Hybrid Neural Network Identifying Complex Dynamic Objects: Comprehensive Modelling and Training Method Modification

Victoria Vysotska[1], Serhii Vladov[2], Ruslan Yakovliev[2] and Alexey Yurko[3]

[1] *Lviv Polytechnic National University, Stepan Bandera Street 12, Lviv, 79013, Ukraine*
[2] *Kremenchuk Flight College of Kharkiv National University of Internal Affairs, Peremohy Street 17/6, Kremenchuk, 39605, Ukraine*
[3] *Kremenchuk Mykhailo Ostrohradskyi National University, University Street 20, Kremenchuk, 39600, Ukraine*

Abstract

The article to the development of mathematical models of complex dynamic objects (using the example of helicopter turboshaft engines) in the form of recurrent neural networks and their use in complex modelling to identify the parameters of automatic control, monitoring, and diagnostic systems is described. For the first time, a concept has been created for constructing neural network models of complex dynamic objects, in which, by increasing the robustness of the functioning of a trained neural network, it becomes possible at its output to increase the reliability of solving problems of identifying complex dynamic objects. The use of a hybrid neural network NARX with a radial-basis nonlinear layer is proposed. The introduction of a radial basis nonlinear layer into the NARX neural network is an effective addition when working with unstructured data such as images, audio signals, or text due to its ability to extract and represent complex patterns and features in these data. This is confirmed by the results of modelling the losses of the neural network, which turned out to be stable over 500 epochs of its training and did not exceed 0.025 (2.5 %). A comprehensive modification of the Levenberg-Marquardt training method is proposed, which consists of a particular application of the Broyden method for calculating the elements of the Hessian matrix, as well as an analytical description of the regularization parameter through the use of control coefficients for increasing or decreasing its value in the event of a neural network training error. The use of the modified Levenberg-Marquardt method made it possible to reduce the average training error of the NARX hybrid neural network with a radial basis layer by 33 % to the level of 0.025.

Keywords

Neural network, helicopters turboshaft engines, complex dynamic objects, hybrid neural network NARX, radial-basis nonlinear layer, Levenberg-Marquardt method, training, Broyden method, loss

1. Introduction

Experimental research and modelling of complex dynamic objects, for example, helicopter turboshaft engines (TE) and their control systems, are constant elements of knowledge of their behaviour throughout the entire life cycle. This starts from the design stage, fine-tuning, and certification, and ending with operation and disposal. Such studies require a special integrated modelling technology creation. It makes it possible to confirm the reliability, operability and required characteristics of systems both before putting them into operation and in operating modes [1, 2]. Today, the development of the industry is based on technologies of digital manufacturing, computer modelling, machine learning, cloud computing, and cyber-physical systems. The *digital twins* concept is being fully implemented. This is a virtual representation of a physical object not only at the stages of design, development, and commissioning but also throughout the entire life cycle, including operation and disposal [3, 4].

CMIS-2024: Seventh International Workshop on Computer Modeling and Intelligent Systems, May 3, 2024, Zaporizhzhia, Ukraine

victoria.a.vysotska@lpnu.ua (V. Vysotska); ser26101968@gmail.com (S. Vladov); director.klk.hnuvs@gmail.com (R. Yakovliev); yurkoalexe@gmail.com (A. Yurko)
0000-0001-6417-3689 (V. Vysotska); 0000-0001-8009-5254 (S. Vladov); 0000-0002-3788-2583 (R. Yakovliev); 0000-0002-8244-2376 (A. Yurko)

The technology of semi-natural modelling of complex control objects has been used for a long time in many industries, where real systems are coupled with mathematical models of control objects. However, the methods and tools for creating such models often remain the same [5, 6].

One of the most pressing and important problems is to ensure the adequacy of the model of a complex dynamic object in the automatic control, monitoring, and diagnostics system. The operation of control, monitoring, and analysis algorithms at the same time can cause various *collisions* that need to be taken into account and modelled when developing and configuring a control system [7]. In addition, in the process, complex dynamic objects gradually exhaust their resources, and their characteristics begin to degrade [8, 9]. In the process of analysis and synthesis of automatic control systems, the need arises to correct and adapt the existing model of a complex dynamic object for its effective operation. To solve this problem, adaptive models are needed that are identified by the real characteristics of the object and its operating conditions.

The widespread application of intelligent technologies utilizing neural networks has been observed in the research and development of sophisticated control and monitoring systems tailored for complex dynamic objects [10, 11], including helicopters TE [12, 13]. However, the task remains of the adequacy and applicability of mathematical models of complex dynamic objects in operating modes, which are mostly presented in the form of fast-calculating piecewise linear dynamic models [14, 15].

The research aim is to increase complex modelling and testing efficiency of a real automatic control system (ACS) for complex dynamic objects, monitoring, and diagnostics through the use of nonlinear dynamic mathematical models and their systems in the form of neural networks (using the example of helicopters TE). A scientific concept is proposed for the neural network model constructing for a complex dynamic object (using the example of helicopters TE), including algorithms for training and identifying a mathematical model of the engine using real data with a choice of the structure and size of the neural network.

2. Related works

Currently, neural networks are an effective means of solving a wide range of problems in complex dynamic objects identifying and their control systems [16, 17]. They are distinguished by the simplicity of the architecture and high representative power. The quality of operation of these networks largely depends on the efficiency of data clustering, as a result of which the centres of activation functions and their dispersion are determined. The works of individual domestic and foreign scientific schools are devoted to the issue of automating the neural network architecture selection. Also, this work focuses both on local modifications of training algorithms [18, 19] and on the use of bionic models [20, 21] to optimize the number of neurons in the hidden layer [22, 23]. The latter has good potential. Because it leads to growing interest in the use of distributed intelligent systems to optimize neural network architecture [24, 25]. An alternative to the described solutions is methods based on special approaches to density clustering. As a result, in hidden layer the optimal neurons number is determined and their key characteristics are established [26, 27]. A common disadvantage of most known solutions is the requirement for the completeness of processed samples. This makes the use of such methods for working in systems with dynamically changing data, for example in control systems, ineffective compared to such specialized neuroarchitectures as Jordan networks [28, 29], Elman networks [30, 31] or recurrent multilayer perceptron [32, 33].

3. Methods and materials

To solve the above task of complex dynamic objects and their control systems, an intelligent system (Fig. 1) can be used that implements the Fault Detection and Identification Method (FDI) [34]. It is based on a neural network mathematical model of the research object and an identification block [35, 36]. Such a system makes it possible to detect and classify abnormal operating modes of the research object, measuring channels and actuators under operating conditions. The output parameters of the mathematical model can be used to diagnose abnormal operating conditions of the research object based on a comparison of the matching the computed parameters with the observed ones. Additionally, they used and also as to restore lost data of measuring channels in the event of their failure being detected. Such a model should have several special properties, the most important of which are [37, 38]:

1. The model must describe the properties of the research object that determine the non-stationary nature of work processes. This means the need to use a dynamic model.
2. The structure of the mathematical model of the research object should provide the practical possibility of its functioning in combination with mathematical models of other elements of the system.

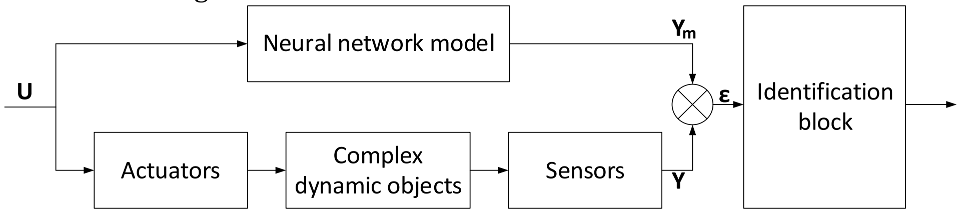


Figure 1: Structure of an intelligent system for identifying complex dynamic objects and their control systems based on the FDI method (author's research, based on [24–36])

The mathematical representation of the nonlinear dynamic model of complex dynamic objects, taking [39] into account, can be represented as a differential equations system:

$$\begin{cases} \dfrac{\Delta\partial y_1(t)}{\partial t} = a_{11} \cdot y_1(t) + \cdots + a_{1n} \cdot y_{n-1}(t) + b_{11} \cdot x_1(t) + \cdots + b_{1m} \cdot x_m(t), \\ \dfrac{\Delta\partial y_2(t)}{\partial t} = a_{21} \cdot y_1(t) + \cdots + a_{2n} \cdot y_{n-1}(t) + b_{21} \cdot x_1(t) + \cdots + b_{2m} \cdot x_m(t), \\ \cdots \\ \dfrac{\Delta\partial y_n(t)}{\partial t} = a_{n1} \cdot y_1(t) + \cdots + a_{nn} \cdot y_{n-1}(t) + b_{n1} \cdot x_1(t) + \cdots + b_{nm} \cdot x_m(t). \end{cases} \quad (1)$$

Currently, neural network models built on linear neural networks, for example, multilayer perceptron, have been acquired for the identification of complex dynamic objects [40, 41]. Neural network models based on linear neural networks (as multi-layer perceptron) may face several disadvantages when modelling complex dynamic objects. First, they are limited in their ability to capture complex nonlinear relations between variables. This can lead to insufficient accuracy in predicting or modelling. In addition, multi-layer perceptron can suffer from the problem of gradient damping when training deep models. This makes training difficult and can lead to low performance in practice. Also, they can be prone to overtraining in the presence of a limited amount of training data. This makes them less effective for processing real dynamic systems. Therefore, it is expedient to use dynamic recurrent neural networks, for example, recurrent multilayer perceptron (NARX). The justification for the transition from a multilayer perceptron to a recurrent multilayer perceptron (NARX) is based on several factors and is given in Table 1, and the scientific features of this transition are in Table 2.

