Citation: Svetunkov S. Elementary image of the Kolmogorov-Gabor polynomial in economic modeling. Technoeconomics. 2024. 3. 2 (9). 4–21. DOI: https://doi.org/10.57809/2024.3.2.9.1

ELEMENTARY IMAGE OF THE KOLMOGOROV-GABOR POLYNOMAL IN ECONOMIC MODELING

Abstract. Today, neural networks are actively used in modeling complex nonlinear dependencies. Amid such a rapid growth of interest in this powerful tool for modeling various objects and processes, research in the natural sciences and engineering, the work on the application of neural networks in economics is vanishingly small. This is explained both by the complexity of the modeling tool itself - neural networks, and by the object of modeling - the evolving economy.

At the dawn of the development of neural networks, the method of modeling processes using Kolmogorov-Gabor polynomials (or Wiener series) was considered as an alternative. For various reasons, this method lost the competitive battle, and neural networks prevail.

The article presents a method and technique for constructing an elementary image of the Kolmogorov-Gabor polynomial and shows that today it can be quite used as an alternative to neural networks in modeling economic processes.

Keywords: economic and mathematical modeling, nonlinear processes, multidimensional dependencies, neural networks, Kolmogorov-Gabor polynomial, N. Wiener series.

Acknowledgements/Funding (if present)

The work was carried out with the financial support of the Russian Science Foundation, grant No. 19-010-00610\19 “Theory, methods and techniques for forecasting economic development using autoregressive models of complex variables.”

Introduction

From the very beginning of the emergence of the task of modeling complex interrelationships, scientists have used mathematical statistics to*LSM* to solve it, directing their efforts to identify causal relationships between variables and to explicitly determine the form of these identified interrelationships. This form of interrelationship took the form of one of the elementary functional dependencies of the indicator y on the factors x_i. It was believed that a correctly chosen function, if it does not express the mathematical expectation of the process, then it is its best approximation.

The main problem that scientists faced when constructing and using a regression model or a set of models was precisely the difficulty of determining the type of functions describing the mathematical expectation, since an error made in this case leads to incorrect modeling results. In the univariate case, this task is solved with a certain degree of success using correlation and regression analysis. But in multivariate dependencies, determining the form and nature of the influence of each factor on the result is a very difficult and rarely successfully solved problem.

Let's demonstrate the complexity of this task with a simple example. Let's generate data for a conditional example. Let the influencing factors change as follows:

$x_{1i} = i, \quad x_{2i} = 0,5x_{1i}+\varepsilon_i = 0,5i+\varepsilon_i, \quad x_{3i} = x_{2i}cos(x_{1i})^{0,1}+x_{1i}.$

And now let's generate the resulting feature using the specified three variables by the formula:

$$y_i = x_{1i} + 2x_{2i} + 0,1x_{3i}^2 \quad (1)$$

We are faced with a nonlinear multifactorial functional dependency, and since it does not contain a random component, we can expect a definitive determination of the form of this dependency. We will change i from 1 to 50 and generate data for a hypothetical task based on this.

Now let's assume that the form of this dependency is unknown to us and, based on the available statistical data about the factors and the modeled variable, it is necessary to find this dependency. The first thing to do is to calculate the correlation matrix. We will obtain:

Table 1. Correlation matrix for a conditional example

	x_{1i}	x_{2i}	x_{3i}	y_i
x_{1i}	1			
x_{2i}	0,8103	1		
x_{3i}	0,9931	0,8674	1	
y_i	0,9746	0,8907	0,9823	1

What conclusion will the researcher draw from these data? Since the pairwise correlation coefficients are above 0.80, these data should be described using a linear multifactorial model. The *LSM* estimates of such a model based on the available data will give the following form of the model:

$$\hat{y}_i = 13{,}16x_{1i} + 5{,}56x_{2i} - 9{,}67x_{3i} - 11{,}34 \quad (2)$$

It has excellent statistical characteristics, which tell the researcher about its significance. However, the obtained model terribly described the original formula (1) and does not correspond at all to the law it describes! And yet, many economists, having received an econometric model of type (2), will try to give an economic interpretation to each coefficient obtained, for example, that an increase of the third factor by one unit will reduce the result by 9.67 units. From the true situation, which is modeled by (1), follows a completely different influence of the third factor – its increase will nonlinearly enhance the result. That is, mathematical statistics have yielded a result opposite to what is the case.

If we now take a closer look at the data in Table 1, we can notice that there is a strong and almost linear relationship between the first and third factors, as the pairwise correlation coefficient between them is r = 0.9931. Therefore, the effect of multicollinearity may be present in the modeling results. One of these two correlated factors should be discarded when building the model. Let it be the factor x_{1i}.

Then, the econometric model, which takes multicollinearity into account, will look like this:

$$\hat{y}_i = 1{,}38x_{2i} + 4{,}47x_{3i} - 18{,}79 \quad (3)$$

It also has excellent statistical characteristics, indicating its statistical significance. But it is as far from the truth as model (2). And if we now give an economic interpretation to the new model (3), we can say that an increase of the third factor by one unit leads to an increase in the result by 4.47 units, which is far from the real influence and opposite to the conclusions of model (2).

It is impossible to find the form of the real dependency (1) using methods of mathematical statistics.

These methodological difficulties in applying structural regression models in economic modeling have been understood for a long time, but there was no alternative until cybernetics appeared. Cybernetics took various steps away from regression, proposing as an alternative the 'black box' model, into which certain factors x_i are fed, and the values $y$are observed at the output. How the input is transformed into the output is of no interest to anyone. In managing the 'black box,' the task was set as follows: to select such values of factors x_i that the output y observed would be close to a predetermined Y. Methods for selecting optimal values of x_i were developed in cybernetics.

In parallel with the tasks of optimal control of complex systems, cybernetics solved another problem – pattern recognition. It differed from the task of optimal control in that, given the input parameters x_i and known values of the output parameter y, it was necessary to adjust the 'black box' so that this correspondence between input and output matched the given pattern

Y. Here, the object of adjustment was not the data, but the 'black box' model, or rather – its coefficients.

Scientists proposed various mathematical models of this 'black box,' and over time, the model of artificial neurons combined into a single neural network proved to be the most accurate and convenient. Inside the 'black box' - the neural network - when solving the task, a multi-iterative recalculation of the network's coefficients occurs so that the output result is obtained with the specified accuracy. The obtained values of the neural network coefficients are of no interest, as they are not statistical characteristics of the process but are parameters of the 'black box.'

