HEURISTIC MODELING OF OBJECTS AND PROCESSES USING DYNAMIC NEURAL NETWORKS

Piotr PRZYSTAŁKA

Silesian University of Technology, Department of Fundamentals of Machinery Design
Konarskiego Street 18a, 44-100 Gliwice, Poland, e-mail: pp@polsl.pl

Summary

The methodology of heuristic modeling is one of the subjects included in the activities developed by the Department of Fundamentals of Machinery Design [4, 6]. Among all the approaches of heuristic modeling some of the most common are artificial neural networks. There are many papers and books devoted to applications of neural networks for modeling dynamic systems [1, 2, 4, 5, 6, 7]. In this paper, known approach basing on dynamic neuron model is presented (dynamic neuron with IIR filter in the activation block [2]) but some developments are introduced. Locally recurrent networks which are composed of dynamic neural units described in [2, 5, 7] are able to model behavior of complex dynamic systems. Nevertheless, they have one major disadvantage, that is, neural networks composed of these neurons are not able to represent stochastic behaviors of some objects [4,6]. By introducing the ARMAX (or ARX) system into dynamic neuron model author has received dynamic neuron unit that never behaves in the same way (it brings an artificial neuron closer and closer to the biological model). In this paper the author presents formal description of dynamic neuron unit with ARMAX system in the feedback block. There are also described a general structure of dynamic neural network composed of these neurons, two known training methods and some commonly used quality measures. At the end of the paper three examples of applications are given.

Keywords: artificial neural networks, locally recurrent neural networks, linear, non-linear and chaotic dynamic systems, quasi-Newton methods, heuristic modeling.

HEURYSTYCZNE MODELOWANIE OBIEKTÓW I PROCESÓW PRZY POMOCY DYNAMICZNYCH SIECI NEURONOWYCH

Streszczenie

Metodologia heurystycznego modelowania obiektów i procesów jest jednym z kierunków badań rozwijanym prze Katedrę Podstaw Konstrukcji Maszyn [4, 6]. Spośród wielu metod modelowania heurystycznego duże znaczenie odgrywają metody bazujące na sztucznych sieciach neuronowych. Można wyróżnić wiele ciekawych prac badawczych prowadzonych w kierunku modelowania systemów dynamicznych z zastosowaniem tego typu narzędzia [1, 2, 4, 5, 6, 7]. W artykule zaprezentowano znane podejście bazujące na dynamicznych neuronach (dynamiczny neuron z filtrem IIR w bloku aktywacyjnym [2]) z pewnymi modyfikacjami. Lokalnie rekurencyjne sieci neuronowe złożone z dynamicznych neuronów opisane w [2, 5, 7] nadają się do modelowania zachowania złożonych systemów dynamicznych. Jednakże, posiadają one jedną główną wadę tzn. nie są zdolne do reprezentowania zachowania losowego niektórych obiektów [4, 6]. Poprzez wprowadzenie systemu typu ARMAX (ARX) do modeli dynamicznych neuronów autor otrzymał dynamiczny model neuronu, który nigdy nie zachowują się w ten sam sposób (przybliża to model sztucznego neuronu do jego biologicznego wzoru). W artykule autor prezentuje formalny opis dynamicznego neuronu z systemem typu ARMAX w bloku sprzężenie zwrotnego. Opisuje również ogólną strukturę dynamicznej sieci neuronowej złożonej z tych neuronów, dwa znane algorytmy trenujące oraz powszechnie stosowane miary jakości. Przykładowe zastosowania opisywanych sieci zaprezentowane są w końcowym fragmencie opracowania.

Słowa kluczowe: sztuczne sieci neuronowe, lokalnie rekurencyjne sieci neuronowe, liniowe, nieliniowe i chaotyczne systemy dynamiczne, metody quasi-Newtonowskie, modelowanie heurystyczne.

