APPLICATIONS OF INTELLIGENT HYBRID SYSTEMS IN MATLAB

Z. Dideková, S. Kajan

Institute of Control and Industrial Informatics, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava, Slovak Republic

Abstract

This paper deals problem of intelligent hybrid systems. Intelligent systems include neural networks (NN), fuzzy systems (FS) and genetic algorithms (GA). Each of these intelligent systems has certain properties (ability of learning, modelling, classifying, obtaining empirical rules, solving optimizing tasks …) fitting specific kind of applications. Combination of these intelligent systems creates neuro-fuzzy system, fuzzy-GA system, neuro-GA system and these systems together are called hybrid intelligent systems (HIS). For these purposes, there was created program in Matlab, where were made several demo applications for several HIS in the field of system modelling and control.

1 Principles of Hybrid intelligent systems

Several modern applications are realised by intelligent technologies as are neural networks (NN), fuzzy systems (FS) and genetic algorithms (GA). Each of these intelligent systems has certain properties (ability of learning, modelling, classifying, obtaining empirical rules, solving optimizing tasks …) fitting specific kind of applications. Table 1 presents a comparison of different intelligent system [1].

Table 1: COMPARISON OF FUZZY SYSTEMS (FS), NEURAL NETWORKS (NN) AND GENETIC ALGORITHMS (GA)

	FS	NN	GA
Knowledge representation	good	bad	rather bad
Uncertainty tolerance	good	good	good
Imprecision tolerance	good	good	good
Adaptability	rather bad	good	good
Learning ability	bad	good	good
Explanation ability	good	bad	rather bad
Knowledge discovery and data mining	rather bad	good	rather good
Maintain ability	rather good	good	rather good

Noticed, that in many real applications we would need not only to acquire knowledge from various sources, but also to combine different intelligent technologies. The need for such a combination has led to the emergence of hybrid intelligent systems (HIS). A hybrid intelligent system is one that combines at least two intelligent systems, i.e. combination of these intelligent systems creates neuro-fuzzy system, fuzzy-GA system, neuro-GA system [1]. The block scheme of HIS is depictured in Fig. 1.

For example, HIS can represent following forms. Neuro-fuzzy system is realized as a neural network, in which fuzzy system parameters are encoded in several layers. Using network learning ability the parameters can be adapted, hence the system is called adaptive neural fuzzy inference system (ANFIS). Fuzzy-GA system provides fuzzy system parameters optimization using GA. Neuro-GA system provides neural network parameters optimization using GA. For example, in Matlab were made several demo applications for several HIS in the field of system modelling and control.

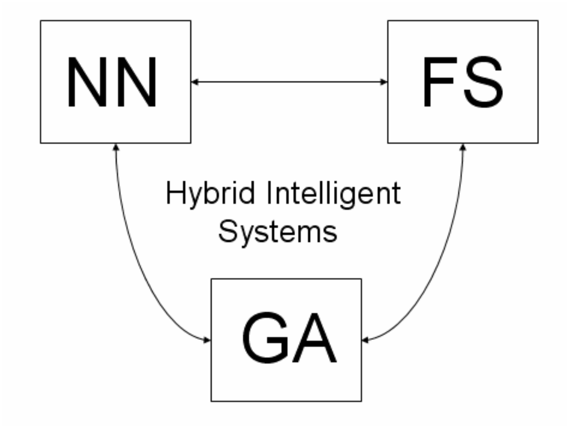


Figure 1: The block scheme of Hybrid Intelligent Systems

2 Genetic algorithms

Genetic algorithm (GA) is a powerful stochastic-based search/optimization approach, which mimics the evolution in the nature. It is described in e.g. [2, 3, 4, 5] and others. A general scheme of a GA can be described by following steps (Fig. 2):

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

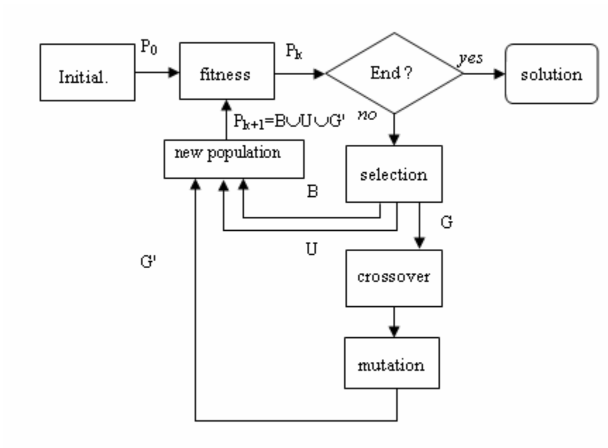


Figure 2: Block scheme of genetic algorithm

Genetic algorithms fall into the optimization techniques, which are able to find global optimum of the function. They can be used to modeling or control linear or non-linear systems using fuzzy logic or neural networks.

When the neural networks are used, the wanted parameters in chromosomes can be connections in the neural network, values of weights and biases or the both.

In the case of fuzzy logic, the wanted parameters are parameters of membership functions, base of rules or the both.

In modeling of a system, the optimized function is the cost function:

$$J = \sum_{i=1}^{N} |e_i| \;=\; \sum_{i=1}^{N} |y_i - y_{mi}|, \tag{1}$$

where y is output from the system, y_m is output from the model of system, e is model error and N is number of patterns.

In control of a system, the optimized function is the cost function:

$$J = \sum_{i=1}^{N} |e_i| \;=\; \sum_{i=1}^{N} |r_i - y_i|, \tag{2}$$

where r is reference variable, y is controlled output, e is control error and N is number of patterns.