Table 1
Factors for the transition from a multilayer perceptron to a hybrid NARX network for the identification of complex dynamic objects (author's research)

Factor	Description
Complexity of the system	Complex dynamical systems have non-linear behaviour and relations between different parameters. Multilayer perceptron may not be flexible enough to model such a complex system.
Taking into account the dynamics	The NARX hybrid network provides an opportunity to take into account the dynamic properties of the system based on time dependencies between inputs and outputs. This is especially important for predicting the parameters of the work process of the research object, which may change over time.
Handling uncertainty	Complex dynamic objects are exposed to various external influences and changes in operating conditions. Hybrid NARX networks can better adapt to data uncertainty and provide more accurate predictions under different conditions.
Use of contextual data	NARX hybrid networks allow you to include both current parameter values and historical data in the model. This can be useful for analyzing previous system states and identifying patterns of parameter changes

Table 2
Scientific substantiation of the peculiarity of the transition from a multilayer perceptron to a hybrid NARX network (author's research)

Factor	Description
Provision no. 1. Innovative modelling of complex dependencies through a combination of nonlinear autoregressive models and radial basis functions	
Combining nonlinear autoregressive models and radial basis functions	Nonlinear autoregressive NARX models are capable of capturing complex dynamic relationships between input and output variables. Nonlinear functions bring flexibility to the model, which allows you to adapt to different forms of data, which is especially important when modelling input parameters with a nonlinear nature.
Effective modeling of complex dependencies	The workflow parameters of complex dynamic objects can exhibit complex, non-linear dependencies that are best described by the hybrid NARX network. This makes it possible to more accurately predict changes in parameters under different operating conditions.
Provision no. 2. Feasibility of solving the problem of identification of complex dynamic objects using the NARX hybrid network and comparing its results with the use of a multilayer perceptron	
Modelling accuracy	Comparison of accuracy coefficients, such as root mean square deviation, on training and test samples.
Generalization of new data	It is estimated how well the model generalizes its knowledge to new data not used during training.
Stability	Analysis of the stability of models in various conditions, including changes in external parameters.
Provision no. 3. Simultaneous use of simulation results using multilayer perceptron and hybrid NARX network in operational conditions	
Using a multilayer perceptron	They are used to quickly assess current input parameters and provide a response to rapidly changing conditions.
Using the hybrid NARX network	They are used for deeper analysis of dynamic changes in parameters, taking into account time dependencies and predicting future values.
Conclusions	
The combined use of a multilayer perceptron and a hybrid NARX network allows combining the advantages of both models, providing more accurate and flexible real-time identification for complex dynamic objects.	

In [40], the use is justified by a modified version of a recurrent multilayer perceptron (NARX). It is a dynamic network characterized by the delay of output/input signals combined in a network input vector, with a radial basis nonlinear and a linear recurrent layer. It should be noted that [40] uses a Gaussian NARX framework with input data regressor selection using a modified gradient method from [41]. This modification is justified based on the outdated NARX models with outdated machine training models [42]. The modified NARX structure proposed in [40] consists of two parts: nonlinear and linear blocks (Fig. 2, where σ_i is i-th element radial function width; $c_{i1}, c_{i2}, \ldots, c_{in}$ are i-th element coordinates centre; $u_1, u_2, \ldots, u_n$ are the input signals).

Such a model in a neural network form with feedback makes it possible to take into account the nonlinear dynamic characteristics of an object and guarantee the structural and parametric adequacy of its analytical model. The vector **u** placed to the input has the form $u(t) = [1, u(t), u(t-1), \ldots u(t-N_u), y(t-1), \ldots, u(t-N_y)]^T$, where N_u – is the input signal delays number, N_y – is the output signal delays number [40]. Depending on the complex dynamic object model, the vector **u** is formed according to the parameters specified in the technical specifications.

According to [33, 34, 37], the network output vector has the following mapping [40]:

$$y(t+1) = f\left(u(t), y(t-1), \ldots, y(t-N_y), u(t-1) \ldots, u(t-N_u)\right), \qquad (2)$$

then the NARX hybrid network is characterized by a set of numbers (N_u, N_y, N_i), where N_i is neurons in the i-th hidden layer number.

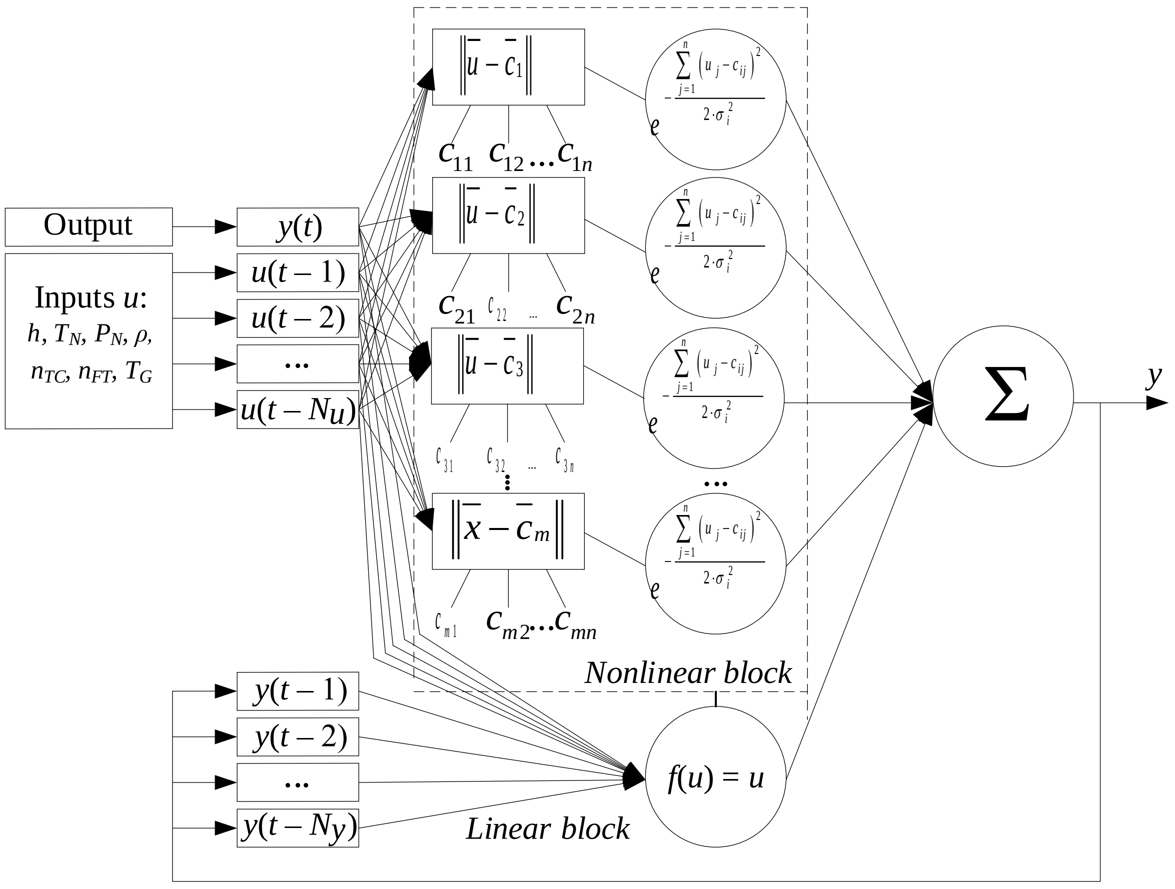


Figure 2: Gaussian NARX architecture using modified neural network (author's research [40])

To the aforementioned, the comprehensive schematic representation of configuring the neural network model parameters for complex dynamic objects (using the example of helicopters TE [40]) is presented in Fig. 3, where ; Δw_{ij} is the neural network synaptic connections increase; $\mathbf{Y} = (y_1, y_2 \ldots, y_m)^{\mathrm{T}}$ is the object output parameters vector; $\mathbf{U} = (u_1, u_2 \ldots, u_m)^{\mathrm{T}}$ is the input influences vector; $\mathbf{Y}^{NN} = (y_1^{NN}, y_2^{NN}, \ldots, y_n^{NN})^T$ is the neural network outputs vector [40].

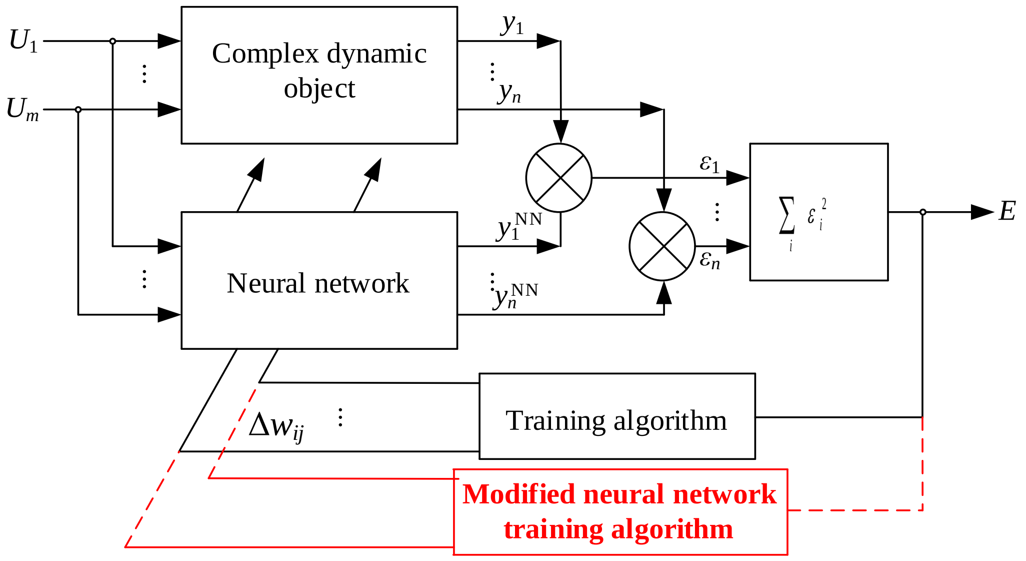


Figure 3: Complex dynamic objects neural network model sheme (author's research [40])

Modifications of training of neural networks are conducted to improve performance or adjust the model for the specific requirements of the task (Fig. 3). Modifying a neural network training includes changing its structure based on adding or removing layers and neurons, changing activation functions, and adjusting hyperparameters. Also, this includes training rate or regularization parameters, and introducing additional training methods, such as data augmentation or based on pre-trained models for transferring training. Modification may also be necessary if the input data characteristics change, the performance requirements change, or the problem that the neural network needs to solve changes. Control influences vector conversion into initial parameters vector is elucidated by operator **F** [40]:

$$\mathbf{Y} = \mathbf{F}(\mathbf{U}). \tag{3}$$

The identifying helicopter ЕУ task using a neural network can be formulated: using the results of the proposed training process for a neural network. It forms vectors ($\mathbf{U}_i$; $\mathbf{Y}_i$) *training set* obtained experimentally for a separate engine instance. Aim is to find operator $\mathbf{F}^{NN}$ within neural network architectures class. The **F** operator approximation by $\mathbf{F}^{NN}$ operator is deemed optimal if a specified functional from the difference ($\mathbf{Y} - \mathbf{Y}^{NN}$) does not surpass a given small value ε_{add}, defining the **F** operator approximation accuracy [40]:

$$E = \|\mathbf{Y} - \mathbf{Y}^{NN}\| = \sum_{i}^{n} \varepsilon_i^2 \leq \varepsilon_{add}. \tag{4}$$

The condition (4) satisfaction is guaranteed by neural network training. It involves finetuning parameters using the training sample {(**U**, **Y**)} and is verified on a meticulously organized *test sample* [40]. A scientific concept of direct neural network model construction based on complex dynamic objects is proposed, which is shown in Table 3.

