Computing and Informatics, Vol. 33, 2014, 1356–1376

ADVANCED INFORMATION SYSTEM FOR SAFETY-CRITICAL PROCESSES

Štefan Kozák, Slavomír Kajan, Ján Cigánek
Viktor Ferencey, Igor Bélai

Institute of Automotive Mechatronics
Institute of Robotics and Cybernetics
Faculty of Electrical Engineering and Information Technology
Slovak University of Technology in Bratislava
Ilkovičova 3
812 19 Bratislava, Slovakia
e-mail: stefan.kozak@stuba.sk, slavomir.kajan@stuba.sk

Abstract. The paper deals with the design and implementation of an intelligent modular information system (IMIS) for modeling and predictive decision making supervisory control of some important critical processes in a nuclear power plant (nuclear reactor) using selected soft computing methods. The developed IMIS enables monitoring critical states, safety impact analysis and prediction of dangerous situations. It also recommends the operator possibilities how to proceed to ensure safety of operations and humans and environment. The proposed complex IMIS has been tested on real data from a nuclear power plant process primarily used as supervisory information for decision making support and management of critical processes. The core of the proposed IMIS is a general nonlinear neural network mathematical model. For prediction of selected process variables an artificial neural network of multilayer perceptron type (MLP) has been used. The effective Levenberg-Marquardt method was used to train the MLP network. Testing and verification of the neural prediction model were carried out on real operating data measurements obtained from the NPP Jaslovské Bohunice.

Keywords: Information system, soft computing methods, neural model, multilayer perceptron (MLP), training methods, critical processes, nuclear reactor

Mathematics Subject Classification 2010: 00-A71

1 INTRODUCTION

In many industrial processes there are important activities whose failure can lead to disasters with huge impact on human life, health and environment (air and water pollution, contamination, etc.). Such safety critical processes can be found in power, chemistry, gas, and transport industries, they include processes with accumulation, distribution and transformation of different kinds of energy (heat, electricity, energy, nuclear, etc.), raw materials processing (production of pulp and paper, steel, gas processes, chemical production, heat processes, electricity and water production, thermal and nuclear power plants, etc.) and all processes where unexpected and uncontrolled change in the on-going dynamics due to disturbances can be dangerous for a normal running operation. In such processes, human failure in the management and decision-making can cause irreparable damages to living organisms, technologies and materials, and even threaten the human existence for a long time. To avoid such adverse situations it is necessary to develop procedures, methods and algorithms to be able to timely identify critical situations and conditions in different processes, and select appropriate strategies and management practices to lead the critical state to operational and safe state. Because industrial processes are complex ones with many inputs, outputs and state variables, optimal decision in critical situations is not trivial, and has to be based on scientific approaches from mathematical modelling that enable to design optimal structure and parameters of the mathematical model. Decision support in critical situations can be based on conventional techniques using deterministic and stochastic modelling methods and measured process data. Conventional approaches are based on modelling and prediction [15] of critical processes primarily based on regression techniques and methods, while newer methods of modelling critical processes used today are based on much more precise techniques and artificial intelligence (fuzzy, neuro-fuzzy, artificial neural networks and genetic algorithms) that can model the static and dynamic processes with high accuracy, and thus provide a more exact estimate of unexpected situations.

Determination of accurate mathematical models from observations of physical processes is an integral part of the general scientific methodology in modeling and control [16]. The use of methods for approximating the behavior of general nonlinear dynamic systems has a long history. Conventional models (e.g. Hammerstein, Walsh, Voltera and Wiener models) played their very important roles during the past twenty years. The emergence of the neural network (NN) paradigm as a powerful tool for learning complex mappings from a set of examples has generated a great deal of excitement in using neural network models for identification and control of dynamical systems with unknown nonlinearities [7, 3]. Due to NN approximation capability as well as their inherent adaptivity features, artificial neural networks offer an appealing representation appropriate for modeling and control of nonlinear static and dynamic systems [5, 6, 8, 10].

Artificial neural networks consist of many simple interconnected nonlinear systems called neurons or processing elements.

Definition 1. The neuron denotes the map $f(U) \to Y$, where U is the input and Y is the output set. Figure 1 shows the structure of the neuron – a typical processing element.

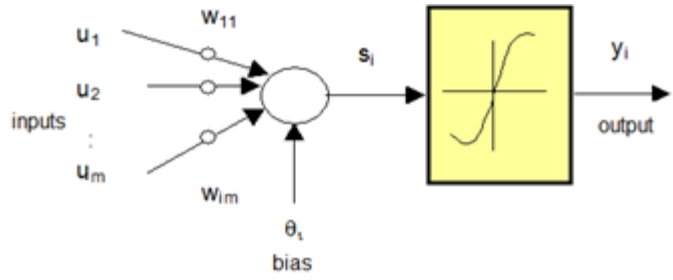

Figure 1. Internal structure of a neuron

The weighted sum of all input signal is the input signal to a nonlinear activation function which generates the network output. The map in Figure 1 can be simply expressed by the equation $y = f(u, w, \theta)$ where u is the input vector, w is the weight vector, θ is the bias vector, and f is a nonlinear function that determines the type of neuron and topology of the network. The activation signal in each processing element is the weighted sum of outputs of the previous layer plus the bias.

According to Figure 1 the activation signal for each layer can be expressed as:

$$s_i = \sum_{j=1}^{m} w_{ji} u_j + \theta_i \tag{1}$$

and output signal from the i^{th} neuron is expressed

$$y_i(u, v, \theta) = f(s_i) = f\left(\sum_{j=1}^{m} w_{ji} u_j + \theta_i\right) \tag{2}$$

where θ_i is the bias, and m is the number of inputs to the i^{th} neuron.

Definition 2. Any function can be approximated by a three-layer neural network (NN). The neuron output variable is defined as $y = f(\sum_{k=1}^{m} w_k u_k + \theta)$ where f is a transient function, m is the number of neuron inputs, w_k is a given neuron parameter (weight) and u_k are neuron inputs.

There are various ANN structures characterized by a network structure, connection patterns (weights), neural activation properties and status-updating or learning rules that modify the weights of the processing elements (neurons). Based on the connection patterns, ANN can be classified in two categories:

- feed-forward networks in which neurons have no loops;
- recurrent (feedback) networks in which loops occur due to feedback connections.

