Empirical Neural Network Studies for Multi-Port Load Calculation by the Input Currents

ALEXANDR PENIN[1], VICTOR COJOCARU[1], MARIA LUPU[1], LUDMILA SIDORENKO[2], ANATOLIE SIDORENKO[3]
[1]Institute of Electronic Engineering and Nanotechnologies “D. Ghitsu”,
Technical University of Moldova,
Chisinau,
MOLDOVA

[2]State University of Medicine and Pharmacy “NicolaeTestemitsanu”,
Chisinau,
MOLDOVA

[3]Moscow Center for Advanced Studies,
Kulakova str. 20, Moscow, 123592,
RUSSIA

Abstract: - A linear multi-port is considered a model of a wire communication line with physical quantities sensors or as a power loads supply line. The problems of known methods are shown to determine the multi-port parameters and the calculation of load resistances by specified or measured input currents. In the present work, the “loads–currents” relationships are approximation tasks of feedforward neural networks. The corresponding input currents are calculated for a particular set of load values, using the multi-port models with one, two, and three loads. This is how the training or input vector (input currents) and the target vector (loads) are composed, the dimension is equal to the amount of input currents or loads, and the size corresponds to the load set. Numerical experiments by the Fit Data package of MATLAB Deep Learning toolbox demonstrate the accuracy of load calculation and capability to generalization. An introduced quantitative index of the quality of training allows us to identify the minimum size of the training vector and the optimal amount of hidden layers’ neurons. The obtained results provide purposeful and fast network training.

Key-Words: - multi-port, wireline, load resistance, feedforward neural network, approximation, relative error.

Received: May 23, 2024. Revised: October 25, 2024. Accepted: November 21, 2024. Published: December 31, 2024.

1 Introduction

Let us consider a linear multi–port with an equal number of inputs and outputs. This multi-port can be used as a model of a wire communication line with physical quantities sensors or as a power loads supply line. Then, the input currents can be calculated for the given loads and known multi-port parameters. This is the direct problem presented in all the textbooks of the electric circuits theory. On the other hand, the loads can be calculated for the given or measurement input currents for the inverse, more complex, problem. In turn, multi-port parameters can be determined experimentally by open and short circuits for the inputs and outputs. In particular, the determination of the two-port parameters is given in [1]. However, manipulations at the inputs and outputs, and even with the connection of a voltage source at the output, complicate this method. It is more convenient when the manipulations are carried out only at the outputs, which corresponds to the formulation of this study.

Various options for organizing experiments at the output are possible, such as short circuit, open circuit, or using test resistive loads, [2]. In this case, the currents can be measured both at the input and output and only at the input. This is due to the number of necessary experiments, the unambiguity of the parameter values, the dimension of the resulting system of equations, and the possibility of solving this system, [3], [4], [5], [6], [7]. Ultimately, such a traditional approach based on the electric circuits theory and analytical expressions is laborious, the solution of intermediate problems is required.

In the present work, to explain the traditional approach, a fairly general case of a multi-port with two mutually influencing loads is considered. Next, the "load–currents" dependencies are not determined but are simply considered as an approximation or regression problem by a feedforward neural network, [8], [9]. The choice of the number of hidden layers and neurons is the subject of many studies, [10], [11], [12], [13], [14], [15], [16]. Grave attention is paid to achieving fast training and calculation simplicity. In general, the choice of multilayer network architecture is a complex and multivariable task. This task is currently poorly formalized, and the effectiveness of the solution depends on the developer's experience, including based of empirical data based on the results of numerical experiments, [17], [18]. In this regard, the studies of both overparameterized networks and networks with a small number of trained parameters are of interest, [19], [20].

Numerical experiments were performed to demonstrate this approximation problem by the Fit Data package with a Shallow Neural Network of MATLAB Deep Learning toolbox. To do this, using the multi-port model (up to three loads), the input currents for a particular set of load values are calculated. This is how the training vector and the target vector are composed. The dimension of the training vector is equal to the amount of input currents. The feedforward neural network of up to three hidden layers is used.

The trained neural network is further being tried on expanded control data to examine the capability to generalization and to verify the specified calculation accuracy for "all possible" load values. The fact is that, for approximation tasks, the training of networks is carried out to minimize the mean square error. Therefore, cases of invalid relative errors for some load values are not excluded. This is important, for example, to physical quantities sensors.