In the both of cases, the minimum of fitness is searching. Fitness is represented by the cost function or in the case of control, by the modified cost function, which can be penalized for example by derivation of process output y, or by measure or derivation of control action u.

3 Modelling of nonlinear function using ANFIS

Neuro-fuzzy system represents connection of numerical data and linguistic representation of knowledge. The system is characterized by transparency as fuzzy systems and learning ability as neural networks. The structure of a neuro-fuzzy system is similar to a multi-layer neural network. In general, neuro-fuzzy system has input and output layers, and three hidden layers that represent membership functions and fuzzy rules. Encoded fuzzy system in several layers of neural network can be in form Mamdani or Sugeno fuzzy interface model. Using network learning ability, the parameters can be adapted, hence the system is called adaptive neural fuzzy inference system (ANFIS) [1, 6]. ANFIS represents Sugeno fuzzy model, which fuzzy rules can be expressed in the following form:

IF x_1 is A_1 AND x_2 is A_2 …. AND x_m is A_m THEN $y=f(x_1,x_2,\dots,x_m)$

where x_1, x_2, …, x_m are input variables, A_1, A_2, …, A_m are fuzzy sets and y is either a constant or a linear function of the input variables. ANFIS is represented by a six-layer feedforward neural network, which architecture is displayed in Fig. 3 [1].

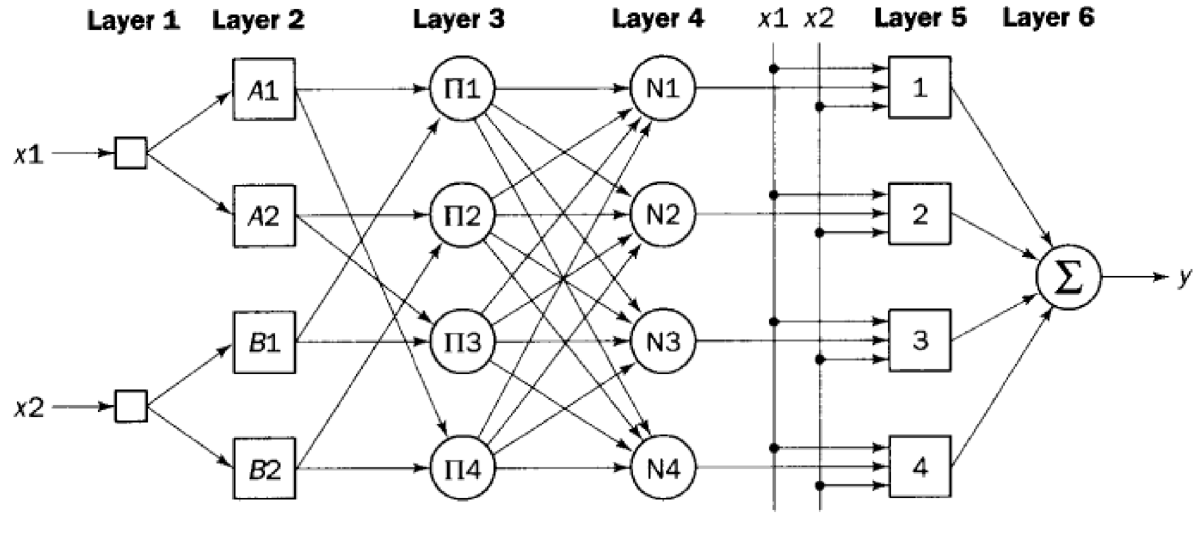


Figure 3: The architecture of ANFIS

ANFIS is trained using I/O data and backpropagation algorithm. By train algorithm are optimized fuzzification neurons parameters, mask of inference system and defuzzification neurons parameters [1]. For example, ANFIS is used for modelling of nonlinear 3D function with name Peaks in following form:

$$z = 3(1-x)^2 e^{\left(-x^2-(y+1)^2-10\left(\frac{x}{5}-x^3-y^5\right)e^{(-x^2-y^2)}-\frac{1}{3}e^{\left(-(x+1)^2-y^2\right)}\right)} \tag{3}$$

Peaks function neuro-fuzzy model has been created in Matlab. There has been made training and testing data set, both in the range $x, y \in \langle -3; 3 \rangle$ and interval between each two consecutive input samples on the both axes was for training data 0.5 and for testing data 0.25 (Fig. 4).

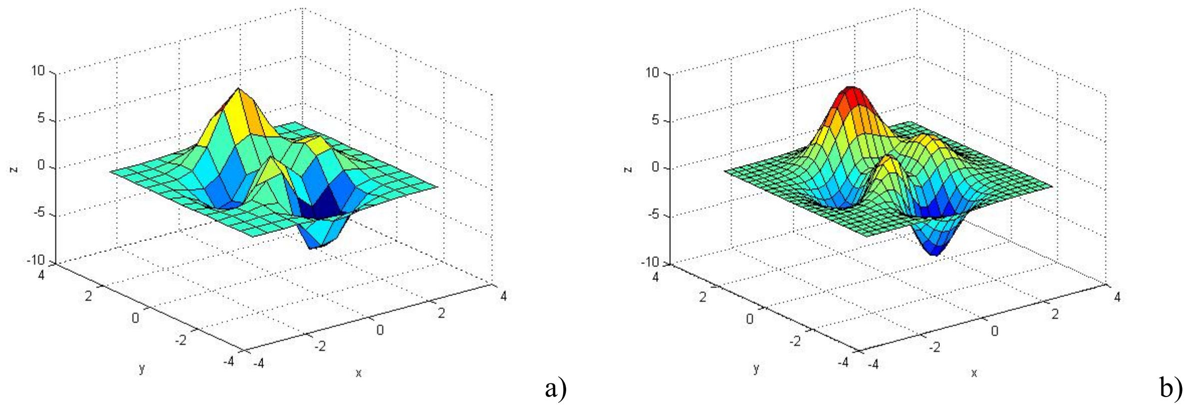


Figure 4: Training data a) and Testing data b)