INTRODUCTION

Nowadays there are many industrial plants that carry out complex processes. In general, a dynamic behaviour of them is difficult to be modelled with the use of classical analytical methods [1, 2, 3, 4, 6]. Sometimes it is easier and faster (in some cases it is only one way) to use selected heuristic methods (soft modeling methods) in order to solve various engineering problems in modeling tasks [4]. The

paper describes locally recurrent neural networks applied to model non-linear dynamic objects and processes. Presented networks consist of artificial neurons with linear dynamic system blocks. The general concept is not new [2, 5], but some further developments are introduced. Generally, neural networks which are composed of those neurons [2, 5] are able to represent dynamic behaviour of some systems but they are not able to model their stochastic behaviour themselves [4, 6]. Therefore, there is need to elaborate much more general neuron models which can be used for modeling both dynamic and stochastic systems simultaneously.

In this paper, the author develops dynamic neuron model with the IIR or FIR filter in the activation block by introducing the ARMAX (or ARX) system into its structure. The behaviour of that neuron is not always deterministic (it is described in Section 1).

The paper is composed as follows. In Section 1 only some equations for formal descriptions of dynamic neurons are shown. Next, in Section 2 a few different variants of a dynamic unit are presented. Section 3 describes applications of two known quasi-Newton methods (the BFGS method and the LM method) for training depicted neural networks. At the end of the paper, there are three examples presented. They show modeling of dynamic systems, fault-tolerant application and modeling of an electric furnace.

1. ARTIFICIAL NEURON WITH IIR/FIR AND ARX/ARMAX DYNAMIC SYSTEMS

Developing dynamic neural units is one of the most common ways to improve the ability of artificial neural networks to model linear, non-linear and chaotic dynamic systems.

Dynamic and stochastic behaviour is embedded in the neuron by introducing the IIR or FIR filter and the ARX or ARMAX system into its structure.

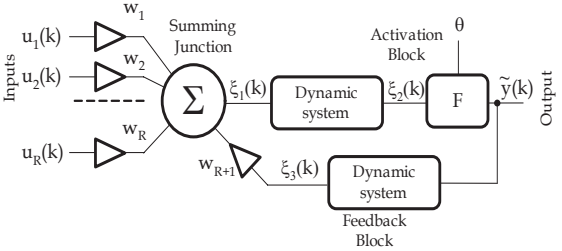

Fig. 1. Formal model of artificial neuron with dynamic systems in the activation and feedback block

The behaviour of the dynamic neural unit under consideration is described by the following equations:

$$\xi_1(k) = \sum_{i=1}^{R} u_i(k) \cdot w_i + \xi_3(k) \cdot w_{R+1}, \qquad (1)$$

where: $\mathbf{\Theta}_1 = [w_1 \;\; w_2 \;\; \ldots \;\; w_{R+1}]$ is a vector with input weights w_i; $u_i(k)$ are neuron inputs, i=1,2,...,R; R is a number of inputs.

$$\xi_2(k) = \sum_{\tau=0}^{B} b_\tau \cdot \xi_1(k-\tau) - \sum_{\tau=1}^{A} a_\tau \cdot \xi_2(k-\tau), \qquad (2)$$

where: $\mathbf{\Theta}_2 = [a_1 \;\; a_2 \;\; \ldots \;\; a_A \;\; b_0 \;\; b_1 \;\; \ldots \;\; b_B]$ is a vector describing the IIR filter placed between the summing junction and the activation block; a_τ and b_τ are the feedback and feed-forward filter parameters; (A,B) is its order, and

$$\xi_3(k) = \sum_{\tau=0}^{C} c_\tau \cdot \tilde{y}(k-\tau) - \sum_{\tau=1}^{D} d_\tau \cdot \xi_3(k-\tau) + \sum_{\tau=0}^{E} e_\tau \cdot \phi(k-\tau), \qquad (3)$$

where: $\mathbf{\Theta}_3 = [\mu \;\; d_1 \;\; d_2 \;\; \ldots \;\; d_D \;\; c_0 \;\; c_1 \;\; \ldots \;\; c_C \;\; e_0 \;\; e_1 \;\; \ldots \;\; e_E \;\; \sigma_e^2]$ is a vector describing the ARMAX system located in the output feedback block; ϕ - vector of a white-noise random process where μ and σ_e^2 are the mean value and variance of a white noise; c_τ , d_τ , e_τ - system parameters; (C,D,E) is the structure representation of ARMAX system. The term $e_\tau \cdot \phi(k-\tau)$ is zero for deterministic dynamic systems and is a white-noise random process for stochastic systems.