The problems of using the neural network lie in determining the minimum set of load values or the size of training vectors and targets, which is important in practice with a technically limited set of loads. Further, a quantitative index of the quality of training is needed, [21], [22]. This index allows us to identify the minimum size of the training vector and the optimal amount of hidden layers' neurons. The obtained results provide purposeful and fast network training.

2 Possible Methods of the four-port Parameters Determination and Calculation of Loads

Let us consider a four-port with all unknown seven resistors r_3, r_{31}, r_1 , r_4, r_{42}, r_2, r_{34} , and load resistors R_{L1}, R_{L2} in Figure 1. Due to the resistance of the common wire, the load interference is manifested. Therefore, this multi-port is of practical interest for determining unknown parameters.

Using the loop current method, the known system of equations has the view for the designated direction of currents:

$$\begin{bmatrix} R_{33} & R_{34} & -R_{31} & 0 \\ -R_{34} & R_{44} & 0 & -R_{42} \\ R_{31} & 0 & -R_{11} & 0 \\ 0 & R_{42} & 0 & -R_{22} \end{bmatrix} \cdot \begin{bmatrix} I_3 \\ I_4 \\ I_1 \\ I_2 \end{bmatrix} = \begin{bmatrix} V_3 \\ V_4 \\ V_1 \\ V_2 \end{bmatrix}, \qquad (1)$$

where
$R_{11} = r_1 + r_{31}$, $R_{22} = r_2 + r_{42}$,
$R_{31} = r_{31}$, $R_{42} = r_{42}$, $R_{34} = r_{34}$, $R_{33} = r_3 + r_{31} + r_{34}$,
$R_{44} = r_4 + r_{42} + r_{34}$

are the four-port parameters. In turn, the load voltages:
$V_1 = R_{L1} I_1$, $V_2 = R_{L2} I_2$.

So, the required experiments are conducted for the following variants to find these seven parameters. Based on the obtained data, a system of the seven equations is compiled and solved.

2.1 Methods of Three Experiments with at the Output and Input Measurement

1) Three pairs of short and open circuit combinations at the outputs. The current and voltage designations are given in Table 1. The voltages at the inputs do not change.

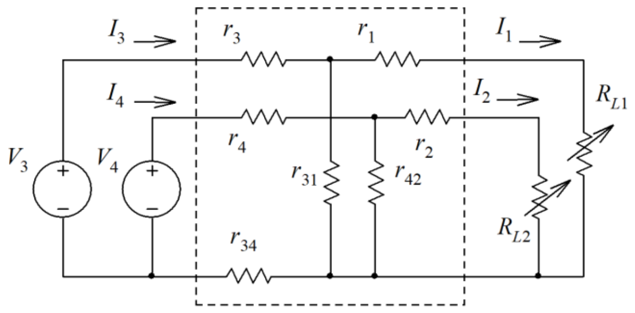


Fig. 1: Four-port with a resistor of the common wire

Table 1. Short and Open Circuit Combination. Designation of Measured Currents and Voltages

Regime of the First and Second Outputs		
Short & short circuit	***Short & open circuit***	***Open & short circuit***
$V_1^{SC}=0$	$V_1^{SCOC}=0$	V_1^{OCSC}
$V_2^{SC}=0$	V_2^{SCOC}	$V_2^{OCSC}=0$
I_3^{SCSC}	I_3^{SCOC}	I_3^{OCSC}
I_4^{SCSC}	I_4^{SCOC}	I_4^{OCSC}
I_1^{SCSC}	I_1^{SCOC}	I_1^{OCSC}=0
I_2^{SCSC}	I_2^{SCOC}=0	I_2^{OCSC}

We must notice that the OCOC combination is redundant and less informative.

