***Представлено новий метод прогнозування часових рядів, який динамічно знаходить ваги для вхідних факторів в залежності від конкретних значень самихфакторів.Запропонованийметод був перевірений на наборі реальних часових рядів і показав кращі результати у порівнянні з методом, що використовувався як базовий***

***Ключові слова: прогнозування часових рядів, лінійна регресія, Байєсівське усереднення моделей, нейронні мережі***

***Представлен новый метод прогнозирования временных рядов, который динамически находит веса для входных факторов в зависимости от конкретных значений самих факторов. Предложенный метод был проверен на наборе реальных временных рядов и показал лучшие результаты по сравнению с методом, который использовался в качестве базового***

***Ключевые слова: прогнозирование временных рядов, линейная регрессия, Байесовское усреднение моделей, нейронные сети***

УДК 004:519.2

# A METHOD FOR BUILDING A FORECASTING MODEL WITH DYNAMIC WEIGHTS

**V. Sineglazov**
Doctor of Technical Sciences, Professor
Department of Aviation Computer-Integrated Systems
Institute of aerospace control systems,
National Aviation University
Komarov Av., 1, Kyiv, Ukraine, 03680
E-mail: svm@nau.edu.ua
**O. Chumachenko**
Candidate of Technical Sciences, Associate Professor*
E-mail: lobach21@mail.ru
**V. Gorbatiuk***
E-mail: vladislav.horbatiuk@gmail.com
*Department of Technical Cybernetics
National Technical University of Ukraine
«Kyiv Polytechnic Institute»
Peremogy Av., 37, Kyiv, Ukraine, 03056

## 1. Introduction

Forecasting has always been one of the most interesting and important problems of mankind. It is also one of the hardest problems, since to solve it we need to deal with the following issues:

a) it is impossible to take into account all the factors that influence the process we are trying to forecast; moreover, their influence can change over time – the factor which was not important today can play a major role tomorrow;

b) there are always a lot (sometimes infinite number) of plausible models that fit the training data well – we have to decide which model or set of models to use, and that's usually a very error-prone decision;

c) it is often hard (if not impossible) to find the optimal complexity of the model.

In this paper we introduce the method that tries to deal with first two issues, i.e. it flexibly determines the set of models to use for the given inputs and takes into account the volatile significance/influence of the factors.

## 2. Problem statement

Let us have a sequence of N data points $x=\{x_1,\ldots,x_N\}$ measured at successive time points $\{t_1,\ldots,t_N\}, t_i - t_{i-1} = T = \text{const}, i = 2\ldots N$. Then the problem of forecasting (Fig. 1) considered in this paper can be stated as follows: using the data we have (Fig. 1, a), build the model of the forecasted process that takes $n$ successive data points $x_{i-n+1},\ldots,x_i$ as input and outputs the forecast for the value $x_{i+k}$ at some future time point $t_{i+k}$ (Fig. 1, b). This model can be represented mathematically as $y = F(x_1,\ldots,x_n)$, where $F$ is some unknown function. One important note is that this model can be defined implicitly or even work as a "black box" – we give it an input, we receive the desired output, which serves as a forecast.

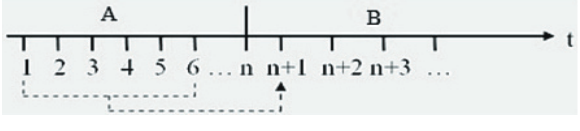


Fig. 1. Graphic representation of the forecasting problem statement: a – known values; b – future values

## 3. Review of existing forecasting methods

The most well-known forecasting method is probably a linear regression [1]. It builds the following linear model:

$$y = F(x_1,\ldots,x_n) = \sum_{i=1}^{n} w_i * x_i + w_0 , \quad (1)$$

where $w_1,\ldots,w_n$ – importance weights of the input variables $x_1,\ldots,x_n$ respectively; $w_0$ – bias term, can be omitted. The weights are usually found by minimizing the mean squared error (MSE) of the model on the training data:

$$\vec{w}^* = \arg\{\min_{\vec{w}} \left\{ \sum_{j=1}^{m} \left( \vec{w}^T * \vec{x}_j + w_0 - y_i \right)^2 \right\}\}, \quad (2)$$

where $\vec{x}_j = \left[ x_{j1},\ldots,x_{jn} \right]^T$ – $j^{th}$ training case; $y_j$ – corresponding known output value.

