Nonlinear Phenomena in Complex Systems, vol. **17**, no. 3 (2014), pp. 327 - 335

Mathematical Models of Complex Systems on the Basis of Artificial Neural Networks

A. N. Vasilyev[*] and D. A. Tarkhov[†]
Saint-Petersburg State Polytechnic University,
29 Politechnicheskaya Str, 195251 Saint-Petersburg, RUSSIA

(Received 23 July, 2014)

Neural networking technique with models based on ordinary/partial differential equations is applied to the known incorrect problems. Solutions to such problems by routine approaches are difficult. The problem approximate solution is found as the artificial neural network output with a prescribed architecture. Network weights are determined in the stepwise network training based on the error functional minimization process in general. The case of the system parameters given in some variation intervals and the parameter set as a part of input data is considered. The construction of robust parameter neural network models is examined using some problems in classical and non-classical statements. The direct problem solution and the inverse problem regularization for the offered neural network approach are constructed uniformly. The neurocomputing results for fixed and growing neural networks are given. The supercomputer use is discussed. The neural network approach advantages and some possible generalizations are mentioned.

PACS numbers: 05.45; 02.30.H, J
Keywords: artificial neural network (ANN), error functional, training, global optimization, interval parameter, heterogeneous data, evolutional algorithm

1. Introduction

Solution to the problem of multiscale process predictive modeling in natural and technical systems by classical methods of mathematical modeling faces a number of fundamental problems.

Most of these methods require the construction of computational grids. Despite the many works dealing with the construction of nets, their generation remains a complex and time consuming task in the case of higher dimensions and areas of complex configuration. Very often the resulting system of linear algebraic equations is ill-conditioned, which leads to an unjustified increase in the cost of its solution. Continuous solution recovery from its discrete approximation is a separate significant task.

Another problem is that differential equations and boundary conditions or initial conditions (or other ones) correspond to the object (phenomenon, process) being modeled approximately. In engineering applications, the accuracy of the differential model of 10-20% is rather common. In such a situation, an approximate solution of the corresponding differential equation with an accuracy of 0.001% is not only unjustified by practical needs, but can cause the researcher harmful illusions about the accuracy of the original object modeling.

To increase the accuracy of modeling it is necessary to consider all information available to researchers, information that can be initially incomplete but recharged during the simulation while the model can be adjusted and tuned in the process of monitoring the functioning real object. An experienced specialist (keyman) has some ill-formalized a priori information about the object under study, and the researcher should be able to take into account similar information in the process of model building.

Parallelizability and scalability of algorithms under consideration are separate problems.

[*]E-mail: `a.n.vasilyev@gmail.com`
[†]E-mail: `dtarkhov@gmail.com`

Their study and solution is a necessary prerequisite for their effective implementation on supercomputers. Often the problems of the computational process parallelization are solved in the programming phase although developing and implementing obviously parallelizable methods and algorithms is more effective.

Application of artificial neural networks (ANN) for modeling [1–5] is justified to solve the mentioned problems. Stability of neural networks with respect to potential errors in the data and the natural parallelization are an important incentive for it.

Hundreds of articles on this topic are being published in the world annually (see review [6]). Different versions of the collocations are employed in most studies, though all variants of the collocation method do not guarantee adequate model construction in the whole considered area. Also only the coefficients of the model are selected. Such an approach leads to the need for guessing the structure or its choice on the basis of general theoretical considerations. Excessive bulkiness of models and unjustifiably high computational costs in their construction are the result.

Methods presented in Russian and foreign literature depend heavily on the characteristics of the problem: shaped borders, nonlinear terms, phase transitions, chemical reactions, etc. Our methods [1] require only a small adaptation, allowing to take into account the specific characteristics of the problem being solved and to solve consistently direct, inverse problems, and the problem of constructing a family of solutions depending on some set of parameters.

Over-cost of supercomputer system time for solving a series of similar problems is the next challenge. Building a model on the supercomputer with its further use on the client computer is more efficient. In this case such a model should give opportunity for its correction on the basis of some additional information accounting. Reusability of the obtained models is very interesting and promising. In this regard parameterized models are of great interest [7, 8].

The task of constructing a parameterized model arises when you want to investigate the behavior of solution depending on a parameter, or to identify the parameter value according to the measurement data, or when the characteristics determining the modeled system are not exactly known, and are given by values distributed in some intervals (interval parameters).