Let us consider (1). We reformat this expression so that the desired parameters represent a vector and the measured currents constitute a matrix. Ultimately, the system of seven equations takes the form

$$I \cdot R = V, \tag{2}$$

$$I=\begin{bmatrix} -I_1^{SCSC} & 0 & 0 & 0 & I_3^{SCSC} & 0 & 0 \\ 0 & -I_2^{SCSC} & 0 & 0 & 0 & 0 & I_4^{SCSC} \\ 0 & 0 & I_3^{SCSC} & 0 & -I_1^{SCSC} & I_4^{SCSC} & 0 \\ 0 & 0 & 0 & I_4^{SCSC} & 0 & I_3^{SCSC} & -I_2^{SCSC} \\ -I_1^{OCSC} & 0 & 0 & 0 & I_3^{OCSC} & 0 & 0 \\ 0 & 0 & I_3^{OCSC} & 0 & -I_1^{OCSC} & I_4^{OCSC} & 0 \\ 0 & -I_2^{SCOC} & 0 & 0 & 0 & 0 & I_4^{SCOC} \end{bmatrix},$$

$$R=\begin{bmatrix} R_{11} \\ R_{22} \\ R_{33} \\ R_{44} \\ R_{31} \\ R_{34} \\ R_{42} \end{bmatrix}, \qquad V=\begin{bmatrix} V_1^{SC} \\ V_2^{SC} \\ V_3 \\ V_4 \\ V_1^{OCSC} \\ V_3 \\ V_2^{SCOC} \end{bmatrix}.$$

The first line of the matrix corresponds to the third equation of (1) for V_1^{SC}, which contains R_{11}, R_{31}. The second line corresponds to the fourth equation of (1) for V_2^{SC}, which contains R_{22}, R_{42}, etc. Further, the system is solved by known methods concerning the parameter vector.

2) *Finite values of the load resistances*. If the short circuit regime results in large current values for the operating values of the input voltages, it is possible to measure the currents for the finite resistances of the loads. Let it be loads with some maximum R_{L1}^{OC}, R_{L2}^{OC} and minimum R_{L1}^{SC}, R_{L2}^{SC} values. The load combinations, designations and measured values of currents and voltages correspond to Table 1 and $V_1 = R_{L1}I_1, V_2 = R_{L2}I_2$.

2.2 Method of Four Experiments with the Input Currents Measurement

3) *Parameter ambiguity case*. At first, we obtain the fractionally linear expressions by (1)

$$R_{L1}(I_3,I_4)=\frac{R_{11}V_3-(R_{11}R_{33}-R_{31}{}^2)I_3-R_{11}R_{34}I_4}{R_{33}I_3+R_{34}I_4-V_3}, \tag{3}$$

$$R_{L2}(I_3,I_4)=\frac{R_{22}V_4+(R_{22}R_{44}-R_{42}{}^2)I_4-R_{22}R_{34}I_3}{R_{44}I_4+R_{34}I_3-V_4}. \tag{4}$$

The first expression (3) contains four unknown parameters $R_{11}, R_{33}, R_{31}, R_{34}$. We reformat (3) as a linear expression so that there are no desired parameters on the right side:

$$\begin{gathered} -R_{11}V_3+(R_{11}R_{33}-R_{31}{}^2)I_3+R_{11}R_{34}I_4+ \\ R_{33}I_3R_{L1}+R_{34}I_4R_{L1}=R_{L1}V_3. \end{gathered} \tag{5}$$

Let us consider four load values $R_{L1}^1,\ R_{L1}^2, R_{L1}^3, R_{L1}^4$ and obtain the corresponding four pairs of currents $I_3^1, I_4^1, \dots\ I_3^4, I_4^4$. Ultimately, the system of four equations (5) take the form:

$$\begin{gathered} -R_{11}V_3+(R_{11}R_{33}-R_{31}{}^2)I_3^i+R_{11}R_{34}I_4^i+ \\ R_{33}I_3^iR_{L1}^i+R_{34}I_4^iR_{L1}^i=R_{L1}^iV_3\ ,\ i=1\dots,4. \end{gathered} \tag{6}$$

We note that (6) contains the multiply and square of the parameters. Therefore, the solution to this system will be ambiguous and give four roots with two different parameter values.

In turn, (4) contains three unknown parameters R_{22}, R_{44}, R_{42} and the already found parameter R_{34}. Similarly, a system of equations is compiled for three pairs of currents from those obtained above. The solution to this system will be ambiguous and give two identical roots.

We can also make a common system of seven equations. Then it turns out eight roots with two different values. This is a problem of experimentally determining parameters.