With the help of neural networks, many diverse tasks have been solved, and we have come close to the task of creating artificial intelligence. However, examples of successful application of neural networks in economics are exceedingly rare.

This can be explained by two main reasons.

The first is that building and using neural networks requires the specialist to have a deep knowledge of several narrowly specialized sections of mathematics and programming. Typically, scientists engaged in economic research do not possess such knowledge. Therefore, at best, they apply templates of existing neural networks to one or another forecasting object. The basis for applying such a template is the known number of inputs (factors) and outputs (indicators).

The second reason is due to the specific properties of the economy as an object of research. The fact is that neural networks were developed as a method of modeling complex objects operating under conditions of homogeneity, characterized by a finite set of properties and features. In contrast, economic processes are mostly heterogeneous and evolving – the set of properties and features of these processes changes over time and is not finite. The exception is perhaps financial and consumer markets under conditions of stable economic conjuncture. Here neural networks show very good results. But such periods of relative stability are eventually interrupted by some external influences on the markets, and the processes become heterogeneous, evolutionary, and chaotic. In such situations, neural networks stop working well.

Thus, the task of modeling complex economic objects is more successfully solved by methods of mathematical statistics than by neural networks or their analogs. Therefore, the task of creating a simple and understandable alternative to neural networks is relevant.

Materials and Methods

A neural network consists of a collection of *j* interconnected neurons, each of which represents a superposition of linear and non-linear functions:

$$\hat{y}_j = f(a_0 + \sum_{i=1}^{n} a_i x_i) \qquad (4)$$

Here::

$\hat{y}_j$ - output signal of *j*-th neuron;

f – activation function or transfer function;

a_i - weight of the *i*-th signal (factor);

x_i – *i*-th component of the input signal or the factor itself;

i = 1, ..., n – neuron input number;

n - number of neuron inputs;

a_0 – coefficient characterizing the displacement.

To avoid problems that may arise with the scales of variables when working with neural networks, all variables are preliminarily normalized.

Two aspects are fundamentally important for the neural network: the form of the function *f* and the structure of the neural network, that is, the number of neurons (4) and their interconnection with each other.

At the very beginning of the use of neural networks, the computational capabilities of their implementation were not high, so the function *f* was often considered as the possibility of activation $y = 1$ or non-activation $y = 0$ of the output from the neuron, and in the case when it was necessary to quantitatively transform the input values x_i, a linear or piecewise-linear function was used. Today, the logistic (sigmoid) function is most often used, allowing for a nonlinear transformation of input signals into output. The logistic is also convenient because its first derivative is easily calculated and computed, which is important when estimating the coefficients of all neurons (4) by numerical methods, since one of the gradient methods [26] is most often used for this.

LSM or other methods of mathematical statistics are not directly suitable for solving this task. Indeed, for example, in a simple two-layer neural network, the variables x_i are considered as inputs to the first layer of *m* neurons of the network. In each neuron of the first layer (4), they are transformed into outputs y_j ($j = 1, 2, \ldots, m$), which are inputs to the second layer neural network. At the output of the second layer, the following is obtained:

$$\hat{y} = f(b_0 + \sum_{j=1}^{m} a_j \hat{y}_j) \qquad (5)$$

If we now substitute (4) into (5), we obtain the following superposition of functions:

$$\hat{y} = f(b_0 + \sum_{j=1}^{m} a_j f_j(a_{0j} + \sum_{i=1}^{n} a_{ij} x_i)) \qquad (6)$$

Real values are described by this model with some error:

$$\varepsilon = y - \hat{y}, \quad (7)$$

the minimization of whose squares will give *LSM* estimates. But it is not possible to directly estimate the coefficients of such a simple two-layer neural network in the case of a nonlinear form of the transfer function, since the calculation of the gradients of the error function (7) by the model coefficients (6) represents a complex task, leading to the need to solve a system of complex nonlinear equations. It is significantly easier and more convenient to solve this problem using numerical methods, most often using the gradient method.

As can be seen from the simple explanation of the essence of building neural networks, their use requires a good knowledge of mathematics and programming skills, as training a neural network is a multi-iterative procedure with many parameters being estimated simultaneously, which can only be done using some advanced programming language.

When a neural network, trained on a sample from the general population, is tested on a validation set from this general population, it should give good results. If this does not happen, the network is complicated and trained again. And this continues until the network is well 'tuned'.

If non-stationary and irreversible processes are modeled, then neural networks demonstrate their inadequacy. And it is such processes that are many of the processes occurring in the economy. Today many scientists are trying to solve this problem by improving neural networks. One of such directions is the theory of neuro-Bayesian methods [18, 23], but this theory so far cannot boast any tangible results and admits that: 'now this area is only at the very beginning of the path and is waiting for new researchers' [19, p. 446]. Some hopes in this direction are pinned on recurrent neural networks [20]. In traditional neural networks, it is assumed that the input factors and output factors are independent, and recurrent neural networks take into account the presence of some influence of previous observations on current observations due to feedback between some neurons.

One of the first publications on this topic among Russian economists is the article by Astrakhantseva I.A., Astrakhantsev R.G., and Kutuzova A.S. [2]. Having identified potential factors of inflation in the Russian Federation, the authors conducted a correlation-regression analysis and determined that inflation can be described by two factors: the exchange rate of the dollar to the ruble and the growth of citizens' debt excluding currency revaluation. A simple recurrent neural network built by them on this data set predicts inflation well for one observation and very poorly predicts all subsequent values.

Physicists Kondratenko V. and Kuprin Y. built a recurrent neural network capable of predicting the sign of price increases in the foreign exchange market with a probability of success just over 50% [13]. For this, they used the logarithm of the ratio of the current price to the previous price of exchange rates of the American dollar, Japanese yen, Swiss franc, British pound, and euro.

Among the few foreign publications on this topic, several articles can be highlighted. Yunze Tao, Xia Sheng presented a method for predicting the exchange rate of the euro to the US dollar using a simple recurrent neural network in which the factors were past daily exchange rates of the euro and the US dollar [29]. It is difficult to assess how well this network works, as no comparison with other forecasting methods is provided in the article.

Zhiguo Qiu, Emese Lazar, and Keiichi Nakata showed comparative results of using models based on recurrent neural networks with state tracking, feedforward neural networks, as well as VAR vector autoregressions and exponential smoothing models [30]. Six asset return time series were modeled over a period of more than 20 years. Recurrent models showed the best results.

