

# Adaptive Forecasting of Non-Stationary Nonlinear Time Series Based on the Evolving Weighted Neuro-Neo-Fuzzy-ANARX-Model

**Zhengbing Hu**

School of Educational Information Technology, Central China Normal University, Wuhan, China

E-mail: hzb@mail.ccnu.edu.cn

**Yevgeniy V. Bodyanskiy**

Kharkiv National University of Radio Electronics, Kharkiv, Ukraine

E-mail: yevgeniy.bodyanskiy@nure.ua

**Oleksii K. Tyshchenko and Olena O. Boiko**

Kharkiv National University of Radio Electronics, Kharkiv, Ukraine

E-mail: lehatish@gmail.com, olena.boiko@ukr.net

**Abstract**—An evolving weighted neuro-neo-fuzzy-ANARX model and its learning procedures are introduced in the article. This system is basically used for time series forecasting. It's based on neo-fuzzy elements. This system may be considered as a pool of elements that process data in a parallel manner. The proposed evolving system may provide online processing data streams.

**Index Terms**—Computational Intelligence, time series prediction, neuro-neo-fuzzy System, Machine Learning, ANARX, Data Stream.

## I. INTRODUCTION

Mathematical forecasting of data sequences (time series) is nowadays well studied and there is a large number of publications on this topic. There are many methods for solving this task: regression, correlation, spectral analysis, exponential smoothing, etc., and more advanced intellectual systems that require sometimes rather complicated mathematical methods and a high qualification of a user. The problem becomes more complicated when analyzed time series are both non-stationary and nonlinear and contain unknown behavior trends as well as quasiperiodic, stochastic and chaotic components. The best results have been shown by nonlinear forecasting models based on mathematical methods of Computational Intelligence [1-3] such as neuro-fuzzy systems [4-5] due to their approximating and extrapolating properties, learning abilities, results' transparency and interpretability. The models to be especially noted are the so-called NARX-models [6] which have the form

$$\hat{y}(k) = f(y(k-1), \dots, y(k-n_y), x(k-1), \dots, x(k-n_x)) \quad (1)$$

where  $\hat{y}(k)$  is an estimate of forecasted time series at discrete time  $k = 1, 2, \dots$ ;  $f(\bullet)$  stands for a certain nonlinear transformation implemented by a neuro-fuzzy system,  $x(k)$  is an observed exogenous factor that defines a behavior of  $y(k)$ . It can be noticed that popular Box-Jenkins AR-, ARX-, ARMAX-models as well as nonlinear NARMA-models can be described by the expression (1). These models have been widely studied; there are many architectures and learning algorithms that implement these models, but it is assumed that models' orders  $n_y$  and  $n_x$  are given a priori. These orders are previously unknown in a case of structural non-stationarity for analyzed time series, and they also have to be adjusted during a learning procedure. In this case, it makes sense to use evolving connectionist systems [7-10] that adjust not only their synaptic weights and activation-membership functions, but also their architectures. There are many algorithms that implement these learning methods both in a batch mode and in a sequential mode. The problem becomes more complicated if data are fed to the system with a high frequency in the form of a data stream [11]. Here, the most popular evolving systems turn out to be too cumbersome for learning and information processing in an online mode.

As an alternative, a rather simple and effective architecture can be considered. It's the so-called ANARX-model (Additive NARX) that has the form [12, 13]

$$\begin{aligned} \hat{y}(k) = & f_1(y(k-1), x(k-1)) + f_2(y(k-2), x(k-2)) + \dots \\ & \dots + f_n(y(k-n), x(k-n)) = \sum_{l=1}^n f_l(y(k-l), x(k-l)) \end{aligned} \quad (2)$$

(here  $n = \max\{n_y, n_x\}$ ), an original task of the forecasting system's synthesis is decomposed into many local tasks of parametric identification for node models with two input variables  $y(k-l)$ ,  $x(k-l)$ ,  $l = 1, 2, \dots, n, \dots$ .

Authors [12, 13] used elementary Rosenblatt perceptrons with sigmoidal activation functions as such nodes. The ANARX-model provided a high forecasting quality, but generally speaking it requires a large number of nodes in its architecture [14].

Some synthesis problems of forecasting neuro-fuzzy [15] and neo-fuzzy [15-17] systems based on the ANARX-models are considered in this work. These systems avoid the above mentioned drawbacks.

Since we consider a case of stochastic nonlinear dynamic signals in this article, the basic novelty has to do with defining a model's delay order in an online mode.

The remainder of this paper is organized as follows: Section 2 describes an architecture of a neuro-fuzzy-

ANARX-model. Section 3 describes an architecture of a neo-fuzzy-ANARX-model. Section 4 presents a weighted neuro-neo-fuzzy-ANARX-model. Section 5 presents several real-world applications to be solved with the help of the proposed system. Conclusions are given in the final section.

## II. A NEURO-FUZZY-ANARX-MODEL

An architecture of the ANARX-model is shown in Fig.1. It is formed by two lines of time delay elements  $z^{-1}$  ( $z^{-1}y(k) = y(k-1)$ ) and  $n$  nodes  $N^{[l]}$  which are simultaneously learned. These nodes are tuned independently from each other. By adding new nodes or removing unnecessary ones, it doesn't have any influence on other neurons, i.e. the evolving process for this system is implemented by changing a number of the nodes.