Even though a linear regression remains one of the most widely used methods due to its simplicity, it has one natural limitation following from its definition: it cannot model complex nonlinear dependencies. To overcome this limitation a lot of nonlinear forecasting methods were developed. Let us mention the most widely used.

1. Group method of data handling (GMDH) [2]. The GMDH is a set of forecasting algorithms which are based on a recursive selection of the best models and the subsequent construction of more complex models using previously selected ones. The forecasting accuracy is improved by increasing a complexity of the models. The selection criterion is based on a model performance on the test set, while model's parameters are determined from the training set. The simplest models also called base functions usually have the following form:

$$F(x_1,\ldots,x_n) = a_0 + \sum_{i=1}^{n} a_i * x_i + \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_i x_j + \cdots. \quad (3)$$

However, any kind of base functions can be used, including harmonic series, exponential series etc.

The GMDH-like algorithms have proven to be really effective on real-life problems mainly because of their use of an external criterion (i.e. models are selected using data that wasn't used for their training).

2. Artificial neural networks (ANN) [3]. An ANN is a system of connected and interacting artificial neurons – mathematical models of biological neural cells. An ANN is not programmed in the usual sense of the word: they are trained. During training, the neural network is able to detect complex relationships between input and output data and perform synthesis. The ability of neural networks to forecast comes directly from their ability to generalize and find the hidden relationships between input and output data. After training, the network is able to predict the future value of a certain sequence on the basis of several previous values and/or any current factors.

Mainly two architectures are used for the forecasting task: feed-forward neural network [4] (Fig. 2) and recurrent neural network [5] (Fig. 3). While feed-forward ANN basically corresponds to very complex function the recurrent ANN adds some dynamics, i. e. it has a finite dynamic response to time series input data.

The main advantage of an ANN over other methods of forecasting is that the network can equally well model practically any functional relationship, whereas most other methods are best suited for modelling some concrete type of functions (obviously, the method of polynomial smoothing is best suited for processes with a polynomial regular component, the method of Fourier series smoothing is best suited for processes with a periodic regular component etc.). Another important advantage of neural networks is the ability to learn.