Ruofan Liao, Petchaluck Boonyakunakorn, Napat Harnpornchai, and Songsak Sriboonchitta used a recurrent neural network to predict the exchange rate of the US dollar to the yuan from 12 other indicators, shifted up to d lags back plus the indicator itself, shifted up to d lag [22]. They compared this network with ARIMA and showed that if ARIMA gave an average forecast error square of 0.211, then their neural network - 0.010.

These results are encouraging but not impressive.

Returning to the origins of the formation of neural networks, it should be noted that the director of the Institute of Cybernetics of the Academy of Sciences of the Ukrainian SSR A.G. Ivakhnenko in the 1970s proposed another path of unstructured modeling of complex objects using complex polynomials. In this regard, he proposed an interesting method of decomposing many complex tasks, the essence of which can be understood from a simple example [9].

Suppose there is a need to build a model of a high-degree polynomial on a small number of observations:

$$\hat{y}_t = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 \quad (8)$$

It is proposed to divide this polynomial into a system of three series.

The first row is:

$$\hat{y}_{1t} = b_0 + b_1 t + b_2 t^2 \quad (9)$$

The second row is:

$$\hat{y}_{2t} = c_1 t^3 + c_2 t^4 \quad (10)$$

It is easy to notice that the first row is a model that includes the first three terms of polynomial (8), and the model of the second row includes two other components of this polynomial.

The coefficients of models (9) and (10) can be easily estimated, for example, using *LSM*. To form the overall polynomial (8), it is proposed to estimate the coefficients of the third row model using *LSM*:

$$\hat{y}_t = d_0 + d_1 \hat{y}_{1t} + d_2 \hat{y}_{2t} \quad (11)$$

Into this model, as you can see, the calculated values of the variable are substituted, which are computed according to (9) and (10). What does this mean? If we substitute into (11) not the calculated values, but the formulas by which they are obtained, that is, models (9) and (10), then we get:

$$\hat{y}_t = d_0 + d_1(b_0 + b_1 t + b_2 t^2) + d_2(c_1 t^3 + c_2 t^4) =$$
$$(d_0 + d_1 b_0) + d_1 b_1 t + d_1 b_2 t^2 + d_2 c_1 t^3 + d_2 c_2 t^4 \qquad (12)$$

From where it is easy to determine the relationship between the coefficients of the original polynomial (8) and the coefficients of the multi-row system (9) - (11):

$$a_0 = d_0 + d_1 b_0,\ \ a_1 = d_1 b_1,\ \ a_2 = d_1 b_2,\ \ a_3 = d_2 c_1,\ \ a_4 = d_2 c_2 \qquad (13)$$

Since the multi-row procedure for estimating the coefficients of the polynomial is a linear superposition of functions linear with respect to unknown coefficients, the estimates (13) will coincide with the *LSM* estimates applied directly to (9). This decomposition method proposed by A.G. Ivakhnenko was suggested for constructing various nonlinear models, and most often it was proposed to use a finite polynomial decomposition of the nonlinear dependency into additive components.

In 1930, V. Volterra published the work 'Theory of Functionals and of Integral and Integro-Differential Equations' where he derived series that allow the study of systems with soft inertial nonlinearities [27]. These series are actively used today in solving technical and engineering tasks of modeling nonlinear processes [16]. In 1958, N. Wiener in the monograph 'Nonlinear Problems in the Theory of Random Processes' published a modification of the Volterra series. He proposed a method of approximating a nonlinear dependency, starting with simple elements, to which new and new nonlinear terms are successively added: 'Our decomposition differs from the usual Fourier decomposition, as we have a countable set of functionals, but the overall task remains the same' [28, p. 50]. Today, mathematicians call this decomposition the Wiener series, and for the discrete case, this series will take the form:

$$y = a_0 + \sum_{i=1}^{m} a_i x_i + \sum_{i=1}^{m}\sum_{j=1}^{m} a_{ij} x_i x_j + \sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{k=1}^{m} a_{ijk} x_i x_j x_k + ... \qquad (14)$$

The same problem was independently solved in 1956 by A.N. Kolmogorov and in 1961 by D. Gabor [7]. Since A.G. Ivakhnenko, who first used series (14), called it the Kolmogorov-Gabor polynomial, this name prevails in domestic science today, and we will also adhere to this naming convention.

Series (14) is indeed very convenient for modeling nonlinear systems with weak nonlinearity based on available statistical data. Moreover, like a neural network, it connects the input variables x_i with the output $y$without defining the nature and form of the relationship between them, that is, it does so in an unstructured way, just like neural networks, which allows it to be considered as an alternative to neural networks. However, unlike neural networks, the structure of the polynomial is fixed and strictly defined. Any researcher with m input variables will always construct the same polynomial (14). Neural networks can connect variables x_i with the output y in many ways – they can be single-layer or multi-layer, vary the connections between neurons, add recursive connections, etc. This means that the dependency between x_i and y can be described using neural networks in many ways – better or worse, simpler or more complex. With series (14), this relationship can only be modeled in the same way.

And if we compare (14) with (6), we can notice an important advantage of the polynomial over the neural network: it represents a linear function in terms of parameters, whose coefficients can be easily found by any statistical method directly, without resorting to numerical methods, by solving a system of linear equations with unknown coefficients.

In the 60s and 70s of the 20th centuries, when the polynomial (14) competed with neural networks in pattern recognition tasks, calculations were performed on analog machines. These machines consisted of electrical devices that transformed electrical signals similarly to mathematical operations. For example, about the first neural networks, A.G. Ivakhnenko reported: "In the first design of the perceptron, automatic potentiometers with servomotors were used as associating elements. The machine used 512 such potentiometers. They were too large and expensive. Now in the perceptron, so-called biaks are used – magnetic elements with ferrite cores. Functionally, biaks reproduce the actions of a two-position polarized relay or trigger [9, p. 159].

At that historical moment in the development of computing technology, simple multipliers and adders in computers for the use of artificial neurons were simpler and cheaper than nonlinear converters for the use of the Kolmogorov-Gabor polynomial, which were also less reliable in operation. This predetermined that neural networks became the main tool for pattern recognition, and the Kolmogorov-Gabor polynomial is only occasionally used for modeling nonlinear dependencies. Mainly these are works in the field of engineering sciences [3, 6, 21]. Such works are not encountered in economics. Conditionally economic can be considered only the article with the results of modeling the relationship between electrical power and a set of technical and economic indicators of the operation of the Ryazan GRES [4], as well as the article on the application of (14) for clustering industrial enterprises [17]. In all these works, not the Kolmogorov-Gabor polynomial is used, but the method of sequential approximation to it - MGUA as developed by A.G. Ivakhnenko.