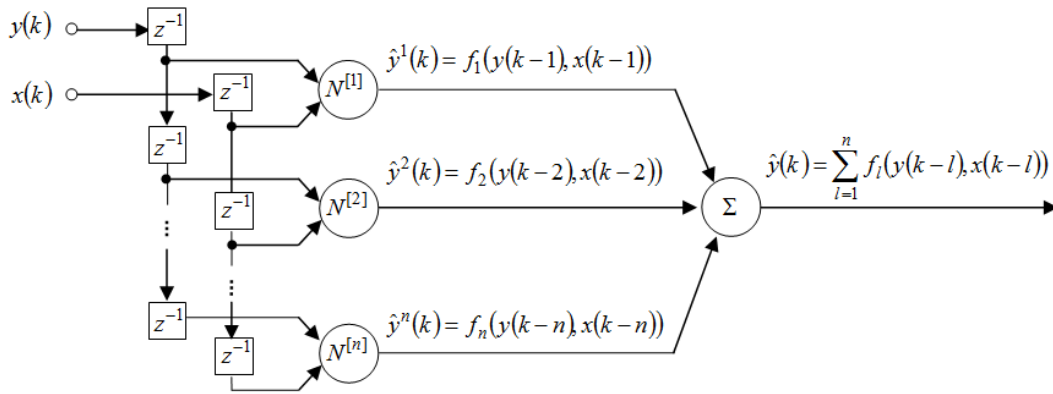


Fig.1. The ANARX-model.

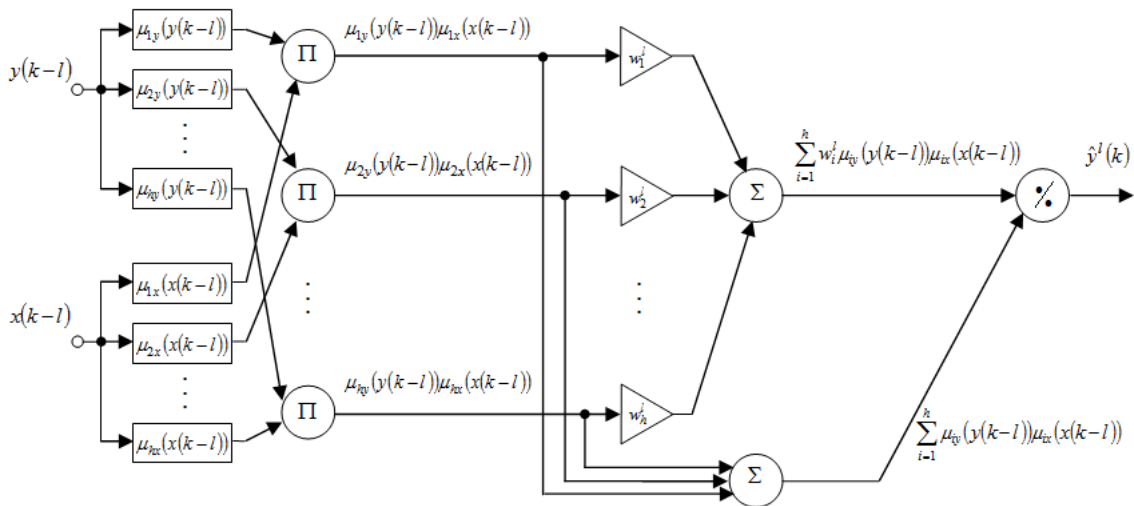


Fig.2. A neuro-fuzzy node of the ANARX-model.

It's recommended to use a neuron with two inputs (its architecture is shown in Fig.2) as a node of this system instead of the elementary Rosenblatt perceptron. As one

can see, this node is the Wang–Mendel neuro-fuzzy system [18] with two inputs. It possesses universal approximation capabilities and is actually the zero-order

Takagi–Sugeno–Kang system [19, 20]. A two-dimensional vector of input signals  $z_l(k) = (y(k-l), x(k-l))^T$  is fed to an input of the node  $N^{(l)}$ ,  $l = 1, 2, \dots, n, \dots$ . The first layer contains  $2h$  membership functions  $\mu_{iy}(y(k-l))$ ,  $\mu_{ix}(x(k-l))$  and fulfills fuzzification of the input variables by calculating membership levels  $0 < \mu_{iy}(y(k-l)) \leq 1$ ,  $0 < \mu_{ix}(x(k-l)) \leq 1$ . The Gaussian functions can be used as membership functions

$$\mu_{iy}(y(k-l)) = \exp\left(-\frac{(y(k-l) - c_{iy})^2}{2\sigma_{iy}^2}\right),$$

$$\mu_{ix}(x(k-l)) = \exp\left(-\frac{(x(k-l) - c_{ix})^2}{2\sigma_{ix}^2}\right)$$

(here  $c_{iy}$  and  $c_{ix}$  stand for parameters that define centers of these membership functions;  $\sigma_{iy}$  and  $\sigma_{ix}$  are width parameters) or other bell-shaped functions with an infinite support for avoiding “gaps” which appear in the fuzzified space.

The second layer of the node provides aggregation of the membership levels which were computed in the first layer. Outputs of the second layer form  $h$  aggregated signals

$$\tilde{z}_i^l(k) = \mu_{iy}(y(k-l))\mu_{ix}(x(k-l)).$$

The third layer contains synaptic weights that are adjusted during a learning procedure. Outputs of the third layer are values

$$w_i^l \mu_{iy}(y(k-l))\mu_{ix}(x(k-l)) = w_i^l \tilde{z}_i^l(k).$$

The fourth layer is formed by two summation units and calculates sums of the output signals in the second and the third layers correspondingly. Outputs of the fourth layer are signals

$$\begin{cases} \sum_{i=1}^h w_i^l \mu_{iy}(y(k-l))\mu_{ix}(x(k-l)) = \sum_{i=1}^h w_i^l \tilde{z}_i^l(k), \\ \sum_{i=1}^h \mu_{iy}(y(k-l))\mu_{ix}(x(k-l)) = \sum_{i=1}^h \tilde{z}_i^l(k). \end{cases}$$

Defuzzification (normalization) is implemented in the fifth (output) layer. Finally, the output signal of the node  $\hat{y}^l(k) = f_l(y(k-l), x(k-l))$  is computed:

$$\begin{aligned} \hat{y}^l(k) &= \frac{\sum_{i=1}^h w_i^l \mu_{iy}(y(k-l))\mu_{ix}(x(k-l))}{\sum_{i=1}^h \mu_{iy}(y(k-l))\mu_{ix}(x(k-l))} = \\ &= \frac{\sum_{i=1}^h w_i^l \tilde{z}_i^l(k)}{\sum_{i=1}^h \tilde{z}_i^l(k)} = \sum_{i=1}^h w_i^l \phi_i^l(k) = w^{lT} \phi^l(k) \end{aligned}$$

where  $\phi^l(k) = (\phi_1^l(k), \phi_2^l(k), \dots, \phi_h^l(k))^T$ ,  $\phi_i^l(k) = \tilde{z}_i^l(k) \left( \sum_{i=1}^h \tilde{z}_i^l(k) \right)^{-1}$ ,  $w^l = (w_1^l, w_2^l, \dots, w_h^l)^T$ .