2. Artificial neural network approach: neurocomputing

Let us explain the essence of our approach on the simplest boundary value problem

$$\begin{cases} A(u) = g, \ u = u(x), \\ x \in \Omega \subset R^p, \ B(u)\,|_{\Gamma} = f \end{cases} \tag{1}$$

where $A(u)$ is some differential operator, $B(u)$ is some operator that allows us to set the boundary conditions, Γ is the boundary of the domain Ω.

When the problem is modified so that its statement includes parameters $r = (r_1, \ldots, r_k)$, changing at some intervals: $r_i \in (r_i^-; r_i^+)$, $i = \overline{1, k}$, then the idea for the approximate solution of problems changes as well:

$$\begin{cases} A(u, r) = g(r), \ u = u(x, r), \\ x \in \Omega(r) \subset R^p, \ B(u, r)\,|_{\Gamma(r)} = f(r). \end{cases} \tag{2}$$

We are looking for an approximate solution of the problem (2) as the output of an artificial neural network (ANN) with a given architecture

$$u(x, r) = \sum_{i=1}^{N} c_i v_i(x, r, a_i).$$

ANN weights (linear input parameters c_i and nonlinear input parameters a_i) are determined in the process of stepwise neural network training based on some error functional minimization. Here the error functional is given in the form

$$J(u) = \sum_{j=1}^{M} |A(u(x_j, r_j)) - g(x_j, r_j)|^2$$

$$+\,\delta\sum_{j=1}^{M'}\left|B(u(x'_j,r'_j))-f(x'_j,r'_j)\right|^2$$

where $\{x_j,r_j\}_{j=1}^{M}$ are the periodically updatable test points in the region $\Omega(r_j)\times\prod_{i=1}^{k}(r_i^-;r_i^+)$, $\{x'_j,r'_j\}_{j=1}^{M'}$ are test points on the boundary $\Gamma(r'_j)$; $\delta>0$ is a penalty parameter.

Let us examine some examples.

1. Analysis of heat and mass balance in a porous catalyst flat pellet in the catalytic chemical reaction leads — in dimensionless variables — to the following nonlinear boundary value problem [9]: it is required to find a solution $y(x)$ of the ordinary differential equation

$$\frac{d^2y}{dx^2}=\alpha(1+y)\exp\left[-\frac{\gamma\beta y}{1-\beta y}\right]$$

with the boundary conditions

$$\frac{dy}{dx}(0)=0,\quad y(1)=0.$$

The above-mentioned neural network approach was applied to this nonlinear problem. Using a heterogeneous neural network with basic neuroelements of the form

$$v(x,\alpha,\beta,\gamma,a_{1i},a_{2i},...,a_{8i})=\exp\{-a_{1i}(x-a_{2i})^2\}\tanh\{-a_{3i}(\alpha-a_{4i})\}$$
$$\times\tanh\{-a_{5i}(\beta-a_{6i})\}\tanh\{-a_{7i}(\gamma-a_{8i})\}$$

showed to be the most effective.

The error functional $J(y)$ being minimized was given in the form

$$\sum_{j=1}^{M}\left(\left|y''(x_j,\alpha_j,\beta_j,\gamma_j)-\alpha_j(1+y(x_j,\alpha_j,\beta_j,\gamma_j))\exp\left[-\frac{\gamma_j\beta_j y(x_j,\alpha_j,\beta_j,\gamma_j)}{1-\beta_j y(x_j,\alpha_j,\beta_j,\gamma_j)}\right]\right|^2\right)$$
$$+\delta\sum_{j=1}^{M}\left(\left|y'(0,\alpha_j,\beta_j,\gamma_j)\right|^2+\left|y(1,\alpha_j,\beta_j,\gamma_j)\right|^2\right).$$

Calculations were performed for the following intervals of parameter variation

$$\alpha\in(0.05;0.15),\ \beta\in(0.4;0.6),\ \gamma\in(0.8;1.2).$$

Optimal values of the weights for the approximate neural network solution $y(x,\alpha,\beta,\gamma)$ were selected on the basis of minimizing the functional$J$ by the variant of dense cloud method which in this case was the most effective. For the neural network having $N=30$ neuroelements with the following parameter values: the size of the cloud $\varepsilon=0.03$, the penalty coefficient $\delta=1$, the number of test points $M=100$, the values of an approximate solution of the problem at the control points differ from the data given in the study [9] less than 2%.