Unfortunately, the Kolmogorov-Gabor polynomial has a significant drawback: as the number of *i* variables *x*, describing the behavior of the variable *y*, increases, the number of terms *N* in the polynomial (14) sharply increases. Indeed, if for $i = 2$ the number of terms in series (14) will be equal to $N = 6$, then for $i = 5$ it becomes equal to $N = 252$. And this is a sharp increase in the dimensionality of the problem being solved.

For example, if modeling the dependence of several variables y_j on *x*, then for $i = 5$ and $j = 4$ we get $N = 1008$ unknown coefficients of the polynomial. And when using a two-layer fully connected feedforward neural network to solve this problem, it is necessary to estimate from twenty to forty unknown coefficients.

Precisely because of the high dimensionality of the problem, this tool is practically not used in solving real modeling tasks. Thus, I.I. Sulyaev, mentioning the Kolmogorov-Gabor polynomial when setting the task of modeling the process of mixing oxygen and air for the oxidation of sulfide copper-nickel raw materials in a metallurgical furnace, pointed out the enormous size of this polynomial and subsequently used a neural network [24].

Pointing out the enormous dimensionality of the problem with many initial variables, A.G. Ivakhnenko proposed a method of step-by-step decomposition of the model - "formation of a multi-row system", the essence of which was outlined earlier in (8) – (13).

For the case of three factors, the full Kolmogorov-Gabor polynomial will be written as follows [12]:

$$\hat{y} = a_0 + a_1x_1 + a_2x_2 + a_3x_3 + a_4x_1^2 + a_5x_2^2 + a_6x_3^2 + a_7x_1x_2 + a_8x_1x_3 +$$
$$a_9x_2x_3 + a_{10}x_1^3 + a_{11}x_2^3 + a_{12}x_3^3 + a_{13}x_1^2x_2 + a_{14}x_1^2x_3 + a_{15}x_1x_2^2 + a_{16}x_2^2x_3 +$$
$$a_{17}x_1x_3^2 + a_{18}x_2x_3^2 + a_{19}x_1x_2x_3 \qquad (15)$$

At the first stage, it is proposed to use partial (support) polynomials with two factors, each of which approximates the modeled indicator *y*with its own approximation error ε_i:

$$y = \hat{y}_1 + \varepsilon_1 = b_0 + b_1x_1 + b_2x_2 + b_3x_1^2 + b_4x_2^2 + b_5x_1x_2 + \varepsilon_1 \qquad (16)$$

$$y = \hat{y}_2 + \varepsilon_2 = c_0 + c_1x_1 + c_2x_3 + c_3x_1^2 + c_4x_3^2 + c_5x_1x_3 + \varepsilon_2 \qquad (17)$$

$$y = \hat{y}_3 + \varepsilon_3 = d_0 + d_1x_2 + d_2x_3 + d_3x_2^2 + d_4x_3^2 + d_5x_2x_3 + \varepsilon_3 \qquad (18)$$

Using the method of least squares, one can easily find the coefficients (16) - (18). After that, using the calculated values of the variables obtained from (7) – (9) as factors, one can find the coefficients of another polynomial using the method of least squares:

$$y = \hat{y}_4 + \varepsilon_4 = e_0 + e_1 y_1 + e_2 y_2 + e_3 y_3 + e_4 y_1 y_2 y_3 + \varepsilon_4 \quad (19)$$

And now, substituting (16), (17) and (18) into (19), we get:

$$z = e_0 + e_1(b_0 + b_1 x_1 + b_2 x_2 + b_3 x_1^2 + b_4 x_2^2 + b_5 x_1 x_2) + e_2(c_0 + c_1 x_1 + c_2 x_3 + c_3 x_1^2 + c_4 x_3^2 + c_5 x_1 x_3) +$$
$$e_3(d_0 + d_1 x_2 + d_2 x_3 + d_3 x_2^2 + d_4 x_3^2 + d_5 x_2 x_3) + e_4(b_0 + b_1 x_1 + b_2 x_2 + b_3 x_1^2 + b_4 x_2^2 + b_5 x_1 x_2) \cdot$$
$$(c_0 + c_1 x_1 + c_2 x_3 + c_3 x_1^2 + c_4 x_3^2 + c_5 x_1 x_3) \cdot (d_0 + d_1 x_2 + d_2 x_3 + d_3 x_2^2 + d_4 x_3^2 + d_5 x_2 x_3) \quad (20)$$

If we now compare the resulting expression with (15), we can see that the coefficient a_0 corresponds to a combination of coefficients of the equations:

$$a_0 = e_0 + e_1 b_0 + e_2 c_0 + e_3 d_0 + e_4 b_0 c_0 d_0 \quad (21)$$

In the same way, correspondences can be found for other coefficients of the complete Kolmogorov-Gabor polynomial (15):

$$a_1 = e_1 b_1 + e_2 c_1 + e_4 b_1 c_0 d_0 + e_4 b_0 c_1 d_0,$$
$$a_2 = e_1 b_2 + e_2 c_2 + e_3 d_1 + e_4 b_2 c_0 d_0 + e_4 b_0 c_0 d_1,$$
$$\ldots$$
$$a_{19} = e_4(b_1 c_2 d_1 + b_2 c_1 d_2 + b_5 c_2 d_0 + b_2 c_5 d_0 + b_1 c_0 d_6 + b_0 c_1 d_5 + b_5 c_0 d_2 + b_0 c_5 d_1). \quad (22)$$