Table 3
The scientific concept of the step-by-step creation of a neural network model of complex dynamic objects (author's research)

Step	Description
1	Development of unique criteria and metrics for evaluating the effectiveness of identification of complex dynamic objects. Justification of the need to define goals to ensure the accuracy and objectivity of the assessment.
2	The rationale for selecting a particular neural network architecture and identifying its integration point within the complex dynamic object identification system.
3	Analysis and justification of the network training algorithm choice considering the specifics of the task to achieve optimal adaptive training.
4	Description of conducting experiments on a digital model using additional resulting results to create a training sample and taking into account new criteria and metrics to improve model accuracy.
5	Network training process description using formed training sample and training algorithm.
6	Justification for the neural network simplifying and reducing to achieve optimal information storage and efficient operation with a minimum number of parameters.
7	Justification of measures aimed at increasing the robustness of the functioning of the trained neural network model, taking into account possible challenges and unexpected situations.
8	Modelling and testing algorithms description for monitoring the operational status and operation for complex dynamic objects management, including ACS based on a neural network.
9	Justification of the choice of software or hardware development of a neural network for implementation in a real system of identification of complex dynamic objects.

The proposed scientific concept of the step-by-step creation of a neural network model of complex dynamic objects defines clear stages of intelligent systems development and implementation. The results of the work of each stage (the development of evaluation criteria, the selection of a network structure, the analysis of training algorithms, and experimental research on a digital model) form a reliable basis for the creation of an effective intelligent monitoring system. In particular, the justified reduction of the neural network and measures to increase the robustness of the model indicate a desire for optimal efficiency. The final stages (modelling and testing algorithms, as well as the choice of software or hardware implementation) emphasize the practical suitability of the concept for implementation in a real intelligent system for the identification of complex dynamic objects.

For the practical implementation of the proposed scientific concept of the step-by-step creation of a neural network model of complex dynamic objects (Table 3), attention should be drawn to the indicator of model robustness (*generalization ability*). This is the stability of modelling results to input data disturbances. Evaluation of the resilience or generalization capability of a neural network model is conducted using an algorithm grounded on incremental multi-criteria training [44].

Complex dynamic object model training using a neural network (Fig. 2) is performed by sequentially presenting pre-prepared delay vectors and corresponding output values while simultaneously adjusting the weights of hidden layers by a certain procedure [45, 46]. The neural network training process includes the application of the Levenberg-Marquardt method. The method combines the steepest descent methods (i.e., minimizing the training error along a gradient) and Newton's method (i.e., using a quadratic model to accelerate the quest for the minimum of the error function [47]. The Levenberg-Marquardt method is intended for optimizing nonlinear regression models of the form $F(\mathbf{u}) = \sum_{i=1}^{N}(f_i(\mathbf{u}) - y_i)^2$. As an optimization criterion, it uses the model mean square error (MSE) on the set training, which it minimizes [40].

The Levenberg-Marquardt method combines the ideas of the Gauss-Newton method and gradient descent. At each iteration, this method updates the parameters $\mathbf{u}$ in this way:

$$\mathbf{u}_{k+1} = \mathbf{u}_k - (J^T J + \lambda_k I)^{-1} J^T \Delta\mathbf{y}, \tag{5}$$

where J – is the Jacobian matrix of $J_{ik} = \frac{\partial f_i}{\partial u_k}$, describing partial derivatives of a model concerning parameters; $\Delta\mathbf{y}$ – is the vector representing the disparity between the present model values and the real data; I – is the identity matrix; λ_k – is the regularization parameter that controls the step size at each iteration.

According to the research of Serhii Parkhomenko [47, 48], most often, to search for the Jacobian matrix, various variations of difference formulas for calculating derivatives are used, including the central difference derivative [49]. According to [47, 48] for the hybrid NARX network y_i values can be written as:

$$y(u,w) = f_1\left(\sum_{i=1}^{N_i} \widetilde{w}_i \cdot f_2\left(\sum_{j=1}^{N_u} w_{ij} \cdot u_j + \sigma_i\right) \cdot f_3\left(\sum_{l=1}^{N_l} w_{il} \cdot u_l + \sigma_i\right) + \tilde{\sigma}\right), \tag{6}$$

where $y(u, w)$ shows the dependence of an output value y_i on input parameters vector values u and corresponding weighting coefficients w, u_j – is the value received by the j-th input neuron, w_{ij} – is the weight coefficient connecting the j-th input neuron with i-th nonlinear hidden layer neuron, σ_i – is the bias coefficient for i-th nonlinear hidden layer neuron, $\widetilde{w}_i$ – is the weight coefficient connecting i-th neuron of the nonlinear hidden nonlinear layer with the output neuron, $\tilde{\sigma}$ – is the bias coefficient for the output neuron, $f_1(\bullet)$, $f_2(\bullet)$, $f_3(\bullet)$ – are the activation functions for the output neuron and neurons of hidden nonlinear and linear layers, respectively, N_u – is the number of the input, N_i and N_l – are the hidden nonlinear and linear layers neurons number, respectively. The radial basis function is chosen as the activation function for neurons of the output and nonlinear hidden layers according to [40] and for neurons of the linear hidden layer – a linear function.

According to [47, 48] when calculating the Hessian matrix, it is convenient to use the formula $\mathbf{H} = J^T J$, which is derived from the premises:

- the function $y(u, w)$ has a low order of nonlinearity: the second derivatives $\frac{\partial^2 y(u,w)}{\partial w_i \partial w_j}$ do not take very large values;
- matrix $\mathbf{H}$ is considered in a small neighbourhood of the minimizing vector w, for which the $y(u, w)$ values are close to the desired $f_i(\mathbf{u})$, i.e. $|f_i(\mathbf{u}) - y(u,w)| \approx 0$.

With an efficient operation of scalar matrix multiplication, the search for $\mathbf{H}$ is fast, while the time for calculating the vector of weight changes depends on the variable number w_{ij}. Experiments show [47, 48] that it is rarely necessary to solve more than three systems per iteration, which does not greatly affect the execution time of the algorithm. Calculating the Jacobian matrix takes up the bulk of the work of the Levenberg-Marquardt algorithm. So reducing the cost of searching for it speeds up the neural network training. One of these methods is to abandon the calculation of the completely accurate matrix J in favour of its approximate version.

For example, the Broyden method calculates J_{n+1} using the matrix J_n calculated at step n according to the formula [47, 48, 50]:

$$J_{n+1} = J_n + h^T \cdot \frac{y(u,w_{n+1}) - y(u,w_n) - J_n \cdot h}{h^T \cdot h}, h = w_{n+1} - w_n, \tag{7}$$

From a theoretical point of view, using this approach at each step of the Levenberg-Marquardt algorithm makes sense. However, in practice, the approximation becomes coarser over time. This affects the $J^T E$ gradient vector and requires re-calculation of the Jacobian matrix using more accurate methods after an unsuccessful selection of a vector of weight changes δ. Analytical calculation of partial derivatives improves the accuracy of calculations. This allows you to shorten the process by reusing intermediate data and reducing the number of calls to complex functions. Similar to [47, 48], the following substitution was made in this work:

$$s_i = \sum_{j=1}^{N_u} w_{ij} \cdot u_j + \sigma_i, \quad S = \sum_{i=1}^{N_i} \tilde{w}_i \cdot f_2(s_i) \cdot f_3(s_i) + \tilde{\sigma}\,; \ 1 \le j \le N_u. \tag{8}$$

Thus, we get:

$$\begin{gathered}\frac{\partial y(u,w)}{\partial w_{ij}} = u_j \cdot [E \cdot C \cdot s_i], \frac{\partial y(u,w)}{\partial \sigma_i} = [E \cdot C \cdot s_i], \frac{\partial y(u,w)}{\partial \tilde{w}_i} = \frac{[E \cdot C'] \cdot [E \cdot s_i]}{1 + [E \cdot s_i]}, \\ \frac{\partial y(u,w)}{\partial \tilde{\sigma}} = [E \cdot C'], C = \frac{\partial^2 C}{\partial S^2} \cdot [E \cdot S], \frac{\partial^2 C}{\partial S^2} = \alpha \cdot [E \cdot C'], \frac{\partial C}{\partial s_i} = \frac{\tilde{w}_i \cdot [E \cdot C'^2] \cdot [E \cdot s_i]}{(1 + [E \cdot s_i])^2}.\end{gathered} \tag{9}$$

The analytical method of calculating the Jacobian matrix is not applicable in all cases, since for each new neural network model it is necessary to revise the formulas. However, it requires less computation than using central difference derivatives while maintaining accuracy. Distributing calculations of the rows of matrix J between threads allows it to be filled in parallel since the elements are calculated independently. This corresponds to the ribbon pattern. According to [48] τ – is the maximum number of threads running simultaneously. For $\tau \ll N$, each thread on average processes about $\frac{N}{\tau}$ rows of matrix J. Row J represents the minimum unit of processing within a parallel block, Ξ – set of numbers of rows J processed by thread t. Since $H^* = J^T J$, then, according to [48], to save memory, you don't have to store the J^T and J matrices separately. For any $n \in [1, N]$ and $m \in [1, M]$, the equality $J_{nm} = J^T_{nm}$ is true, which allows you to get elements J^T and J by swapping the indices. To calculate the Hessian matrix element according to [48]

$$H^*_{pq} = \sum_{i=1}^{N_u} J^T_{qi} \cdot J_{ip} = \sum_{i=1}^{N_u} J_{iq} \cdot J_{ip}\,; 1 \le p \le M; \ 1 \le q \le M. \tag{10}$$

you need to know all the elements of the p-th and q-th columns of J. The simplest way is to wait for an entire matrix J to be calculated, but this causes a synchronization point where all processes must wait for the calculation to complete before continuing. Decomposing the matrix H^* into a sum allows us to avoid this [48]:

$$H^* = \sum_{t=1}^{\tau} \left({}^{[t]}H\right) = {}^{[1]}H + {}^{[2]}H + \dots + {}^{[\tau]}H = \left\{{}^{[t]}H_{pq}\right\}_{M \times M}, \tag{11}$$

where ${}^{[t]}H$ – is the matrix of the cumulative sum of flow t, ${}^{[t]}H_{pq}$ – is the element of this matrix.

