the IEEE Proceedings on Neural Network, 1994
[6] Z. Dideková, S. Kajan. *Adaptive PID Controller Design Base on Genetic Algorithms and Neural Networks.* In: ELITECH´11: 13th Conference of Doctoral Students Faculty of Electrical Engineering and Information Technology. Bratislava, STU in Bratislava FEI, 2011. pp. 1-6.
[7] S. Kajan, M. Hypiusová. *Robust Controller Design Using Genetic Algorithm*, In: ATP Journal plus. no. 1 : Systémy automatického riadenia, 2011 , pp. 18-21.
[8] S. Kajan. *Comparison of Some Neural Control Structures for Nonlinear Systems.* In Journal of Cybernetics and Informatics. Vol. 8, 2009
[9] I. Sekaj , S. Kajan, Z. Dideková. *Neural Controller for Nonlinear System Designed by Genetic Algorithm.* In MENDEL 2012 : 18th International Conference on Soft Computing. June 27-29, 2012, Brno, Czech Republic. Brno : University of Technology, 2012, p. 268-274.
[10] D. E. Goldberg, *Genetic Algorithms in Search, Optimisation and Machine Learning* (Addisson-Wesley, 1989).
[11] I. Sekaj. *Evolučné výpočty a ich využitie v praxi* (Iris, Bratislava, 2005, in slovak).
[12] I. Sekaj, *Evolutionary Based Controller Design*, In: Evolutionary Computation, Book edited by: Wellington Pinheiro dos Santos, pp. 239-260, October 2009, In-Tech, Vienna, Austria. 2009.
[13] R. C. Dorf. *Modern Control Systems* (Addison-Wesley publishing Company, 5th edition, 1990).

---

Ing. Slavomír Kajan, PhD.:
Institute of Robotics and Cybernetics, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, Slovak Republic, Ilkovičova 3, 812 19 Bratislava, E-mail: slavomir.kajan@stuba.sk

Assoc Prof. Ing. Ivan Sekaj, PhD:
Institute of Robotics and Cybernetics, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, Slovak Republic, Ilkovičova 3, 812 19 Bratislava, E-mail: ivan.sekaj@stuba.sk