ATLANTIS PRESS

Advances in Intelligent Systems Research, volume 166

7th Scientific Conference on Information Technologies for Intelligent Decision Making Support (ITIDS 2019)

Investigation of the Functional Stability of Neural Network Algorithm for Solving the Ordinary Differential Equations

Irina Bolodurina
Orenburg State University
Federal Research Centre of Biological Systems and Agrotechnologies RAS
Orenburg, Russia
prmat@mail.osu.ru

Lubov Zabrodina
Department of Applied Mathematics
Orenburg State University
Orenburg, Russia
zabrodina97@inbox.ru

***Abstract*—The paper analyzes the neural network approach for solving the Cauchy problem of ordinary differential equations of the first order, based on the representation of the function as a superposition of elementary functions. The use of neural network approach allows obtaining the desired solution in the form of a functional dependence, which satisfies the required conditions of smoothness. On the basis of a two-layer perceptron, a model of neural network solution of the problem and a numerical algorithm implementing the search for a solution are constructed. The software-algorithmic solution of the Cauchy problem is obtained. To determine the stability of the neural network approach, a series of experiments were conducted to find a solution to a particular Cauchy problem of ordinary differential equation of the first order with an analytical solution. The study shows that the considered neural network algorithm has no functional stability. This may be due to the problems of weights minimization, scalability in network training and other factors.**

Keywords—ordinary differential equations, artificial neural network, optimization methods, functional stability

I. INTRODUCTION

Currently, neural networks are widely used in various fields of science. Such networks are used for solving problems of pattern recognition, time series prediction, network optimization, packet routing, etc. This is due to the fact that neural networks represent a flexible set of tools for solving a variety of data processing and analysis tasks.

Artificial neural networks are universal approximants of complex (nonlinear) functional dependencies. Their most important feature is the stability of the neural network model in relation to data errors. Such errors include boundary perturbations, calculation errors, etc. Another important feature of artificial neural networks is the possibility of parallelizing the solution of the problem, as well as the use of a set of networks (including networks of different types). In this matter, the use of neural network technologies in solving various problems is an object of great interest. The effectiveness of using this approach for solving ordinary differential equations (ODE) is expressed in the possibility of representing a solution in the functional dependence form, which satisfies smoothness conditions.

Currently, the apparatus of solving ODE by numerical methods is widely developed: the finite difference method, the finite element method, the finite volume method, the boundary element method, etc. Numerical methods allow obtaining the desired solution in a discrete form, which is not always sufficient. The main difference between neural network and numerical approaches is that neural network technology allows obtaining a functional representation of the desired solution.

The study analyzes the functional stability of the algorithm based on the use of modern neural network technologies for solving the problem Cauchy of ODE of the 1st order.

The rest of the paper is organized as follows. Section 2 presents the results of a re-view of the literature devoted to the consideration of various approaches to solving differential equations based on artificial neural networks. Section 3 describes the basic idea of the neural network approach. Section 4 is devoted to the description of the mathematical formulation of the problem and the functional representation of the desired solution. In Section 5, the initial formulation of the problem reduces to the problem of minimizing the error in the approximation of the resulting solution. Section 6 presents a step-by-step algorithm for the neural network approach for solving the first-order Cauchy problem. Section 7 is devoted to describing the practical implementation of the presented algorithm and evaluating the results.

II. RELATED WORK

The neural networks have properties that allow performing optimization procedures. Thus, in the article [1] the authors Bolodurina I. P. et al. use the neural network model to optimize the network operation in the infrastructure of the virtual data center. In this regard, we consider the works that allow for neural network optimization of the search for an approximate solution of differential equations.

For example, scientists J. Mead et al. consider the construction of a direct and non-iterative feedforward neural network with approximate arbitrary coefficients for the solution of ordinary differential equations. The developed method uses a hard limit transfer function, but the construction requires imposing certain constraints on the values of the input, bias, and output weights, and the attribution of certain roles to each of these parameters [2].