Many combinations of artificial neurons result in a useful sub-class of neural network systems.

A multilayer perceptron (MLP) is a feedforward artificial neural network model that maps sets of input data onto a set of appropriate output. An MLP consists of multiple layers of nodes in a directed graph, with each layer fully connected to the next one. Except for the input nodes, each node is a neuron (or processing element) with a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training the network. MLP is a modification of the standard linear perceptron and can distinguish data that is not linearly separable. A multilayer perceptron (MLP) with a recurrent structure with three layers and adjustable weights is depicted in Figure 2.

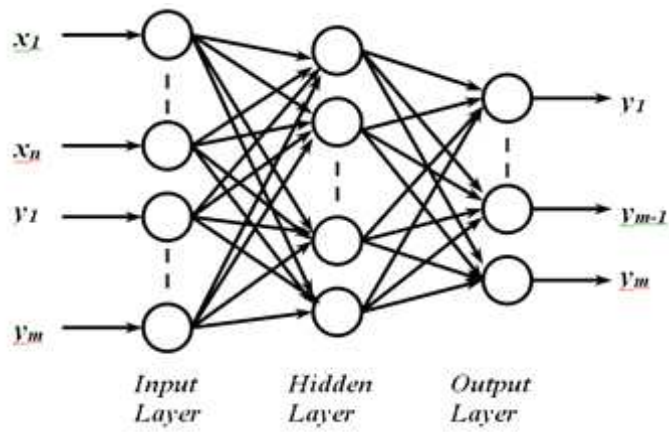

Figure 2. Multilayer neural network structures

In this paper we consider the problem of approximating of nonlinear discrete-time dynamic processes using ANN structure. For single-input single-output (SISO) discrete-time, time invariant nonlinear dynamic system it is expressed by the difference equation

$$y(k) = f(y(k-1), \ldots, y(k-n_y), u(k-1), \ldots, u(k-n_u)) \tag{3}$$

where $k \in T$ is the discrete time, $u(k) \in R^1$, $y(k) \in R^1$ are the input and output, at time k, respectively, and $f : R^{n_y} \times R^{n_u} \to R^1$ is an unknown smooth mapping defined on an open set $R^{n_y} \times R^{n_u}$.

Positive integers n_y and n_u denote the maximum lag in the system output and input, respectively. For notational convience, let $n_z = n_y + n_u$ and

$$z(k-1) = [y(k-1), \ldots, y(k-n_y), u(k-1), \ldots, u(k-n_u)]^T \in R^{n_z}. \tag{4}$$

In the network formulation, the variable $z \in R^{n_z}$ is the input to the ANN network and $w \in R^w$ is a set of adjustable parameters (weights) in the vector form.

As f is an unknown nonlinear function, the objective is to approximate $f(z)$ using some appropriate type of network approximator $N^A(z, w)$. By changing weighting parameters w, it is possible to change the input/output time response of the ANN. The training process can run on-line or off-line, i.e. during the control process or prior to it.

Multilayer neural networks are nonlinearly parametrized approximators. Theoretical works [5] have shown that such networks can uniformly approximate any function $f \in C(z)$ to any degree of accuracy provided n is sufficiently or equivalently large, provided the network has a sufficient number of neurons.

For the training of neural network we used of one modified Levenberg-Marquard method [5, 12, 9]. The Levenberg-Marquardt algorithm [12, 11] provides a numerical solution to the problem of minimizing a nonlinear function. It is fast and has stable convergence. In the artificial neural-networks field, this algorithm is suitable for training small- and medium-sized problems.

The Levenberg-Marquardt algorithm blends the steepest descent method and the Gauss-Newton algorithm. Fortunately, it inherits the speed advantage of the Gauss-Newton algorithm and the stability of the steepest descent method. It is more robust than the Gauss-Newton algorithm, because in many cases it can converge well even if the error surface is much more complex than the quadratic situation. Although the Levenberg-Marquardt algorithm tends to be a bit slower than Gauss-Newton algorithm (in convergent situation), it converges much faster than the steepest descent method.

The basic idea of the Levenberg-Marquardt algorithm is that it performs a combined training process: around the area with complex curvature, the Levenberg-Marquardt algorithm switches to the steepest descent algorithm, until the local curvature is proper to make a quadratic approximation; then it approximately becomes the Gauss-Newton algorithm, which can speed up the convergence significantly.

Let us introduce some commonly used indices for Levenberg-Marquardt weights training algorithm:

- i is the index of patterns, from 1 to N, where N is the number of patterns,
- j is the index of outputs, from 1 to M, where M is the number of outputs,
- i and j are the indices of weights, from 1 to N, where N is the number of weights,
- k is the index of iterations.

Sum square error (SSE-criterion) is defined to evaluate the training process. For all training patterns and neural network output, it is calculated by quadratic criterion

$$E(x, w) = 0.5 \sum_{i=1}^{N} \sum_{j=1}^{M} e^2(i, j) \tag{5}$$

where u is input vector, w is the weight vector, $e(i, j)$ is the training vector at

output j applying pattern i and it is defined $e(i,j) = y_p(i,j) - y_M(i,j)$ where y_p is measure output vector, y_M is actual output ANN model vector.

For the computation of weights we use the iterative recursive form of algorithm in the form

$$w_{k+1} = w_k - H_k^{-1} g_k \tag{6}$$

where g_k is the gradient

$$g_k = \frac{\partial E(x,w)}{\partial w} = \left[\frac{\partial E}{\partial w_1} \frac{\partial E}{\partial w_2} \cdots \frac{\partial E}{\partial w_N} \right]^T \tag{7}$$

and matrix H is Hessian matrix

$$H = \begin{bmatrix} \frac{\partial^2 E}{\partial w_1^2} & \frac{\partial^2 E}{\partial w_1 \partial w_2} & \cdots & \frac{\partial^2 E}{\partial w_1 \partial w_N} \\ \frac{\partial^2 E}{\partial w_2 \partial w_1} & \frac{\partial^2 E}{\partial w_2^2} & \cdots & \frac{\partial^2 E}{\partial w_2 \partial w_N} \\ \cdots & \cdots & \cdots & \cdots \\ \frac{\partial^2 E}{\partial w_N \partial w_1} & \frac{\partial^2 E}{\partial w_N \partial w_2} & \cdots & \frac{\partial^2 E}{\partial w_N^2} \end{bmatrix} \tag{8}$$

In order to make sure that the approximated Hessian matrix $J^T J$ is invertible, Levenberg-Marquardt algorithm [13, 3] introduces another approximation to Hessian matrix:

$$H \cong J^T J + \mu I \tag{9}$$

where μ is always positive, called combination coefficient and I is the identity matrix.