Some scientists suggest considering this correspondence as estimates of the Kolmogorov-Gabor polynomial. However, (20) is not identical to (15), since in addition to the 20 terms of the Kolmogorov-Gabor polynomial, polynomial (20) contains many other terms that are not present in (15). If we expand the brackets (20) and group the resulting terms of the polynomial, we will obtain a significantly more complex formation. In order not to clutter the description of the obtained polynomial with its coefficients, let's assume that in the final polynomial they are all equal to one. Then we will get:

$$z = 1 + x_1 + x_2 + x_3 + x_1^2 + x_2^2 + x_3^2 + x_1^3 + x_2^3 + x_3^3 + x_1^4 + x_2^4 + x_3^4 +$$
$$x_1 x_2 + x_1 x_3 + x_2 x_3 + x_1 x_2^2 + x_1 x_3^2 + x_1 x_2^3 + x_1 x_3^3 + x_1 x_2^4 + x_1 x_3^4 + x_1^2 x_2 + x_1^2 x_3 +$$
$$x_1^2 x_2^2 + x_1^2 x_3^2 + x_1^2 x_2^3 + x_1^2 x_3^3 + x_1^2 x_2^4 + x_1^2 x_3^4 + x_1^3 x_2 + x_1^3 x_3 + x_1^3 x_2^2 + x_1^3 x_3^2 +$$
$$x_1^3 x_2^3 + x_1^3 x_3^3 + x_1^3 x_2^4 + x_1^3 x_3^4 + x_1^4 x_2 + x_1^4 x_3 + x_1^4 x_2^2 + x_1^4 x_3^2 + x_1^4 x_2^3 + x_1^4 x_3^3 +$$
$$x_2 x_3^2 + x_2 x_3^3 + x_2 x_3^4 + x_2^2 x_3 + x_2^2 x_3^2 + x_2^2 x_3^3 + x_2^2 x_3^4 + x_2^3 x_3 + x_2^3 x_3^2 + x_2^3 x_3^3 + x_2^3 x_3^4 + x_2^4 x_3 + x_2^4 x_3^2 + x_2^4 x_3^3 +$$
$$x_1 x_2 x_3 + x_1 x_2^2 x_3 + x_1 x_2 x_3^2 + x_1 x_2^3 x_3 + x_1 x_2 x_3^3 + x_1 x_2^4 x_3 + x_1 x_2 x_3^4 + x_1 x_2^2 x_3^2 + x_1 x_2^2 x_3^3 + x_1 x_2^3 x_3^2 +$$
$$x_1^2 x_2 x_3 + x_1^2 x_2^2 x_3 + x_1^2 x_2 x_3^2 + x_1^2 x_2^2 x_3^2 + x_1^2 x_2^3 x_3 + x_1^2 x_2 x_3^3 +$$
$$x_1^3 x_2 x_3 + x_1^3 x_2^2 x_3 + x_1^3 x_2 x_3^2 + x_1^4 x_2 x_3 + x_1 x_2 x_3^2 x_2^2 + x_1 x_2^2 x_3 x_2^2 \quad (23)$$

This polynomial contains 80 terms, unlike the Kolmogorov-Gabor polynomial (15), which has 20 terms. That is, using the approach of A.G. Ivakhnenko, the researcher constructs not the Kolmogorov-Gabor polynomial, but a new polynomial with a different structure, which has four times more terms. A.G. Ivakhnenko wrote that under certain conditions "... the coefficients of non-existent real connections turn out to be zero (or very small)" [11, p. 177]. However, it turned out that this is not the case: "Testing the classical GMDH (Group Method of Data Handling) by solving control tasks with artificially formed initial data shows that its selecting abilities are not high enough: in some examples, arguments not included in the formula defining the process were in the list of arguments of the model of the process" [4, p. 39].

A.G. Ivakhnenko later abandoned the idea of multi-stage estimation of the coefficients of the Kolmogorov-Gabor polynomial as a whole and considered another task - the sequential complication of models, starting with reference polynomials, and gradually complicating the form of the model, approaching the form of the full Kolmogorov-Gabor polynomial, but not reaching it. At each stage of complicating the model, its statistical characteristics (for example, the variance of the approximation error) are evaluated, which are compared with the same characteristics of less complex models. The process of complicating the model stops when the measured statistical characteristic ceases to improve. This method was named by him "Group Method of Data Handling" (GMDH) and it is used today in solving some practical problems [1, 19], including in combination with neural networks [13, 15].

Research has shown that the Kolmogorov-Gabor polynomial in terms of accuracy can actively compete with neural networks used in economics, especially today with the availability of different computational capabilities than fifty years ago [26], but for this, an effective method is needed to overcome the "curse of dimensionality". The Ivakhnenko method (16) – (22) does not solve this problem.

A simple method for constructing the full Kolmogorov-Gabor polynomial, which overcomes the "curse of dimensionality," is outlined below.

Let's consider this method first on the example of the dependence of y on three factors x_1, x_2, and x_3 (15), and then make the necessary generalizations.

At the first stage, for example, using the method of least squares, it is necessary to find the coefficients of a simple linear model that includes all factors:

$$\hat{y}' = b_0 + b_1x_1 + b_2x_2 + b_3x_3 \quad (24)$$

And at the second and last stage, the same least squares method should be used to estimate the coefficients of the cubic polynomial, substituting the calculated values into it as a factor (24):

$$\hat{y} = c_0 + c_1\hat{y}' + c_2(\hat{y}')^2 + c_3(\hat{y}')^3 \quad (25)$$

It’s all. The model is built. If we now substitute (24) into (25), we get:

$$\hat{y} = c_0 + c_1(b_0 + b_1x_1 + b_2x_2 + b_3x_3) + c_2(b_0 + b_1x_1 + b_2x_2 + b_3x_3)^2 + c_3(b_0 + b_1x_1 + b_2x_2 + b_3x_3)^3 \quad (26)$$

Opening the brackets and grouping, we obtain complete correspondence of the structure of the polynomial (26) to the structure of the Kolmogorov-Gabor polynomial (15) - it contains exactly 20 terms.

Now we can find the correspondence of coefficients (25) - (26) to the coefficients of the Kolmogorov-Gabor polynomial (15):

$$a_0 = c_0 + c_1b_0 + c_2b_0^2 + c_3b_0^3,$$

$$a_1 = c_1b_1 + 2c_2b_0b_1 + 2c_2c_3b_0^2b_1,$$

$$\ldots$$

$$a_{19} = 2c_3(b_1 + b_2 + b_3) \quad (27)$$

It should be noted that with such a simple method, we will not obtain the true values of the polynomial coefficients. The least squares estimate (LSE) applied directly to the Kolmogorov-Gabor polynomial (15) and the LSE applied to the proposed method of stepwise decomposition (25) - (26) will differ from each other. This is easily understood because, in the first case, 20 independent coefficients are estimated, while in the second case, 8 coefficients are estimated, of which only 4 coefficients of the linear multifactor model (25) are completely independent of the other coefficients.

Therefore, a simplified model of the Kolmogorov-Gabor polynomial is obtained, which we will call the "elementary form" of the Kolmogorov-Gabor polynomial.

Is it possible to obtain a more accurate representation of the Kolmogorov-Gabor polynomial? Yes, it is.

In relation to the task at hand, the process of constructing such a more complete representation will consist of three stages.

At the first stage, a multifactor linear model (24) is constructed.