M. Kjaramonte et al. in their research describe the need for more complex activation functions to solve the ODE. The approach used makes it possible to obtain results comparable in error with numerical methods; however, the complete algorithm for solving ODE by means of neural networks is not presented [3].

Other researches use the Kansa method to solve partial differential equations (PDE) by using neural networks, the trust region method of solving problems of minimizing the quadratic error functional is expounded. Yu.N. Zemskova et al. conducted the experiments that combine the above methods for nonlinear parameters. The results of the operation of the algorithms show the effectiveness of the application of neural networks with radial basis functions in the solution of PDE [4].

In the study of A.N. Kovalenko et al. are used methods for solving partial differential equations using radial basis function networks (RBF networks), feedforward neural networks and a modified neural network. The stages of the neural network approach, the adjustment of weights and the problems arising during the research were studied [5].

In the monograph "Principles and Techniques of Neural Network Modeling" by A.N. Vasilyev et al. outlined the neural network approach, which makes it possible to solve problems in a uniform manner, without fundamentally reconstructing the algorithms. The problem of choosing a functional basis of a neural network is solved, methods for selecting parameters and structure of a neural network model are described. The authors consider the practical application of the approach to the heat equation [6].

In the study of the N. Yadav et al. develop an artificial neural network trained by a modified error back propagation algorithm, which is capable of adjusting parameters for modeling the dynamics of the heat equation even with large changes in the boundary conditions without knowledge of system equations [7].

In the study of A.N. Vasilyev et al. others considered in detail the problem of finding a function for which the dynamic equation in a certain region and its values in a certain set of points are known. The contribution of the authors of the article was to build a mathematical model of the neural network on heterogeneous data, including differential equations and experimental observations [8].

In the other work, A.N. Vasilyev et al. was research the multicomponent systems in the study of regions of a composite type with discontinuous coefficients are pre-sented in the. Authors examined neural network approaches to solving boundary problems of mathematical physics in multidimensional composite domains [9].

In the one more work, A.N. Vasilyev et al. considered a method for solving problems in mathematical physics based on the use of normalized radial basis function networks, using the example of elliptic and parabolic problems with known analytical solutions. To solve non-stationary problems, a hybrid algorithm that combines neural network approximation in space with finite-difference time-domain approximation is used [10].

A review of studies has shown that the neural network approach is actively used to solve various kinds of differential equations. Currently, there are methods for solving specific types of problems and a General idea of the solution scheme for the problems, but there is still a need for detailed analysis of the developed algorithms. In this regard, this research aims to study the algorithm for solving the Cauchy problem for ODEs of the first-order.

In the article [11] I. E. Lagaris and others stated the basic idea of neural network approach to the solution of both ODE and DU in partial derivatives. The authors consider the practical application of the developed approach to different types of problems and confirm the effectiveness of its use. The difference between this study and [11] is the analysis of functional stability of the neural network algorithm for solving the Cauchy problem for the first-order ODE and the study of the network overtraining problem, including determining the dependence of the solution on the parameters used.

In the article [12] S. A. Fedosin and others analyzed various methods of neural network training, including iterative gradient minimization algorithms. The main factors affecting the performance of the algorithm in the study are the local minimum of the function, the step size, and the network paralysis, which is caused by too large values of the weight coefficients, while most neurons function in the area where the growth of the function is small. Therefore, it is necessary to take into account the influence of various factors on the stability of the solution while developing the algorithm.

A review of studies has shown that the neural network approach is actively used to solve various kinds of differential equations. Currently, there are methods for solving specific types of problems and a General idea of the solution scheme for the problems as a whole, but a detailed analysis of the developed algorithms is required. In this regard, this paper is aimed at studying the functional stability of the algorithm for solving the Cauchy problem for the first-order ODEs.

III. WORK BASE

The basis for the application of artificial neural networks as universal approximations is Kolmogorov's theorem that any function can be represented as a superposition of continuous functions [13].

In terms of the theory of neural networks, this theorem is formulated as follows: Any continuous function of several variables can be realized with any accuracy by means of a two-layer neural network.