From Equation (9), one may notice that the elements on the main diagonal of the approximated Hessian matrix will be larger than zero. Therefore, with this approximation it can be sure that matrix H is always invertible.

Levenberg-Marquardt iterative algorithm can be expressed as

$$w_{k+1} = w_k - (J_k^T J_k + \mu I)^{-1} J_k e_k. \tag{10}$$

2 NUCLEAR REACTOR PROCESS DESCRIPTION

Using artificial neural network (ANN), measured data for modeling are collected from the technology of the nuclear power plant (NPP) (Figure 4). The main object of modeling is a nuclear reactor, in which fission of the nuclear fuel (most commonly used is the isotope of uranium – ^{235}U) is running. In fission, energy is released and most of it is converted into heat. This energy is transported into the heat exchanger (steam generator) using heat transfer medium (for example H_2O, CO_2). The hot water circulation circuit (HCC) provides heat removal from their act ort of the steam generator and conversely, the cold water circulation circuit (CCC) together with the main coolant pump (MCP) provides reactor cooling. The steam generator is a heat exchanger that uses energy released from the reactor for the production of saturated steam. The produced steam is supplied into the separator and steam

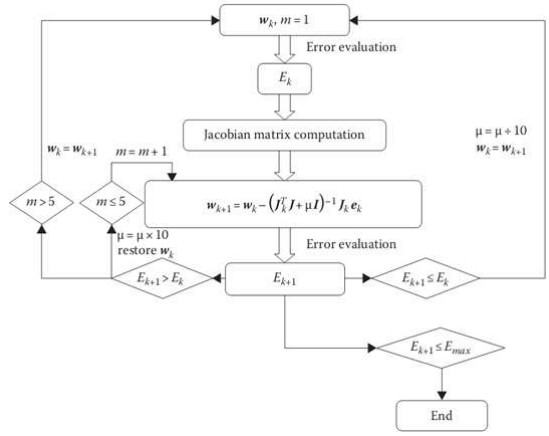

Figure 3. Block diagram for training using Levenberg-Marquardt algorithm (w_k is the current weight, w_{k+1} is the next weight, E_{k+1} is the current total error, and E_k is the last total error)

heater to reduce moisture, and is further preheated to a temperature above the temperature of saturation to increase effectiveness of the work cycle. Then, the steam is supplied to the turbine that drives a synchronous generator and produces electricity.

For modeling the nuclear reactor using ANN, the measured output process variables are neutron flux and output water temperature in HCC. Measured temperature of the cooling water inflow in CCC, pressure within the core and differential pressure in MCP are process input variables.

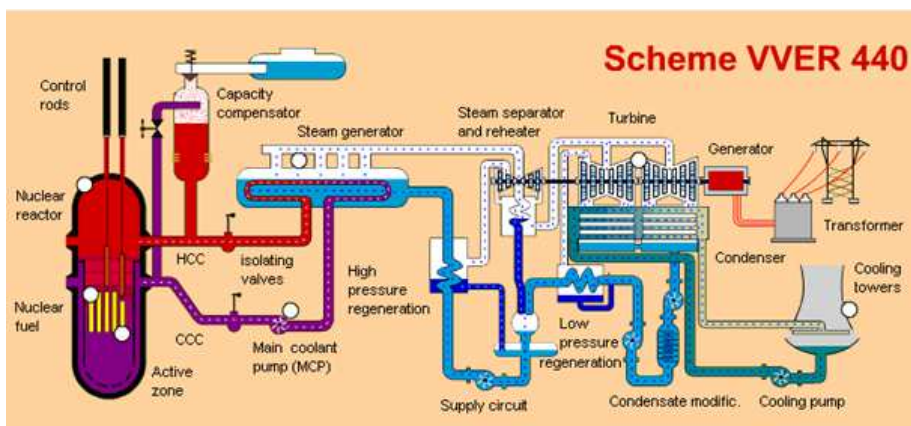

Figure 4. Functional scheme of electricity production in nuclear reactor

3 MODELING OF PROCESSES IN NUCLEAR REACTOR

Recently, ANN have been used increasingly in various application areas dealing with modeling of nonlinear processes [1]. Due to very good approximation properties of a multilayer perceptron (MLP) networks, the neural models can be successfully applied for modeling of nonlinear processes. Using neural models for nuclear reactor process modeling enables to predict behavior of parameters of the reactor which are important for its safety [2].

One of the main functions of ANN is to predict future outputs [17] based on past measured inputs and outputs. Neural networks can learn to perform a variety of predictive tasks. Network training is performed by repeating the following procedure. Each step of the training procedure consists of the following steps:

1. Provide input data values to nodes of the input layer.
2. Propagate the presented input values towards the output layer (forward process).
3. Compare the values of output nodes with the actual training data values.
4. Correct the differences of outcome and propagate towards the input layer (backward process).

Normally, training data records are to be applied many times before the network is actually used. Network training is a repetitive process. At the beginning, networks are trained roughly. Then, they are refined by repeated application of input data. After the network reaches a certain maturity level, they are used or deployed for value prediction.

For modeling and prediction of variables in the nuclear reactor, a neural model based on multilayer perceptron network was created, whereby the neural model represents a MIMO system with 3 inputs and 2 outputs. The ANN approximates dynamics of the modeled system and is described by following nonlinear difference equation:

$$\begin{aligned}[y_1(k), y_2(k)] \;=\; & f(y_1(k-1), \ldots, y_1(k-n_1), y_2(k-1), \ldots, y_2(k-n_2), \\ & u_1(k-d_1), \ldots, u_1(k-d_1-m_1), u_2(k-d_2), \ldots, \\ & u_2(k-d_2-m_2), u_3(k-d_3), \ldots, u_3(k-d_3-m_3))\end{aligned} \tag{11}$$

where $k = t/T_{vz}$ is discrete time (T_{vz} is sampling time), $y_j(k)$ are outputs of the modeled system, $u_i(k)$ are inputs of the modeled system, n_i are orders of outputs $y_j(k)$, m_i are orders of inputs $u_i(k)$, d_i are time delays of inputs $u_i(k)$, $f(\ldots)$ is unknown nonlinear function.