4) *Case of unambiguous generalized parameters*. Let us consider (5). We exclude multiples and squares of the parameters. To do this, the following generalized parameters are introduced:

$$d_1=R_{11}R_{33}-R_{31}{}^2,\ \ d_2=R_{11}R_{34}. \tag{7}$$

Then, together with the remaining initial parameters, the expression with five parameters is obtained:

$$\begin{gathered} -R_{11}V_3+d_1I_3+d_2I_4+R_{33}I_3R_{L1}+ \\ R_{34}I_4R_{L1}=R_{L1}V_3.. \end{gathered} \tag{8}$$

We consider five load values $R_{L1}^1,\ R_{L1}^2, R_{L1}^3, R_{L1}^4$, R_{L1}^5 and the corresponding five pairs of currents

$I_3^1, I_4^1, \dots\ I_3^5, I_4^5$. Ultimately, the system of five equations (8) take the matrix form:

$$\begin{bmatrix} -V_3 & I_3^1 & I_4^1 & R_{L1}^1 I_3^1 & R_{L1}^1 I_4^1 \\ -V_3 & I_3^2 & I_4^2 & R_{L1}^2 I_3^2 & R_{L1}^2 I_4^2 \\ -V_3 & I_3^3 & I_4^3 & R_{L1}^3 I_3^3 & R_{L1}^3 I_4^3 \\ -V_3 & I_3^4 & I_4^4 & R_{L1}^4 I_3^4 & R_{L1}^4 I_4^4 \\ -V_3 & I_3^5 & I_4^5 & R_{L1}^5 I_3^5 & R_{L1}^5 I_4^5 \end{bmatrix} \cdot \begin{bmatrix} R_{11} \\ d_1 \\ d_2 \\ R_{33} \\ R_{34} \end{bmatrix} = V_3 \begin{bmatrix} R_{L1}^1 \\ R_{L1}^2 \\ R_{L1}^3 \\ R_{L1}^4 \\ R_{L1}^5 \end{bmatrix}. \quad (9)$$

The generalized parameters for the second load (4) are introduced similarly.

2.3 Discussion of the Presented Methods

As we can see, these methods for determining parameters are rather laborious and take time. Especially everything becomes more complicated with an increase in the amount of loads. So, for three loads, the system of twelve equations is obtained. In addition, the generalized parameters lead to a different physical circuit model. However, having determined the parameters by one method or another, we can calculate the load resistances from the measured currents. For that, it is convenient to use the explicit (3, 4, 8).

3 Calculation of Two-Port Load

Feedforward neural networks are good at fitting functions, [8]. For this task a set of examples of proper network behavior as the input data and target or output data is required. Such data are found experimentally or calculated according to the accepted model.

3.1 Preparing the Training Data and Expanded Control Data

Let us consider a two-port in Figure 2. For such a simple circuit, the expression $I_0(R_L)$ has the following form:

$$I_0(R_L) = V_0 \frac{R_L + R_{11}}{R_{00}R_L + R_{00}R_{11} - R_{10}^{\ 2}}, \quad (10)$$

where $R_{00} = r_0 + r_{10}, R_{11} = r_1 + r_{10}, R_{10} = r_{10}$. In turn, the load voltage $V_1 = R_L I_1$.

We consider the specific example of this two-port. Let us take the following dimensionless parameter values $V_0 = 12, r_0 = 4, r_1 = 1, r_{10} = 9$. Let the load resistance R_L change over a sufficiently large range 5…,17 in 0.8 increments. Therefore, a set of 16 samples is obtained. This set represents the target vector t_16. The corresponding set of input current samples I0_16 is the training vector. In turn, the increment of 0.5 in the same range yields 25 samples and the training values I0_25, and t_25. The presented variants allow us to determine the minimum amount of samples based on the results of using expanding control data.

To check the neural network during or after training, we prepare two expanded control data, whose values should be within the range of changing parameters for training, but not coincide with them.

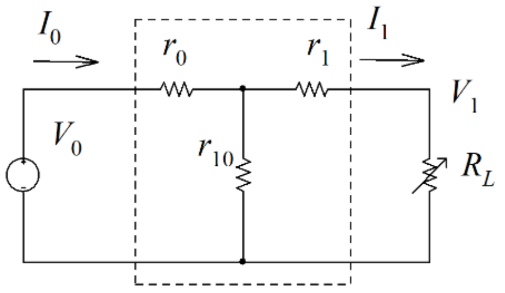


Fig. 2: Two-port with the load resistor R_L

Let the first control data I0_225, t_225 of 225 samples cover the entire range of change with the increment of 0.053. To compare the results of checking, let us take even more samples. Let the second control data I0_625, t_625 will be 625 samples with the increment of 0.0192.