2. Initial boundary value problem for the heat conduction equation, on the assumption that the thermal diffusivity r is known approximately, has the following form

$$u_t = ru_{xx},\ (x;t) \in (0;1) \times (0;T),$$
$$r \in (r^-;r^+),\ u(x,0,r) = \phi(x),\ x \in (0;1),$$
$$u(0,t,r) = u(1,t,r) = 0,\ t \in [0;T].$$

We seek an approximate solution of the problem in the form of a neural network output

$$u(x,t,r) = \sum_{i=1}^{N} c_i e^{-a_i(x-x_i)^2 - b_i(x-x_i)(t-t_i) - d_i(t-t_i)^2}$$

$$\times \tanh(p_i(r - r_i))$$

where $r \in (r^-;r^+)$.

Network training was carried out by minimizing the error functional:

$$J(u) \overset{def}{=} J(w) = J_1(w) + \delta_b J_b(w) + \delta_C J_C(w)$$

where $w = (w_1,..,w_N)$ is the vector of network weights;

$$J_1(w) = \sum_{j=1}^{N_1} \{u_t(\xi_j,\tau_j,\eta_j) - r_j u_{xx}(\xi_j,\tau_j,\eta_j)\}^2$$

is the term corresponding to the equation;

$$J_b(w) = \sum_{j=1}^{N_b} \left\{u^2(0,\tau_j,\eta_j) + u^2(1,\tau_j,\eta_j)\right\}$$

is the term corresponding to the boundary conditions;

$$J_C(w) = \sum_{j=1}^{N_C} \{u(x_j,0,r_j) - \phi(x_j)\}^2$$

is the term corresponding to the initial conditions (the same for samples with different values of the thermal diffusivity r); δ_b, $\delta_C > 0$ are penalty coefficients.

The variant of the method of growing networks with the rejection of some added elements is used for the selection of the ANN structure [1]. Neurocomputing results show that in the case of noisy Cauchy's data the continuation of the solution by the parameter r in a rather wide interval of variation also is possible.

3. Let us consider constructing a neural network model of the temperature field from the experimental data under the condition that cooling samples have different thermal diffusivity. In this case the initial temperature distribution is uniform.

The formal statement of the problem appears as follows:

$$u_t = ru_{xx},\ (x;t) \in (0;1) \times (0;T),\ r \in (r^-;\ r^+),$$
$$u(0,t,r) = u(1,t,r) = 0,\ t \in [0;T],$$
$$u(x_i,t_i,r_i) = f_i,\ i = \overline{1,\,p}.$$

A solution of the problem was sought in the form

$$u(x,t,r) = \sum_{i=1}^{N} c_i e^{-a_i(x-x_i)^2 - b_i(x-x_i)(t-t_i) - d_i(t-t_i)^2}$$

$$\times \tanh(p_i(r - r_i))$$

where $r \in (r^-;\ r^+)$. The ANN weight selection was accomplished through the minimization of the error functional which was obtained from the functional in the previous example by the replacement of last term with the following one

$$J_d(w) = \sum_{j=1}^{N_d} \{u(x_j,t_j,r_j) - f_j\}^2 .$$

In this problem neurocomputing results are similar to the results of the previous two examples.

3. Unified process

Building ANN solution to complex problems of mathematical modeling is based on the previously mentioned methodology [10]. This process can also be used to construct approximate non-neural network models. The main stages of this process are the following ones:

1. *Quality of the model is characterized by a functional (set of functionals).* This step is based on information about the models of the phenomena studied. (These models are specified in the process of solution building, object constructing and functioning.) This step can be implemented by an expert in the subject area.

2. *A functional basis (set of bases) is selected.* This step may be performed both by an expert in the subject area, using the information about the nature of the simulated phenomena, and automatically, using evolutionary algorithms (see works [1–3]). The neural network bases are effective in solving various problems and weakly depend on the characteristics of the problem itself.

3. *Methods of selecting parameters and structure of the model are chosen and implemented.* This stage can be fully automated and does not require participation of an expert in the subject area, although some available approximate expert information about the behavior of the object can be easily taken into account when constructing the model.

4. *Methods for verifying and refining models of objects during their functioning and appropriate tuning of control algorithms for the constructed models are implemented.* Principles of such methods construction are discussed below. These methods and algorithms can be implemented in a software package that will be used by specialists in application areas for a wide range of tasks.