Considering that the output signal  $\hat{y}^l(k)$  of each node depends linearly on the adjusted synaptic weights  $w_i^l$ , one can use conventional algorithms of adaptive linear identification [21] for their tuning which are based on the quadratic learning criterion.

If a training data set is non-stationary [22], one can use either the exponentially weighted recurrent least squares method

$$\begin{cases} w^l(k) = w^l(k-1) + \\ \frac{P^l(k-1)(y(k) - w^{lT}(k-1)\phi^l(k))\phi^l(k)}{\alpha + \phi^{lT}(k)P^l(k-1)\phi^l(k)}, \\ P^l(k) = P^l(k-1) - \\ \frac{P^l(k-1)\phi^l(k)\phi^{lT}(k)P^l(k-1)}{\alpha + \phi^{lT}(k)P^l(k-1)\phi^l(k)}, 0 < \alpha \leq 1, \end{cases} \quad (3)$$

or the Kaczmarz–Widrow–Hoff optimal gradient algorithm (in a case of the “rapid” non-stationarity)

$$w^l(k) = w^l(k-1) + \frac{y(k) - w^{lT}(k-1)\phi^l(k)}{\phi^{lT}(k)\phi^l(k)} \phi^l(k). \quad (4)$$

In fact, it is possible to tune  $4h$  membership functions’ parameters  $c_{iy}$ ,  $c_{ix}$ ,  $\sigma_{iy}$ ,  $\sigma_{ix}$  of each node, but taking into consideration the fact that the signal  $\hat{y}^l(k)$  depends nonlinearly on these parameters, a learning speed can’t be sufficient for non-stationary conditions in this case.

### III. A NEO-FUZZY-ANARX-MODEL

If a large data set to be processed is given (within the “Big Data” conception [23]) when a data processing speed and computational simplicity come to the forefront, it seems reasonable to use neo-fuzzy neurons that were proposed by T. Yamakawa and his co-authors [15-17]

instead of the neuro-fuzzy nodes in the ANARX-model. An architecture of the neo-fuzzy neuron as a node of the ANARX-model is shown in Fig.3. The neo-fuzzy neuron's advantages are a high learning speed, computational simplicity, good approximating properties and abilities to find a global minimum of a learning criterion in an online mode.

Structural elements of the neo-fuzzy neuron are nonlinear synapses  $NS_y$ ,  $NS_x$  that implement the zero order Takagi-Sugeno fuzzy inference, but it's easy to notice that the neo-fuzzy neuron is much simpler constructively than the neuro-fuzzy node shown in Fig.2.

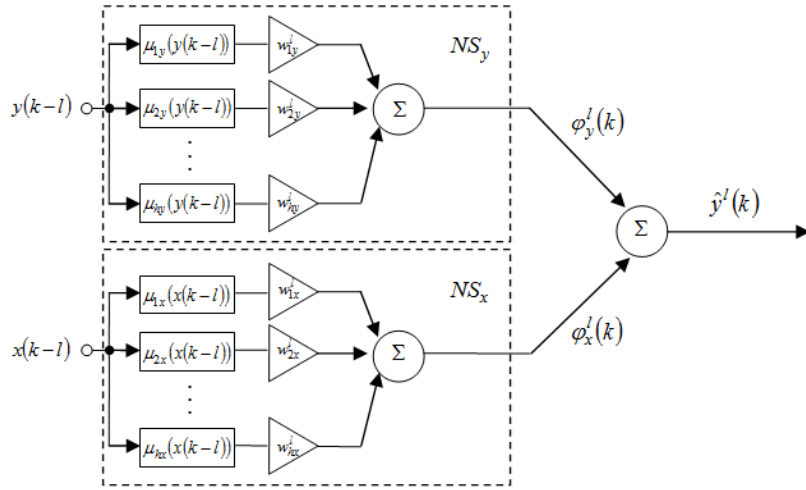


Fig.3. A neo-fuzzy node for the ANARX-model.

An output value of this node is formed with the help of input signals  $y(k-l)$ ,  $x(k-l)$

$$\hat{y}^j(k) = \varphi_y^j(k) + \varphi_x^j(k) = \sum_{i=1}^h w_{iy}^j \mu_{iy}(y(k-l)) + \sum_{i=1}^h w_{ix}^j \mu_{ix}(x(k-l)) \quad (5)$$

and an output signal of the ANARX-model can be written down in the form

$$\hat{y}(k) = \sum_{l=1}^n \left( \sum_{i=1}^h w_{iy}^l \mu_{iy}(y(k-l)) + \sum_{i=1}^h w_{ix}^l \mu_{ix}(x(k-l)) \right),$$

which means that since the neo-fuzzy neuron is also an additive model [24], the ANARX-model based on the neo-fuzzy neurons is twice additive.

Triangular membership functions are usually used in the neo-fuzzy neuron. They meet the unity partitioning conditions

$$\sum_{i=1}^h \mu_{iy}(y(k-l)) = 1, \quad \sum_{i=1}^h \mu_{ix}(x(k-l)) = 1,$$

which make it possible to simplify the node's architecture by excluding the normalization layer from it.