There are three different ways of calculating its output:

- by using Gaussian activation function:

$$\tilde{y}(k) = \mathrm{F}(\xi_2, \mathbf{\theta}) = e^{-\frac{1}{2}\left(\frac{\xi_2(k)-c}{\sigma}\right)^2}, \qquad (4)$$

- by using hyperbolic activation function:

$$\tilde{y}(k) = \mathrm{F}(\xi_2, \mathbf{\theta}) = = \frac{2}{1+\exp[-\beta \cdot (\xi_2(k) + \delta u)]} + b - 1, \qquad (5)$$

- by using linear activation function:

$$\tilde{y}(k) = \mathrm{F}(\xi_2, \mathbf{\theta}) = a \cdot \xi_1(k), \qquad (6)$$

where: $\mathbf{\theta} = [\sigma \;\; c \;\; \beta \;\; \delta u \;\; b \;\; a]$ - a vector depending on selected activation function.

2. DYNAMIC NEURAL NETWORK

This class of artificial neural networks (with dynamic neuron models) is well-known as *locally recurrent globally feed-forward networks* [2,5]. As we can read in [2] that is *somewhere in between a feed-forward and a globally recurrent architecture.* The dynamic unit described in Sec. 1 may be used differently.

Figure 2 shows three units: the first one is a static neuron (for B=0, b_0=1, A=1, a_1=0, C=c_0=D= =d_1=E=e_0=0); the second unit is a completely dynamic neuron model with white-noise random process and hyperbolic (tan for short) activation function; the last one represents also

a dynamic neuron model with the finite impulse response filter and there is Gaussian (gauss for short) activation function. The topology of a dynamic neural network (described in Fig. 3) which has been used in this part of the research consists of three layers (a general structure was chosen based on information in the literature). The first layer has simple static neurons with a non-linear activation function. Hidden layer includes dynamic neurons with a non-linear activation function (dynamic neurons $iir^{(A,B)}armax^{(C,D,E)}$-*tan*, $fir^{(B)}arx^{(C,D)}$ -*gauss*, $iir^{(A,B)}arx^{(C,D)}$-*tan*, etc. were tested). The last layer consists of simple static units but the activation function is linear.

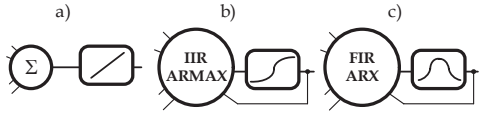

Fig. 2. Example of dynamic neural units (short notation): a) static neuron; b,c) dynamic neurons with the IIR or FIR block and ARMAX or ARX block

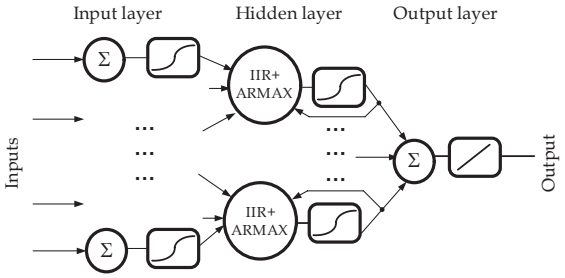

Fig. 3. Structure of a dynamic neural network

All unknown network parameters may be represented by the vectors: ${}^{L}\boldsymbol{\Theta} = \left[{}^{L}\boldsymbol{\Theta}_1^n \quad {}^{L}\boldsymbol{\Theta}_2^n \quad {}^{L}\boldsymbol{\Theta}_3^n \quad {}^{L}\boldsymbol{\theta}^n \right]$, where: L is a number of the layer, n is a unit number. For a three-layered network we are able to define: $L \in \{1,2,3\}$ and for each layers there are $n=n_L=\{1,2,...,NL\}$, NL - the number of units in the layer L.