5. *Databases of models, algorithms and programs are being replenished.*

4. Advantages and generalizations

The hierarchical parallelism of resulting algorithms is the undoubted advantage of the neural network approach.

At the lowest level the application of the learning algorithm requires the computations of neural network function values, usually (if gradient optimization algorithms are used) it is necessary to calculate the derivatives of neural network function according to its weights; the application of higher order algorithms may require the computations of relevant higher derivatives. These operations are parallelized in a natural way for neural network functions, which makes them highly scalable to multi-core computing systems. One should separately emphasize the possibility of adapting the structure of the network under the configuration of the computer system.

Applied to the next level, parallelization techniques of optimization algorithms are quite well researched, therefore we will not dwell on this issue.

At the third level there are the evolutionary algorithms allowing us to combine the selection of network weights and adapting its structure. Several of these algorithms are discussed below. Such algorithms as a rule tend to not only allow good parallelization, but they also can be effectively implemented in distributed systems. This issue is discussed further in the description of each algorithm.

The problem of predictive modeling of multi-scale processes in natural and technical systems posed above admits the following generalization comprising a lot of standard and non-standard tasks.

Let there be given a set of conditions

$\{A_q(u_1, u_2, \ldots, u_r)|_{\Omega_q} = 0\}_{q=1}^{Q}$ where Ω_q is a set on which the corresponding condition must be fulfilled and u_s, $s = \overline{1,r}$, are some unknown functions.

Operators A_q set equations (usually these are partial differential equations or their discrete approximation when considering the net-point method) and the boundary and other conditions such as conservation laws, equations of state, symmetry requirements or data derived from experiment, as well as other requirements for the solution of the problem. During training, operators A_q can vary, for example, include pieces of information incoming newly into examination or in the transition from the functional representation to the grid ones and vice versa.

We will seek every unknown function as an expansion:

$$u_s(x) = \sum_{i=1}^{N_s} c_{i,s}\phi_s(x; a_{i,s}),\ s = \overline{1,\, r}$$

where weights (parameters $a_{i,s}$) and coefficients $c_{i,s}$ are selected by minimizing the error functional composed of terms of the form

$$\sum_{j=1}^{M_q} \left| A_q(u_1, u_2, \ldots, u_r)(x_{j,q})|_{\Omega_q} \right|^2,$$

each entering into the sum with a weight factor $\delta_q > 0$ usually being fixed beforehand or recalculated from time to time on a certain procedure. Corresponding test points $x_{j,q} \in \Omega_q$ are selected randomly after a certain number of steps of the optimization algorithm. Terms in the functional need not be quadratic; they can be taken in a different form.

To resolve this problem, one can apply several fundamentally different options for the algorithm.

1. It is possible to compose a single functional using all conditions and to seek all parameters at once. This option is very demanding of computational resources.

2. You can make several functionals based on different sets of conditions, and adjust parts of weights minimizing each of them alternately. With a rational organization of calculations this option allows to speed up calculations, but a separate problem arises: a reasonable choice of the structure of the models used.

3. It is possible to apply one of the algorithms combining the selection of model parameters and its structure. This option allows you to get the most accurate and adequate solution to the problem assigned. Five approaches of this type are discussed below.

5. Evolutionary or metaheuristic algorithms

5.1. Generalized algorithm of clustering errors

Step 1. We are looking for a starting set of functions, so we perform several steps to minimize the error functional.

Step 2. We calculate the error for every function package on a test set, meanwhile the test sets are regenerated after each phase of training (passing a certain number of steps in the process of minimizing the functional). If the condition is not defined at some point, we assume that the error is equal to 0.

Step 3. We perform clustering collection of test point sets and corresponding errors in the appropriate space.

Step 4. For every cluster we build approximation giving the minimum error for the restriction of the error functional.

Step 5. We add the terms constructed at the preceding step to the functions required and repeat Step 1 with the set received.

Step 6. If the functional is insufficiently small, then we replenish the population of collections approximating the clusters and repeating Steps $1-5$ using a new sample.

It should be noted that the method proposed makes any special demands neither to a region form (connectedness, the possibility of decomposition) nor to the equation (linearity, real coefficients). However, the complication of such model data as the area shape and the equation leads to the difficulty of selecting the initial values of model parameters, to increase of the number of functions required to achieve the desired accuracy of the solution, and to corresponding slow down of the process of nonlinear optimization.