To apply this property to the solution of the Cauchy problem for ODE of the first order, we consider a two-layer neural network with m inputs and one output.

For definiteness, assume that the transfer functions of the first and output layers are sigmoid.

If any numbers $x_1, \dots, x_n$ are input, then at the output we get the value of some function $y = F(x_1, \dots, x_n)$, which is the response of the neural network. It is clear that the result depends on the input signal, and the value of internal parameters of the network, which are the weight coefficients of the neurons.

The exact form of the function:

$$F(x_1, \dots, x_n) = \sum_{i=1}^{m} w_i^{(1)} \cdot \sigma\left(\sum_{j=1}^{m} x_j \cdot w_{ji}^{(0)}\right), \qquad (1)$$

$$\sigma(s) = \frac{1}{1 + e^{\alpha s}}, \quad (2)$$

where $\sigma(s)$ is the sigmoid activation function; $w_{ji}^{(0)}$ are the weight coefficients of the input layer; $w_i^{(1)}$ are the weight coefficients of the hidden layer; x_j are input signals.

IV. Neural Network Approach To Solving The Problem

In the proposed approach a recurrent neural network for constructing an approximate solution is used, and a parameter vector $\vec{p}$ corresponds weights and additional parameters of the neural architecture.

We seek the solution $u_t(\vec{t})$ in such a way that it immediately satisfies the boundary conditions from the construction. This condition is achieved by representing the solutions as a sum of two terms:

$$u_t(\vec{t}) = A(\vec{t}) + F\left(\vec{t}, N(\vec{t}, \vec{p})\right), \quad (3)$$

where $N(\vec{t}, \vec{p})$ is the output of a single-layer recurrent network with the parameter vector $\vec{p}$ and n input signals determined by the input vector $\vec{t}$.

The function $A(\vec{t})$does not contain adjustable parameters and satisfies the boundary conditions. The second term in the representation of the solution is that the function F is constructed in such a way that it does not contribute to the boundary conditions, since the function $u_t(\vec{t})$ must satisfy them. This term uses the neural network, weights and rules, which must be corrected when solving the minimization problem.

At this stage, the problem is reduced from the initial problem of bounded optimization to unbounded optimization because of the choice of the form of the desired solution, which satisfies the boundary conditions by construction.

V. Introduction To The Problem Of Minimization

The effective minimization can be considered as a training procedure for a neural network, where the error corresponding to each input vector $\vec{t}_i$, is equal to the value, which must be identically equal to zero. The calculation of this error value includes not only the output of the network (as in the case of conventional training), but also the derivative of the output with respect to any of its input signals. Therefore, when calculating the error gradient with respect to the weight coefficients, it is necessary to calculate the gradient of the output function of the neural network in the domain as well as the gradient with respect to its input data (parameters).

Consider in general a multilayer perceptron with n input signals, one hidden layer with m sigmoid activation functions and a linear output block. Of course, one can expand the neural network into more than one hidden layer, to solve ordinary differential equations of a higher order.

For a given vector of input signals $\vec{t} = (t_1, \dots, t_n)$, the output of the neural network is:

$$N = \sum_{i=1}^{m} w_i^{(1)} \cdot \sigma(z_i), \quad (4)$$

and z_i is the total input sign of the neural network, having the form:

$$z_i = \sum_{j=1}^{N} w_{ij}^{(0)} \cdot t_j + \tau_i, \quad (5)$$

where $w_{ij}^{(0)}$ denotes the weight from the neuron of the input layer i to the neuron of the hidden layer j; $w_i^{(1)}$ denotes the weight from the neuron of the hidden layer j directed to the output of the network result; τ_i means the offset of the signal of hidden layer i; $\sigma(z_i)$ is a sigmoid activation function.