It was proposed to use B-splines as membership functions for the neo-fuzzy neuron [25]. They provide a higher approximation quality and also meet the unity partitioning conditions. A B-spline of the  $q$ -th order can be written down:

$$\mu_{iy}^q(y(k-l)) = \begin{cases} 1, & \text{if } c_{iy} \leq y(k-l) < c_{i+1,y}, \\ 0 & \text{otherwise} \end{cases} \quad \text{for } q = 1$$

$$\mu_{iy}^q(y(k-l)) = \begin{cases} \frac{y(k-l) - c_{iy}}{c_{i+q-1,y} - c_{iy}} \mu_{iy}^{q-1}(y(k-l)) + \\ \frac{c_{i+q,y} - y(k-l)}{c_{i+q,y} - c_{i+1,y}} \mu_{i+1,y}^{q-1}(y(k-l)), & \text{for } q > 1 \end{cases}$$

$$i = 1, \dots, h - q,$$

$$\mu_{ix}^q(x(k-l)) = \begin{cases} 1, \text{ if } c_{ix} \leq x(k-l) < c_{i+1,x}, \\ 0 \text{ otherwise} \end{cases} \text{ for } q = 1$$

$$\mu_{ix}^q(x(k-l)) = \begin{cases} \frac{x(k-l) - c_{ix}}{c_{i+q-1,x} - c_{ix}} \mu_{ix}^{q-1}(x(k-l)) + \\ \frac{c_{i+q,x} - x(k-l)}{c_{i+q,x} - c_{i+1,x}} \mu_{i+1,x}^{q-1}(x(k-l)), \text{ for } q > 1 \end{cases}$$

$$i = 1, \dots, h - q.$$

It should be mentioned that B-splines are a sort of generalized membership functions: when  $q = 2$  one gets traditional triangular membership functions; when  $q = 4$  one gets cubic splines, etc.

Introducing a vector of variables  $\varphi^l(k) = (\mu_{1y}(y(k-l)), \dots, \mu_{hy}(y(k-l)), \mu_{1x}(x(k-l)), \dots, \mu_{hx}(x(k-l)))^T$ ,  $w^l = (w_{1y}^l, \dots, w_{hy}^l, w_{1x}^l, \dots, w_{hx}^l)^T$ , the expression (5) can be rewritten in the form

$$\hat{y}^l(k) = w^{lT} \varphi^l(k).$$

Either the algorithms (3), (4) or the procedure [26]

$$\begin{cases} w^l(k) = w^l(k-1) + r_l^{-1}(k)(y(k) - w^{lT}(k-1)\varphi^l(k))\varphi^l(k) \\ r_l(k) = \alpha r_l(k-1) + \varphi^{lT}(k)\varphi^l(k), 0 \leq \alpha \leq 1, \end{cases} \quad (6)$$

can be used for the neo-fuzzy neuron's learning. This procedure possesses both filtering and tracking properties. It should also be noticed that when  $\alpha = 1$  the equation (6) coincides completely with the Kaczmarz–Widrow–Hoff optimal algorithm (4).

Evolving systems based on the neo-fuzzy neurons demonstrated its effectiveness for solving different tasks (especially forecasting tasks [25-30]). The twice additive system is the most appropriate choice for data processing in Data Stream Mining tasks [11] from a view point of implementation simplicity and a processing speed.

#### IV. A WEIGHTED NEURO-NEO-FUZZY-ANARX-MODEL

Considering that every node  $N^{[l]}$  of the ANARX-model is tuned independently from each other and represents an individual neuro(neo)-fuzzy system, it is possible to use a combination of neural networks' ensembles [31] in order to improve a forecasting quality. This approach results in an architecture of a weighted neuro-neo-fuzzy-ANARX-model (Fig.4).

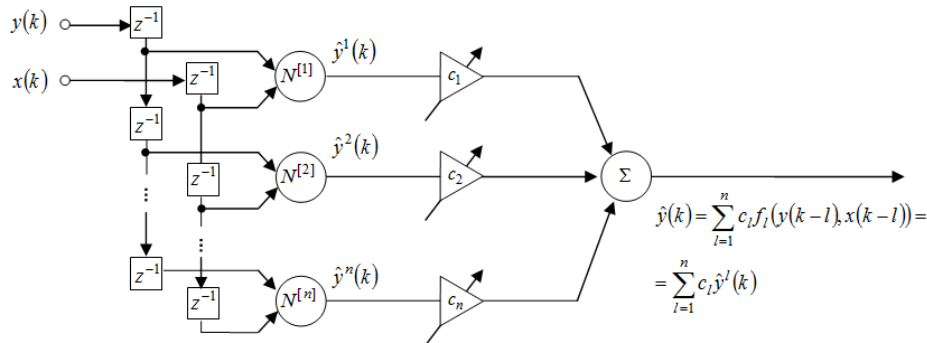


Fig.4. The weighted ANARX-model

An output signal of the system can be written in the form

$$\hat{y}(k) = \sum_{l=1}^n c_l \hat{y}^l(k) = c^T \hat{y}(k)$$

where  $\hat{y}(k) = (\hat{y}^1(k), \hat{y}^2(k), \dots, \hat{y}^n(k))^T$ ,  $c = (c_1, c_2, \dots, c_n)^T$  is a vector of adjusted weight coefficients that define proximity of the signals  $\hat{y}^l(k)$  to the forecasted process  $y(k)$  and meet the unbiasedness condition

$$\sum_{l=1}^n c_l = c^T I_n = 1$$

where  $I_n$  is a  $(n \times 1)$ -vector of unities.