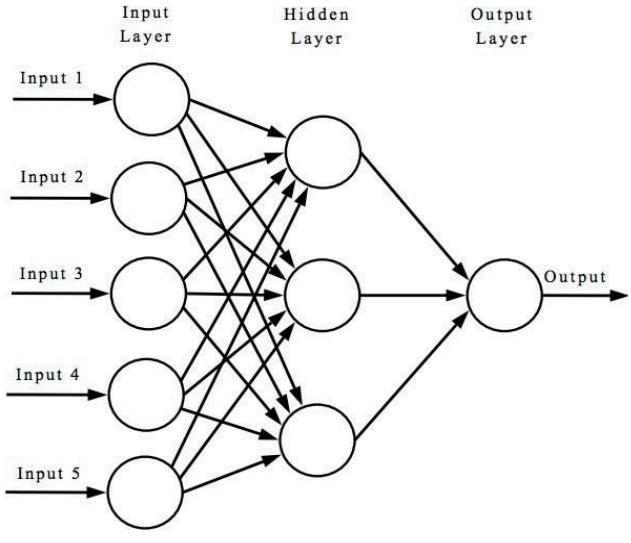


Fig. 2. Feedforward neural network architecture

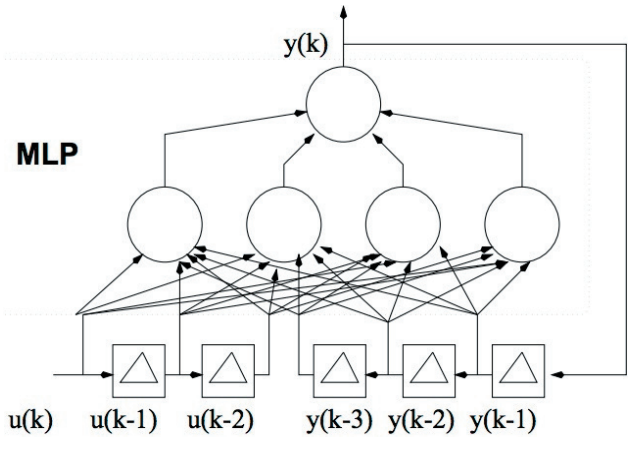


Fig. 3. Recurrent neural network architecture

3. Wavelet-based time series forecasting [6]. Many time series exhibit non-stationarity in their statistics. While the series may contain dominant periodic signals, these signals can vary in both amplitude and frequency over long periods of time. Ideally, one would like to separate the shorter period oscillations from the longer. Wavelet analysis attempts to solve these problems by decomposing the time series into time/frequency space simultaneously. One gets information on both the amplitude of any "periodic" signals within the series and how this amplitude varies with time.

The wavelet-based forecasting suggests the use of a discrete wavelet transform [7] to obtain the corresponding wavelet coefficients and the subsequent prediction of the future values using these coefficients as inputs.

One step of discrete wavelet transform produces so-called detail coefficients and approximation coefficients given by:

$$y_{appr}[n] = \sum_{k=-\infty}^{\infty} x[k] g[2n-k], \quad (4)$$

$$y_{detail}[n] = \sum_{k=-\infty}^{\infty} x[k] h[2n-k], \quad (5)$$

where $g[2n-k]$ and $h[2n-k]$ is an impulse response of the low-pass filter and high-pass filter respectively. Usually, the

approximation coefficients get decomposed further multiple times (Fig. 4).

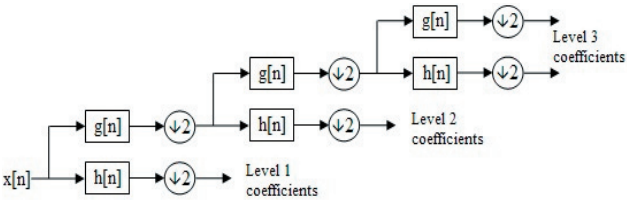


Fig. 4. Graphic representation of wavelet decomposition

4. Various combinations of multiple methods. For instance, a combination of GMDH and ANN was suggested in [8]: instead of using predefined base functions small feedforward neural networks can be used, thus eliminating the issue with selecting the most appropriate type of base functions.

Despite of the variety of existing forecasting methods, most of them can be generalized using the following equation:

$$\vec{w}^{*} = \arg\{\min_{\vec{w}}\{E[F(\vec{w},\vec{x}),X,\vec{y}]\}\}, \quad (6)$$

where $E$ is some error function that is minimized; $F(\vec{w},\vec{x})$ – function that represents a forecasting model (linear or nonlinear); $X$ – matrix of training cases; $\vec{y}$ – vector of known output values for the training cases. It's clear that such an approach ignores the issues (a) and (b) given in the introduction – it uses a single model and assumes that input variables have constant influence.

## 4. Overview of the suggested forecasting method

The main idea of the method is to 'dynamically' find a set of weights for given inputs rather than use a single 'static' set of weights; in other words, the inputs are used for both finding the appropriate weights and predicting the output using these weights.

We suggest naming the method "linear regression with dynamic weights (LRDW)".

The method's inputs are the matrix of training cases $X \in \mathbb{R}^{m\times n}$ and the vector of known output values $y \in \mathbb{R}^{m\times 1}$ where m is a number of training cases and n is a number of input variables.

The preprocessing stage needed to obtain these matrices from the raw time series $x = \{x_i\}, i = 1\ldots N$ is left outside the scope of this method for the sake of simplicity (we suggest normalizing the time series values to the range $[-1;1]$ and then using an embedding technique [8] with an appropriate embedding dimension and the horizon of prediction to obtain these matrices).