For each calculated row of matrix J in [48], all its elements are multiplied by each other, which leads to obtaining a matrix term:

$${}^{[t]}H^+_k\Big|_{ij} = J^T_{ik} \cdot J_{kj} = J_{ki} \cdot J_{kj}; k \in \Xi, 1 \le i \le M; 1 \le j \le M, \tag{12}$$

where ${}^{[t]}H^+_k\Big|_{ij}$ – is the element of the matrix ${}^{[t]}H^+_k$.

By combining the matrices of one stream, a cumulative sum matrix is formed in the form:

$${}^{[t]}H = \sum_{k \in \Xi_i} {}^{[t]}H^+_k, \tag{13}$$

which, when added with matrices from other streams outside the parallel region, results in a finite Hessian matrix H^*.

Thus, combining the calculations of row J and the matrix H within one parallel block was achieved by distributing the calculations of a scalar matrix product over ${}^{[t]}H$. Calculation $g = J^T \tilde{\varepsilon}(w)$ also requires all elements of the p-th column of matrix J [48]:

$$g_p = \sum_{i=1}^{N_u} J_{pi}^T \tilde{\varepsilon}(w) = \sum_{i=1}^{N_u} J_{ip} \tilde{\varepsilon}(w), 1 \leq p \leq M. \tag{14}$$

By decomposing the vector g into cumulative vectors ${}^{[t]}g$ for each thread, and then decomposing them into term vectors ${}^{[t]}g_k^+$ for all rows J processed within a single thread, we obtain a similar calculation method used to calculate H^*:

$${}^{[t]}g_k^+\Big|_i = J_{ik}^T \tilde{\varepsilon}(w) = J_{ki} \tilde{\varepsilon}(w), g = \sum_{k \in \Xi_i} {}^{[t]}g_k^+, g = \sum_{t=1}^{\tau} {}^{[t]}g, \; k \in \Xi_i, 1 \leq i \leq M. \tag{15}$$

After processing of the string J and all related calculations (10–15) is completed, it loses its relevance, and, therefore, storing it in RAM, as well as the entire matrix J as a whole, becomes redundant. It is enough to carry out a line-by-line search in several threads to obtain an array of partial derivatives for weighting coefficients, using pairs of input and expected values, which provides significant memory savings.

In the dynamic approach, to select the regularization parameter λ_k at each iteration of the Levenberg-Marquardt method, it is proposed to use an algorithm that adaptively adjusts its value depending on the change in error at the current and previous iterations. Let ΔE_k be the change in error between the current and previous iterations, that is, $\Delta E_k = E_k - E_{k-1}$, where E_{k-1} is error function value at previous iteration $k - 1$and E_k is the error function value at current iteration k,. Then we can determine the new value of Λ_k as follows:

$$\Lambda_k = \begin{cases} \alpha \cdot \lambda_k, \text{if } \Delta E_k > 0, \\ \beta \cdot \lambda_k, \text{if } \Delta E_k < 0, \\ \lambda_k, \quad \text{if } \Delta E_k = 0, \end{cases} \tag{16}$$

where α and β – are the coefficients that control the increase or decrease in the value of Λ_k in the event of a change in error. For example, $\alpha > 1$, $0 < \beta < 1$. This approach allows the regularization parameter Λ_k to be adaptively changed depending on the direction of error change at each iteration. This can help speed up the convergence of the method and improve its efficiency.

An efficient choice of the initial value of the regularization parameter λ_k in the Levenberg-Marquardt method can be essential to ensure fast convergence and avoid potential problems such as overfitting or underconvergence. Mathematically, it can be done this way:

1. The analytical method is based on c analytically estimating the initial value of the regularization parameter. Also, it takes into account the characteristics of the error function and gradient. For example, if the error function has large values, the initial value of λ_k can be chosen relatively large to ensure the stability of the algorithm. If the error function has small values, the initial value of λ_k can be chosen relatively small to allow the algorithm to converge quickly.
2. Heuristic method – heuristics or rules of thumb can be used to select the initial value λ_k. For example, the initial value of λ_k may be chosen to be a small fixed value based on an assumption of typical parameter scales or error function values.
3. Optimization method – optimization methods can also be used to select the optimal initial value of λ_k that minimizes the error function at the initial stage. For example, one can use grid search or optimization methods such as gradient descent to select the initial value of λ_k in such a way as to minimize the error function.

Choosing an effective initial value for the regularization parameter λ_k can significantly impact the Levenberg-Marquardt method performance. So, it is important to pay attention to this and apply appropriate mathematical techniques to ensure the optimal choice.

Since an error function sometimes contains several local minima or has different scales of change, it is advisable to use the multiscale optimization method to adaptively adjust the step in different parameter directions. At each iteration of the optimization algorithm, a base step size η is selected and used to update the parameters based on an error function gradient. For each parameter u_i, its characteristic scale of change Δu_i is calculated, for example, as the standard deviation or scale of a change in the parameter at previous iterations. The step size η is adaptively adjusted for each parameter u_i by its characteristic scale of change, allowing us to take into account scale differences between parameters and adapt the step size for each parameter:

$$\eta_i = \frac{\eta}{\Delta u_i}. \quad (17)$$

These parameters are updated using the adapted step size η_i:

$$u_i^{(k+1)} = u_i^{(k)} - \eta_i \frac{\partial E}{\partial u_i}. \quad (18)$$

This process allows the step size to be adaptively varied in different parameter directions, which can improve the convergence of the optimization algorithm and help avoid getting stuck in local minima or jumps in the error function due to large-scale parameter differences. Taking into account the above, (5) is rewritten as:

$$\mathbf{u}_{k+1} = \mathbf{u}_k - \left({}^{[t]}H + \Lambda_k I\right)^{-1} J^T \Delta\mathbf{y} + \eta_i \frac{\partial E}{\partial u_i}. \quad (19)$$

Expression (19) is the modified Levenberg-Marquardt method. The Levenberg-Marquardt algorithm adjusts the neural network weights using a quadratic approximation of the error surface. This approximation ensures that the minimum is quickly found, but the risk of hitting a local extremum on the training surface increases.

4. Experiment

To conduct a computational experiment, the TV3-117 TE was selected as the research object. The helicopter TE model parameters include atmospheric parameters (ρ – is the air density, P_N – is the pressure, T_N – is the temperature, and h – is the flight altitude). The helicopter onboard parameters (T_G – is the gas temperature in compressor turbine front, n_{FT} – is free turbine rotor speed, n_{TC} – is the gas generator rotor r.p.m) are normalized to absolute values (Table 4) based on V. Avgustinovich theory [51, 52].

Table 4
The training set part (author's research [51, 52])

Number	n_{FT}	n_{TC}	T_G
1	0.943	0.929	0.932
2	0.982	0.933	0.964
3	0.962	0.952	0.917
4	0.987	0.988	0.908
5	0.972	0.991	0.899
…	…	…	…
256	0.981	0.973	0.953

In complex dynamic object identification problems, error surfaces with numerous plateaus and valleys are often encountered, which makes the local minima task one of the main difficulties in achieving maximum efficiency. To overcome this problem, two heuristic approaches were developed and tested to prevent the search process from becoming trapped in local minima [53].

According to [53], the first heuristic requires the algorithm to take *risky* steps in a random direction along the error surface with increasing step length, to local minimum *jump out*. After several unsuccessful attempts, in this case, 4, the minimum found is considered the smallest, and the algorithm completes its work by the rules of the original algorithm. This heuristic was tested on 4 data sets, including 64 input numerical variables and 1 output variable. For each data set, the optimal neural network architecture is determined, including the number of neurons in the hidden layer, while minimizing the error. This architecture was identified by an exhaustive search of options, varying from 1 to 20 neurons in the nonlinear layer and from 1 to 10 neurons in the linear layer. After this, similarly [53], 50 test runs were carried out. Each of these included 10 neural network training with different initialization of weights for each data set, without using heuristics. At the final stage, another 50 test runs were performed, within each of which 10 networks were trained with different initialization of weights for each data set, using the heuristics proposed in [53]. The results of testing the neural network are shown in Table 5.

Table 5
Neural network testing results (author's research, based on [53])

Data set number	Standard deviation of traditional Levenberg-Marquardt method			The standard deviation of the modified Levenberg-Marquardt method		
	Minimum	Maximum	Average	Minimum	Maximum	Average
Data set 1	6.95	18.38	12.67	2.84	16.12	9.48
Data set 2	7.30	22.42	14.86	5.98	19.76	12.87
Data set 3	0.22	0.64	0.43	0.14	0.20	0.17
Data set 4	4.54	7.12	5.83	3.68	5.22	4.45

Thus, the use of heuristics increases the likelihood of finding successful solutions. However, it is worth noting that in some cases this can lead to a loss of a previously found optimal solution, followed by *getting stuck* in a local minimum discovered later. This disadvantage can be surmounted by augmenting the random step number, which, in turn, requires additional computing resources [53]. The recommended 4 attempts provide the optimal balance for our subject area. According to [53], the second heuristic is to change the neural network weights when a local minimum is reached, calculating the weighting coefficients using the formula:

$$w_{ij} = w_{ij} + \theta \cdot w_{ij}, \tag{20}$$

where $\theta_1 \leq \theta \leq \theta_2$ – random number. According to [53], the θ_1 and θ_2 values are chosen through empirical means as the outcome of an experiment. Various values were tested in the range [–0.5; 0.5] with steps of 0.05 [54]. As a result, the following optimal values were obtained: $\theta_1 = -0.035$, $\theta_2 = 0.035$.