The method of undetermined Lagrange multipliers can be used for finding the vector  $c$  in a batch mode. So, a sequence of forecasting errors

$$\begin{aligned} v(k) &= y(k) - \hat{y}(k) = y(k) - c^T \hat{y}(k) = c^T I_n y(k) - c^T \hat{y}(k) = \\ &= c^T (I_n y(k) - \hat{y}(k)) = c^T V(k), \end{aligned}$$

the Lagrange function

$$\begin{aligned} L(c, \lambda) &= \sum_k c^T V(k) V^T(k) c + \lambda (c^T I_n - 1) = \\ &= c^T R c + \lambda (c^T I_n - 1) \end{aligned} \quad (7)$$

(here  $\lambda$  is an undetermined Lagrange multiplier,  $R = \sum_k V(k) V^T(k)$  stands for an error correlation matrix) and the Karush–Kuhn–Tucker system of equations

$$\begin{cases} \nabla_c L(c, \lambda) = 2Rc + \lambda I_n = \vec{0}, \\ \frac{\partial L}{\partial \lambda} = c^T I_n - 1 = 0 \end{cases} \quad (8)$$

are introduced. A solution of the system (8) can be written down in the form

$$\begin{cases} c = R^{-1} I_n (I_n^T R^{-1} I_n)^{-1}, \\ \lambda = -2 I_n^T R^{-1} I_n \end{cases} \quad (9)$$

where a value of the Lagrange function (7) at the saddle point is

$$L^*(c, \lambda) = (I_n^T R^{-1} I_n)^{-1}.$$

An implementation procedure of the algorithm (9) can meet some problems in a case of data processing in an online mode and a high level of correlation between the signals  $\hat{y}^l(k)$  that leads to the ill-conditioned matrix  $R$  which has to be inverted at every time step  $k$ .

The Lagrange function (7) can be written down in the form

$$L(c, \lambda) = \sum_k (y(k) - c^T \hat{y}(k))^2 + \lambda (c^T I_n - 1)$$

and a gradient algorithm for finding its saddle point based on the Arrow–Hurwicz procedure [27, 32] can be written in the form

$$\begin{cases} c(k) = c(k-1) - \eta_c(k) \nabla_c L(c, \lambda), \\ \lambda(k) = \lambda(k-1) + \eta_\lambda(k) \frac{\partial L(c, \lambda)}{\partial \lambda} \end{cases}$$

or

$$\begin{cases} c(k) = c(k-1) + \eta_c(k) \times \\ \times \left( 2(y(k) - c^T(k-1) \hat{y}(k)) \hat{y}(k) - \lambda(k-1) I_n \right) = \\ = c(k-1) + \eta_c(k) \left( 2v(k) \hat{y}(k) - \lambda(k-1) I_n \right), \\ \lambda(k) = \lambda(k-1) + \eta_\lambda(k) (c^T(k) I_n - 1) \end{cases} \quad (10)$$

where  $\eta_c(k)$  and  $\eta_\lambda(k)$  are learning rate parameters.

The Arrow–Hurwicz procedure converges to the saddle point with rather general assumptions on the values  $\eta_c(k)$ ,  $\eta_\lambda(k)$ , but these parameters can be optimized to speed up the learning process as that is particularly important in Data Stream Mining tasks.

That's why the first ratio in the equation (10) should be multiplied by  $\hat{y}^T(k)$

$$\begin{aligned} \hat{y}^T(k) c(k) &= \hat{y}^T(k) c(k-1) + \\ &+ \eta_c(k) \left( 2v(k) \|\hat{y}(k)\|^2 - \lambda(k-1) \hat{y}^T I_n \right) \end{aligned}$$

and an additional function that characterizes a criterial convergence is introduced

$$\begin{aligned} (y(k) - \hat{y}^T(k) c(k))^2 &= v^2(k) - \\ - 2\eta_c(k) v(k) \left( 2v(k) \|\hat{y}(k)\|^2 - \lambda(k-1) \hat{y}^T I_n \right) + \\ + \eta_c^2(k) \left( 2v(k) \|\hat{y}(k)\|^2 - \lambda(k-1) \hat{y}^T I_n \right)^2 \end{aligned}$$

A solution for the differential equation

$$\begin{aligned} \frac{\partial (y(k) - \hat{y}^T(k) c(k))^2}{\partial \eta_c(k)} &= \\ - 2v(k) \left( 2v(k) \|\hat{y}(k)\|^2 - \lambda(k-1) \hat{y}^T I_n \right) + \\ + 2\eta_c(k) \left( 2v(k) \|\hat{y}(k)\|^2 - \lambda(k-1) \hat{y}^T I_n \right)^2 &= 0 \end{aligned}$$

gives the optimal learning rate values  $\eta_c(k)$  in the form

$$\eta_c(k) = \frac{v(k)}{2v(k)\|\hat{y}(k)\|^2 - \lambda(k-1)\hat{y}^T I_n}$$

Applying this value to the expression (10), final equations can be written down

$$\begin{cases} c(k) = c(k-1) + \frac{v(k)(2v(k)\hat{y}(k) - \lambda(k-1)I_n)}{2v(k)\|\hat{y}(k)\|^2 - \lambda(k-1)\hat{y}^T I_n}, \\ \lambda(k) = \lambda(k-1) + \eta_\lambda(k)(c^T(k)I_n - 1). \end{cases} \quad (11)$$

It's easy to notice that when  $\lambda(k-1) = 0$  the procedure (11) coincides with the Kaczmarz–Widrow–Hoff algorithm (4).

### V. EXPERIMENTS

In order to prove the effectiveness of the proposed system, a number of experiments should be carried out.

#### A. Electricity Demand

This data set describes 15 minutes averaged values of power demand in 1997 (a full year). Generally speaking, this data set contains 15000 points but we took only 5000 points for the experiment. 3000 points were selected for a training stage and 2000 points were used for testing. Prediction results of the ANARX and weighted ANARX systems are in Fig.5 and Fig.6 correspondingly (signal values are marked with a blue color; prediction values are marked with a magenta color; and prediction errors are marked with a grey line).