The method's parameters are numbers $K \in \{1,2,\ldots,m\}$ and $\gamma \in (0;\infty)$ which will be described later.

Main steps of the method are:

1. Subtract row mean from each row of the matrix X (i.e. make the set of its rows zero-mean):

$$x_{ij} = x_{ij} - \bar{x}_j, i = 1\ldots m, j = 1\ldots n, \quad (7)$$

where $\bar{x} = [\bar{x}_1,\ldots,\bar{x}_n]$ - row mean vector of the matrix X.

2. Find the initial 'static' weights vector $\vec{w}^{(in)} = [w_1^{(in)},\ldots,w_n^{(in)}]^T$ using standard linear regression with an error function $E = \sum_{i=1}^{m}\left[\sum_{j=1}^{n}\left(w_j^{(in)} * x_{ij}\right) - y_i\right]^2$. It is important to omit a bias term - in practice, models without bias (given that training input vectors are zero-mean) usually have better prediction error on the whole data set.

3. To perform the next step we should introduce new error functions, one for each training case:

$$E_i = \alpha * \left[\sum_{j=1}^{n}\left(w_{ij} * x_{ij}\right) - y_i\right]^2 + \beta * \sum_{j=1}^{n}\left(w_{ij} - w_j^{(in)}\right)^2, \quad (8)$$

where $\alpha$ and $\beta$ are some constants, $\alpha,\beta \in (0;\infty)$; $\vec{w}_i = [w_{i1},\ldots,w_{in}]^T$ – new, 'dynamic' weights vectors, one for each training case. We need to make the squared prediction error for the $i^{th}$ training case small by choosing the appropriate weights $\vec{w}_i$, and to keep these weights close to the static ones $\vec{w}^{(in)}$ in order to minimize the particular error function $E_i$. The tradeoff between how much to reduce the error and how close should the weights $\vec{w}_i$ be to the initial ones is controlled by the parameters $\alpha$ and $\beta$; if we set them to be $\alpha > \beta$ we want to improve the error more than to keep the weights and vice versa. To reduce the number of method's parameters we can divide all error functions by $\beta$ and let $\gamma = \frac{\alpha}{\beta}$; now we can see the meaning of the second parameter $\gamma$: choosing $\gamma > 1$ is equivalent to choosing $\alpha > \beta$ and $\gamma < 1 \Leftrightarrow \alpha < \beta$. When the input values lie in the range $[-1;1]$ the suitable choice of $\gamma$ is somewhere between 0.1 and 0.3.

4. Find the optimal set of weights $\vec{w}_i^{*}$ for each error function $E_i$ by solving the following linear system, obtained as a result of finding the partial derivatives w.r.t. corresponding weights and equating them to 0:

$$A_i * \vec{w}_i^{*} = b_i, \quad (9)$$

where

$$A_i = \begin{bmatrix} \gamma x_{i1}^2 + 1 & \gamma x_{i1}x_{i2} & \cdots & \gamma x_{i1}x_{in} \\ \gamma x_{i2}x_{i1} & \gamma x_{i2}^2 + 1 & \cdots & \gamma x_{i2}x_{in} \\ \cdots & \cdots & \cdots & \cdots \\ \gamma x_{in}x_{i1} & \cdots & \cdots & \gamma x_{in}^2 + 1 \end{bmatrix},$$

$$b_i = \begin{bmatrix} \gamma y_i x_{i1} + w_1^{(in)} \\ \vdots \\ \gamma y_i x_{in} + w_n^{(in)} \end{bmatrix}, \vec{w}_i^{*} = \begin{bmatrix} w_{i1}^{*} \\ \vdots \\ w_{in}^{*} \end{bmatrix}. \quad (10)$$

Thus, the set of weights $\vec{w}_i^{*}$ that minimizes the error function $E_i$ can be found as $\vec{w}_i^{*} = A_i^{-1} * b_i$.