3. LEARNING ALGORITHMS AND QUALITY COEFFICIENTS

3.1. Learning methods

There are a lot of optimization problems in many engineering areas which can be formulated as follows: given a real-valued loss (cost) function $f : \Re^n \to \Re^1$, find a global minimum,

$$\boldsymbol{\Theta}^* = \arg\min_{\Theta \in C} f(\boldsymbol{\Theta}), \tag{7}$$

where: C is the predefined parameters space, usually a compact set in $\Re^n$; $\boldsymbol{\Theta}^*$ is the vector with optimal network parameters.

There are many solutions to this problem, but in general two kinds of high-performance algorithms are widely used: based on standard numerical optimization techniques (like gradient methods, quasi-Newton methods) or based on heuristic techniques (like recursive random search, simulated annealing, genetic algorithms or even simple momentum technique in the backpropagation algorithm). In this paper the Broyden, Fletcher, Goldfarb, and Shanno (BFGS for short) quasi-Newton algorithm and the Levenberg-Maquardt (LM for short) algorithm are used. For the cost function that is described by the following equation:

$$f(k;\boldsymbol{\Theta}) = \left\| y(k) - \hat{y}(k;\boldsymbol{\Theta}) \right\|, \tag{8}$$

should be minimized basing on a given set of input-output patterns. All the line-search methods explore along the line containing current point $\boldsymbol{\Theta}(k)$, parallel to the search direction $\mathbf{d}(k)$ (Eq. 9).

$$\boldsymbol{\Theta}(k+1) = \boldsymbol{\Theta}(k) + \alpha \cdot \mathbf{d}(k), \tag{9}$$

where: α is a scalar step length parameter. In the next part of this Section the two ways of definition of a direction will be discussed.

For the BFGS method the direction is given as follows:

$$\mathbf{d}(k) = -\mathbf{H}^{-1}(k;\boldsymbol{\Theta}) \cdot \nabla f(k;\boldsymbol{\Theta}), \tag{10}$$

where: the gradient information $\nabla f(k;\boldsymbol{\Theta})$ is derived by partial derivatives using the numerical differentiation method via finite differences (i.e. like in [3]). To avoid a large amount of computation during calculating $\mathbf{H}$ numerically the formula of BFGS for its approximation is used:

$$\mathbf{H}(k+1) = \mathbf{H}(k) + \frac{\mathbf{q}(k) \cdot \mathbf{q}^T(k)}{\mathbf{q}^T(k) \cdot \mathbf{s}(k)} - \frac{\mathbf{H}^T(k) \cdot \mathbf{s}^T(k) \cdot \mathbf{s}(k) \cdot \mathbf{H}(k)}{\mathbf{s}^T(k) \cdot \mathbf{H}(k) \cdot \mathbf{s}(k)}, \tag{11}$$

where: $\mathbf{H}(k=0)$ can be a set as the identity matrix $\mathbf{I}$.

$$\begin{aligned} \mathbf{s}(k) &= \boldsymbol{\Theta}(k+1) - \boldsymbol{\Theta}(k) \\ \mathbf{q}(k) &= \nabla f(k+1;\boldsymbol{\Theta}) - \nabla f(k;\boldsymbol{\Theta}) \end{aligned}. \tag{12}$$

On the other hand, for the LM method the direction can be calculated as follows:

$$\mathbf{d}(k) = -\left[\mathbf{J}^T(k;\boldsymbol{\Theta}) \cdot \mathbf{J}(k;\boldsymbol{\Theta}) + \lambda \cdot \mathbf{I} \right]^{-1} \cdot \nabla f(k;\boldsymbol{\Theta}), \tag{13}$$

where: the Jacobian information is derived using also a numerical differentiation method, and scalar λ controls both the magnitude and direction.

A very important thing is that the term $e_\tau \cdot \phi(k-\tau)$ at each *k-th algorithm step* has to be fixed (for both algorithms).