We used for comparison multilayer perceptrons (MLPs), radial-basis function neural networks (RBFNs), ANFIS and two proposed systems ANARX and weighted ANARX (both based on neo-fuzzy nodes). The comparison of the systems' results is shown in Table 1. Since MLP can't work in an online mode, it was processing data in two different modes. It had just one epoch in the first case (something similar to an online case), and it had 5 epochs in the second case (the MLP architecture had almost the same number of adjustable parameters when compared to the proposed systems). A number of MLP inputs was equal to 4 and a number of hidden nodes was equal to 7 in both cases. A total number of parameters to be tuned was 43 in both modes. It took about two times more time to compute a result in the second case but a prediction quality was like almost two times higher. MLP (case 2) demonstrated the best result in this experiment.

Speaking of RBFNs, there are two cases. The first-case RBFN was taken really close in the sense of parameters' number to our systems and the second-case RBFN's architecture was chosen to show the best performance. In the first case, RBFN had 3 inputs and 7 kernel functions. In the second case, it had 3 inputs as well but 12 kernel

functions which generally led to the higher prediction quality (+30% precision compared to RBFN in the first case) but took longer to compute the result. A number of parameters to be tuned was 36 in the first case and 61 in the second case.

ANFIS showed one of the best prediction qualities in this experiment. It had 4 inputs, 55 nodes and it was processing data during 5 epochs. It contained 80 parameters to be tuned.

Table 1. Comparison of the systems' results

Systems	Parameters to be tuned	RMSE (training)	RMSE (test)	Time, s
MLP (case 1)	43	0.0600	0.0700	0.4063
MLP (case 2)	43	0.0245	0.0381	0.9219
RBFN (case 1)	36	0.0681	0.0832	0.6562
RBFN (case 2)	61	0.0463	0.0604	1.0250
ANFIS	80	0.0237	0.0396	0.7031
ANARX (neo-fuzzy nodes)	37	0.0903	0.0922	0.4300
Weighted ANARX (neo-fuzzy nodes)	36	0.0427	0.0573	0.3750

The proposed ANARX system based on the neo-fuzzy nodes had 2 inputs, 2 nodes, 9 membership functions, and its  $\alpha$  parameter was equal to 0.62. This system had 37 parameters. Its prediction quality was rather high and it was definitely one of the fastest system on this data set. We should also notice that we used B-splines (q=2, which means that we used triangular membership functions) as membership functions for the proposed systems.

The proposed weighted ANARX system based on the neo-fuzzy nodes had 2 inputs, 2 nodes, 8 membership functions; its  $\alpha$  parameter was equal to 0.9. It had 37 adjustable parameters. This system demonstrated better performance when compared to ANARX and the fastest results.

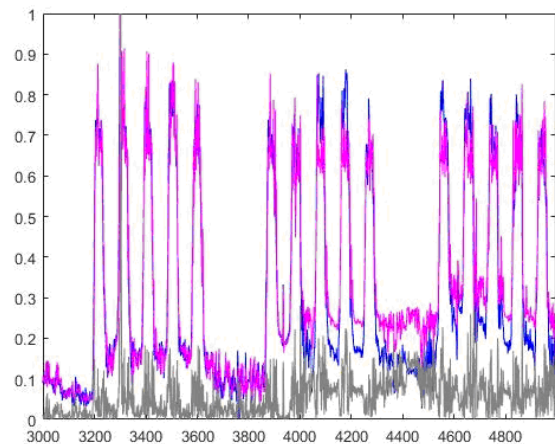


Fig.5. Identification of a nonlinear system. Prediction results of the ANARX system.



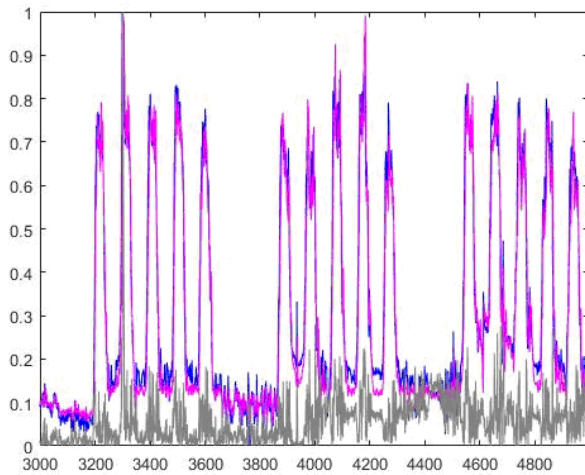


Fig.6. Identification of a nonlinear system. Prediction results of the weighted ANARX system.

### B. Monthly sunspot number

This data set was taken from datamarket.com. The data set describes monthly sunspot number in Zurich. It was collected between 1749 and 1983. It contains 2820 points. 2256 points were selected for a training stage and 564 points were used for testing. Prediction results of the ANARX and weighted ANARX systems are in Fig.7 and Fig.8 correspondingly (signal values are marked with a blue color; prediction values are marked with a magenta color; and prediction errors are marked with a grey line).

We used a set of systems which is similar to the previous experiment to compare results. The comparison of the systems' results is shown in Table 2. MLP\_1 (MLP for the first case) had 3 inputs, 6 hidden nodes and it was processing data during just one epoch. MLP\_2 basically had the same parameter settings except the fact that it was processing data during 5 epochs. Both MLP systems had 31 parameters to be tuned. MLP\_1 was 2 times faster than MLP\_2 but its prediction quality was also almost 2 times worse.

Let's denote our RBFN architectures as RBFN\_1 (RBFN for the first case) and RBFN\_2 (RBFN for the second case). Both of them had 3 inputs. RBFN\_1 had 4 kernel functions unlike RBFN\_2 which had 19 hidden nodes. A number of parameters to be tuned was 21 in the first case and 96 in the second case. RBFN\_2 showed a better prediction quality but it was much slower than RBFN\_1.

ANFIS had 4 inputs and 55 nodes. It was processing data during 3 epochs. It had 80 adjustable parameters.

The proposed ANARX system based on the neo-fuzzy nodes had 2 inputs, 2 nodes, 4 membership functions, and its  $\alpha$  parameter was equal to 0.9. This system contained 17 parameters. Its prediction quality was rather high and it was definitely one of the fastest systems on this data set.