ANFIS structure has been created using order *genfis1* in Matlab [6]. The order designs initial Sugeno type fuzzy inference system using a grid partition. There has been chosen 7 gbell membership functions for both of the inputs *x* and *y*.

```
    data = [x y z];
    in_fis = genfis1(data,[7 7],char('gbellmf','gbellmf'));
```

There has been used matlab order *anfis* for neuro-fuzzy system training, which uses hybrid learning algorithm to identify the membership function parameters of single-output, Sugeno type fuzzy inference systems (FIS). A combination of least-squares and backpropagation gradient descent methods are used for training FIS membership function parameters to model a given set of input/output data [6]. There has been employed 200 training epochs for training.

```
    epoch_n = 200;
    out_fis = anfis(data,in_fis,epoch_n);
```

Trained neuro-fuzzy system (ANFIS) next has been tested on training and testing data. There has been used order *evalfis* to enumerate output from the system [6].

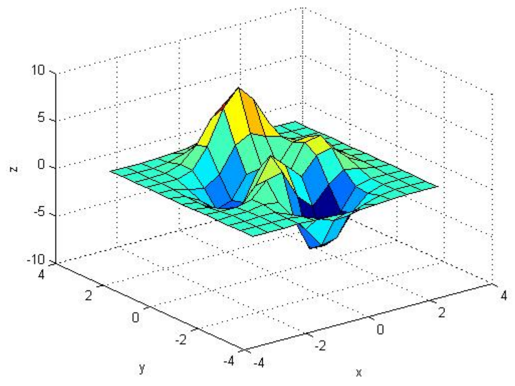
```
    outfissim = evalfis([x y],out_fis);
```

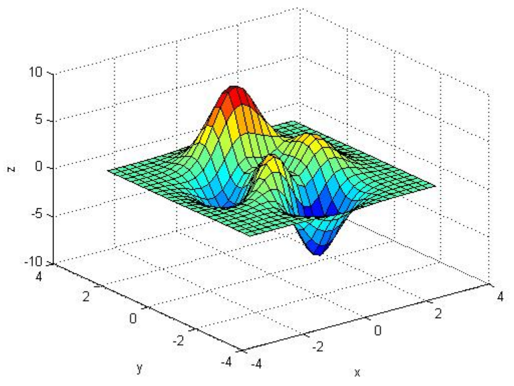
There has been calculated sum of least-squares errors of outputs from modelled system and Peaks function for training and testing data.

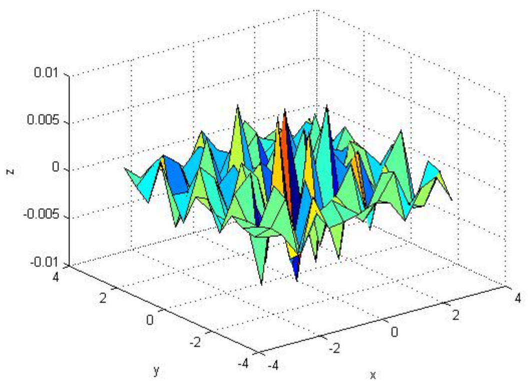
$$e = \sum_{i=1}^{N} (z_i - z\_sim_i)^2 \ , \tag{4}$$

Where *z* is output from Peaks function, *z_sim* is output from modelled neuro-fuzzy system and *N* is number of samples.

There has been attained these error values: training error *e_tren* = 0.0015 and testing error *e_test* = 11.1034. On Fig. 5, there are shown output from ANFIS for training a) and testing b) and errors of training c) and testing d).


a)


b)


c)

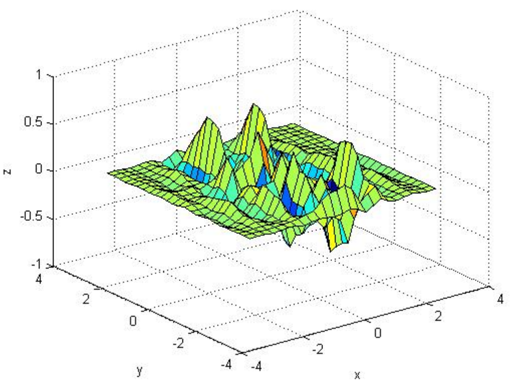

d)

Figure 5: Output from ANFIS for training a) and testing b), Errors of training c) and testing d)

4 Fuzzy PI controller with membership functions optimization using GA

In combination of fuzzy system and genetic algorithm, GA is used in the design of fuzzy system parameters, particularly for generating fuzzy rules and adjusting membership functions of fuzzy sets. For example of neuro-GA system application is fuzzy PI controller with membership functions optimization using GA.

Simulation scheme of the controlled process is on Fig. 6 and its step response is on Fig. 7.

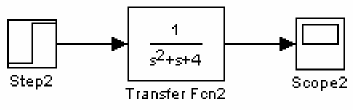


Figure 6: Scheme of the controlled process

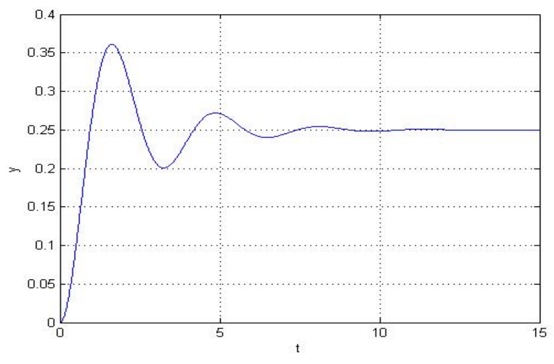


Figure 7: Step response of the process

Simulation scheme of the controlled process with fuzzy PI controller is on Fig. 8 and its step response (fuzzy PI controller has uniform setting of membership functions) is on Fig. 9.