The proposed neural model of the nuclear reactor will be implemented off-line using measured data. A block scheme of the ANN process model with one input and one output is depicted in Figure 5, where the neural model is located in parallel to the process, and the prediction error is used as the network training signal for the learning algorithm. The Levenberg-Marquardt learning method has been used

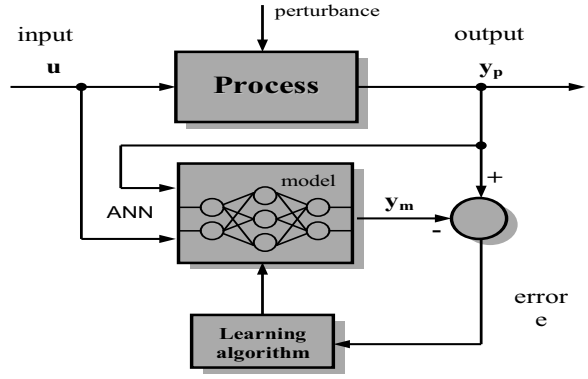

Figure 5. Block scheme of process modeling and learning using ANN

for training the MLP network. The modeling process was implemented in MATLAB [4] using measured process data. The neural model structure for the system with 3 inputs and 2 outputs is shown in Figure 6. Table 1 shows a description of inputs and outputs of the neural model, whereby the variables were normalized to range within $(-1, 1)$.

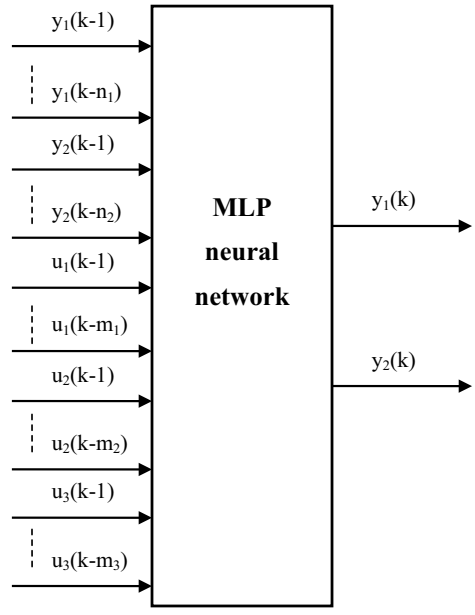

Figure 6. Structure of MLP network for process modeling

Name and type of the variable	**Description**	**Unit**	**Range**
Output y_1	NEUTRON FLUX FROM AKR-02R 1.CHANNEL	A	$(5.8e^{-5}, 6.8e^{-5})$
Output y_2	TEMPERATURE IN HCC 1/TPGN1/1	$^\circ$C	$(290, 300)$
Input u_1	TEMPERATURE IN CCC 1/TPCHN1/1	$^\circ$C	$(265, 270)$
Input u_2	PRESSURE IN ACTIVE AREA	MPa	$(12, 12.2)$
Input u_3	DIFFERENTIAL PRESSURE IN MCP	MPa	$(0.43, 0.45)$

Table 1. Description of measured input and output variables of the nuclear reactor

4 MODELING AND PREDICTION OF PROCESSES IN NUCLEAR REACTOR

For implementation of the ANN prediction model of the nuclear reactor processes a modular intelligent program system was created in Matlab-Simulink. The block structure is shown in Figure 7. After starting the program, the main window (Figure 8) is opened. The program system consists of the following modules: *data collection module, dynamic models module, prediction module, parameter settings module* and the *module of recognition* of critical states.

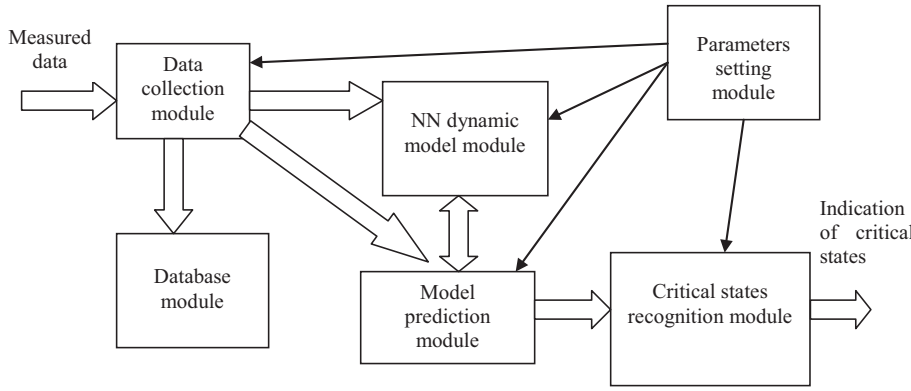

Figure 7. Modular intelligent information system for modeling and prediction of critical states in nuclear reactor processes

The data collection module has to collect and preprocess the measured data for modeling and prediction. The module of dynamic models contains the MLP neural network trained on measured data from the reactor. The prediction module predicts future outputs in defined forward steps on the basis of past models inputs and outputs. Based on prediction of reactor variables from the neural model, the module for recognition of critical states indicates the state variables and their exceeding of technical and safety limits.

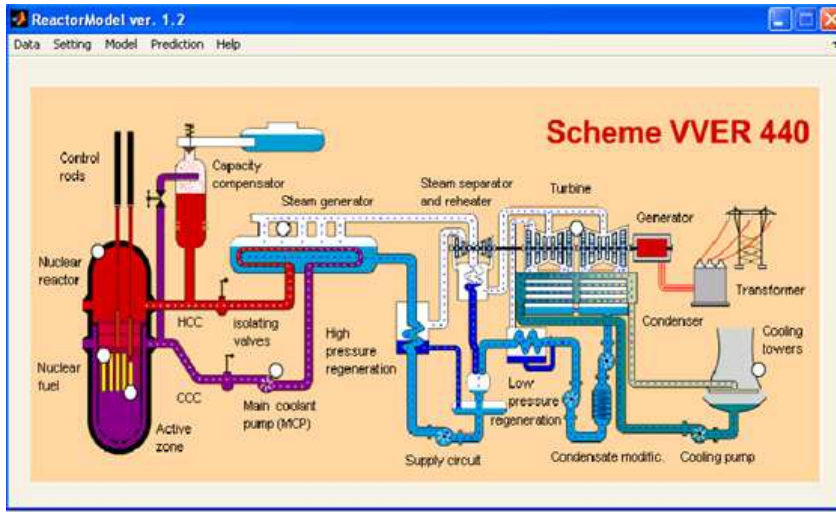

Figure 8. Main window of the *Reactor Model* program for modeling and prediction of nuclear reactor processes

The program system processes the measured data in the program through the *Data* menu (Figure 9), enabling loading, saving, filtering, normalizing and displaying measured data.