Thus, when a local minimum is reached, the weights of the neural network undergo random changes by an amount not exceeding θ in both directions. Large values of θ often reset the previous training phase, requiring a neural network to recover and start training again. This, in essence, leads to the fact that the heuristic goes into the mode of a series of tests with different initial weights. Small values of θ may prevent the network from exiting the local minimum. And it will cycle back to an original minimum without benefiting from the heuristic. After *shaking* the scales, training continues according to the rules of the original algorithm. If the new minimum is not reached, a random change is made again. Through experimentation, it was revealed that the optimal number of random changes should not exceed three. To evaluate the effectiveness of the 2nd heuristic, the same data sets were used as the first. The optimal neural network architectures found at the previous stage were also used. Similarly, 50 test runs were carried out: 10 trainings of neural networks with different initialization of weights for each data set, using the second heuristic. The results of testing the neural network are shown in Table 6.

Table 6
Neural network testing results (author's research, based on [53])

Data set number	Standard deviation of traditional Levenberg-Marquardt method			The standard deviation of the modified Levenberg-Marquardt method		
	Minimum	Maximum	Average	Minimum	Maximum	Average
Data set 1	6.95	18.38	12.67	2.84	16.12	9.48
Data set 2	7.30	22.42	14.86	5.98	19.76	12.87
Data set 3	0.22	0.64	0.43	0.14	0.20	0.17
Data set 4	4.54	7.12	5.83	3.68	5.22	4.45

Thus, the second heuristic, like the first, significantly increases the probability of detecting a global minimum. But it has the same drawback as the first – in some instances, it may result in the loss of a previously found optimal solution. In addition, its use requires an empirical selection of boundaries for changing the parameter θ.

5. Results

Labview created specialized software that uses a modified Levenberg-Marquardt algorithm and automates the recording of experimental results (Fig. 4, 5). The program implements the following single-threaded and parallel options: the traditional Levenberg-Marquardt method with the calculation of *J* using the central difference derivative formula; the modified Levenberg-Marquardt method with the calculation of *J* by the Broyden method once every two epochs with the two heuristic approaches described above; the traditional Levenberg-Marquardt method with the calculation of *J* using analytically derived formulas.

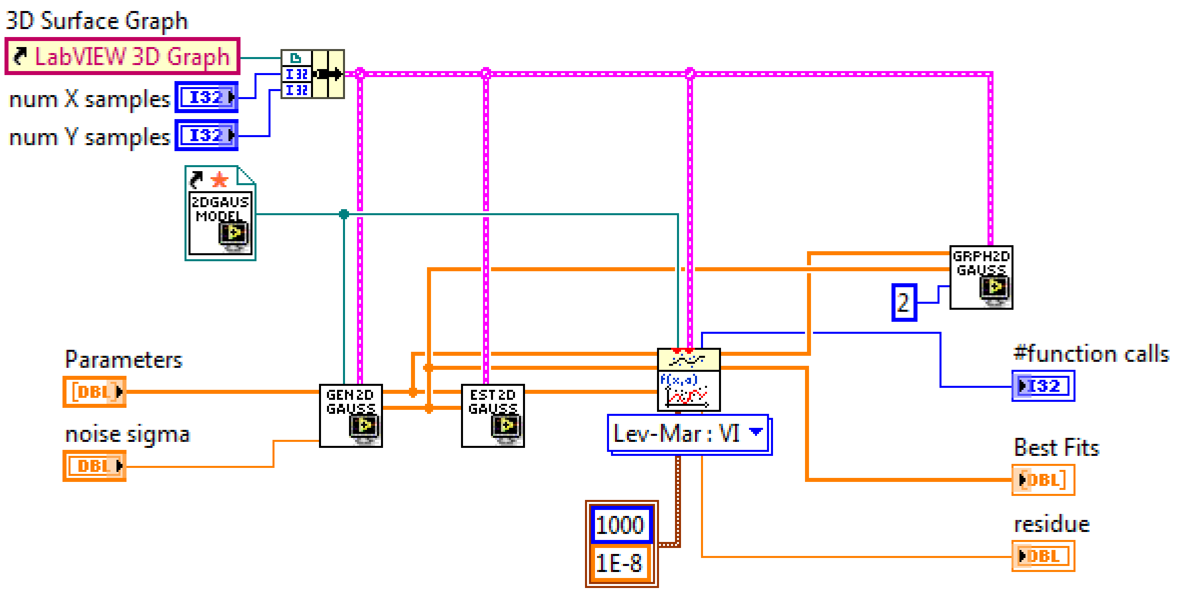


Figure 4: Diagram of complex dynamic objects neural network model (author's research)

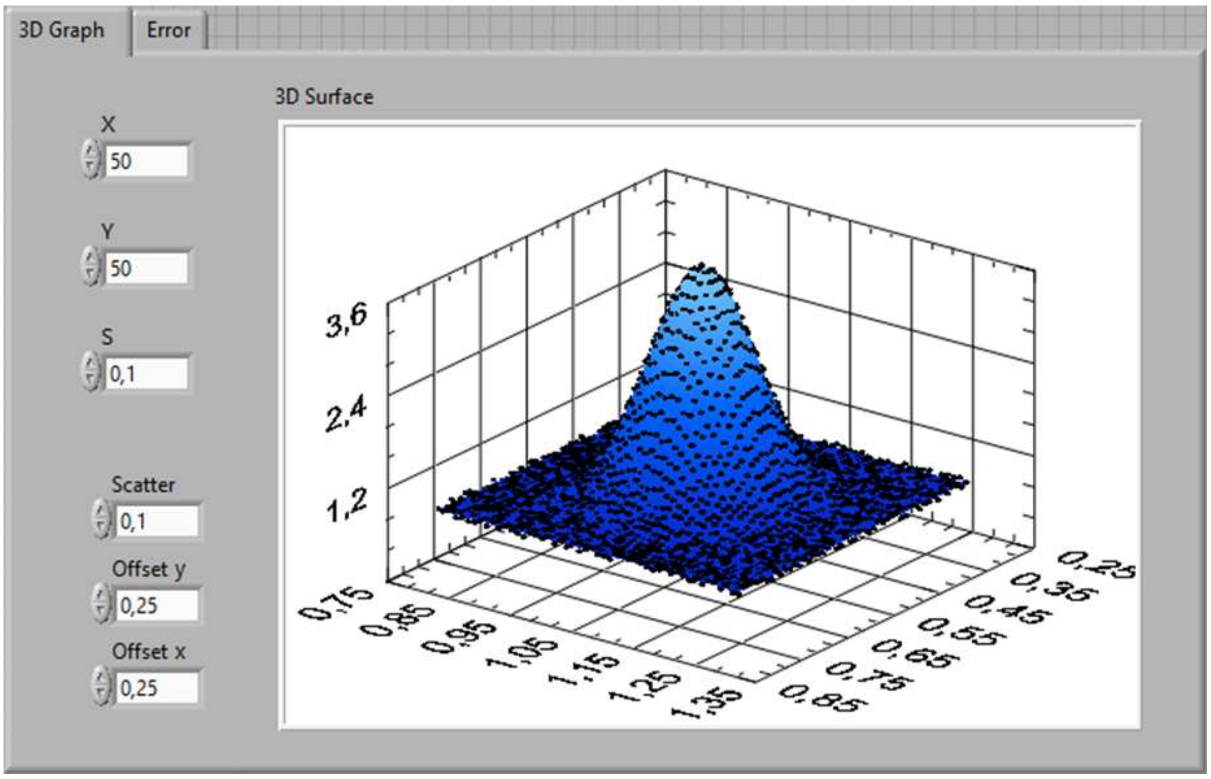


Figure 5: Developed software user interface (author's research)

For testing, we used a personal computer running the GNU/Linux operating system with an AMD Ryzen 5 5600 processor. It has 6 cores with 12 threads operating at a frequency of 3.3 GHz and 32 GB RAM for DDR-4. The computational experiment aims to assess the execution time of each method, considering the proposed methods for calculating the Jacobian matrix. For training, a NARX hybrid network (Fig. 2) with 7 inputs, in linear layer 5 neurons, in nonlinear layer 20 neurons, and in output layer 1 neuron is used [40]. The instantaneous functional $\mathbf{u}_{k+1}$ (19) is used to assess training according quality to [48]. The calculation results are presented in Table 7.