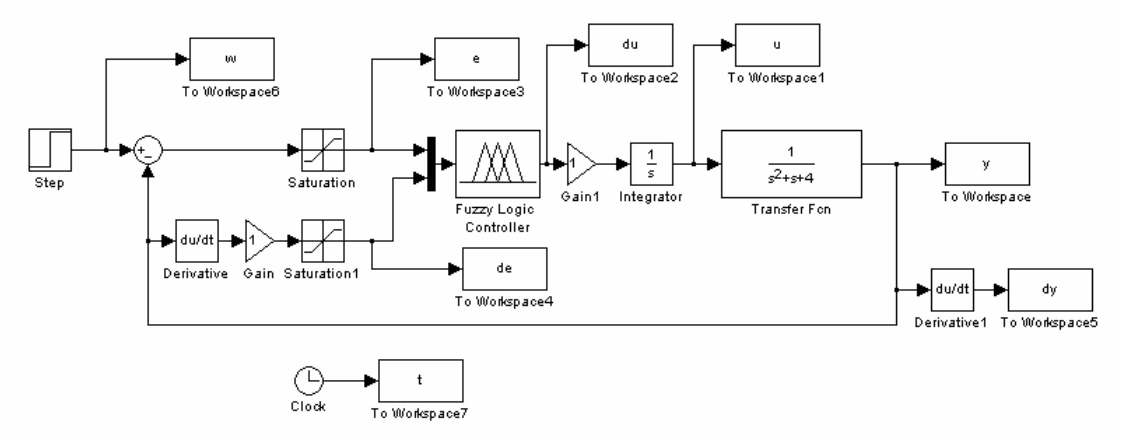


Figure 8: Scheme of the controlled process with fuzzy PI controller

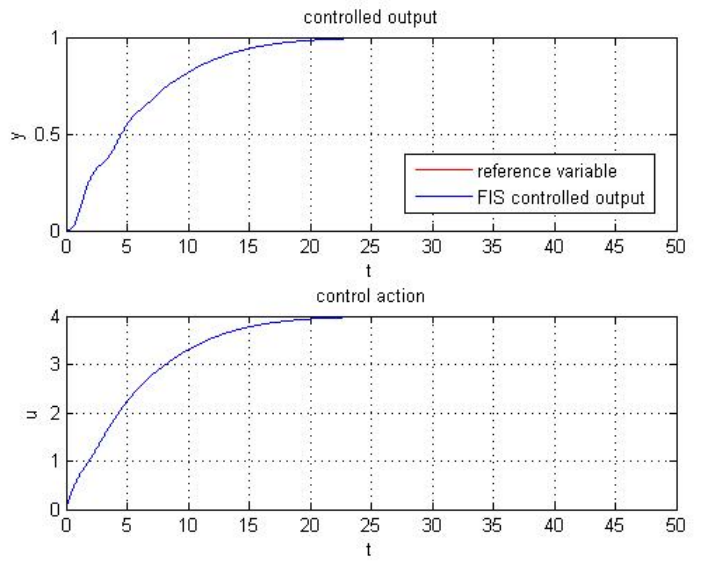


Figure 9: Step response of the controlled process with initial fuzzy PI controller

Inputs to the fuzzy PI controller are control error *e* and derivation of control error *de*. Output from the controller is derivation of control action *du*. Each input and output has 5 membership functions. Optimized is position of 2 membership function tops and spread of 3 membership functions. Each variable is symmetric. The shape of membership function is triangle. There are 2 unknown parameters for each variable. First and fifth membership functions are fixed, second membership function start, where is the top of first membership function and fourth membership function end, where is the top of fifth membership function. Top of the third membership function is in the middle of the range of the variable and there is also end of second and start of fourth membership function. Unknown are top of second and start of third membership function and end of third and top of fourth membership functions are symmetric. Together, there are 6 unknown parameters.

One of initial adjustments has uniform setting of membership functions (Fig. 10) and base of rules is fixed (Table 2). The sum of absolute control errors (SAE) with fuzzy PI controller with uniform setting of membership functions was SAE = 589.6102. Size of population was set to 30 and number of generations to 200.

On Fig. 11, there is shown the best fitness of population in dependence on generation and on Fig. 12, there is depicted the best adjustment. The step response is shown on Fig. 13. The sum of absolute control errors with optimized fuzzy PI controller was SAE = 346.7841.