Gradient of the network output function in the direction of the input signals of the network will have the form of:

$$\frac{\partial^k N}{\partial t_j^k} = \sum_{j=1}^{m} w_i^{(1)} \cdot {w_{ij}^{(0)}}^k \cdot \sigma_i^k, \quad (6)$$

where $\sigma_i^k = \sigma(z_i)$ and σ^k denotes the k-th order derivative of the sigmoid.

In addition, the gradient of the network output function with the respect to all input signals has the form:

$$\frac{\partial^{\lambda_1}}{\partial t_1^{\lambda_1}} \frac{\partial^{\lambda_2}}{\partial t_2^{\lambda_2}} \dots \frac{\partial^{\lambda_n}}{\partial t_n^{\lambda_n}} N = \sum_{j=1}^{m} w_i^{(1)} \cdot P_i \cdot \sigma_i^{(\Lambda)}, \quad (7)$$

where

$$P_i = \prod_{k=1}^{n} {w_{ik}^{(0)}}^{\lambda_k} \quad (8)$$

$$\Lambda = \sum_{i=1}^{n} \lambda_i. \quad (9)$$

The expression (7) shows that the derivative of the network output function at any of its inputs is equivalent to a recurrent neural network $N_g(\vec{t}_i)$ with one hidden layer which has the same values for the weights $w_{ij}^{(0)}$ and interference τ_i, and with each weight $w_i^{(1)}$ replaced by $w_i^{(1)} \cdot P_i$. In addition, the activation function of each neuron of the hidden layer is replaced by a Λ-th derivative of the sigmoid.

Therefore, the gradient N_g with the respect to the parameters of the original network can be easily obtained in the form of:

$$\frac{\partial N_g}{\partial w_i^{(1)}} = P_i \cdot \sigma_i^{(\Lambda)}, \quad (10)$$

$$\frac{\partial N_g}{\partial w_i^{(1)}} = P_i \cdot \sigma_i^{(\Lambda)}, \quad (11)$$

$$\frac{\partial N_g}{\partial w_{ij}^{(0)}} = t_j \cdot w_i^{(1)} \cdot P_i \cdot \sigma_i^{(\Lambda+1)} + w_i^{(1)} \cdot \lambda_j \cdot {w_i^{(1)}}^{\lambda_j - 1} \left(\prod_{k=1, k \neq j}^{n} {w_{ik}^{(0)}}^{\lambda_k} \right) \sigma_i^{(\Lambda)}. \quad (12)$$

Once the derivative of the error calculation function with respect to the network parameters is determined, then it is enough to use almost any minimization method. For example, you can use the steepest descent method, the conjugate gradient method, and other methods proposed in the theory of optimization methods. In this paper, we used a gradient projection method that is quadratic convergent and it demonstrates high performance.

The gradient projection method is based on finding the extreme point of the function in the direction of the anti-gradient. The application of this method to the quadratic functions in the field of real numbers defines a finite number of steps.

It should also be noted that for a given grid of points, the derivatives of the output function of the neural network (or gradient network) with the respect to the parameters can be obtained even in the case of algorithms for parallelizing computations. Also, in the case of an error back propagation algorithm, an online or batch mode for updating the weight can be used.

VI. Algorithm Neural Network Approach For Solving An Ode Of The 1st Order

We consider the application of the neural network approach described above to solve the Cauchy problem for the firs-order ODE, which have the form:

$$\begin{cases} \frac{du}{dt} = f(t,u) \\ u(a) = A \\ t \in [a,b] \end{cases}. \quad (13)$$

We will search the solution of the problem (13), in accordance with the theory above, in the form (3):

$$u_t(t) = A + t \cdot N(t,\vec{p}), \quad (14)$$

where $N(t,\vec{p})$ is the output of the recurrent neural network with one input signal t and the parameters of the network $\vec{p}$, which includes the weight coefficients, the learning rate, the number of input signals and the gradient descent multiplier. Note that $u_t(t)$ satisfies the Cauchy problem in construction.