Calculations in Step 4 are independent for every cluster, so they can be implemented on different nodes of a computer system.

5.2. Generalized Schwarz's method

In this case, we essentially use the possibility to decompose the original domain into subdomains that intersect only at the parts of the boundaries.

Step 1. We act just as it was described in the previous approach **I** for the entire region: we build its own approximation for the solution in each subdomain using by definition of the error functional a corresponding part of the conditions, meanwhile placing the control point sets on the boundary where boundary conditions are known.

Step 2. Approximation for the unknown part of the boundary conditions at the boundaries of each subdomain occurs after a certain number of stages of training for each set of neural networks.

Step 3. Data exchange occurs — we introduce additional terms into each of the error functionals; these summands are determined by the information about a solution on the part of the boundary of subdomains where solutions have not been set. This piece of information is the solution built on another subdomain.

Step 4. The computational procedure is repeated a predetermined number of times or until the required level of accuracy is reached.

Modification of the Schwarz method seems more interesting in the case when the subdomains can not only have a common interface, but also intersect in sets of positive measure. In this case, information about some solution mismatch is introduced into the error functionals in Step 3 of the algorithm where the sampling points are taken at the intersection of the corresponding subdomains (this gives a smoother interfacing).

Calculation of approximate solutions for subdomains and data exchange in the construction of the solution for the entire domain can be implemented within a framework of grid-technologies. In this case, the solution for each subdomain is selected on its own computer taking into account the approximations of solutions at the intersection with neighboring areas. Pieces of information concerning these approximate solutions are gathered from time to time from corresponding computers.

5.3. Approach based on the group method of data handling (GMDH)

Step 1. We choose some set of suites of basis functions for each condition.

Step 2. We consider linear combinations of such suites and choose their coefficients by minimizing the error functionals built on the basis of sampling points from the corresponding subdomains.

Step 3. We select the best of the resulting functions in the sense of the best values of the error functionals that are based on sampling points from the subdomains intersecting (close to) the subdomains used in the previous step. It is possible to use several of these functionals for such a selection.

Step 4. We examine the linear combinations of resulting functions and repeat the previous steps tuning parameters of the models by some functionals and selecting the best performing models by other functionals until the error is sufficiently small.

Step 2 for different models is implemented independently, therefore it can be effectively implemented in a multiprocessor and distributed system.

5.4. Specialized genetic algorithm for constructing a set of neural networks using domain decomposition

Step 1. We build an ANN-population via some given number of solution approximations for each subdomain using the corresponding part of the conditions by the error functional definition; in this case, as in the second approach (**II**), the boundary conditions are used only on the part of the boundary: control point sets are taken on the border where the boundary conditions are known.

Step 2. For further work we select the best model of each group on the basis of the error functional minimum; the control points in the functional are taken from the complement to the area used in the previous step.

Step 3. We produce random mutations of the models; the probability of these mutations is the greater, the bigger are errors on their own area and on their complement area. These mutations can be of different types: removal of a term with a minimum coefficient of the basis function in the sum for the given model or deleting a random term, addition of the basis function with random internal parameters, random change of network weights to a certain value, etc.

Step 4. We carry out crossing — we take a specified number of best models (in the sense of minimizing the corresponding functional), we choose two of them, and we take some terms from one model and additional part of terms from the other model, the result is a new model called *a descendant* that replenishes a set of selected models. This operation is repeated with some set of pairs of such models. In this case some part of descendants is produced from models of one population (models trained on the same set), other part is produced from models of different populations. The resulting descendants complement each population to its former size.

Step 5. We repeat the previous steps a certain number of times or until the error functional throughout entire region for some collection of models becomes sufficiently small.

This genetic approach is easily adapted to distributed computing. The most natural variant arises if each collection of models is selected in its node. In this case only some models (sets of parameters and information about the structure) or parts of models for the crossing must be transferred between nodes. In addition, some information may be sent, for example, the values of the optimizable functionals. If there are few nodes, several populations can be trained on a single node; it seems to be reasonable implementing a more intense hybridization between them than between populations from different nodes. If there are many nodes, then some part of the population or even individual elements can be placed on each of them. This will not too strongly affect the speed of calculations, especially in the case of large models, as the most time-consuming operation (parameter selection) is implemented locally.