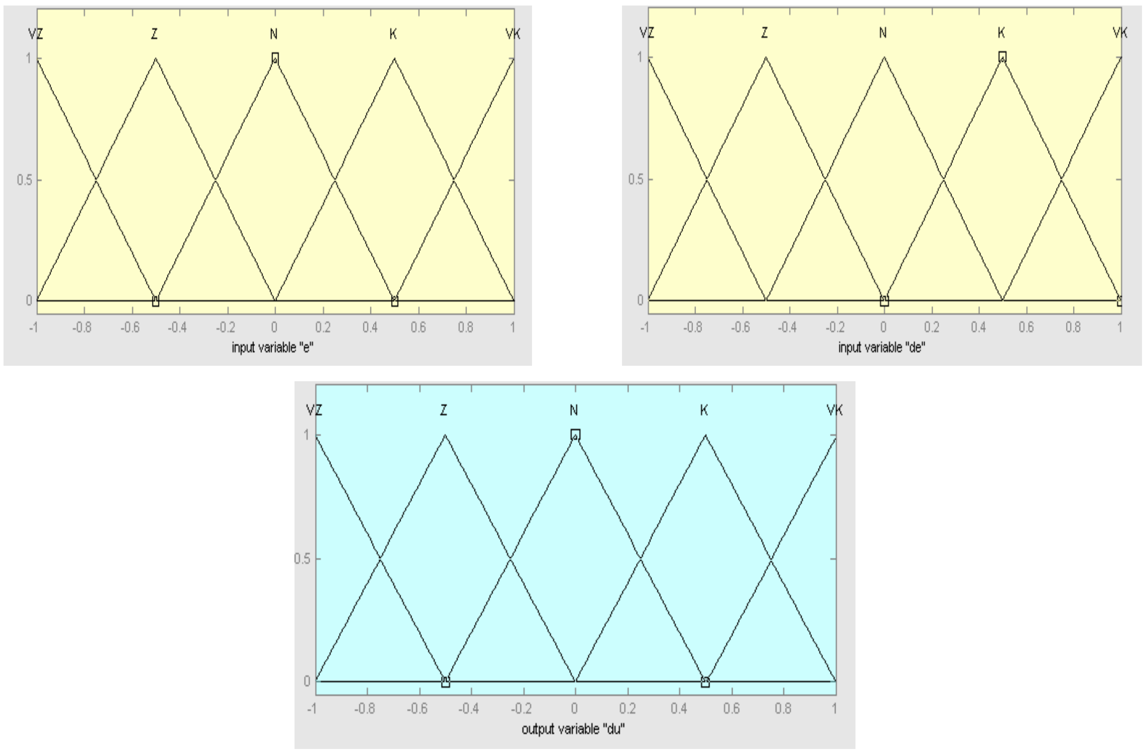


Figure 10: Initial adjustment of the inputs *e, de* and the output *du*

Table 2: BASE OF RULES

de\e	VZ	Z	N	K	VK
VZ	VZ	VZ	VZ	Z	N
Z	VZ	VZ	Z	N	K
N	VZ	Z	N	K	VK
K	Z	N	K	VK	VK
VK	N	K	VK	VK	VK

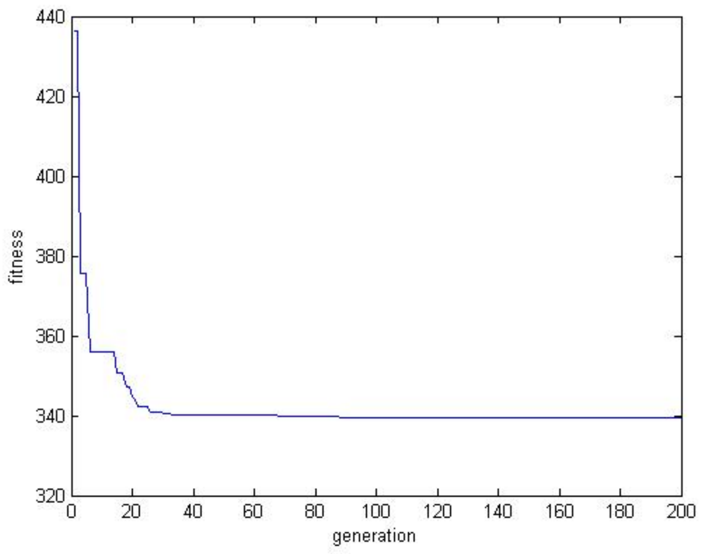


Figure 11: Best fitness of population in dependence on generation

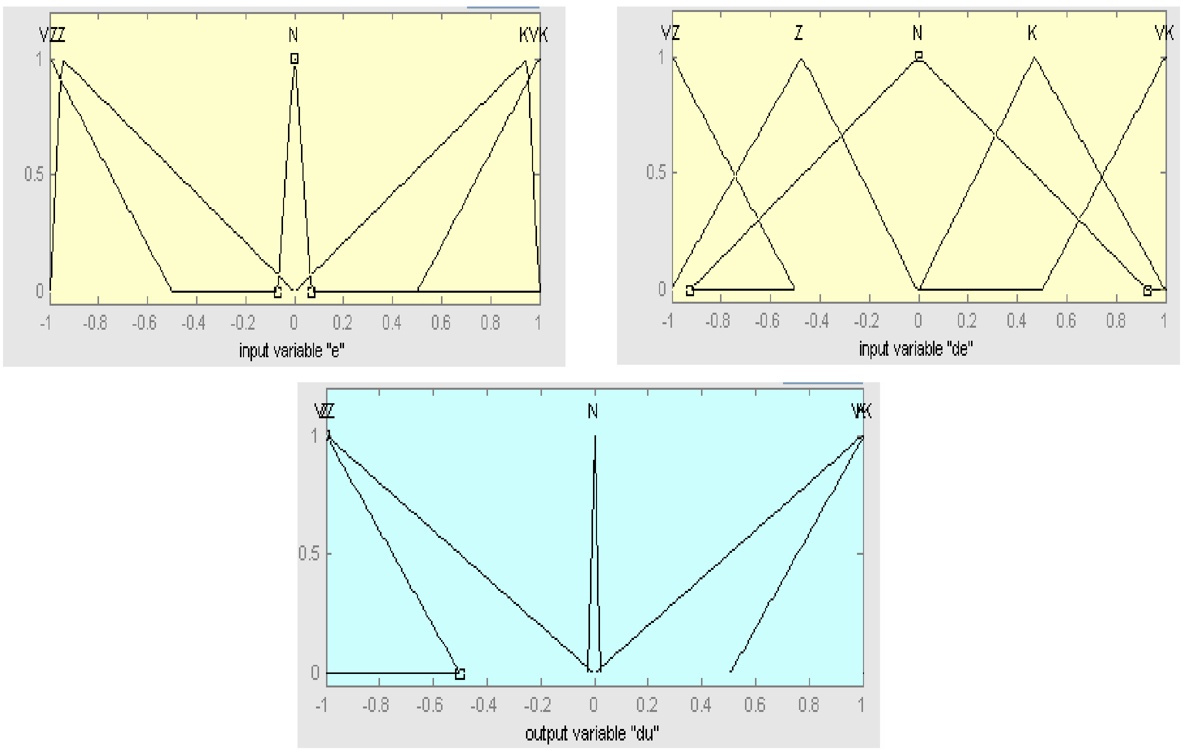


Figure 12: Best adjustment of the inputs *e, de* and the output *du*

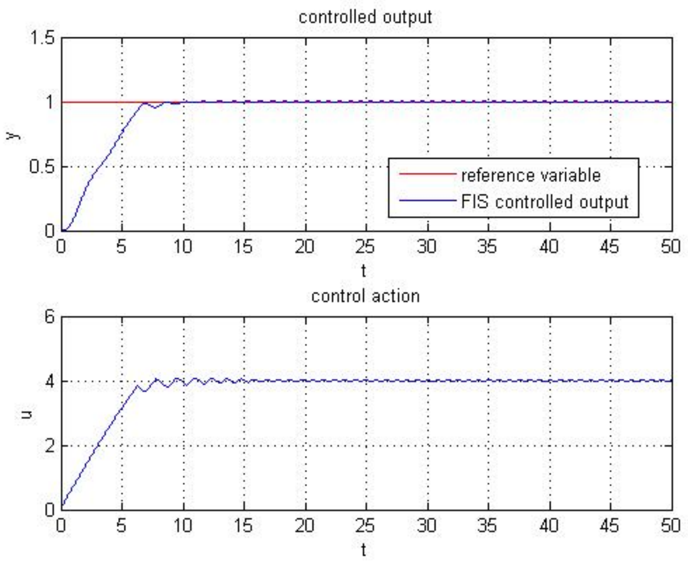


Figure 13: Step response of the controlled process with optimized fuzzy PI controller

5 Neuro-predictive controller design based on genetic algorithms

The block scheme of the used controller structure is depicted in Fig. 14. Without loss of generality, in case of the controlled process model a MLP (multilayer perceptron) artificial neural network is considered. The number of layers is 3 and the number of neurons in the input and hidden layer and their interconnections are the objects of the design/optimisation process. As the learning rule the off-line version of the Error-back-propagation method with Levenberg–Marquardt modification is considered [9]. The model predictive control method is based on the receding horizon technique [7]. The neural network model predicts the plant response over a specified time horizon. The predictions are used by a numerical optimization routine to determine the control value *u(k)* in each control period *k* that minimizes the following performance criterion *J* over the specified horizon.