3.2 Some Information about the Feedforward Neural Network

Let us consider the Fit Data package with a Shallow Neural Network of MATLAB Deep Learning toolbox, [8]. It is generally best to start with the graphical user interface GUI by the command *nftool*. This GUI allows us to go through all the stages of selection, train a feedforward neural network, and get a Script file for further training and verification of the neural network with original data. The neural network architecture for our example is shown in Figure 3.

At first, we put 3 neurons for the hidden layer. This number corresponds to the formula $2n + 1$, where $n = 1$ defines the input vector dimension, [12].

The training data I0_16, and t_16 are usually divided randomly into a training set of 70%, a validation set of 15%, and a test set of 15%. When every neuron's initial weight w and biases b are randomly initialized, the network is ready for training.

The input layer or node receives the input vector of the first training sample I0_16(1) and transmits it to the neurons of the hidden layer. Inside these neurons, this sample is assigned weights w_h and biases b_h and then subjected to a non-linear sigmoid activation function, which is a hyperbolic tangent. So, the outputs of the hidden layer are calculated by

$$g(1)_h = \tanh[w_h I0\_16(1) + b_h], h = 1\ldots,3. \quad (11)$$

The output layer is with the linear activation function. So, the output of the network as the predicted value is calculated by:

$$\hat{t}(1) = \sum_{o,h=1}^{3} [w_o g(1)_h + b_o]. \quad (12)$$

Similarly, the validation set and test set samples are sent to the network input. In the same way, the network output for all other samples is calculated, which is an epoch.

The default performance function for feedforward networks is a cost factor C or mean square error MSE as the average squared error between the network outputs $\hat{t}$ and the target outputs t. It is defined as follows for the training set with amount of samples Ntr

$$C = MSE = \frac{1}{Ntr}\sum_{i=1}^{Ntr}(\hat{t}(i) - t(i))^2. \quad (13)$$

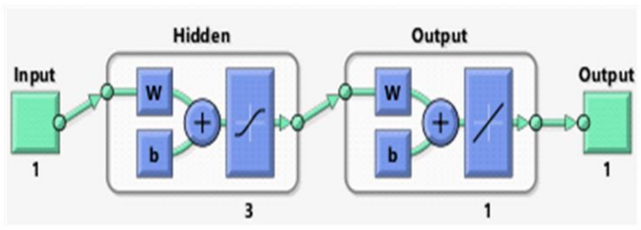


Fig. 3: Feedforward neural network with one hidden layer.

Similarly, MSE is calculated for the validation and test sets.

The process of training a neural network involves tuning the values of the weights and biases of the network to minimize MSE. The changes in updated values were dependent on the MSE change concerning previous w_h, w_o and b_h, b_o values. A decreasing error produced a smaller change from preceding values and vice versa. If the desired accuracy was not achieved, the algorithm updated w_h values in the next iteration as:

$$w_{h+1} = w_h + L\,{}^{\partial C}/_{\partial w_h},$$

where $L \ll 1$ is the learning rate that determines the amount of weight adjustment. The other values as w_o, b_h, and b_o followed the same upgradation criteria.

The presented procedure is repeated for the next epochs. Naturally, the training MSE decreases faster than for the validation set because the validation set is not used to adjust. The training is terminated when validation MSE begins to increase or the amount of epochs is over. In turn, if the test MSE is greater than necessary, then the training process is repeated.

3.3 Neural Network Training and Evaluation

We continue the above example with the following:

Training and evaluation algorithm

the initial weights *w* and biases *b* are randomly selected

↓

from 50 to 100 training experiments are conducted for 500 epochs

↓

expanded control data are submitted to the input

↓

only extreme relative error values are used

↓

the error values are distributed up to 0.5%, 1%, 2%, 5%, and 10% ranges

↓

we display the number of errors by ranges as the repeatability from the total amount of extreme values

Thus, introduced repeatability, as a probabilistic index of accuracy or generalizing capability shows the training quality changes. If the repeatability approaches 100%, then the neural network is guaranteed to calculate the load with the appropriate accuracy or error.

So, let us consider the training data of 25 samples. After the first ten retraining, we got the results of Table 2 and the corresponding “plotperform” plots for some characteristic cases.