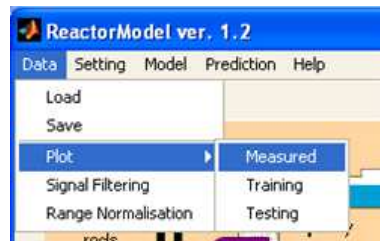

Figure 9. *Data* menu of the main window of the *ReactorModel* program

Measured input and output variables of the model (according to Table 1) are shown in Figures 10 and 11. For neural model training, 2 500 measured data samples were selected (from 50 501 to 53 000 samples) previously smoothed by averaging filter, and normalized to range (-1,1). Training data used for modeling are shown in Figure 10. For model verification, the full range of measured data has been used.

For model parameters setting, saving into and loading from a file, training and testing of model, the *Model* menu of the *ReactorModel* program can be used (Figure 9).

For setting prediction horizons of inputs and outputs, and for model prediction, the menu *Prediction* was created.

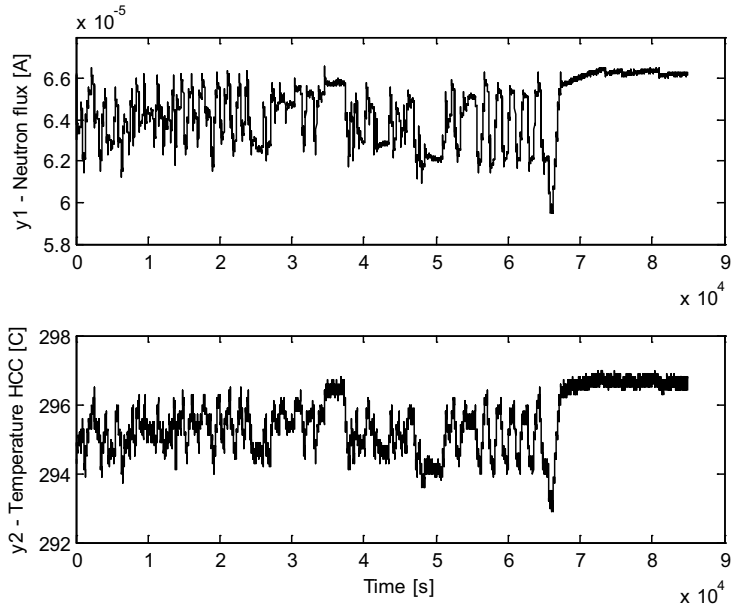

Figure 10. Measured output variables data of the neural model (y_1-neutron flux, y_2-temperature in HCC)

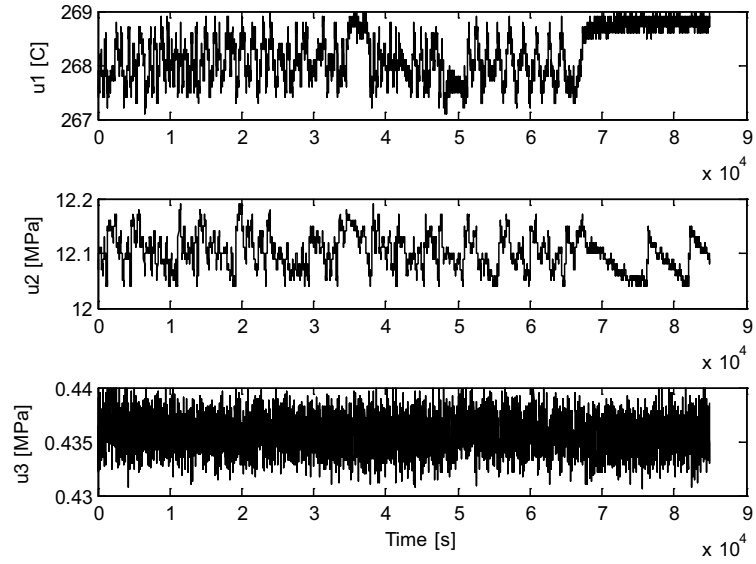

Figure 11. Measured input variables data of the neural model (u_1-temperature in CCC, u_2-pressure AZ, u_3-dif. pressure in MCP)

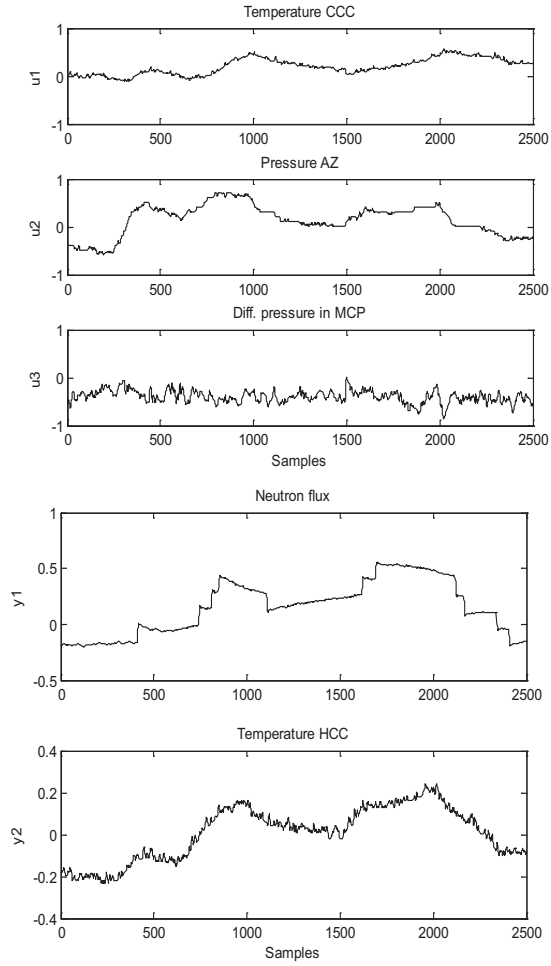

Figure 12. Training data (input and output variables) of the neural model

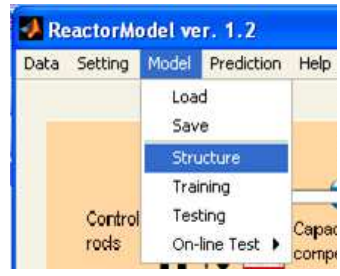

Figure 13. *Model* menu of the main window of program *ReactorModel*

5 NEURAL MODEL IMPLEMENTATION: SIMULATION RESULTS

The neural model was created using MLP NN network with one hidden layer which contains 15 neurons with the *tansig* type activation function. Previous samples of individual variables were used as input data to the neural model; in total, the neural model has 16 inputs and 2 outputs. Setting parameters of the neural model structure is shown in Figure 14.