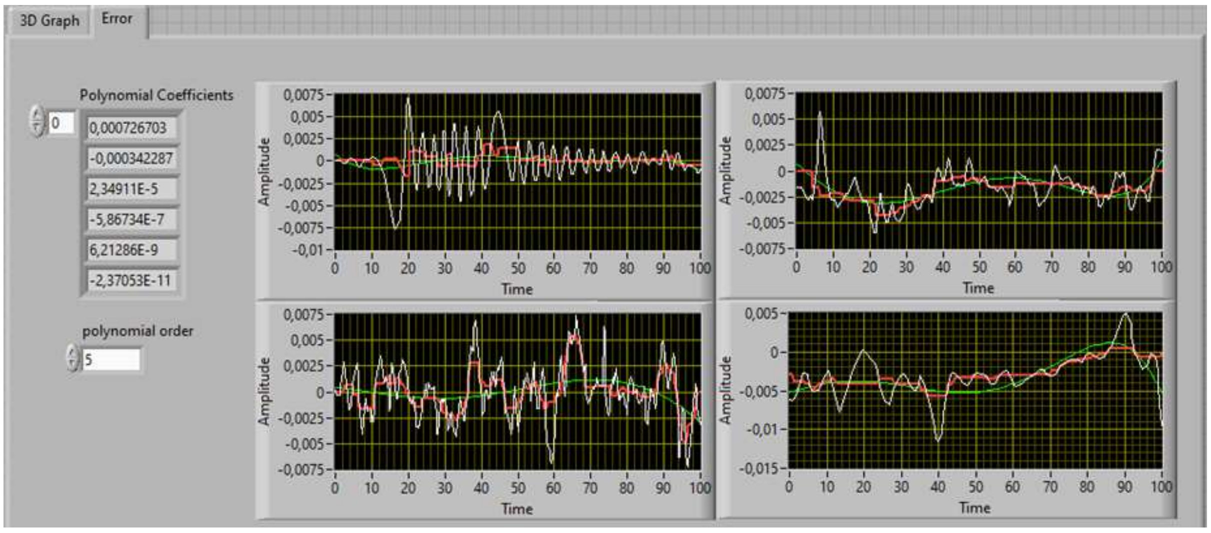
Table 7
Neural network training results (author's research, based on [42])

Data set	Central difference		Broyden method		Analytically derived	
number	t, seconds	$\mathbf{u}_{k+1}$	t, seconds	$\mathbf{u}_{k+1}$	t, seconds	$\mathbf{u}_{k+1}$
1 data stream, N = 256 (total training sample size [45, 46])						
Data set 1	121.382	0.988	11.785	0.985	58.639	0.983
Data set 2	121.371	0.976	11.559	0.973	58.072	0.971
Data set 3	120.989	0.995	10.082	0.992	56.537	0.990
Data set 4	121.295	0.969	11.776	0.966	57.759	0.964
6 data streams, N = 256 (total training sample size [45, 46])						
Data set 1	40.193	0.988	3.902	0.985	19.417	0.983
Data set 2	40.323	0.976	3.840	0.973	19.250	0.971
Data set 3	39.669	0.995	3.306	0.992	18.281	0.990
Data set 4	40.298	0.969	3.912	0.966	19.190	0.964
12 data streams, N = 256 (total training sample size [45, 46])						
Data set 1	20.298	0.988	1.971	0.985	9.806	0.983
Data set 2	20.262	0.976	1.930	0.973	9.695	0.971
Data set 3	20.098	0.995	1.675	0.992	9.262	0.990
Data set 4	20.216	0.969	1.963	0.966	9.627	0.964

Based on the results of the computational experiment, we can state the following:

1. In hidden layer neurons number increases and the number of steps is limited, the value of E(w) increases and more steps are required to correct the parameters.
2. Using the Broyden method, it was possible to reduce the computation time by approximately 10…12 times compared to the central difference derivative, but $\mathbf{u}_{k+1}$ increased.
3. Direct calculations made it possible to reduce the calculation time by approximately 2.07…2.14 times compared to the central difference derivative. With a small training sample size, direct calculations usually yield the minimum $\mathbf{u}_{k+1}$.
4. Parallel versions of methods work on average about 3…6 times faster than the sequential implementation.
5. $\mathbf{u}_{k+1}$ almost does not change when the number of threads changes, which indicates the correct implementation of parallel processing. The maximum acceleration is approximately 61.38…120.71 times.

The next stage of the computational experiment is devoted to obtaining and analyzing the error in the operation of the trained neural network in the created software product on the identified parameters. Using the training sample (Table 4), identification errors were obtained for the following parameters of the TV3-117 TE: increase degree at compressor pressure (Fig. 6, top left), compressor turbine shaft power (Fig. 6, top right), compressor turbine operation (Fig. 6, bottom left), in combustion chamber fuel consumption (Fig. 6, bottom right), where *yellow line* is error obtained by the NARX hybrid neural network with the classical Levenberg-Marquardt method [40], *red line* is error obtained by the NARX hybrid neural network with the modified Levenberg-Marquardt method, *green line* is approximation line. Analysis of the results of processing the average error of training the hybrid neural network NARX using the modified Levenberg-Marquardt method showed a decrease in the average error value by 33 %, to a level of 0.025.



Polynomial Coefficients

0

0,000726703
-0,000342287
2,34911E-5
-5,86734E-7
6,21286E-9
-2,37053E-11

polynomial order

5

Figure 6: Neural network training error calculating results (author's research)

At the final stage of the computational experiment, the loss was calculated and analyzed in the hybrid neural network NARX (Fig. 7), which serves as an indicator of the variance between the model's predicted values and target variable during training. Loss reflects the degree of error in the model of the research object and is used in the training process to adjust the neural network parameters for minimize this error. The expression for calculating the loss L in a neural network usually depends on the type of problem (such as regression or classification) and the loss function used. For regression problems, the squared error is often used for classification tasks, cross-entropy, or other loss functions. The general formula can be represented as the loss sum over all examples of the training set. In this work, used [54, 55]:

$$L = \frac{1}{n} \cdot \sum_{i=1}^{n} w_i \cdot (y_i - \tilde{y}_i)^2, \tag{21}$$

where n is examples number, y_i is actual value of the target variable, $\tilde{y}_i$ is predicted value by the model, ω_i is the weight assigned to each error, allowing their significance to be taken into account in the final loss function.

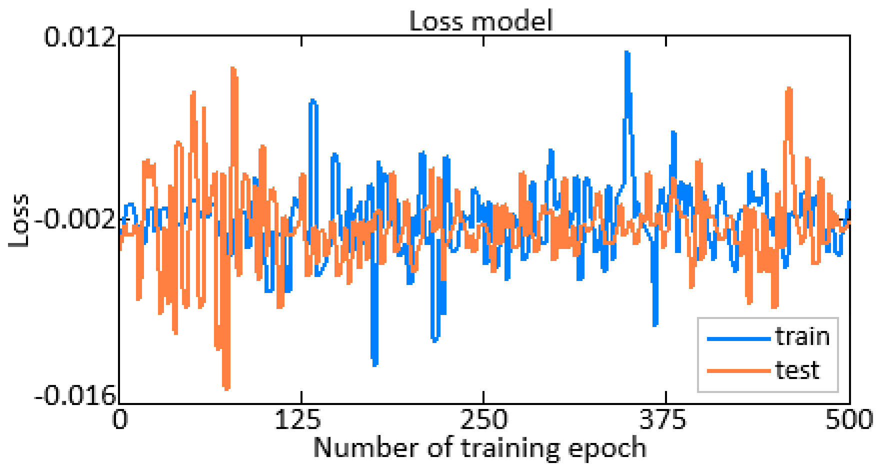


Figure 7: Loss rates during model training and testing on the input dataset (according to Table 4): *blue curve* is train; *orange curve* is validation (author's research)

6. Discussions

Fig. 7 shows that the loss function of the hybrid neural network NARX over 500 training epochs is generally stable and does not go beyond the limit of 0.025 (2.5 %), which indicates acceptable losses in problems of identifying complex dynamic objects [56, 57].

Table 8 contains comparative analysis results for thermogas-dynamic parameters identification accuracy in the engine operating process using a neural network and classical methods for each parameter in TV3-117 TE model [40].

Table 8
Absolute error calculation results (author's research, comparisons with [40])

Model	Absolute error, %			
	Fuel consumption in combustion chamber	Compressor turbine operation	Compressor turbine shaft power	Increase degree dependence on compressor pressure
Classical	1.95	1.95	1.96	1.95
Neural network:				
three-layer perceptron	0.65	0.68	0.64	0.66
[49, 50]	0.41	0.43	0.41	0.42
Gaussian NARX-model				
[34]	0.26	0.28	0.26	0.27
Gaussian NARX-model with modified Levenberg-Marquardt method				

We introduced supplementary noise to the dataset to assess neural networks resilience to variations in input information (Table 4). This noise was incorporated into each parameter by integrating white noise with a standard deviation of σ_i = 0.025 and a mean of zero. For each parameter this corresponds to maximum value2.5 %. Table 9 illustrates the outcomes of a comparative assessment of the precision in implementing the technique for discerning thermogas-dynamic parameters in TV3-117 TE operating process using both neural networks and traditional approaches.

Table 9
Absolute error (with white noise) calculation results (author's research, comparisons with [40])

Model	Absolute error, %			
	Increase degree dependence on compressor pressure	Compressor turbine shaft power	Compressor turbine operation	Fuel consumption in combustion chamber
Classical	3.11	3.15	3.14	3.15
Neural network:				
three-layer perceptron [55, 56]	1.13	1.09	1.17	1.11
Gaussian NARX-model [40]	0.74	0.72	0.73	0.71
Gaussian NARX-model with modified Levenberg-Marquardt method	0.43	0.41	0.42	0.40

An examination of Table 9 demonstrates that under the stated noise conditions, the error in identification remains within specific limits: for Gaussian NARX model with modified Levenberg-Marquardt method is 0.43 %, for Gaussian NARX model is 0.71 %, for three-layer perceptron structured as 7–53–36 is 1.09 % [40, 58, 59], and for thermogas-dynamic TV3-117 TE model is 3.15 %. Due to the maximum absolute error and white noise presence in applying the identification technique for the thermogas-dynamic parameters using the least squares method rose from 1.96 % to 3.15 %. This error increased from 0.64 % to 1.09 % [40, 58, 59] for the three-layer perceptron structured as 7–53–36, , from 0.28 % to 0.43 % for the gaussian NARX model with modified Levenberg-Marquardt method, and it went up from 0.43 % to 0.74 % for Gaussian NARX model. To evaluate the dependability of the neural network approach in discerning the thermogas-dynamic parameters of TV3-117 TE operating process [40, 58, 59], the following formulations can be employed [60, 61]:

$$K_{error} = \frac{T_{error}}{T_0} \cdot 100\%,$$
$$K_{quality} = \left(1 - \frac{T_{error}}{T_0}\right) \cdot 100\%, \quad (22)$$

where K_{error} and $K_{quality}$ represent coefficients for erroneous and quality identification, respectively [62]; T_{error} indicates the cumulative time of segments associated with misclassification, while T_0 denotes test sample duration (in this context, T_0 = 5 s) [63].

Table 10 presents coefficients computing outcomes for parameters quality identification and both erroneous [40, 60–64], including the relations between the increase degree dependence in compressor turbine operation, compressor turbine shaft power, the total compressor pressure, and fuel consumption in the combustion chamber.