Let us consider experiment no.1 and plot “plotperform” in Figure 4(a). We pay attention to the small value for the Validation MSE curve when the training process has ended by 206 epochs.

Table 2. Results of the first ten experiments

No. Exper.	**Actual number of epochs**	**Maximum relative errors, %**
1	206	0.67
2	500	0.064
3	15	0.74
4	500	0.013
5	500	0.18
6	8	5.93
7	1	28.7
8	500	0.25
9	80	0.22
10	132	0.48

At the same time, the value for the Test MSE curve is much larger. But still, the maximum relative error “max error” is equal to the small value of 0.6705 due to the small value for the Validation MSE curve. As it turned out, the extreme values for 225 samples are close to the values of 625 samples. Therefore, the control data can be limited to 225 samples.

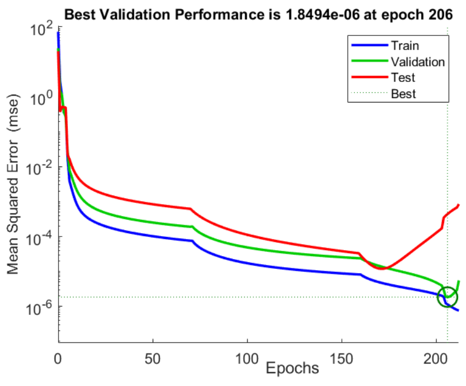
Next, experiment no. 2 results in an even lower max error of 0.0644. For this case, the value of the Test MSE curve is much less than for the Validation MSE curve, as shown in Figure 4(b).

These experiments no. 1 and no. 2 show that the Best Validation Performance value ambiguously determines our relative error. This justifies the use of expanded control data.

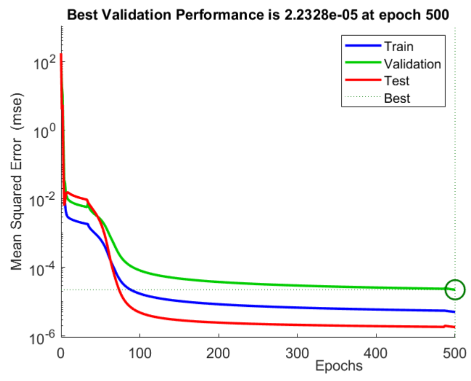
As can be seen from this table, there are 6 errors up to 0.5%, and the number of errors up to 1% is 8 or 80%. We will consider this a high result. Therefore, we calculate the repeatability of the specified error. The corresponding tendency to change the training quality of such averaged data is shown in Table 3.

Table 3. Repeatability of 1 to 10 experiments

Specified errors, %	0.5	1	2	5	10
Repeatability, %	60	80	80	80	90



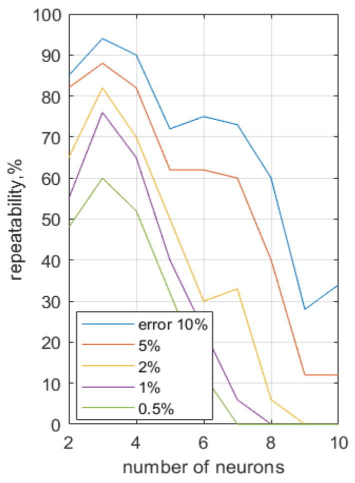
(a)



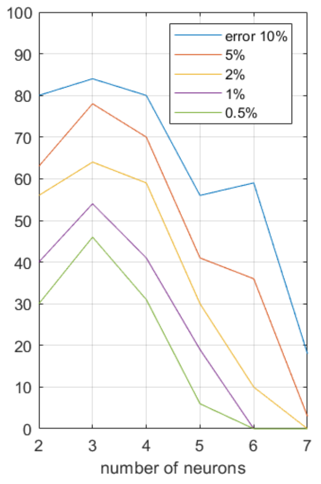
(b)

Fig. 4: Change of the MSE in the training process: (a) the Test MSE is significantly larger than the Validation MSE at the training end; (b) the Test MSE is significantly less than the Validation MSE at the training end

To confirm the statistics, we carry out the following ten experiments and calculate the average repeatability from 1-20, 1-30, etc. For 50 experiments, the repeatability became close to that of 40 ones. The achieved steady-state repeatability value is a criterion for the ending of this multiple retraining, which is important in practice.