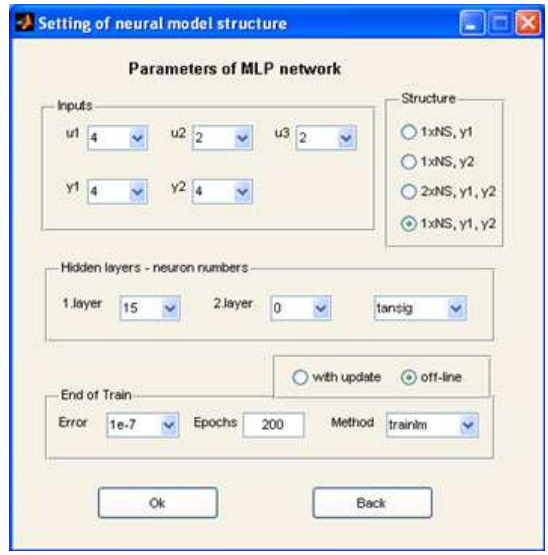

Figure 14. Window for setting of neural model structure

The MLP network was trained on training data from Figure 12 with set parameters from Figure 14. Comparison of outputs from the neural model and the process for training data is shown in Figures 15 and 16. Testing of neural model was realized using all measured data; comparison of outputs from the neural model and the process for training data is shown in Figures 17 and 18. The comparison shows that the model output for neutron flux fits measured data very well. The HCC temperature model output also fits the temperature dynamics; however errors occur in some measurement intervals due to disturbance variables not included in the model. Numerical evaluation of errors between measured and model outputs are provided in Table 2.

The neutron flux prediction was tested using the created neural model, and compared with the measured data. Testing was performed on a data set of $N = 30\,000$ samples (Figure 19). Errors for prediction horizons (N_p) 5, 7 and 10 are recorded in Table 3. Figure 19 and Table 3 show that the predictive model follows the dynamics of the process very well, only in a few intervals small deviations between the output

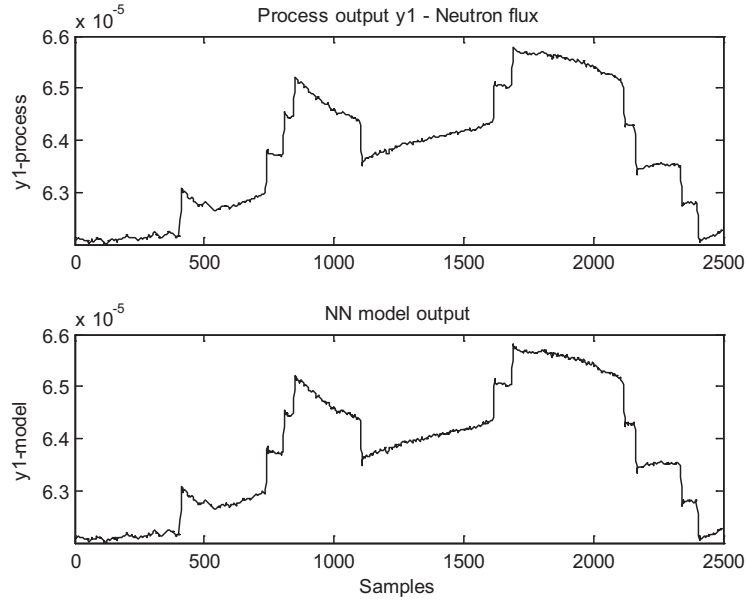

Figure 15. Comparison of outputs from the neural model and the process (neutron flux – training data)

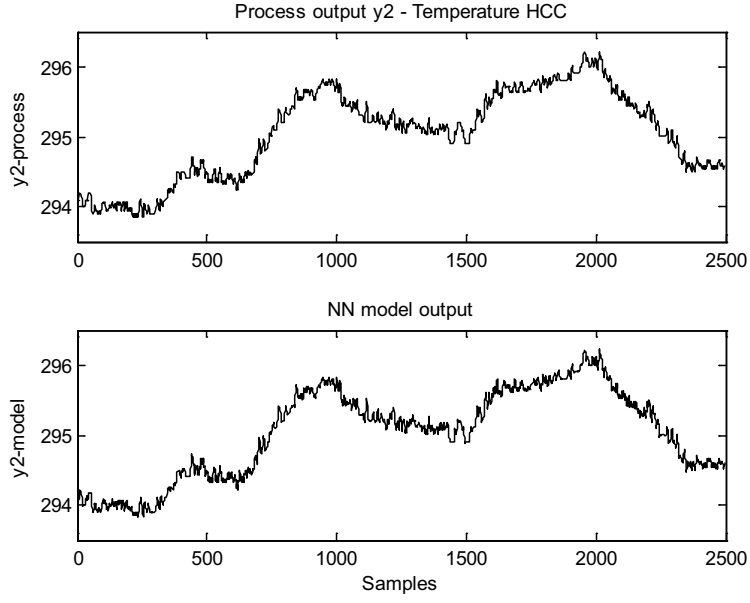

Figure 16. Comparison of outputs from the neural model and the process (temperature in HCC – training data)