5. Remember the matrix of discrete derivatives for each training case:

$$V = \begin{pmatrix} x_{12} - x_{11} & \cdots & x_{1n} - x_{1n-1} \\ \vdots & \ddots & \vdots \\ x_{m2} - x_{m1} & \cdots & x_{mn} - x_{mn-1} \end{pmatrix}. \quad (11)$$

6. The forecast for a new input vector $\vec{x} = [x_1,\ldots,x_n]^T$ is formed as follows:

a. Find the vector of derivatives $\vec{v} = [x_2 - x_1,\ldots,x_n - x_{n-1}]$.

b. Find K nearest neighbors of $\vec{v}$ in the matrix V, and remember:

– their indices in sorted order - from nearest to furthest: $idx = [idx_1,\ldots,idx_K]$

– distances to neighbors $\vec{d} = [d_1,\ldots,d_K], d_1 \le d_2 \le \ldots \le d_K$

– total distance $D = \sum_{i=1}^{K} d_i$ .

c. Find $K$ stored weights vectors for the corresponding training cases:

$$W = \begin{pmatrix} w^*_{idx_1,1} & \cdots & w^*_{idx_K,1} \\ \vdots & \ddots & \vdots \\ w^*_{idx_1,n} & \cdots & w^*_{idx_K,n} \end{pmatrix} = \begin{pmatrix} \vec{w}^*_{idx_1} & \cdots & \vec{w}^*_{idx_K} \end{pmatrix}. \qquad (12)$$

d. Find the weights vector that will be actually used for the prediction. To do this, we should average the found weights vectors depending on the distance from $\vec{v}$ to the vector of discrete derivatives for the corresponding training case. The averaging weights are calculated as $\vec{\omega} = \left[\frac{d_K}{D},\ldots,\frac{d_1}{D}\right]^T$, i. e. the weights vector for the nearest neighbor gets the biggest weight.

e. Finally, the forecast is calculated as:

$$y = (W * \vec{\omega})^T * \vec{x} . \qquad (13)$$

To sum up, the method finds a separate weights vector for each training case and then calculates the forecast for new inputs by finding the weights for $K$ nearest training cases (nearest in the sense of Euclidean distance between the vectors of derivatives), weighting them based on the distance to produce a single set of weights and then applying these weights to the inputs. It is obvious that using this approach the weights ⇔ importance of the input variables will be different for different input vectors. Also, since each set of weights defines the corresponding forecasting model, we are not using a constant set of models – instead, we find the most appropriate set depending on the input. The parameter $K$ plays a 'smoothing' role - the bigger the $K$ the more weights vectors will be averaged the closer to the weights of a linear regression the average will be.

When searching for the nearest neighbors, vectors of derivatives are used instead of original vectors because for the time series forecasting problem the dynamics (i.e. how values change over time) is usually much more important and representative than the exact values of the forecasted process – for other problems, where inputs are not the successive points of some time series we should use original vectors.

The method is somewhat similar to Locally Linear Regression (LLR) [9] and Bayesian Model Averaging (BMA) [10]: it finds some kind of a local model for each training case similar to LLR and averages multiple models for the given inputs just like in BMA. However, LLR loses global information ('static' weights of a linear regression) while building local regressions - as a result, these local models can overfit badly. And opposite to BMA, where the set of averaged models is constant and we average the models' outputs, the proposed method selects models to average depending on the inputs and averages the models themselves, not their output (there is no difference in a linear case, but in general these two averaging methods are not equivalent).

## 5. Testing performance of the proposed method

To test the performance of the proposed method the set of 11 publicly available ([11, 12]) time series was used. Linear regression and GMDH were used for comparison. All methods shared the following parameters:

– time series embedding dimension ⇔ number of input variables $n = 5$

– horizon of prediction $h = 2$ (predicting the value 2 time steps ahead)

– training set size to full data set size ratio $r = 0.5$ (half of the cases were used for model training).

A bias term was omitted for both the LRDW and the linear regression; default parameters values of the specific GMDH implementation [13] were used; the method's parameters were set to $\gamma = 0.2$ and $K = 1$.