The proposed weighted ANARX system based on the neo-fuzzy nodes had 2 inputs, 2 nodes, 4 membership functions; its  $\alpha$  parameter was equal to 0.9. This system contained 20 parameters to be tuned. This system demonstrated better performance when compared to

ANARX and one of the fastest results.

Table 2. Comparison of the systems' results

Systems	Parameters to be tuned	RMSE (training)	RMSE (test)	Time, s
MLP (case 1)	31	0.1058	0.1407	0.2813
MLP (case 2)	31	0.0600	0.0808	0.5938
RBFN (case 1)	21	0.1066	0.2155	0.2219
RBFN (case 2)	96	0.0702	0.1638	0.9156
ANFIS	80	0.0559	0.1965	0.8906
ANARX (neo-fuzzy nodes)	17	0.1297	0.1350	0.2252
Weighted ANARX (neo-fuzzy nodes)	20	0.0784	0.1081	0.2981

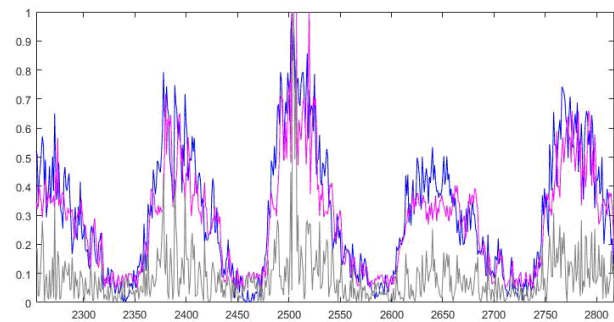


Fig.7. Identification of a nonlinear system. Prediction results of the ANARX system.

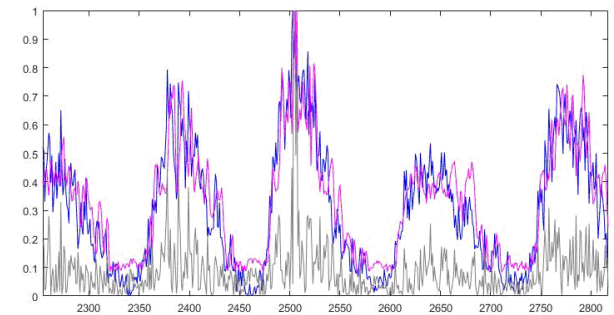


Fig.8. Identification of a nonlinear system. Prediction results of the weighted ANARX system.

## VI. CONCLUSION

The evolving forecasting weighted neuro-neo-fuzzy-twice additive model and its learning procedures are proposed in the paper. This system can be used for non-stationary nonlinear stochastic and chaotic time series' forecasting where time series are processed in an online mode under the parametric and structural uncertainty. The proposed weighted ANARX-model is rather simple from a computational point of view and provides fast data stream processing in an online mode.

So, the proposed evolving forecasting model has



demonstrated its efficiency for solving real-world tasks. A number of experiments has been performed to show high efficiency of the proposed neuro-neo-fuzzy system.

#### ACKNOWLEDGMENT

The authors would like to thank anonymous reviewers for their careful reading of this paper and for their helpful comments.

This scientific work was supported by RAMECS and CCNU16A02015.