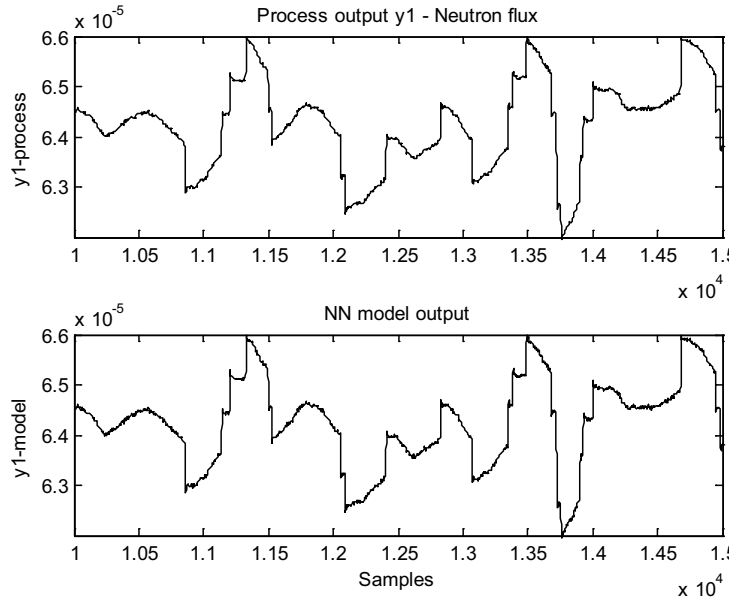

Figure 17. Comparison of outputs from the neural model of the neutron flux and the process (testing data)

Model testing criterion	Training data for y_1, $N = 2\,500$	Training data for y_2, $N = 2\,500$	Testing data for y_1, $N = 70\,000$	Testing data for y_2, $N = 70\,000$
SSE (sum of squared errors) $\sum_{k=1}^{N}(y_p - y_m)^2$	0.0034	0.0366	0.7014	6.9957
MSE (mean squared error) $\frac{1}{N}\sum_{k=1}^{N}(y_p - y_m)^2$	$0.0136e^{-4}$	$0.1467e^{-4}$	$0.01e^{-3}$	$0.5285e^{-3}$

Table 2. Numerical evaluation of neural model errors

and the process model occurred. The model errors increase with prediction horizon increasing N_p.

The results obtained by numerical simulation have shown a high accuracy and high model quality in the use of MLP ANN models and improved learning Levenberg-Marquardt methods. The deviation between the measured and modeled output variables is ranged up to 10^{-3}. To determine the optimal structure, the authors of the proposed paper developed a new algorithm for optimal ANN structure determination directly from the measured data, which can be significantly improve quality modeling and realization of modeling in the real time operation.

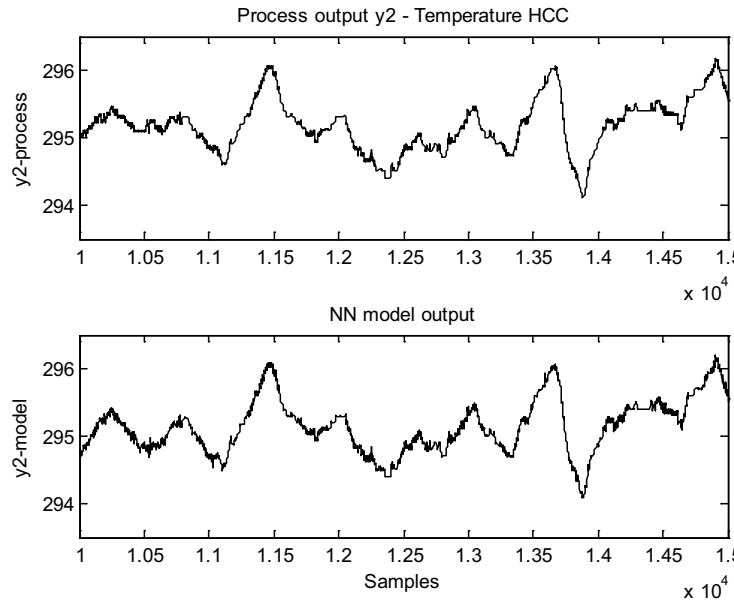

Figure 18. Comparison of outputs from the neural model of HCC temperature and the process (testing data)

Testing data for y_1, $N = 30\,000$	$N_p = 3$	$N_p = 5$	$N_p = 7$	$N_p = 10$
SSE	10.305	79.315	348.205	1 045.37
MSE	$0.343e^{-3}$	$2.645e^{-3}$	$11.611e^{-3}$	$34.862e^{-3}$

Table 3. Numerical evaluation of prediction errors using neural model

6 CONCLUSIONS

The paper deals with the design of an intelligent modular complex information system for modeling of selected critical processes in a nuclear reactor. Testing and verification of the developed program system using the neural prediction model was performed in Matlab-Simulink using real data measured from the nuclear power reactor. The obtained graphical and numerical results confirm high quality of modeling highly nonlinear processes. The developed complex information system is elaborated in most general form and can be used for identification and prediction of critical situations and states in a wide class of industrial processes. Intelligent and interactive programming system (IMIS) has a modular interactive structure and can be applied as supervisory control systems for managing of critical processes by operators and process managers.

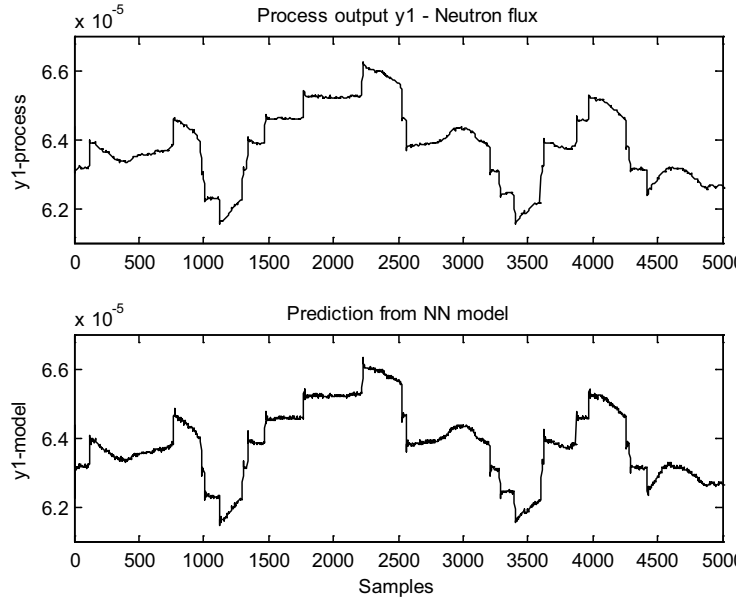

Figure 19. Comparison of the model prediction and the process output (neutron flux with prediction horizon $N = 3$)

Acknowledgement

We thank for the support of the European Operating Program Reasearch and Development project ITMS code 26240220060, of the VEGA project No. 1/0937/14 and of the Slovak Research and Development Agency under grant No. APVV-0772-12.