5.5. Training collective of ANN-models

Step 1. We select a specified number of sets of neural network functions minimizing the single error functional.

Step 2. For each subdomain we choose a set of models for which the error functional built on the basis of sampling points corresponding to this subdomain is the minimal one.

Step 3. We select each suite on its own subset Ω_q including terms that are responsible for the mismatch at the joints in the error functional minimized (see approach **II**).

The result of training is the approximate solution which is given in each subregion by the corresponding model.

This algorithm allows such a modification in which the area decomposition is not given a priori and it is produced naturally in the course of the algorithm, i.e., the subregion includes the points for which the error functional for a given set of functions is smaller than the functional for other sets of functions in the population.

Just like the previous one, the approach **V** admits a reasonably efficient distributed implementation. The simplest option is to train a separate set of functions, which turned out to be the best on some subset, on each node. In this case you only need to send on the information about the behavior of the approximation on the joints and only from those nodes which correspond to the region with the mating area for this node. It is also a possible variant of the algorithm when one node corresponds to several sets of functions. If you have got many nodes, then you can teach one set of functions on several nodes implementing some distributed algorithm.

If all the sets which the sampling points are taken from are finite, the different variants of the net-point method are obtained by an appropriate choice of the operators defining the conditions, and this fact forms the basis for the development of hybrid methods.

6. Conclusion

The above evolutionary approaches (approaches **IV** and **V** in the first place) allow us to use models built for close typical problems as elements of an initial population. Accumulation of data base of built models (model hierarchies) or trained neural networks for common tasks should be one of the most important activities in the field of modeling. We can use data from such a storage ready-made or refine them to solve new problems taking these basic models as initial approximation.

Approach which examines the whole *hierarchy of models* at once as both differential and functional including all available background information, allowing the evolution of the models at any level and the ability to enter new data into consideration, can be considered the most promising.

Natural parallelism of models considered and methods for constructing such models allows to effectively implement them on modern GPUs on the basis of CUDA technology.

References

[1] A.N. Vasilyev, D. A. Tarkhov. *Neural Network Modeling. Principles. Algorithms. Applications.* (SPbSPU Publishing House, Saint-Petersburg, 2009). (in Russian)

[2] D.A. Tarkhov. *Neural Networks: Models and Algorithms. Applications. Vol. 18.* (Radiotechnika Publishing House, Moscow, 2005). (in Russian)

[3] D. A. Tarkhov. Neural networks as a mathematical modeler. Neurocomputers: development, application. No. 2, 3-48 (2006). (In Russian)

[4] A.N. Vasilyev, D.A. Tarkhov. New neural network technique to the numerical solution of mathematical physics problems. I: Simple problems. Optical Memory and Neural Networks (Information Optics). **14**, no. 1, 59-72 (2005).

[5] A.N. Vasilyev, D.A. Tarkhov. New neural network technique to the numerical solution of mathematical physics problems. II: Complicated and nonstandard problems. Optical Memory and Neural Networks (Information Optics). **14**, no. 2, 97-122 (2005).

[6] K. Manoj, Y. Neha. Multilayer perceptrons and radial basis function neural network methods for the solution of differential equations: A survey. Computers and Mathematics with Applications. **62**, no. 10, 3796-3811 (2011).

[7] A. N. Vasilyev, D.A. Tarkhov. Parametric neural network models of classical and non-classical problems for heat conduction equation. St. Petersburg State Polytechnical University Journal: Physics and mathematics. **153**, no. 3, 136-144 (2012). (In Russian)

[8] A.N. Vasilyev, D.A. Tarkhov. Mathematical models of systems with interval representation of parameters on the basis of heterogeneous neural networks. Temperature field continuation – classical problem statement. Neurocomputers: development, application. No. 11, 56-59 (2012). (in Russian)

[9] T.Y. Na. *Computational methods in engineering boundary value problems.* (ACADEMIC PRESS, A Subsidiary of Harcourt Brace Jovanovich, Publishers, New York, London, Toronto, Sydney, San Francisco, 1979).

[10] A.N. Vasilyev, V.P. Osipov, D.A. Tarkhov. Unified process of constructing neural network model hierarchy and adequacy problem in mathematical modeling. Neurocomputers: development, application. No. 7, 20-28 (2010). (in Russian)