3.2. Quality measures

The quality coefficients of neural network models under consideration are given below:

- Mean Absolute Percentage Error (*MAPE* for short):

$$MAPE = \frac{100}{n} \cdot \sum_{n=1}^{N} \left| \frac{y(n) - \hat{y}(n)}{y_{\max} - y_{\min}} \right|, \tag{14}$$

- Thiel's statistic (U for short):

$$U = \sqrt{\frac{\sum_{n=1}^{N}[y(n)-\hat{y}(n)]^2}{\sum_{n=1}^{N}[y(n)-y(n-1)]^2}}, \quad (15)$$

- conforming with Obuchowicz [2] is measure (*J* for short):

$$J = \frac{\sum_{n=1}^{N}[y(n)-\hat{y}(n)]^2}{\sum_{n=1}^{N}[y(n)]^2}, \quad (16)$$

where: N is the number of patterns; $\hat{y}(n)$ is the network output.

These measures may be employed for different tasks (i.e. the first and second coefficients are meaningful for control systems whilst the last one could be used for comparing with those described in the literature).

4. EXPERIMENTAL RESULTS

In the following parts of the paper some examples of applications will be presented. The first and second subsections show the results obtained using models which have been determined in the Matlab environment like Simulink and some other toolboxes. The last example describes the application of discussed neural structures to data gathered from the real process.

4.1. Modeling of dynamic systems

Problem 1

Given input-output pairs *{u(k),y(k)}* generated by the non-linear system defined by (Narendra K, Parthasarathy K.):

$$f(x_1,x_2,x_3,x_4,x_5)=\frac{y(k-1)\cdot y(k-2)\cdot u(k)\cdot u(k-2)\cdot[u(k)-1]+u(k-1)}{1+y(k-2)^2+u(k)^2}$$

identify the topology of the network and its parameters.

For this task several different variants of dynamic and static neural structures were examined. Table 1 presents some results (one can compare different neural structures).

Table 1. Results obtained for feed-forward and locally recurrent neural networks

Type	No. Iter.	J	MAPE	U
	[-]	[-]	[%]	[-]
LM method				
ffbp	200	12e-05	2,76e-01	2,28e-01
ffbptd	200	8,35e-05	2,28e-01	1,19e-01
$iir^{(2,1)}arx^{(1,1)}$	200	9,12e-05	2,56e-01	2,11e-01
BFGS method				
ffbp	200	32e-04	1,47	1,19
ffbptd	200	22e-04	9,47e-01	7,8e-01
$iir^{(2,1)}arx^{(1,1)}$	200	35e-04	7,21e-01	8,1e-01

There are results for three-layered perceptron *ffbp* (the input layer has 16 neurons with hyperbolic activation function; 8 neurons in the hidden layer with hyperbolic function; 1 neuron in the output layer), for time delay feedforward network *ffbptd* with the same architecture as previous one but also with a delay block (0,1,2) in the input layer. The last one is dynamic neural structure $iir^{(2,1)}arx^{(1,1)}$ (discussed in Sec. 2, three-layered network 5-12-1 with dynamic neurons and hyperbolic activation function in the hidden layer). As we can see locally recurrent network usually requires much less neurons to obtain almost the same quality result as in the case of a simple feed-forward network with static neurons. However, we must remember that dynamic neuron model is described by much more parameters than straightforward neural unit.

In the training stage the input-output patterns were represented as follows:

$$\{(\mathbf{P}(k),\mathbf{T}(k))\mid k=1,\ldots,N\}$$
$$\mathbf{P}(k)=[u(k)\ \ u(k-1)\ \ u(k-2)\ \ y(k-1)\ \ y(k-2)]^T, \quad (17)$$
$$\mathbf{T}(k)=[y(k)]$$

and *u(k)* was initialized with random values (1000 samples with a uniform distribution in the range <-2; 1.2>). In the testing stage three different input signals were used: polyharmonic signal, hard-limit signal and pseudo-random binary sequence (in Table 1 there are errors for the first signal only).

4.2. Fault-tolerant application

Problem 2

Design a virtual sensor $\hat{L}_2$ that could serve as a plain fault detection and identification block (based on the non-linear analytical redundancy technique).