$$\Delta u(k) = ?; J \to \min \tag{5}$$

$$J = \sum_{i=N_1}^{N_2} [r(k+i) - \hat{y}(k+i)]^2 + \rho . \sum_{i=1}^{N_u} [\Delta u(k+i-1)]^2 \tag{6}$$

$$u(k) = u(k-1) + \Delta u(k) \tag{7}$$

Here the r is the reference signal, $\hat{y}$ is the controlled value prediction, Δu is the control value change, k is the control step, N_1 is the lower and N_2 the upper output prediction horizon, N_u is control horizon and ρ is a weight constant. The control performance depends on the values N_1, N_2, N_u and ρ. The block scheme of the neural model is in Fig. 15. The number of inputs y_{k-1}, y_{k-2}, …, y_{k-n}, u_k, u_{k-1}, …, u_{k-m} (number of past samples of y and u) and the number of neurons in the hidden layer are normally set by the designer using experience. But often the chosen model architecture is not optimal because of the modelling accuracy and on the other hand from point of view computation time.

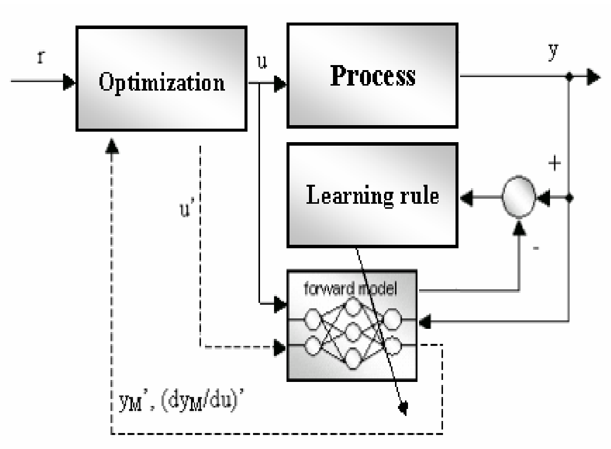


Figure 14: Block scheme of neuro-predictive controller

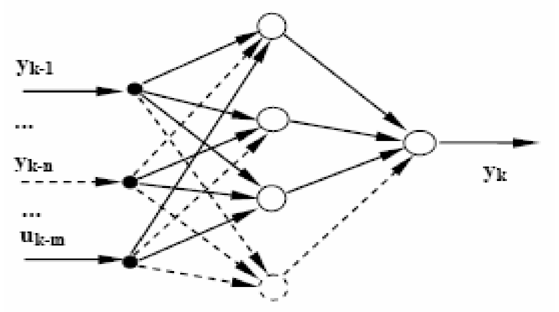


Figure 15: Neural network structure for process modelling

GA's are used for search for the optimal neural model architecture and for the predictive controller parameter optimisation.