It is natural that the desired solution in the form of (14) will have its accuracy. The error representing the difference between the found solution and the exact one will be determined as follows:

$$E[\vec{p}] = \sum_i \left\{ \frac{du_t(t_i)}{dt} - f(t_i, u_t(t_i)) \right\}^2, \quad (15)$$

where t_i are the points on the interval [a,b].

Since the following holds:

$$\frac{du_t(t)}{dt} = N(t,\vec{p}) + t \cdot \frac{dN(t,\vec{p})}{dt}, \quad (16)$$

then it is possible to calculate the gradient of the network output function with respect to the vector of its parameters $\vec{p}$ using (7) – (12), depending on the neural network structure used to solve the ODE. The same applies to all subsequent task models.

Algorithm of neural network approach

Step 1:

Select the test set $\{t_j\}$of the input signals by dividing the definition area into n equidistant points. Generate random weight coefficients of layers $w_{ji}^{(0)}, w_i^{(1)}$.

Step 2:

Calculate the total input feature of the system:

$$\Sigma_i = \sum_j w_{ji}^{(0)} \cdot t_j. \quad (17)$$

Pass the total feature and the set of input signals $\{t_j\}$ to the output layer.

Step 3:

For updating weight coefficients ($w_{ji}^{(0)}, w_i^{(1)}$), use gradient descent for minimization of calculation errors:

$$E[\vec{p}] = \sum_i \left\{ \frac{du_t(t_i)}{dt} - f(t_i, u_t(t_i)) \right\}^2 \to \min. \quad (18)$$

Thus, if the error of the neural network increased (the network was overtrained), then finish the series of gradient descent.

Step 4:

We obtain an analytic representation of $u_t(t)$ in the form (14) and the approximation error $E[\vec{p}]$.

End.

VII. Result Of Implementation

Let us investigate the developed neural network algorithm for solving the first-order Cauchy problem for the error of the obtained result, comparing it with the fourth-order Runge-Kutta method, and also the dependence of the neural network approach on the network parameters used.

Consider specific examples of the Cauchy problem for ODE of the 1st order with the analytical solution.

A. The problem of retraining the neural network on a particular example

When developing a neural network algorithm for solving the Cauchy ODE problem of the 1st order, the condition for the exit from the gradient descent (search for weight coefficients) is of great importance. This is due to the fact that the usual convergence condition of the gradient method can not be applied, since in this case the neural network can well recognize examples from the training set, but does not acquire the generalization property. In this case, it is said that the neural network is retrained. Consider this property on a particular example of the Cauchy ODE problem of the 1st order.

Let it be necessary to solve the Cauchy problem:

$$\begin{cases} \frac{du}{dt} + 2tu = e^{-t^2} \\ u(0) = 10 \\ t \in [0,1] \end{cases}. \quad (19)$$

Analytical solution of the problem (19) has the form:

$$u = te^{-t^2} + 10e^{-t^2}. \quad (20)$$

As a result of the neural network algorithm, changing the condition of the exit from the gradient descent, we obtained:

1) for $Count_{N41} = 150$:

$$\Delta u_{41} = 0{,}24960104, \qquad Neur = 4.$$

where Δu_{41} - neural network error; $Count_{N41}$ - number of iterations;$Neur$ - number of neurons (figure 1).

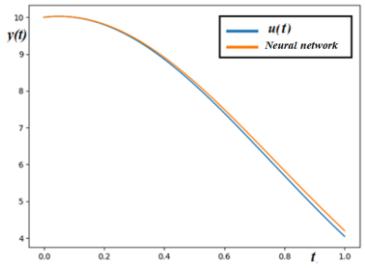

Fig. 1. Graphs of the obtained solutions (1)

2) for $Count_{N42} = 250$:

$$\Delta u_{42} = 25{,}21320845, \quad Neur = 4.$$

where Δu_{24} - neural network error; $Count_{N24}$ - number of iterations; $Neur$ - number of neurons (figure 4).