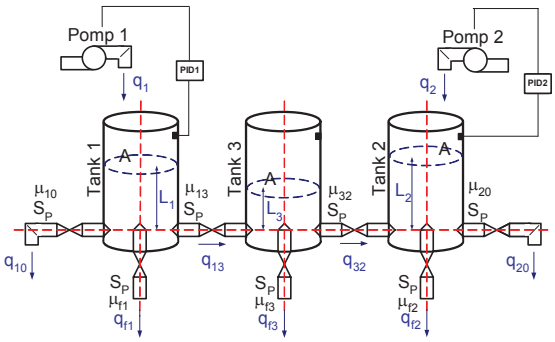

Fig. 5. The three-tank system

The three-tank system model as shown in figure 5 is written using well-known mass balance equations. The system can be represented like in [2] by:

$$\begin{cases} A\cdot\dfrac{\mathrm{d}[L_1(t)+\Delta L_1(t)]}{\mathrm{d}t}=q_1(t)-q_{13}(t)-q_{f1}(t)-q_{10}(t) \\ A\cdot\dfrac{\mathrm{d}[L_2(t)+\Delta L_2(t)]}{\mathrm{d}t}=q_2(t)+q_{32}(t)-q_{f2}(t)-q_{20}(t) \\ A\cdot\dfrac{\mathrm{d}[L_3(t)+\Delta L_3(t)]}{\mathrm{d}t}=q_{13}(t)-q_{32}(t)-q_{f3}(t) \end{cases}, \quad (18)$$

where: q_{ij} represents the water flow rate from tank *i* to *j* which is given by Torricelli's rule:

$$q_{ij}(t) = \mu_i \cdot S_p \cdot sign\left[L_i(t) - L_j(t) + \Delta L_{ij}(t)\right] \cdot \sqrt{2 \cdot g \cdot \left|L_i(t) - L_j(t) + \Delta L_{ij}(t)\right|}, \quad (19)$$

where: ΔL_i represents fault of the measuring channel i, q_{fi} is undesirable leakage from the tank i.

The measured signals are: streams of the medium q_1, q_2 that flow into the first and second tank; control signals U_1, U_2; levels in tanks L_1, L_2, L_3.

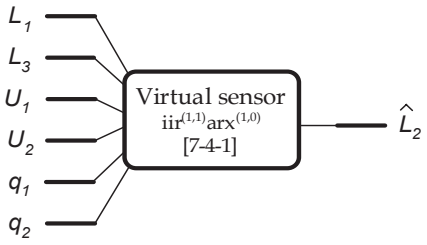

Fig. 6. Virtual sensor for fault detection and identification block

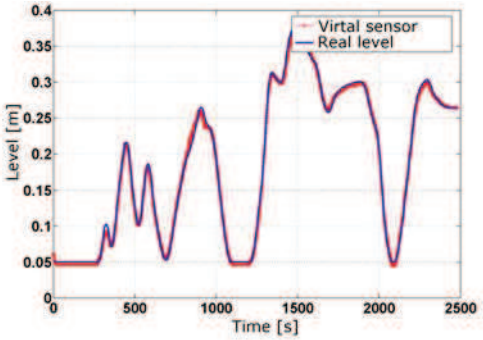

Fig. 7. Virtual and real sensor outputs (results for testing stage: *MAPE<5%)*

In order to create a virtual sensor the three-layered dynamic neural network was used (7 neurons with hyperbolic transfer function in the input layer, 4 hidden neurons with the $iir^{(1,1)}arx^{(1,0)}$ blocks and hyperbolic function, one output neuron with linear activation function, Fig. 6). The training and testing patterns were obtained by simulating model (Eq. 18) in the Matlab environment. The error (*MAPE*) was less than 5%, therefore it was possible to use the model in the fault detection and control modules what is shown in Figures 8 and 9.