At the second stage, a multifactor nonlinear quadratic model is constructed:

$$\hat{y}'' = c_0 + c_1 x_1^2 + c_2 x_2^2 + c_3 x_3^2 . \qquad (28)$$

And based on these two models, the coefficients of the final model are estimated:

$$\hat{y} = d_0 + d_1 \hat{y}' + d_2 (\hat{y}')^2 + d_3 (\hat{y}')^3 + d_4 y'' \qquad (29)$$

After substituting (24) and (28) into (29) and grouping the terms, the image of the Kolmogorov-Gabor polynomial is obtained again, in the construction of which not 4, but 8 independent coefficients (24) and (28) are estimated, as well as five dependent coefficients (29). Of course, the new image will be somewhat more accurate than the elementary image (26), and at the same time, the estimation of its parameters is still simpler than the direct estimation of the coefficients of the Kolmogorov-Gabor polynomial (15). The feasibility of using the elementary or full images of the Kolmogorov-Gabor polynomial is determined by practical needs.

This simple method of constructing images of the Kolmogorov-Gabor polynomial can be extended to the case of any number of variables x_i, i=1, 2, …, m. For the elementary image of the Kolmogorov-Gabor polynomial of degree m we obtain:

$$\begin{cases} \hat{y}' = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_m x_m, \\ \hat{y} = c_0 + c_1 \hat{y}' + c_2 (\hat{y}')^2 + \ldots + c_m (\hat{y}')^m . \end{cases} \qquad (30)$$

As can be seen, it is necessary to estimate step by step (m+1) unknown coefficients, which is a routine task. Therefore, the "curse of dimensionality," which A.G. Ivakhnenko repeatedly pointed out, is completely overcome, and with the help of the indicated method, an elementary image of the Kolmogorov-Gabor polynomial can be constructed for any m.

The system (30) can be represented in a more compact mathematical form:

$$\hat{y} = b_0 + \sum_{j=1}^{m} b_j (a_0 + \sum_{i=1}^{m} a_i x_i)^j \qquad (31)$$

If a researcher is interested in a more accurate approximation to the Kolmogorov-Gabor polynomial, then its full image at x_i, i=1, 2, …, m will be formed like this:

$$\begin{cases} \hat{y}' = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_m x_m, \\ \hat{y}'' = c_0 + c_1 x_1^2 + c_2 x_2^2 + \ldots + c_m x_m^2, \\ \ldots \\ \hat{y}^{m-1} = w_0 + w_1 x_1^{m-1} + w_2 x_2^{m-1} + \ldots + w_m x_m^{m-1} \\ \hat{y} = z_0 + z_1 \hat{y}' + z_2 (\hat{y}')^2 + \ldots + z_2 (\hat{y}')^m + \ldots + z_m \hat{y}^{m-2} + z_{m+1} \hat{y}^{m-1} . \end{cases} \qquad (32)$$

Research conducted on numerous hypothetical and real examples confirms the conclusion that the elementary image of the Kolmogorov-Gabor polynomial models various nonlinear processes only slightly worse than the full image of this polynomial. Since the Kolmogorov-Gabor polynomial is suitable for describing dependencies with weak nonlinearity [5], which most of the nonlinear multifactor economic processes are, the elementary image can be used as the primary model for describing economic nonlinear dependencies.

A.G. Ivakhnenko, having defined the Kolmogorov-Gabor polynomial as a certain limit, believed that there is no particular sense in reaching it, since the partial polynomials (the first parts of the Kolmogorov-Gabor polynomial) can cope with the task of modeling nonlinearity quite successfully and there is no need to "multiply entities beyond necessity." To find a model of optimal complexity, he proposed the "Group Method of Data Handling" (GMDH), which involves the sequential complication of the model, following the structure of the Kolmogorov-Gabor polynomial for this set of variables [11, pp. 46 - 47].

It is quite possible that the elementary image of the Kolmogorov-Gabor polynomial may also be excessively complex for the modeled object, and therefore it also makes sense to find a polynomial of optimal complexity through the sequential use of partial elementary images, starting with the simplest linear image.

For this, the first equation of system (30) is initially constructed - a multifactor linear model, which is considered as a partial image of the first-degree polynomial. The coefficients found for this model are the basis for calculating the variance σ_1^2.

Then, the coefficients of the partial image of the second-degree Kolmogorov-Gabor polynomial are estimated, which has the form:

$$\hat{y}_2 = c_{02} + c_{12}\hat{y}_1 + c_{22}\hat{y}_1^2 . \quad (33)$$

For it, the variance σ_2^2 is also calculated.

If $\sigma_1^2 \leq \sigma_2^2$, then complicating the polynomial is pointless, and a simple linear multifactor model should be used for modeling. However, if this condition is not met and a reduction in variance is observed, the process of complicating the polynomial image continues, and a partial image of the third-degree polynomial is constructed:

$$\hat{y}_3 = c_{03} + c_{13}\hat{y}_1 + c_{23}\hat{y}_1^2 + c_{33}\hat{y}_1^3 \quad (34)$$

and its variance σ_3^2 is estimated. It is compared with the previous variance σ_2^2.

If the condition $\sigma_2^2 \leq \sigma_3^2$ is satisfied again then the model becomes more complicated until a complete elementary image of the Kolmogorov-Gabor polynomial is constructed:

$$\hat{y}_m = c_{0m} + c_{1m}\hat{y}_1 + c_{2m}\hat{y}_1^2 + ... + c_{mm}\hat{y}_1^m \quad (35)$$

with dispersion σ_m^2.

Results and Discussion

Let's demonstrate with an example how this procedure can be used to select the optimal image of the Kolmogorov-Gabor polynomial. Suppose that for some research purposes, it became necessary to model the dependence of the number of divorces in the Russian Federation from 1999 to 2022 based on five factors: population size, birth rate, GDP per capita, cost of one square meter of housing, as well as the number of marriages and divorces. These data can be taken from the open statistics of the Russian Federation and are not provided here to save space.

Since a nonlinear dependence of one indicator on six is being modeled, the full Kolmogorov-Gabor polynomial that could be constructed from these data should contain 954 coefficients that need to be estimated from the available statistical data. Constructing such a polynomial is a complex task. However, constructing an elementary image of this polynomial is a simple task. Let's find the partial image of the Kolmogorov-Gabor polynomial of optimal complexity using the procedure described above.

The linear multifactor model of this dependence has the maximum variance of all, which we will take as 100%. How the variance of the approximation error of this dependence changes with the complication of the partial images of the Kolmogorov-Gabor polynomial is shown in Table 2.