Let consider the non-linear system, which is described by the differential equation:

$$y'' + 0{,}7y' + 0{,}2y + 0{,}3y^3 - u = 0 \tag{8}$$

The block scheme of the non-linear system Eq. (8) is depicted in Fig. 16.

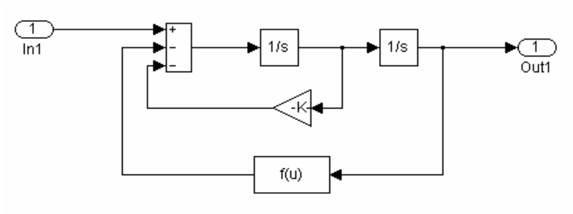


Figure 16: Simulation block scheme of non-linear system

5.1 Optimisation of neural model structure

For modelling of the non-linear system neural model structure with 6 inputs and 10 hidden neurons (Fig. 17) is used. In the case of the neural model, the number of neurons in input and hidden layer, and their interconnections have been optimised [8]. The interconnection map of the net is coded into the chromosome of the GA (Table 3). The length of the chromosome is 70 genes (Table 4). The genes can be either values 0 or 1, where 1 represents connection between neurons and 0 no connection.

TABLE 3: THE INTERCONNECTION MAP OF THE NEURAL NETWORK BETWEEN NEURONS

	Input Neurons					Hidden Neurons					ON
	1	2	.	5	6	7	8	.	15	16	17
1	0	0	.	0	0	0	0	.	0	0	0
2	0	0	.	0	0	0	0	.	0	0	0
.	.	.	.	.	.	.	.	.	.	.	.
5	0	0	.	0	0	0	0	.	0	0	0
6	0	0	.	0	0	0	0	.	0	0	0
7	1	1	.	1	1	0	0	.	0	0	0
8	1	1	.	1	1	0	0	.	0	0	0
.	.	.	.	.	.	.	.	.	.	.	.
15	1	1	.	1	1	0	0	.	0	0	0
16	1	1	.	1	1	0	0	.	0	0	0
17	0	0	.	0	0	1	1	.	1	1	0

TABLE 4: THE CHROMOSOME REPRESENTATION

1-10		11-20		·	41-50		51-60		61-70	
1	1	1	1	.	1	1	1	1	1	1

Over the chromosomes in the population, genetic operations crossover and mutation are applied. After these operations, unrealizable neural network structures can appear. Hence, in case of input neuron with no connection and hidden neuron with no connection to input or to output, such neurons are omitted from the net structure.

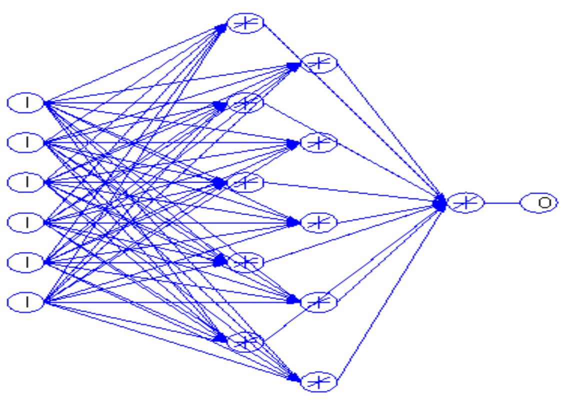


Figure 17: Neural model structure with full connections

For searching optimal neural model structure with GA, the used cost function (fitness) was considered in the form

$$F = MSE + (1-\alpha)*(w/wn)*10^{-5} + \alpha*(n/nn)*10^{-5} \tag{9}$$

MSE – mean square error of the neural model (in comparison with the modelled object)

α – weight constant, which was set to 0.7

w – number of weighted interconnections between the input and hidden layer

w_n – maximal number of all interconnections

n – number of neurons in network

n_n – maximal number of neurons in network

10^{-5} – weight constant

The optimised neural network structure is shown in Fig. 18. Using testing data the obtained error was MSE=0.4216e^{-6}.

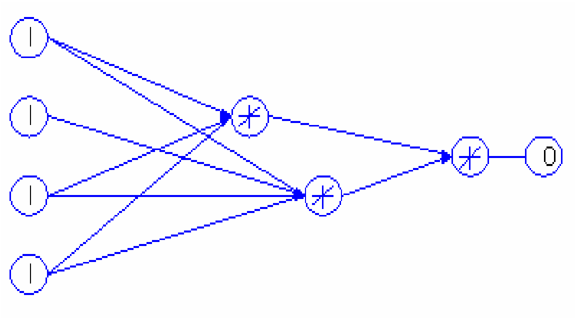


Figure 18: Neural model with optimal structure

5.2 Optimisation of neural controller parameters

The second design step was the search for the predictive controller cost function coefficients N_1, N_2, N_u and ρ. For this purpose another GA has been used, where the fitness consists of the closed-loop simulation and the performance index evaluation Eq. (10). An example of a simple performance index is as follows:

$$I_{AE} = \int_0^T |e(t)| dt \qquad (10)$$

where e is the control error. This performance index has been minimised, and it represents the controller performance. The obtained results after these two design steps, which were performed off-line are demonstrated in Fig. 19. It is the time-response of the controlled system (green) after the reference signal steps (blue). The control value time-response is in Fig. 20. In Fig. 21 the trend of the fitness function is depicted, which is the graph of the current best individual of the actual population vs. generation number.