Normalized squared error (NSE) given by the formulae $E = \frac{\sum_{j=1}^{m} (F(x_1,\ldots,x_n) - y_i)^2}{\sum_{j=1}^{m} y_i^2}$ was used as a model performance indicator. It was calculated on the full data set. The obtained results are given in the Table 1.

Table 1

NSE of tested models on the full data set

| Name of time series | Linear regression | LRDW | GMDH |
|---|---|---|---|
| Australian electricity production | 0.017662 | 0.019721 | 0.012685 |
| CATS benchmark [14] | 0.002894 | 0.002696 | 0.002901 |
| Dollar to euro exchange rate | 0.063802 | 0.05511 | 0.062086 |
| Dollar to pound exchange rate | 0.055874 | 0.050277 | 0.058154 |
| Consumer price index (CPI) | 5.50E-05 | 0.007696 | 2.22E-05 |
| Spanish electric energy demand | 0.019363 | 0.024104 | 0.017655 |
| Spanish mean interest rates | 0.055512 | 0.048002 | 0.053009 |
| Spanish stock exchange index | 0.002652 | 0.005721 | 0.002495 |
| Sunspots per month | 0.5811 | 0.45099 | 0.17865 |
| US aviation shipments | 0.20121 | 0.15927 | 0.13734 |
| Winter NAO index | 1.0566 | 0.98757 | 1.0009 |
| Total error | 2.0567 | 1.8112 | 1.5259 |

Short analysis of the obtained results:

– the LRDW has better NSE on most but not all time series – so we need to carefully choose its parameters, especially $\gamma$;

– the average improvement in error is about 12 % relative to the NSE of a linear regression (and the biggest improvement is $\approx 22.3\%$ for the 'Sunspots per month' time series);

– in general, GMDH performs better than the LRDW – however, the approach we used to obtain LRDW from a linear regression can be easily applied to other forecasting methods, including GMDH – and it can possibly boost their performance as well;

– there are several time series for which LRDW performed even better than GMDH.

The graphical example of the LRDW model producing better forecasts than the linear regression is given on the Fig. 5 ('US aviation shipments' time series).

As you can see, the forecast of a LRDW method is very similar to the one, obtained by linear regression, but for some

cases the proposed method gives much more accurate predictions (the training cases were selected randomly).

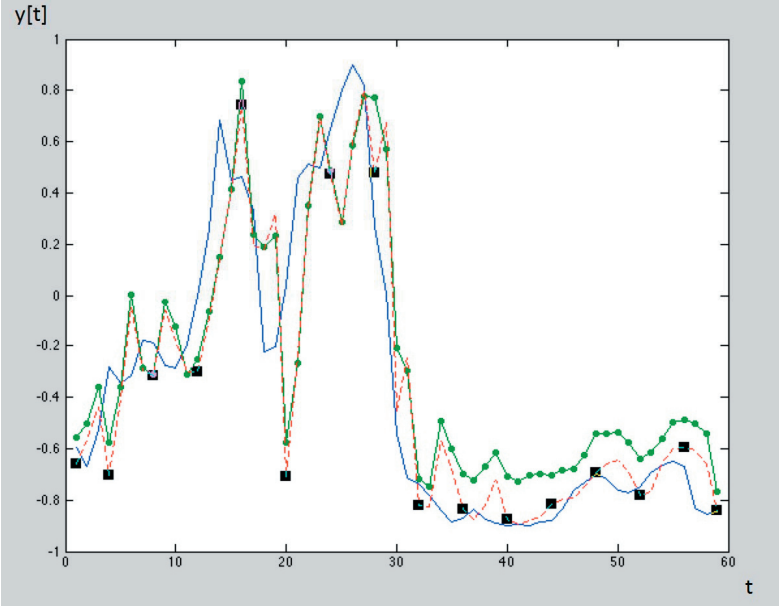


Fig. 5. Forecasts, obtained by two different methods: solid line — original time series, dotted line — LRDW forecast, line with markers — linear regression forecast

## 6. Conclusion

The proposed method was tested on real data and its performance (measured using NSE criterion) is usually better than the performance of the method it 'originated' from - linear regression. Hence, we believe that applying the same approach to other methods, including nonlinear ones like GMDH or neural networks can improve their performance also.