Table 2. Change in the variance of the approximation error with a change in the complexity of the elementary image of the Kolmogorov-Gabor polynomial

View of the elementary image of the Kol-mogoro-va-Gabora polynomial	Approximation error variance, % of maximum variance
Particular elementary image of a polynomi-	*100,00*

al of a first degree	
Particular elementary image of a polynomi-al of a second degree	*98,78*
Particular elementary image of a polynomi-al of a third degree	*95,62*
Particular elementary image of a polynomi-al of ta fourth degree	*82,07*
Particular elementary image of a polynomi-al of a fifth degree	*72,83*
Elementary image of a polynomial	*72,84*

The model of optimal complexity for the considered example is a partial elementary image of a fifth-degree polynomial. In the artificial neuron (4), the first stage involves a linear convolution of the input variables into a single parameter *s*, which is then transformed into the output *y* through the transfer function *f*(*s*) at the second stage.

In the elementary image of the polynomial (31), the first stage also involves a linear convolution of the same input variables into the calculated value y_1. However, here, unlike the neuron model, the first adjustment of the image occurs when the function of the discrepancy between the result of the linear convolution and the actual value *y* is minimized. The second stage involves "fine-tuning" the image in the form of a simple power series.

The strength of neural networks is determined by the fact that its neurons are interconnected according to the principle: the output from one neuron simultaneously becomes the input to the next neuron (or several subsequent neurons). The number of neurons is determined by the researcher, and by varying them, the researcher can complicate the network until it begins to describe the modeled dependence between input and output in the best way.

Elementary images of the polynomial can also be combined into a certain network when the output from one elementary image of the polynomial simultaneously becomes the input to the next elementary image of another polynomial (or several subsequent elementary images of polynomials). Then a network of polynomials is formed, which describes the dependence between input and output data differently and with different accuracy than a neural network. Let's call such a network polynomial.

It is important to note one very important difference between the polynomial network and the neural network. In the neuron model, only the form of the transfer function *f*(*s*) can change. It can be logistic, linear, piecewise-linear, or hyperbolic tangent. Other types of transfer functions in the neuron are rare and can be called exotic. But all these transfer functions model the same type of sigmoidal nonlinear transformation: linear and piecewise-linear functions somewhat worse, logistic and hyperbolic tangent better. Replacing the type of transfer function during the training process is impossible – the type of transfer function determines the mathematical algorithms for training the network.

In the model of the elementary image of the polynomial, the fine-tuning function does not necessarily have to represent the full elementary image of the Kolmogorov-Gabor polynomial. These can be partial elementary images, ranging from a simple linear to a full elementary image of the polynomial. When training such a network, you can change these very fine-tuning functions – complicating them from simple to full or vice versa – simplifying them from full to simple. Thus, polynomial networks acquire an additional training tool without changing the network structure.

What should we expect from the new polynomial network compared to neural networks? Since the computational power of an artificial neuron is significantly lower than the computational power of the elementary image of the polynomial, more accurate modeling results can be expected from the polynomial network.

Let's confirm this statement with a simple example. Let's build a neural network and a polynomial network based on data from 1990 to 2022 about the GDP of the United Kingdom (y) depending on the gross capital accumulation (x_1), the size of the economically active population (x_2), expenditures on scientific research and development (x_3), and the size of the UK's GDP for the previous period (x_4).

Since the influence of the first three factors on the indicator is approximately the same, and the influence of the GDP of the previous period on the current value is somewhat different, let's build a neural network and a polynomial network in such a way:

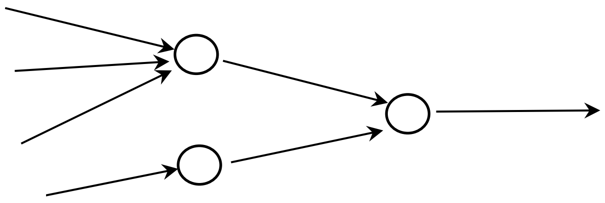

Figure 1. Graphical model of neural network and polynomial network used

Here, for the neural network, each circle represents an artificial neuron, and for the polynomial network – an elementary image of the Kolmogorov-Gabor polynomial. We have before us a simple two-layer feedforward model, the complexity of which is optimal for the considered example. Since after normalization the original data become both negative and positive, the transfer function of the last neuron was represented as a hyperbolic tangent, which allows working with negative values.

Let's include in the list of models for comparison of the neural and polynomial networks also an elementary image of the Kolmogorov-Gabor polynomial. The comparison results are presented in Table 3.

Table 3. Results of approximation of UK data (1990 – 2020)

Model type	Neural network	Elementary image of the Kolmogorov-Gabor polynomial	Polynomial network
Average sum of squares	0,01388	0,00865	0,00846

As can be seen from the table, the neural network gave the worst results out of the three models, while the network of elementary images of the Kolmogorov-Gabor polynomial gave the best result. This, of course, does not mean that neural networks will always be worse than polynomial networks. Moreover, it was previously discussed that recurrent neural networks are best suited for modeling economics, not feedforward networks. But our task was not to build the best model with the available data, but to compare the neural network and the polynomial network. And the given example suggests that polynomial networks can be a worthy alternative to neural networks, and if necessary, polynomial networks can also be made recurrent.

But another conclusion follows from the example: the elementary image of the Kolmogorov-Gabor polynomial showed very good approximating properties. Its average sum of squares of approximation error is less than that of a simple neural network and slightly worse than that of a polynomial network. This once again confirms the conclusion that in a sufficiently large number of cases of modeling nonlinear economic interrelations, the elementary image of the

Kolmogorov-Gabor polynomial copes well with this task, and it is possible to do without forming a polynomial network.

Conclusions.

The elementary image of the Kolmogorov-Gabor polynomial, presented in this article, is quite simple to construct based on existing statistical data, and its coefficients can be estimated for a fairly large number of influencing factors without compromising the dimensionality of the problem. It can flexibly change depending on the modeled process, as partial elementary images of the polynomial of a lower degree than the full elementary image can be used instead of its full form.

Even though elementary images of the Kolmogorov-Gabor polynomial model various nonlinear dependencies very well, a polynomial network like neural networks can be created from them to model even more complex dependencies. The question of which networks are better to use for economic modeling - neural or polynomial - requires further research.