Table 10
Erroneous and qualitative coefficients calculating results (author's research, comparisons with [40])

Parameter	Gaussian NARX-model [40]		Gaussian NARX-model with modified Levenberg-Marquardt method	
	K_{error}	$K_{quality}$	K_{error}	$K_{quality}$
Compressor turbine operation	0.528	99.873	0.393	99.923
Compressor turbine shaft power	0.523	99.871	0.389	99.921
Increase degree dependence on compressor pressure	0.521	99.872	0.386	99.925
Fuel consumption in combustion chamber	0.526	99.872	0.390	99.922

As depicted in Table 10, the rates of erroneous identification coefficients remain below 0.393 %, while the minimum coefficients for accurate identification rate reach 99.925 %.

The main area of practical application of the developed method is the helicopter TE monitoring and operation controlling neural network on-board expert system [65]. The developed method can be included as a neural network module for helicopter TE parameters identification, which provides continuous and engine operation accurate monitoring in real-time, and also increases the level of safety and flight efficiency.

7. Conclusions

For the first time, a concept has been created for constructing neural network models of complex dynamic objects, in which, by increasing the robustness of the functioning of a trained neural network, it becomes possible at its output to increase the reliability of solving problems of identifying complex dynamic objects.

The universal neural network model of complex dynamic objects was further developed in the form of a hybrid neural network NARX with a radial-basis layer, in which, through the use of a modified Levenberg-Marquardt training method, a reduction in the maximum absolute identification error by almost 2 times is achieved – from 0.74 to 0.43 %.

The transition from linear neural networks (multilayer perceptron) to nonlinear ones (hybrid neural network NARX with a radial basis layer) in identification tasks of complex dynamic objects is scientifically substantiated, providing more accurate and flexible identification of parameters of complex dynamic objects in real-time.

The method of calculating the elements of the Hessian matrix, a component of the analytical expression of the Levenberg-Marquardt method, was further developed, which, by taking into account the weight connections of neurons of both nonlinear and linear layers, allowed for to reduction of the calculation time by approximately 10...12 times compared to the central difference derivative. The use of direct calculations made it possible to reduce the calculation time by approximately 2.07...2.14 times compared to the central difference derivative.

For the first time, an analytical description of the regularization parameter has been proposed, which is based on control coefficients of increasing or decreasing its value in the event of a change in error, in the mathematical expression of the Levenberg-Marquardt method, which significantly affects its performance.

The proposed complex modification of the Levenberg-Marquardt method made it possible to experimentally select the optimal structure of the neural network, reduce the average value of the training error of a NARX hybrid neural network with a radial basis layer by 33 % for the level of 0.025, and also ensure the stability of the loss function of the neural network throughout 500 training epochs, which does not exceed 2.5 %.

References

[1] R. Voliansky, A. Pranolo, Parallel mathematical models of dynamic objects, International Journal of Advances in Intelligent Informatics 4:2 (2018) 120–131. doi: 10.26555/ijain.v4i2.229.

[2] V. Sherstjuk, M. Zharikova, I. Didmanidze, I. Dorovskaja, S. Vyshemyrska, Risk modeling during complex dynamic system evolution using abstract event network model, CEUR Workshop Proceedings 3101 (2022) 93–110.

[3] A. Sharma, E. Kosasih, J. Zhang, A. Brintrup, A. Calinescu, Digital Twins: State of the art theory and practice, challenges, and open research questions, Journal of Industrial Information Integration 30 (2022) 100383. doi: 10.1016/j.jii.2022.100383.

[4] M. Fore, M. O. Alver, J. A. Alfredsen, A. Rasheed, T. Hukkelas, H. V. Bjelland, B. Su, S. J. Ohrem, E. Kelasidi, T. Norton, N. Papandroulakis, Digital Twins in intensive aquaculture – Challenges, opportunities and future prospects, Computers and Electronics in Agriculture 218 (2024) 108676. doi: 10.1016/j.compag.2024.108676.

[5] A. Becue, E. Maia, L. Feeken, P. Borchers, I. Praca, A New Concept of Digital Twin Supporting Optimization and Resilience of Factories of the Future, Applied Sciences 10 (13) (2020) 4482. doi: 10.3390/app10134482.

[6] D. Galar, U. Kumar, Digital Twins: Definition, Implementation and Applications, Advances in Risk-Informed Technologies (2024) 79–106.

[7] A. Nikiforov, Automatic Control of the Structure of Dynamic Objects in High-Voltage Power Smart-Grid, Automation and Control (2020). doi: 10.5772/intechopen.91664. URL: https://www.intechopen.com/chapters/71513

[8] O. Maksymov, E. Malakhov, V. Mezhuyev, Model and method for representing complex dynamic information objects based on LMS-trees in NoSQL databases, Herald of Advanced Information Technology 4:3 (2021) 211–224. doi: 10.15276/hait.03.2021.1.

[9] D. Kahl, M. Kschischo, Searching for Errors in Models of Complex Dynamic Systems, Frontiers in Physiology 11 (2020). doi: 10.3389/fphys.2020.612590. URL: https://www.frontiersin.org/journals/physiology/articles/10.3389/fphys.2020.612590/full

[10] Y. Li, Application Analysis of Artificial Intelligent Neural Network Based on Intelligent Diagnosis, Procedia Computer Science 208 (2022) 31–35. doi: 10.1016/j.procs.2022.10.006.

[11] A. Kupina, D. Zubov, Y. Osadchuka, R. Ivchenkoa, V. Saiapin, Intelligent Neural Networks Models for the Technological Separation Processes, CEUR Workshop Proceedings 3373 (2023) 76–86.

[12] S. S. Talebi, A. Madadi, A. M. Tousi, M. Kiaee, Micro Gas Turbine fault detection and isolation with a combination of Artificial Neural Network and off-design performance analysis, Engineering Applications of Artificial Intelligence, vol. 113 (2022) 104900. doi: 10.1016/j.engappai.2022.104900.

[13] S. Kim, J. H. Im, M. Kim, J. Kim, Y. I. Kim, Diagnostics using a physics-based engine model in aero gas turbine engine verification tests, Aerospace Science and Technology, vol. 133 (2023) 108102. doi: 10.1016/j.ast.2022.108102.

[14] J. Zeng, Y. Cheng, An Ensemble Learning-Based Remaining Useful Life Prediction Method for Aircraft Turbine Engine, IFAC-PapersOnLine, vol. 53, issue 3 (2020) 48–53. doi: 10.1016/j.ifacol.2020.11.009.

[15] B. Li, Y.-P. Zhao, Y.-B. Chen, Unilateral alignment transfer neural network for fault diagnosis of aircraft engine, Aerospace Science and Technology, vol. 118 (2021) 107031. doi: 10.1016/j.ast.2021.107031.

[16] Y. Shen, K. Khorasani, Hybrid multi-mode machine learning-based fault diagnosis strategies with application to aircraft gas turbine engines, Neural Networks, vol. 130 (2020) 126–142. doi: 10.1016/j.neunet.2020.07.001.
[17] R. Chen, X. Jin, S. Laima, Y. Huang, H. Li, Intelligent modeling of nonlinear dynamical systems by machine learning, International Journal of Non-Linear Mechanics, vol. 142 (2022) 103984. doi: 10.1016/j.ijnonlinmec.2022.103984.
[18] M. Soleimani, F. Campean, D. Neagu, Diagnostics and prognostics for complex systems: A review of methods and challenges, Quality and Reliability Engineering, vol. 37, issue 8 (2021) 3746–3778 doi: 10.1002/qre.2947.
[19] Z. Huang, Automatic Intelligent Control System Based on Intelligent Control Algorithm, Journal of Electrical and Computer Engineering, vol. 7 (2022) 1–10. doi: 10.1155/2022/3594256
[20] S. Tian, J. Zhang, X. Shu, L. Chen, X. Niu, Y. Wang, A Novel Evaluation Strategy to Artificial Neural Network Model Based on Bionics, Journal of Bionic Engineering, 19 (1) (2022) 224–239. doi: 10.1007/s42235-021-00136-2.
[21] L. Qian, C. Liu, J. Yi, S. Liu, Application of hybrid algorithm of bionic heuristic and machine learning in nonlinear sequence, Journal of Physics: Conference Series 1682 (2020) 012009. doi: 10.1088/1742-6596/1682/1/012009.
[22] H. Lin, C. Wang, J. Sun, X. Zhang, Y. Sun, H. H. C. Iu, Memristor-coupled asymmetric neural networks: Bionic modeling, chaotic dynamics analysis and encryption application, Chaos, Solitons & Fractals 166 (2023) 112905. doi: 10.1016/j.chaos.2022.112905
[23] J. Sun, S. Sathasivam, M. Khan, Analysis and Optimization of Network Properties for Bionic Topology Hopfield Neural Network Using Gaussian-Distributed Small-World Rewiring Method, IEEE Access 10 (2022) 95369–95389. doi: 10.1109/ACCESS.2022.3204821.
[24] S. Vladov, Y. Shmelov, R. Yakovliev, Optimization of Helicopters Aircraft Engine Working Process Using Neural Networks Technologies, CEUR Workshop Proceedings 3171 (2022) 1639–1656.
[25] R. Abdulkadirov, P. Lyakhov, N. Nagornov, Survey of Optimization Algorithms in Modern Neural Networks, Mathematics 11 (11) (2023) 2466. doi: 10.3390/math11112466
[26] J. Chen, Y. Liu, Neural optimization machine: a neural network approach for optimization and its application in additive manufacturing with physics-guided learning, Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 381: 2260 (2023). doi: 10.1098/rsta.2022.0405. URL: https://royalsocietypublishing.org/doi/10.1098/rsta.2022.0405
[27] F. Mehmood, S. Ahmad, T. K. Whangbo, An Efficient Optimization Technique for Training Deep Neural Networks, Mathematics 11 (6) (2023) 1360. doi: doi.org/10.3390/math11061360
[28] T. Asrav, E. Aydin, Physics-informed recurrent neural networks and hyper-parameter optimization for dynamic process systems, Computers & Chemical Engineering 173 (2023), 108195. doi: 10.1016/j.compchemeng.2023.108195
[29] A. Merabet, S. Kanukollu, A. Al-Durra, E. F. El-Saadany, Adaptive recurrent neural network for uncertainties estimation in feedback control system, Journal of Automation and Intelligence 2:3 (2023), 119–129. doi: 10.1016/j.jai.2023.07.001
[30] M. F. Ab Aziz, S. A Mostafa, C. F. M. Foozy, M. A. Mohammed, M. Elhoseny, A. Z. Abualkishik, Integrating Elman recurrent neural network with particle swarm optimization algorithms for an improved hybrid training of multidisciplinary datasets, Expert Systems with Applications 183 (2021), 115441. doi: 10.1016/j.eswa.2021.115441
[31] Q. Ali M. N. Mahdi, M. Ali, M. N. Atta, A. Khan, S. A. Lashari, D. A. Ramli, Training Learning Weights of Elman Neural Network Using Salp Swarm Optimization Algorithm, Procedia Computer Science 225 (2023), 1974–1986. doi: 10.1016/j.procs.2023.10.188
[32] Y. Li, Z. Wang, R. Han, S. Shi, J. Li, R. Shang, H. Zheng, G. Zhong, Y. Gu, Quantum recurrent neural networks for sequential learning, Neural Networks 166 (2023), 148–161. doi: 10.1016/j.neunet.2023.07.003
[33] Y. Wang, W. Zhou, H. Feng, L. Li, H. Li, Progressive Recurrent Network for shadow removal, Computer Vision and Image Understanding 238 (2024), 103861. doi: 10.1016/j.cviu.2023.103861
[34] H. Wang, W. Jiang, X. Deng, J. Geng, A new method for fault detection of aero-engine based on isolation forest, Measurement 185 (2021) 110064. doi: 10.1016/j.measurement.2021.110064