#### REFERENCES

- [1] Rutkowski L. Computational Intelligence. Methods and Techniques. Springer-Verlag, Berlin-Heidelberg, 2008.
- [2] Kruse R, Borgelt C, Klawonn F, Moewes C, Steinbrecher M, Held P. Computational Intelligence. Springer, Berlin, 2013.
- [3] Du K-L, Swamy M N S. Neural Networks and Statistical Learning. Springer-Verlag, London, 2014.
- [4] Jang J-S, Sun C-T, Mizutani E. Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence. Prentice Hall, Upper Saddle River, 1997.
- [5] Rezaei A, Noori L, Taghipour M. The Use of ANFIS and RBF to Model and Predict the Inhibitory Concentration Values Determined by MTT Assay on Cancer Cell Lines. International Journal of Information Technology and Computer Science(IJITCS), 2016, 8(4): 28-34.
- [6] Nelles O. Nonlinear System Identification. Springer, Berlin, 2001.
- [7] Kasabov N. Evolving fuzzy neural networks – algorithms, applications and biological motivation. Proc. “Methodologies for the Conception, Design and Application of Soft Computing”, Singapore, 1998:271-274.
- [8] Kasabov N. Evolving fuzzy neural networks: theory and applications for on-line adaptive prediction, decision making and control. Australian J. of Intelligent Information Processing Systems, 1998, 5(3):154-160.
- [9] Kasabov N. Evolving Connectionist Systems. Springer-Verlag, London, 2003.
- [10] Lughofer E. Evolving Fuzzy Systems – Methodologies, Advanced Concepts and Applications. Springer, Berlin, 2011.
- [11] Bifet A. Adaptive Stream Mining: Pattern Learning and Mining from Evolving Data Streams. IOS Press, 2010, Amsterdam.
- [12] Belikov J, Vassiljeva K, Petlenkov E, Nömm S. A novel Taylor series based approach for control computation in NN-ANARX structure based control of nonlinear systems. Proc. 27th Chinese Control Conference, Kunming, China, 2008:474-478.
- [13] Vassiljeva K, Petlenkov E, Belikov J. State-space control of nonlinear systems identified by ANARX and neural network based SANARX models. Proc. WCCI 2010 IEEE World Congress on Computational Intelligence, Barcelona, Spain, 2010:3816-3823.
- [14] Cybenko G. Approximation by superpositions of a sigmoidal function. Math. Control Signals Systems, 1989, 2:303-314.
- [15] Yamakawa T, Uchino E, Miki T, Kusanagi H. A neo fuzzy neuron and its applications to system identification and prediction of the system behavior. Proc. 2nd Int. Conf. on Fuzzy Logic and Neural Networks “IIZUKA-92”, Iizuka, Japan, 1992:477-483.
- [16] Uchino E, Yamakawa T. Soft computing based signal prediction, restoration, and filtering. In: Intelligent Hybrid Systems: Fuzzy Logic, Neural Networks, and Genetic Algorithms, 1997:331-349.
- [17] Miki T, Yamakawa T. Analog implementation of neo-fuzzy neuron and its on-board learning. In: Computational Intelligence and Applications, 1999:144-149.
- [18] Wang L-X, Mendel J M. Fuzzy basis functions, universal approximation, and orthogonal least-squares learning. IEEE Trans. on Neural Networks, 1992, 3(5):807-814.
- [19] Takagi T, Sugeno M. Fuzzy identification of systems and its applications to modeling and control. IEEE Trans. on Systems, Man, and Cybernetics, 1985, 15:116-132.
- [20] Sugeno M, Kang G T. Structure identification of fuzzy model. Fuzzy Sets and Systems, 1998, 28:15-33.
- [21] Ljung L. System Identification: Theory for the User. Prentice Hall, Inc., Upper Saddle River, 1987.
- [22] Polikar R, Alippi C. Learning in nonstationary and evolving environments. IEEE Trans. on Neural Networks and Learning Systems, 2014, 25(1):9-11.
- [23] Jin Y, Hammer B. Computational Intelligence in Big Data. IEEE Computational Intelligence Magazine, 2014, 9(3):12-13.
- [24] Friedman J, Hastie T, Tibshirani R. The Elements of Statistical Learning. Data Mining, Inference and Prediction. Springer, Berlin, 2003.
- [25] Bodyanskiy Ye, Kolodyazhniy V. Cascaded multiresolution spline-based fuzzy neural network. Proc. Int. Symp. on Evolving Intelligent Systems, Leicester, UK, 2010:26-29.
- [26] Bodyanskiy Ye, Kokshenev I, Kolodyazhniy V. An adaptive learning algorithm for a neo-fuzzy neuron. Proc. 3rd Int. Conf. of European Union Soc. for Fuzzy Logic and Technology (EUSFLAT'03), Zittau, Germany, 2003:375-379.
- [27] Bodyanskiy Ye, Otto P, Pliss I, Popov S. An optimal algorithm for combining multivariate forecasts in hybrid systems. Lecture Notes in Artificial Intelligence, 2003, 2774:967-973.
- [28] Bodyanskiy Ye, Viktorov Ye. The cascade neo-fuzzy architecture using cubic-spline activation functions. Int. J. Information Theories and Applications, 2009, 16(3):245-259.
- [29] Bodyanskiy Ye, Teslenko N, Grimm P. Hybrid evolving neural network using kernel activation functions. Proc. 17th Zittau East-West Fuzzy Colloquium, Zittau/Goerlitz, Germany, 2010:39-46.
- [30] Bodyanskiy Ye V, Tyshchenko O K, Kopalani D S. A multidimensional cascade neuro-fuzzy system with neuron pool optimization in each cascade. Int. J. Information Technology and Computer Science, 2014, 6(8):11-17.
- [31] Sharkey A J C. On combining artificial neural nets. Connection Science, 1996, 8:299-313.
- [32] Bodyanskiy Ye, Deineko A, Stolnikova M. Adaptive generalization of neuro-fuzzy systems ensemble. Proc. of the Int. Conf. “Computer Science and Information Technologies”, Lviv, Ukraine, November 16-19, 2011:13-14.

## Authors' Profiles



**Zhengbing Hu:** Ph.D., Associate Professor of School of Educational Information Technology, Central China Normal University, M.Sc. (2002), Ph.D. (2006) from the National Technical University of Ukraine "Kiev Polytechnic Institute". Postdoc (2008), Huazhong University of Science and Technology, China. Honorary Associate Researcher (2012), Hong Kong University, Hong Kong. Major interests: Computer Science and Technology Applications, Artificial Intelligence, Network Security, Communications. Data Processing, Cloud Computing, Education Technology.

Based on the Evolving Weighted Neuro-Neo-Fuzzy-ANARX-Model", International Journal of Information Technology and Computer Science (IJITCS), Vol.8, No.10, pp. 1-10, 2016. DOI: 10.5815/ijitcs.2016.10.01



**Yevgeniy V. Bodyanskiy** graduated from Kharkiv National University of Radio Electronics in 1971. He got his PhD in 1980. He obtained an academic title of the Senior Researcher in 1984. He got his Dr.Habil.Sci.Eng. in 1990. He obtained an academic title of the Professor in 1994.

Prof. Bodyanskiy is the professor of Artificial Intelligence Department at KhNURE, the Head of Control Systems Research Laboratory at KhNURE. He has more than 600 scientific publications including 40 inventions and 10 monographs. His research interests are Hybrid Systems of Computational Intelligence: adaptive, neuro-, wavelet-, neo-fuzzy-, real-time systems that have to do with control, identification, and forecasting, clustering, diagnostics and fault detection.

Prof. Bodyanskiy is an IEEE Senior Member and a member of 4 scientific and 7 editorial boards.



**Oleksii K. Tyshchenko** graduated from Kharkiv National University of Radio Electronics in 2008. He got his PhD in Computer Science in 2013. He is currently working as a Senior Researcher at Control Systems Research Laboratory, Kharkiv National University of Radio Electronics.

His current research interests are Evolving, Reservoir and Cascade Neuro-Fuzzy Systems; Computational Intelligence; Machine Learning.



**Olena O. Boiko** graduated from Kharkiv National University of Radio Electronics in 2013. She is a PhD student in Computer Science at Kharkiv National University of Radio Electronics. Her current interests are Time Series Forecasting, Fuzzy Clustering, Evolving Neuro-Fuzzy Systems.

Manuscript received June 27, 2016; revised August 3, 2016; accepted October 8, 2016.

**How to cite this paper:** Zhengbing Hu, Yevgeniy V. Bodyanskiy, Oleksii K. Tyshchenko, Olena O. Boiko, "Adaptive Forecasting of Non-Stationary Nonlinear Time Series