REFERENCES

[1] Jadlovská, A.: Modelling and Control of Dynamic Processes Using Neuron Networks. FEI TU Košice, 2003, ISBN 80-8894122-9 (in Slovak).

[2] Kajan, S.—Hypiusová, M.: Application of Artificial Neural Network in Modelling and Simulation a Power Engineering Process. Control of Power and Heating Systems 2006, Zlín 2006.

[3] Hagan, M. T.—Menhaj, M. B.: Training Feedforward Networks with the Marquardt Algorithm. IEEE Transactions on Neural Networks, Vol. 5, 1994, No. 6, pp. 989–993.

[4] The Mathworks. Neural Network Toolbox, User's Guide, 2002.

[5] Hornik, M.—Stinchcombe, M.—White, H.: Multilayer Feedforward Networks Are Universal Approximators. Neural Networks, Vol. 2, 1989, No. 5, pp. 359–366.

[6] Lippmann, R. P.: Pattern Classification Using Neural Networks. IEEE Communications Magazine, Vol. 27, 1989, pp. 47–64.

[7] Rumelhart, D. E.—Hinton, G. E.—Williams, R. J.: Learning Representations by Back-Propagating Errors. Nature, Vol. 323, 1986, pp. 533–536.

[8] Siegelmann, H. T.: Neural Networks and Analog Computation: Beyond the Turing Limit. Birkhauser, Boston 1999.

[9] Andersen, T. J.—Wilamowski, B. M.: A Modified Regression Algorithm for Fast One Layer Neural Network Training. World Congress of Neural Networks, Washington, DC, USA, July 17–21, 1995, Vol. 1, pp. 687–690.

[10] Werbos, P. J.: Back-Propagation: Past and Future. Proceedings of International Conference on Neural Networks, San Diego, CA, USA, 1988, Vol. 1, pp. 343–354.

[11] Levenberg, K.: A Method for the Solution of Certain Problems in Least Squares. Quarterly of Applied Mathematics, Vol. 2, 1944, No. 2, pp. 164–168.

[12] Marquardt, D.: An Algorithm for Least-Squares Estimation of Nonlinear Parameters. SIAM Journal on Applied Mathematics, Vol. 11, 1963, No. 2, pp. 431–441.

[13] Valo, R.—Kozák, Š.: Effective Application of Levenberg-Marquardt Teaching. In: Kozák, Š., Kozáková, A., Rosinová, A. (Eds.): Kybernetika a informatika. Medzinárodná konferencia SSKI SAV, STU Bratislava, 2012, pp. 107–111 (in Slovak).

[14] Ionescu, M.—Sburlan, D.: Some Applications of Spiking Neural P Systems. In Computing and Informatics, Vol. 27, 2008, No. 3, pp. 515–528.

[15] Korenčiak, D.—Gutten, M.: Opportunities for Integration of Modern Systems Into Control Processes in Intelligent Buildings. Przeglad Elektrotechniczny, Electrical Review, Vol. 88, 2012, No. 2, pp. 266–269, ISSN 0033-2097.

[16] Kashif, Z.—Rauf, A.—Rauf, B.: Multiple Route Generation Using Simulated Niche Based Particle Swarm Optimization. Computing and Informatics, Vol. 32, 2013, No. 4, pp. 697–721.

[17] Korošec, P.—Šilc, J.: Using Stigmergy to Solve Numerical Optimization Problems. Computing and Informatics, Vol. 27, 2008, No. 3, pp. 377–402.

Štefan Kozák obtained his M. Sc. degree from the Slovak University of Technology in Bratislava in 1970 and the Ph. D. degree in technical cybernetics from the Slovak Academy of Sciences in 1978. He worked at the Institute of Technical Cybernetics in the field of control algorithms design and was a leader of a research team at the Institute of Applied Cybernetics in Bratislava. Since 1984 he had been with the Department of Automatic Control Systems at the Faculty of Electrical Engineering and Information Technology in Bratislava (between 1998 and 2006 he lead the department). Currently, he is with the Institute of Automotive Mechatronics at the Faculty of Electrical Engineering and Information Technology, STU in Bratislava. His research interests include system theory, linear and nonlinear control methods, numerical methods and software for modeling, control, signal processing and embedded intelligent systems. He published over 250 research papers in conference proceedings and international journals, and organized four IFAC events held in Slovakia.

Slavomír Kajan received his diploma and Ph. D. degree in automatic control from the Faculty of Electrical Engineering and Information Technology, Slovak University of Technology (FEI STU) in Bratislava, in 1997 and 2006, respectively. He is now an Assistant Professor at the Institute of Control and Industrial Informatics, FEI STU in Bratislava. His research interests include servo-systems, soft-computing control methods and robust control.

Ján Cigánek received his diploma and Ph. D. degree in automatic control from the Faculty of Electrical Engineering and Information Technology, Slovak University of Technology (FEI STU) in Bratislava, in 2005 and 2010, respectively. He is now an Assistant Professor at the Institute of Automotive Mechatronics, FEI STU in Bratislava. His research interests include optimization, robust control design, computational tools, and hybrid systems.

Viktor Ferencey is now a Full Professor at the Institute of Automotive Mechatronics, Faculty of Electrical Engineering and Information Technology in Bratislava. His research work includes optimization of energy sources and of power systems for electric drives. He conducts research in energy intensity for fuel cells in the capacity of power source for an electric vehicle. In pedagogical work, he focuses on teaching of mechatronics systems for engines and managing the dynamics of movement of electric vehicles. In cooperation with automotive industry, he addresses issues of research and development of hybrid and electric propulsion systems for different types of vehicles. He is author of three monographs, four university books, several textbooks and over 150 scientific and professional publications.

Igor Bélai graduated in electronic computers from Faculty of Electrical Engineering, Slovak University of Technology in Bratislava, Slovakia. He completed doctoral studies in automation and control. He is currently an Assistant Professor at the Institute of Automotive Mechatronics, FEI STU in Bratislava. His research interests include control of electrical drives and industrial communication systems.