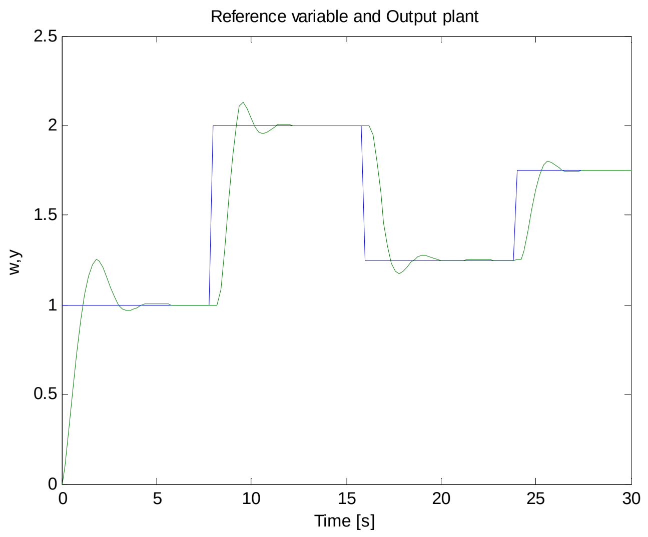


Figure 19: Time-response of the controlled system

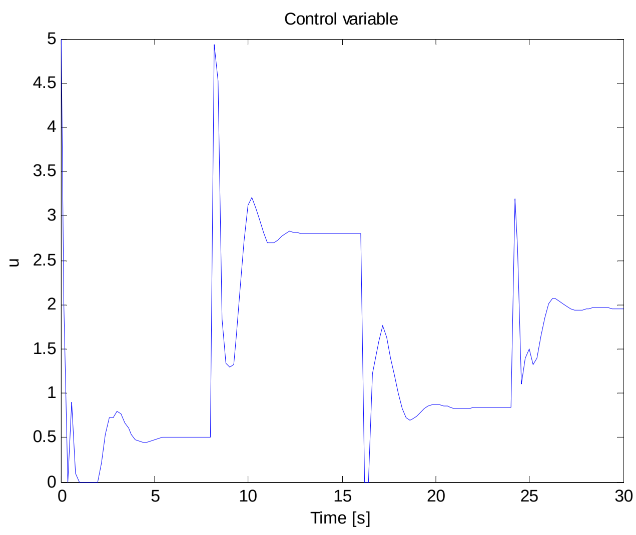


Figure 20: Control value of neural predictive controller

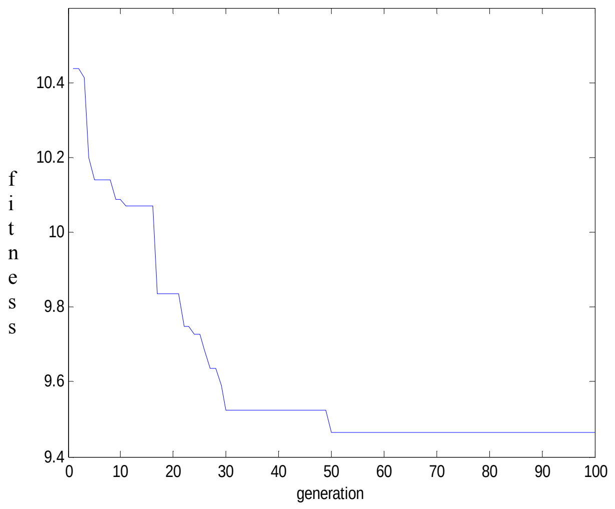


Figure 21: Evolution of the fitness function

6 Conclusion

The main objective of this article was to demonstrate very good properties of hybrid intelligent systems. In several examples from modelling and control domain we showed good efficiency of hybrid intelligent systems. We used Matlab Simulink to evaluate quality modelling and control criteria. Due to properties as the ability of learning, modelling, classifying, obtaining empirical rules, solving optimizing tasks, hybrid intelligent systems are applying in many applications.

Acknowledgement

The work has been supported by the grants agency VEGA no. 1/0544/09 and no. 1/0690/09.

References

[1] M. Negnevitsky, Artificial Intelligence. Pearson Education Limited, 2005

[2] D.E.Goldberg, Genetic Algorithms in Search, Optimisation and Machine Learning. Addisson-Wesley, 1989

[3] K.F. Man, K.S. Tang, S. Kwong, Genetic Algorithms, Concepts and Deign. Springer, 2001

[4] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolutionary Programs. Springer, 1996

[5] I.Sekaj, Evolučné výpočty a ich využitie v praxi, Iris 2005, Bratilava (in slovak)

[6] The Mathworks. Fuzzy Toolbox,User's Guide, documentation on CD

[7] D. Soloway, P.J. Haley, "Neural Generalized Predictive Control", In: *Proceedings of the 1996 IEEE International Symposium on Intelligent Control*, 1996, pp. 277-281

[8] P. Sartoris, Generovanie optimálnej štruktúry neurónových sietí pomocou genetických algoritmov, Diploma thesis, FEI STU Bratislava, 2008

[9] The Mathworks, Matlab Release 2006b, documentation on CD

[10] The Mathworks. Neural Network Toolbox,User's Guide, documentation on CD

Ing. Zuzana Didekova, E-mail: zuzana.didekova@stuba.sk
Institute of Control and Industrial Informatics, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava

Ing. Slavomír Kajan, PhD, E-mail: slavomir.kajan@stuba.sk
Institute of Control and Industrial Informatics, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology in Bratislava