To use neural networks for modeling economic dynamics, it is necessary to complicate them to the form of recurrent networks. This is not required with the elementary image of the Kolmogorov-Gabor polynomial. Moreover, in the elementary image of the Kolmogorov-Gabor polynomial, the influence and role of each coefficient are easily determined, unlike the parameters of a neural network. This means that both Bayesian methods and various adaptive methods successfully used in economic forecasting, such as the method of stochastic approximation [25], can be applied to them.

Interest is the use of the elementary image in the form of autoregressive dependence - when, in this image, instead of input factors xi, previous values of the modeled indicator y_t, y_{t-1}, y_{t-2}, ..., $y_{t-\tau}$ are used. In this case, it opens up the possibility of simple construction of unstructured nonlinear autoregressions. Their use in process analytics is limited, but in predictive analytics, they can be indispensable.

Summarizing all the above, it should be concluded that the introduction of the elementary image of the Kolmogorov-Gabor polynomial into scientific circulation opens new interesting prospects for economic researchers in the field of economic modeling.

REFERENCES

1. Artemenko M.V., Kalugina N.M., Shutkin A.N. 2016. Formation of a set of informative indicators based on the approximating Kolmogorov–Gabor polynomial and the maximum gradient of functional differences. Proceedings of the South-West State University 1 (18), 116 – 124.
2. Astrakhantseva I.A., Kutuzova A.S., Astrakhantsev R.G. 2020. Recurrent neural networks for fore-casting regional inflation. Scientific works of VEO of Russia: Materials of MA-EF-2020 223, 420 – 431. doi: 10.38197/2072/2060-2020-223-3-420-431
3. Balasanyan S.Sh., Gevorgyan E.M. 2016. Comparative analysis of regression methods and group accounting of arguments in modeling mineral processing processes. Georesources Engineering 327 (4), 23–34.
4. Belov V.V., Chistyakova V.I. 2008. Modeling and forecasting business processes using algorithms for self-organization of formal descriptions. Business-Informatics 4 (6), 37–45.
5. Busgang J.J, Ehrman L., Graham J.W. 1974. Analysis of Nonlinear Systems with Multiple Inputs. IEEE 62, 1088.
6. Dyachkov M.Yu. 2017. Inductive modeling of objects and phenomena by the method of group accounting of arguments: shortcomings and ways to eliminate them. Bulletin of the Russian Peoples' Friendship University. Series: Mathematics. Computer science. Physics 25 (4), 323–330.

7. Gabor D., Wilby W.R., Woodcock R.A. 1961. A universal nonlinear filter, predictor and simulator which optimizes itself by a learning process. Proc. Inst. Electr. Engrs. 108 (40), 85—98.
8. Geets V.M. ., Klebanova T.S., Chernyak O.I., Ivanov V.V., Kizim M.O., Dubrovina N.A., Stavitsky A.V Models and methods of socio-economic forecasting. Kh.: VD "IN-ZHEK", 2008. 396 p.
9. Ivakhnenko A.G. Self-learning systems with positive feedback. K.: Publishing House of the Academy of Sciences of the Ukrainian SSR, 1963.
10. Ivakhnenko A. G. Systems of heuristic self-organization in technical cybernetics. K.: “Technology”, 1971. 372 p
11. Ivakhnenko A.G. Long-term forecasting and control of complex systems. K.: Tekhnika, 1975, 312 p.
12. Ivakhnenko A.G., Muller J.A. Self-organization of predictive models. K.: Technika, 1985; Berlin: FEB Verlag Technik, 1984. 223 pp.
13. Kiselev A.V., Petrova T.V., Degtyaryov S.V., Rybochkin A.F., Filist S.A., Shatalova O.V., Mishustin V.N. 2018. Hybrid deciding modules with virtual streams for classification and prediction of functional state of complex systems. Proceedings of the Southwest State University 22(4), 123-134. doi:10.21869/2223-1560-2018-22-4-123-134
14. Kondratenko V. V., Kuperin Yu. A. Using Recurrent Neural Networks To Forecasting of Forex. Papers cond-mat/0304469, arXiv.org
15. Lusia D.A., Ambarwati A. 2018. Multivariate Forecasting Using Hybrid VARIMA-Neural Network in JCI Case, Proceedings of SAIN, 11-14.
16. Maas S. What you need to know about the method of analysis based on Volterra series. Engineering micro-electronics, 2000, No. 1. pp. 45 – 51.
17. Mekhovich S.A., Akhiezer E.B., Dunaevskaya O.I. 2014. Economic and mathematical model of zoning of industrial enterprises. Energy saving, energy, energy audit, 8 (126), 39-49.
18. Neal R.M. 1993. Probabilistic inference using. Technical Report CRGTR,140.
19. Nikolenko S.I., Kadurin A.A., Arkhangelskaya E.O. Deep learning, St. Petersburg: Publishing house "Peter", 2022. 480 p.
20. Patterson D. 2018. Deep learning from a practitioner's perspective, M.: DMK Press, 2018. 482 p
21. Petrov K.E., Deineko A.A., Chalaya O.V., Panferova I.Yu. 2020. Method for ranking alternatives during the procedure of collective expert evaluation. Radioelectronics, informatics, management 2, 84-94.
22. Ruofan L., Petchaluck B. 2020. Forecasting the Exchange Rate for USD to RMB using RNN and SVM. Journal of Physics: Conference Series 1616, 012050. doi:10.1088/1742-6596/1616/1/012050
23. Schmidhuber J. 2015. Deep learning in neural networks: An overview. Neural Networks 61, 85-117.
24. Sulyaev I.I. 2014. Neuromodel of the technological process of preparing an oxygen-air mixture as a control object. News of St. Petersburg Electrotechnical University "LETI" 7, 20 – 25.
25. Svetunkov S.G. 1993. Econometric methods for demand forecasting M.: MSU, 1993. 123 p.
26. Svetunkov S.G., Chernyagin A.S. 2024. Neural network and Kolmogorov-Gabor polynomial in modeling complex economic processes. Modern Economy Success 4, 153 – 157.
27. Volterra V. Theory of Functionals and of Integral and Integro-differential Equations. Blackie & Son Limited, 1930. 226 p.

28. Wiener N. Nonlinear problems in random theory. The Technology Press of The Massachusetts Institute of Technology and John Wiley & Sons, Inc., New York. 1958. 138p
29. Yunze T., Sheng X. 2024. Exchange rate forecast between Euro and US dollar based on recurrent neural network. Highlights in Science, Engineering and Technology 85, 663 – 669.
30. Zhiguo Q., Emese L., Keiichi N. 2024. VaR and ES forecasting via recurrent neural network-based stateful models. International Review of Financial Analysis 92, 103102.