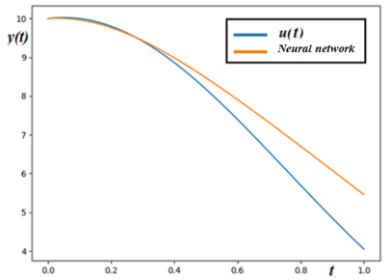

Fig. 2. Graphs of the obtained solutions (2)

Thus, it can be concluded that it is necessary to further study the dependence of the parameters of the neural network for each solved Cauchy ODE problem. The parameter can serve as the number of neurons, organized within the constraints, and other factors, whose change exposes the result of even a small oscillation.

In view of the possible retraining of the neural network, in the gradient descent when obtaining the weight coefficients, it is necessary to take into account the error of the deviation of the solution found in parallel to the error of the coefficients. If the neural network reaches the optimal solution, the gradient descent cycle is completed.

B. Functional stability check of the obtained solution

Due to the fact that neural networks vary depending on the initial conditions and adapt to the solution, it is quite clear that even if one condition changes, the results of the algorithm will be different. Thus, the question arises: how stable is the solution?

To study this algorithm of neural network approach, it was decided to analyze the cumulative quadratic errors of deviation of the obtained functions from the exact solution.

The quadratic error function takes the form:

$$\mathrm{Err} = \sum_i \{u_t(t_i) - u(t_i)\}^2, \quad (21)$$

where $u_t(t_i)$ is the solution found, $u(t_i)$ is the exact solution.

To determine the stability, a series of 40 experiments was carried out. In these experiments all the parameters of the neural network remained the same, only the weight coefficients were changed. Weight coefficients were generated randomly to start the network operation.

It is obvious that a normal distribution law for the quadratic deviations is required for the stability of the solutions. For the obtained errors, we check the presence of the Gauss distribution in the Statistica data analysis package.

The normal distribution law is not fulfilled because the probability p confirming the normal distribution hypothesis is less than 0.5. In this regard, it is not possible to talk about the stability of the neural network solution of the 1st order ODE.

TABLE I. EXPERIMENTS OF SOLUTION ERRORS MEASURING

N	Δu	N	Δu	N	Δu	N	Δu
1	0,100276	11	0,59066	21	0,100018	31	0,957556
2	0,100242	12	0,623046	22	0,413206	32	0,617833
3	0,606373	13	0,353352	23	0,479846	33	0,586435
4	0,95674	14	0,504536	24	0,691906	34	0,145875
5	0,632925	15	0,304263	25	0,102284	35	0,106547
6	0,573263	16	0,365975	26	0,549262	36	0,449785
7	0,157128	17	0,934656	27	0,522658	37	0,467890
8	0,254465	18	0,537054	28	0,426408	38	0,276640
9	0,683996	19	0,483135	29	0,52114	39	0,265545
10	0,158925	20	0,419194	30	0,48198	40	0,397358

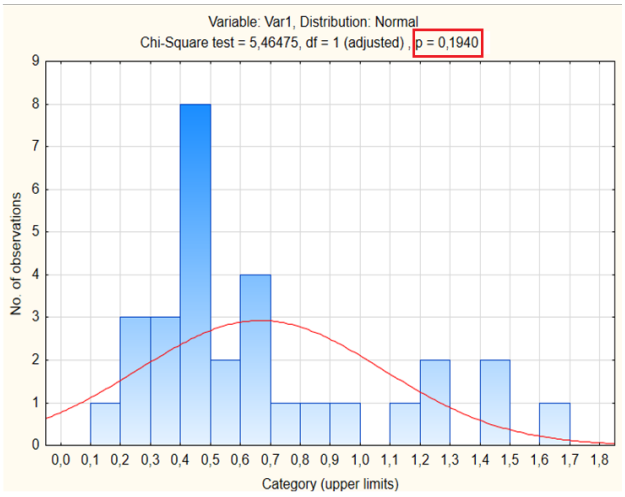

Fig. 3. Checking the Normal distribution law

Thus, the studied neural network algorithm for solving ODE of the 1st order has no functional stability, which may be due to the problems of weights minimization, scalability

in network training and other factors. Currently, there are many modifications of gradient methods: the conjugate gradient method, the Levenberg-Makar method, the fast propagation method, the delta-bar-delta, etc. However, all of them are appropriate for a specific solution model depending on the structure of the neural network and its characteristics. In this regard, it is necessary to take into account the stability problems of the solution while using the neural network algorithm in practice.