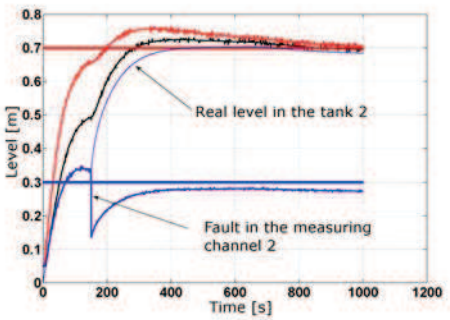

Fig. 8. Fault in the measuring channel 2 (without analytical redundancy)

The fault of the measuring channel no. 2 is detected by comparing value from the sensor L_2 with estimated value $\hat{L}_2$ obtains from the virtual sensor. When absolute difference between these values is greater than 10% of scale range then control system change its rules (Fig.9).

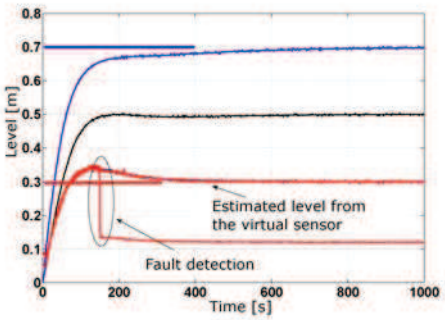

Fig. 9. Fault-tolerant control (with analytical redundancy)

4.3. Modeling of industrial furnace

Problem 3

The database concerning process of copper reduction from slag has been collected by SCADA system. There are more than 180 attributes collected: parameters of power supply, temperatures in many points of the furnace, and state parameters such as mass of charge, and its chemical analysis determined three times a cycle. In order to create a submodel of the analyzed process a non-linear discrete difference equation is proposed (Eq. 20, in the first stage of research inputs and output were selected based on information from staff maintaining this object and technical documentation dealing with this process):

$$\hat{y}_{Cu}(k+\Delta k) = \hat{f}\left[\mathbf{I}(k), \mathbf{L}(k), \hat{\mathbf{I}}(k), \hat{\mathbf{L}}(k), p(k), y_{Cu}(k)\right], \quad (20)$$

where: $\mathbf{I}(k)$ is a vector of three currents (for electrodes *R, S, T*), $\mathbf{L}(k)$ is a vector determines that positions of three electrodes; $p(k)$ is a position of a claw of the transformer; $y_{Cu}(k)$ is a current state parameter describing the copper concentration in slag; Δk is the time horizon of prediction (for this example it equals 5, 10, 15 or 20 min.); $\hat{f}$ represents a non-linear input-output relation of the neural network; $\hat{\mathbf{I}}(k), \hat{\mathbf{L}}(k)$ are suitably preprocessing currents and electrodes positions (detailed description was omitted in the paper).

In the training stage 8000 samples were used. The testing of the model was carried out using another set of 4000 samples. In Table 2 the results received for two neural networks are presented. A simple feed-forward with time delay lines network is presented *ffbptd* (12 neurons with time delay lines (0,1,2) and hyperbolic activation function in the input layer, 7 neurons in the hidden layer with hyperbolic activation function; one output neuron with linear transfer function). There are also results for dynamic neural network $fir^{(2)}arx^{(1,0)}$

(6 neurons with hyperbolic activation function in the input layer; 5 hidden neurons with the $fir^{(2)}$ filter and the $arx^{(1,0)}$ system and hyperbolic function; one output neuron with linear transfer function).

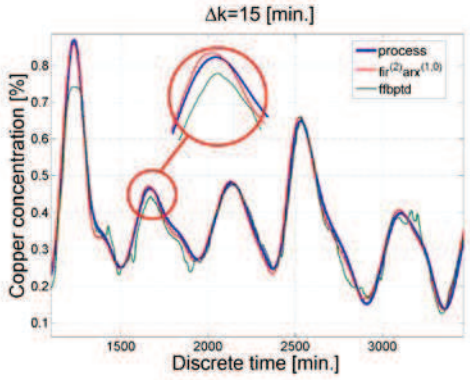

Fig. 10. Example of process and model outputs obtained for different neural structures