[35] S. Zhernakov, A. Gilmanshin, New onboard gas turbine engine diagnostic algorithms based on neural-fuzzy networks, Aviation and rocket and space technology 19: 2 (68) (2015) 63–68.
[36] S. Zhernakov, A. Gilmanshin, Realization of hybrid gas turbine engine control and diagnostics algorithms using modern on-board computing devices, in: Proceedings of the VII International conference “Actual problems of mechanical engineering”, March 25–27, 2015, pp. 765–769.
[37] B. Mokin, V. Mokin, O. Mokin, O. Mamyrbayev, S. Smailova, The synthesis of mathematical models of nonlinear dynamic systems using Volterra integral equation, Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Środowiska 12:2 (2022) 15–19. doi: 10.35784/iapgos.2947.
[38] A. Slipchuk, P. Pukach, M. Vovk, O. Slyusarchuk, Study of the dynamic process in a nonlinear mathematical model of the transverse oscillations of a moving beam under perturbed boundary conditions, Mathematical Modeling and Computing 11:1 (2024) 37–49. doi: 10.23939/mmc2024.01.037
[39] A. Abdulnagimov, G. Ageev, Neural network technologies in hardware-in-the-loop simulation: principles of gas turbine digital twin development, Informatics, Computer Science and Management 23: 4 (86) (2019) 115–121.
[40] S. Vladov, R. Yakovliev, O. Hubachov, J. Rud, Y. Stushchanskyi, Neural Network Modeling of Helicopters Turboshaft Engines at Flight Modes Using an Approach Based on “Black Box” Models, CEUR Workshop Proceedings 3624 (2024) 116–135.
[41] S. Vladov, I. Dieriabina, O. Husarova, L. Pylypenko, A. Ponomarenko, Multi-mode model identification of helicopters aircraft engines in flight modes using a modified gradient algorithms for training radial-basic neural networks, Visnyk of Kherson National Technical University 4 (79) (2021) 52–63. doi: 10.35546/kntu2078-4481.2021.4.7
[42] G. Alcan, M. Unel, V. Aran, M. Yilmaz, C. Gurel, K. Koprubasi, Diesel Engine NOx Emission Modeling Using a New Experiment Design and Reduced Set of Regressors, IFAC-PapersOnLine 51:15 (2018) 168–173. doi: 10.1016/j.ifacol.2018.09.114
[43] S. Vladov, Y. Shmelov, R. Yakovliev, Modified Searchless Method for Identification of Helicopters Turboshaft Engines at Flight Modes Using Neural Networks, in: Proceedings of the 2022 IEEE 3rd KhPI Week on Advanced Technology, Kharkiv, Ukraine, October 03–07, 2022, pp. 257–262. doi: 10.1109/KhPIWeek57572.2022.9916422
[44] S. Novikova, E. Kremleva, Increasing the robustness of the neural network model for monitoring gas turbine engines based on reduction, Aircraft, aircraft engines and methods of their operation 3 (2019) 17–26.
[45] J. Bill, B. A. Cox, L. Champagne, A comparison of quaternion neural network backpropagation algorithms, Expert Systems with Applications 232 (2023) 120448. doi: 10.1016/j.eswa.2023.120448
[46] G. Xing, J. Gu, X. Xiao, Convergence analysis of a subsampled Levenberg-Marquardt algorithm, Operations Research Letters 51:4 (2023) 379–384. doi: 10.1016/j.orl.2023.05.005
[47] S. Parkhomenko, A Levenberg-Marquardt algorithm execution time reducing in case of large amount of the data, International Research Journal 1 (20) part 1 (2014) 80–83.
[48] S. Parkhomenko, T. Ledeneva, Training neural networks using the Levenberg-Marquardt method in conditions of a large amount of data, System analysis and information technology 2 (2014) 98–106.
[49] N. Marumo, T. Okuno, A. Takeda, Majorization-minimization-based Levenberg–Marquardt method for constrained nonlinear least squares, Computational Optimization and Applications 84 (2023) 833–874. doi: 10.1007/s10589-022-00447-y.
[50] A. O. Umar, I. M. Sulaiman, M. Mamat, M. Y. Waziri, N. Zamri, On damping parameters of Levenberg-Marquardt algorithm for nonlinear least square problems, Journal of Physics: Conference Series 1734 (2021) 012018. doi: 10.1088/1742-6596/1734/1/012018
[51] S. Vladov, Y. Shmelov, R. Yakovliev, Modified Helicopters Turboshaft Engines Neural Network On-board Automatic Control System Using the Adaptive Control Method, CEUR Workshop Proceedings 3309 (2022) 205–224.
[52] S. Vladov, Y. Shmelov, R. Yakovliev, M. Petchenko, Modified Neural Network Fault-Tolerant Closed Onboard Helicopters Turboshaft Engines Automatic Control System, CEUR Workshop Proceedings, vol. 3387 (2023) 160–179.

[53] K. Makhotilo, D. Voronenko, Modification of the Levenberg-Marquardt algorithm to improve the accuracy of predictive models of connected energy consumption in everyday life, Bulletin of the National Technical University "KhPI" A series of "Information and Modeling" 56 (2005) 83–90.

[54] W. Zonghui, H. Jian, S. Xiaodan, Study on Robust Loss Function for Artificial Neural Networks Models in Reliability Analysis, Procedia Structural Integrity 52 (2024), 203–213. doi: 10.1016/j.prostr.2023.12.021

[55] S. Zhang, L. Xie, Leader learning loss function in neural network classification, Neurocomputing 557 (2023), 126735. doi: 10.1016/j.neucom.2023.126735

[56] S. Vladov, Y. Shmelov, R. Yakovliev, Helicopters Aircraft Engines Self-Organizing Neural Network Automatic Control System, CEUR Workshop Proceedings 3137 (2022) 28–47. doi: 10.32782/cmis/3137-3

[57] Y. Wang, H. Li, Y. Zheng, J. Peng, A fractional-order visual neural network for collision sensing in noisy and dynamic scenes, Applied Soft Computing 148 (2023), 110897. doi: 10.1016/j.asoc.2023.110897

[58] S. Vladov, Y. Shmelov, R. Yakovliev, M. Petchenko, S. Drozdova, Neural Network Method for Helicopters Turboshaft Engines Working Process Parameters Identification at Flight Modes, in: Proceedings of the 2022 IEEE 4th International Conference on Modern Electrical and Energy System (MEES), Kremenchuk, Ukraine, 2022, pp. 604–609. doi: 10.1109/MEES58014.2022.10005670

[59] S. Vladov, Y. Shmelov, R. Yakovliev, M. Petchenko, S. Drozdova, Helicopters Turboshaft Engines Parameters Identification at Flight Modes Using Neural Networks, in: Proceedings of the IEEE 17th International Conference on Computer Science and Information Technologies (CSIT), Lviv, Ukraine, 2022, pp. 5–8. doi: 10.1109/CSIT56902.2022.10000444

[60] S. Marton, Stefan Ludtke, C. Bartelt, Explanations for Neural Networks by Neural Networks, Applied Sciences 12(3) (2022), 980. doi: 10.3390/app12030980

[61] C. Yuan, J. Y. Wang, C. E. Lee, K.-N. Chiang, Equation Informed Neural Networks with Bayesian Inference Improvement for the Coefficient Extraction of the Empirical Formulas, in: Proceedings of the 2023 24th International Conference on Thermal, Mechanical and Multi-Physics Simulation and Experiments in Microelectronics and Microsystems (EuroSimE), Graz, Austria, 2023. doi: 10.1109/EuroSimE56861.2023.10100752

[62] F. Munoz, J. M. Valdovinos, J. S. Cervantes-Rojas, S. S. Cruz, A. M. Santana, Leader–follower consensus control for a class of nonlinear multi-agent systems using dynamical neural networks, Neurocomputing 561 (2023), 126888. doi: 10.1016/j.neucom.2023.126888

[63] V. Makarov, The neural network to identify an object by a sequential training mode, Procedia Computer Science 190 (2021), 532–539. doi: 10.1016/j.procs.2021.06.062

[64] H. Taherdoost, Deep Learning and Neural Networks: Decision-Making Implications, Symmetry 15(9) (2023), 1723. doi: 10.3390/sym15091723

[65] Y. Shmelov, S. Vladov, Y. Klimova, M. Kirukhina, 60. Expert system for identification of the technical state of the aircraft engine TV3-117 in flight modes, in: Proceedings of the System Analysis & Intelligent Computing : IEEE First International Conference on System Analysis & Intelligent Computing (SAIC), Kyiv, Ukraine, 2018, pp. 77–82. doi: 10.1109/SAIC.2018.8516864