VIII. CONCLUSION

The neural network approach, based on the representation of the function in the form of a superposition of elementary functions for the Cauchy problem of first-order ODE, as well as the algorithm for solving the problem were discussed. The functional stability of the obtaining solution was examined.

The following areas of further research are promising:

- study of the impact of the structure, parameters and types of activation functions of the neural network on the solution;
- improvement of the presented algorithm at the stage of finding coefficients (providing the solution to the overfitting problem and stability of the obtaining result).

ACKNOWLEDGMENTS

The researches were carried out in accordance with the plan of research work in 2019-2020 years of FSSI «Federal Research Centre of Biological Systems and Agro-technologies of the RAS» (No. 0761-2019-0004).

REFERENCES

[1] I.P. Bolodurina, D.I. Parfenov, “Neural network model for optimize network work in the infrastructure of the virtual data center”, 25th Telecommunications forum TELFOR : materials of forum, pp. 21-22, Belgrade, Serbia , 2017.

[2] Jr. Meade, A. Fernandez, “The Numerical Solution of Linear Ordinary Differential Equations by Feedforward Neural Networks”, Mathematical and Computer Modelling, vol. 19, Issue 12, 1994, pp. 1–25 .

[3] M. Chiaramonte, M.Kiener, “Solving differential equations using neural networks”, Machine Learning Project, 2013, pp. 1-5 .

[4] Yu. Zemskova, V. Gorbachenko and E. Artyukhina, “Using the method of confidence neighborhoods for solving partial differential equations on neural networks with radial basis activation functions, ” Scientific reports. № 9 (258), 2017, pp. 103-110 (in Russian).

[5] A.N. Kovalenko, A.A. Chernomorets, M.A. Petina, “On the neural networks application for solving of partial differential equations, ” Scientific reports. Economy. Informatics, № 9(258), 2017, pp. 103-110.

[6] A.N. Vasilyev, D.A. Tarkhov, “Principles and techniques of neural network modeling, ” SPb.: Nestor-History, 2014, 218 pp. - ISBN 978-5-4469-0474-7 (in Russian).

[7] N.Yadav, A. Yadav, M. Kumar, “An Introduction to Neural Network Methods for Differential Equations,” SpringerBriefs in Applied Sciences and Technology, 2015, 114 pp. ISBN-13: 978-9401798150.

[8] A.N. Vasilyev, D.A. Tarkhov, “Construction of approximate neural network models from heterogeneous data”, Mathematical modeling, 19(12), 2007, pp. 43-51 (in Russian).

[9] A.N. Vasilyev, “Neural network approaches to solution of boundary problems in multidimensional composite areas”, News TSURE. № 9, 2004, pp. 80 – 89 (in Russian).

[10] A.N. Vasilyev, D.A. Tarkhov, “Neural network as a new universal approach to the numerical solution of problems of mathematical physics”, Neurocomputers: development, application. M.: Radio Engineering, №7-8, 2004, pp. 111-118 (in Russian).

[11] I.E. Lagaris, “Artificial Neural Networks for Solving Ordinary and Partial Differential Equations”, IEEE Transactions on Neural Networks № 5, 1998 , pp. 9

[12] S.A. Fedosin, D.A. Ladyaev, O.A. Maryina, "Analysis and Comparison of Teaching Methods for Neural Networks", University Bulletin of Mordovia, № 4, 2010, pp. 79-88 (in Russian).

[13] A.N. Kolmogorov, “On the representation of continuous functions of several variables in the form of superpositions of continuous functions of one variable and addition, ” Dokl. T. 114, № 5, 1957.- pp. 953-956 (in Russian).