There are also possible improvements to the approach itself:

• instead of finding dynamic weights for each training case it is possible to find them for some clusters of training cases to improve the method's runtime efficiency;

• the suitable choices for method's parameters can possibly be determined from the training data – for example a value of the $\gamma$ parameter can somehow depend on the ratio between the total magnitude of static weights (i.e. sum of their values) and the magnitude of an error for this training case (when using these static weights);

• instead of finding nearest neighbors and averaging the corresponding dynamic weights we can build a model to predict the weights values from the inputs values using any suitable forecasting method.

References

1. Cook, R. D. Influential Observations in Linear Regression [Text] / R. D. Cook // Journal of the American Statistical Association. – 1979. – № 74. – P. 169–174.
2. Stepashko, V. S. GMDH Algorithms as Basis of Modeling Process Automation after Experimental Data [Text] / V. S. Stepashko // Sov. J. of Automation and Information Sciences. – 1988. – № 21 (4). – P. 43–53.
3. Rosenblatt, F. The Perceptron: A Probalistic Model For Information Storage And Organization In The Brain [Text] / F. Rosenblatt // Psychological Review. – 1958. – № 65 (6). – P. 386–408.
4. Auer, P. A learning rule for very simple universal approximators consisting of a single layer of perceptrons [Text] / P. Auer, B. Harald, M. Wolfgang // Neural Networks. – 2008. – № 21 (5). – P. 786–795.
5. Elman, J. L. Finding Structure in Time [Text] / J. L. Elman // Cognitive Science. – 1990. – № 14 (2). – P. 179–211.
6. Benaouda, D. Wavelet-based nonlinear multi-scale decomposition model for electricity load forecasting [Text] / D. Benaouda, F. Murtagh, J. L. Starck, O. Renaud // Neurocomputing. – 2006. – № 70. – P. 139–154.
7. Akansu, A. N. Wavelet Transforms in Signal Processing: A Review of Emerging Applications [Text] / A. N. Akansu, W. A. Serdijn, I. W. Selesnick // Physical Communication, Elsevier. – 2010. – № 3 (1). – P. 1–18.
8. Sineglazov, V. An algorithm for solving the problem of forecasting [Text] / V. Sineglazov, E. Chumachenko, V. Gorbatiuk // Aviation. – 2013. – № 17 (1). – P. 9–13.
9. Cleveland, W. S. Robust Locally Weighted Regression and Smoothing Scatterplots [Text] / W. S. Cleveland // Journal of the American Statistical Association. – 1979. – № 74 (368). – P. 829–836.
10. Hoeting, J. A. Bayesian Model Averaging: A Tutorial [Text] / J. A. Hoeting, D. Madigan, A. E. Raftery, C. T. Volinsky // Statistical Science. – 1999. – № 14 (4). – P. 382–401.
11. U.S. General Aviation Aircraft Shipments and Sales [Electronic resource] / Barr Group Aerospace & AeroWeb / Available at: http://www.bga-aeroweb.com/database/Data3/US-General-Aviation-Aircraft-Sales-and-Shipments.xls. – 2014.
12. Data Sets for Time-Series Analysis [Electronic resource] / Evolutionary and Neural Computation for Time Series Prediction Mini-site. – Available at: http://tracer.uc3m.es/tws/TimeSeriesWeb/repo.html - 2005.
13. Jekabsons, G. GMDH-type Polynomial Neural Networks for Matlab [Electronic resource] / Gints Jekabsons. Regression software and datasets. – Available at: http://www.cs.rtu.lv/jekabsons/ - 2013.
14. Lendasse, A. Time Series Prediction Competition: The CATS Benchmark [Text] / A. Lendasse, E. Oja, O. Simula, M. Verleysen // International Joint Conference on Neural Networks, Budapest (Hungary), IEEE. – 2004. – P. 1615–1620.