Table 2. Results obtained for locally recurrent neural networks

Type	Δk [min.]	J [-]	MAPE [%]	U [-]
LM method				
ffbptd	5	8,70e-04	1,17	1,58
$fir^{(2)}arx^{(1,0)}$	5	5,34e-04	0,25	0,50
ffbp	10	2,00e-03	1,79	1,13
$fir^{(2)}arx^{(1,0)}$	10	1,1e-03	0,91	0,55
ffbp	15	5,00e-03	2,09	1,14
$fir^{(2)}arx^{(1,0))}$	15	1,64e-03	1,20	0,49
ffbp	20	9,28e-03	3,82	1,14
$fir^{(2)}arx^{(1,0)}$	20	1,4e-03	0,90	0,26

As we can observe, all the results ($MAPE \approx 1\%$) for each example are quite similar to each other. However, in the case of the $fir^{(2)}arx^{(1,0)}$ network *Thiel's coefficients (U)* are lesser than in the *ffbptd* network. Consequently, these models may be useful in fault detection systems or even in control systems.

5. SUMMARY

This work demonstrates theoretical and practical aspects of heuristic modeling based on locally recurrent neural networks. As one can see it is possible to introduce into artificial neuron structure some modules which represent its dynamic and stochastic behaviour (it brings an artificial neuron closer and closer to the biological model). Moreover, some known local optimization algorithms may be used in order to train dynamic network networks. Nevertheless, these algorithms are still too slow for the reason that the gradient information is computed using a numerical differentiation method.

6. TO DO

Many problems are still open to solution, such as the automatic identification of dynamic neural topology. However, the most important tasks are to find the gradient information using an analytical method and to apply some global optimization algorithms (e.g. genetic algorithms or recursive random search) for preliminary adjusting network parameters.

ACKNOWLEDGMENTS

Paper presents a selected part of the research supervised by Prof. W. Moczulski. This research has been partially supported by the Ministry of Education and Scientific Research under grant No. 4 T07B 018 27.

REFERENCES

[1] Gobbak Anil K., Raghavendran H., Tapas Anand M.: *Internal Feedback Neuron Networks for Modeling of an Industrial Furnace*. The 1997 Ieee International Conference on Neural Networks. Volume: 2, On page(s): 700-705 vol.2. ISBN: 0-7803-4122-8.

[2] Korbicz J., Koscielny J. M., Kowalczuk Z., Cholewa W. (Eds.): *Fault diagnosis. Models, Artificial Intelligence, Applications*. Springer-Verlag Berlin Heidelberg 2004 New York. ISBN 3-540-40767-7.

[3] Majchrzak E., Mochnacki B.: *Metody numeryczne. Podstawy teoretyczne, aspekty praktyczne i algorytmy*. Wydawnictwo Politechniki Śląskiej, Gliwice 2004, ISBN 83-7335-231-7 (In Polish).

[4] Moczulski W.: *Methodology of Heuristic Modelling of Dynamic Objects and Processes for Diagnostics and Control.* Recent Developments in Artificial Intelligence Methods. AI-METH 2005, p. 123 – 126.

[5] Sinha N.K., Gupta M.M. and Rao D. H.: *Dynamic Neural Networks: An Overview*. Proceedings of IEEE International Conference on Industrial Technology 2000 (IEEE Cat. No.00TH8482). Volume: 1, On page(s): 491-496 vol.2. ISBN: 0-7803-5812-0.

[6] Przystałka P., Moczulski W.: *Koncepcja metodyki tworzenia heurystycznych modeli obiektów przemysłowych.*: *Pomiary Automatyka Kontrola,* 2005, Agencja Wydawnicza SIMP, ISSN 0032-4110, materiały VII Krajowej Konferencji Naukowo-Technicznej Diagnostyka Procesów Przemysłowych. Rajgród 2005, str. 136-138 (in Polish).

[7] Yazdizadeh, A.; Khorasani, K.: *Identification of a class of non-linear systems using dynamic neural network structures*. Proceedings of International Conference on Neural Networks (ICNN'97). Page(s): 194-198 vol.1. Digital Object Identifier 10.1109/ ICNN.1997.61162.