(a)



(b)

Fig. 5: Repeatability of specified errors: (a) 25 samples; (b) 16 samples

Next, we repeat all the experiments for 2, 4, etc. up to 10 hidden layer neurons. The obtained results are presented by the family of plots in Figure 5(a). As can be seen, 3 neurons provide the greatest repeatability. For 2 neurons, the result is somewhat worse. This suggests that there are not enough neurons to approximate. The result is also worse in the case of 4 and 5 neurons. This may indicate insufficient training data sets. The repeatability is monotonously reduced. However, for 6 or 7 neurons, oscillations in the repeatability begin to appear. Similar results for 16 samples are presented in

Figure 5(b). Then, the repeatability in this case is lower.

Hence the training methodology follows. The amount of neurons is selected for the prepared training data. We run several training attempts. If the repeatability is by the required error, then we are approaching the optimal network structure. If the errors are large and there is no repeatability, then it does not make sense to realize the retraining for a long time. For example, we can take 16 samples or less, 2 or 5 neurons, and repeat retraining for a long time, while we get an error of no more than 1%. After all, the statistics of the repeatability of such a result show only 40% and 20%. It is better to change the number of neurons, and the repeatability of the result will immediately show the tendency. In this example, according to Figure 5, the amount of samples from 16 to 25 is the optimal value for 3 neurons that corresponds to the theory.

4 Calculation of Two Loads

4.1 Preparing the Training Data and Expanded Control data

We consider the specific example of the above four-port in Figure 1. Let us take the following dimensionless parameter values:

$$V_3 = 12,\ V_4 = 10,\ r_3 = 3, r_{31} = 9,\ r_1 = 1,$$
$$r_4 = 4, r_{42} = 12,\ r_2 = 2, r_{34} = 1.225.$$

Then the expressions for the input currents $I_3(R_{L1}, R_{L2})$, $I_4(R_{L1}, R_{L2})$, as the inverses of (3, 4), have the MATLAB form

```
    I3=2*(3889*RL2 + 19886).*(RL1 + 10)./det;
     I4=2*(2351*RL1 + 7310).*(RL2 + 14)./det;
```
(14)

```
     det=50552*RL1 + 34711*RL2 +
           9052*RL1.*RL2 + 190754;
```

Now, we calculate the training data with the different amounts of samples according to Table 4.

Note that the amounts of load samples are the same, and the total amounts of samples are formed due to the mutual search of the values for each load.

Therefore, the calculation code uses the *for* loop to the rL1 value, the nested *for* loop to the rL2, and the user-defined functions (14).

Similarly, we calculate the expanded control data with two different amounts of samples according to Table 5.

Table 4. Training data

Amount of samples	rL1	9	7
	rL2	9	7
Total amount		81	49
Range and increment of load changes	rL1	5:1.5:17	5:2:17
	rL2	7:2.625:28	7:3.5:28
Input vector		I0_81	I0_49
Target vector		t_81	t_49

Table 5. Control data

Total amount		256=16^2	676=26^2
Range and increment of load changes	rL1	5.0:0.8:17.0	5.0:0.48:17.0
	rL2	7.0:1.4:28.0	7.0:0.84:28.0
Input vector		I0_256	I0_676
Target vector		t_256	t_676

4.2 Neural Network Training and Evaluation

We repeat all the actions similarly to one load for the given samples by different numbers of the hidden layer neurons. The neural network architecture is shown in Figure 6.

The initial number of these neurons equals 5 and corresponds to the formula 2n+1, where n=2 defines the input vector dimension. Let us consider the training data of 81 samples. As it turned out, the extreme values for 256 samples are close to the values of 676 samples. Therefore, the control data can be limited to 256 samples. The results obtained are presented by the family of plots in Figure 7(a). Similar results for 49 samples are presented in Figure 7(b). But, the repeatability in this case is lower. The required amount of samples and the number or amount of neurons is determined by the given repeatability.

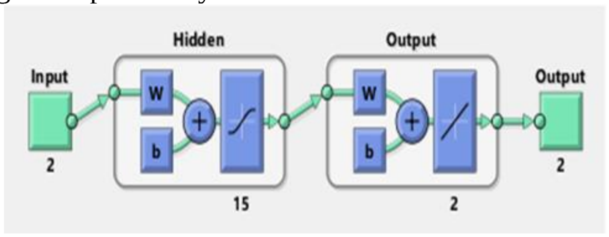
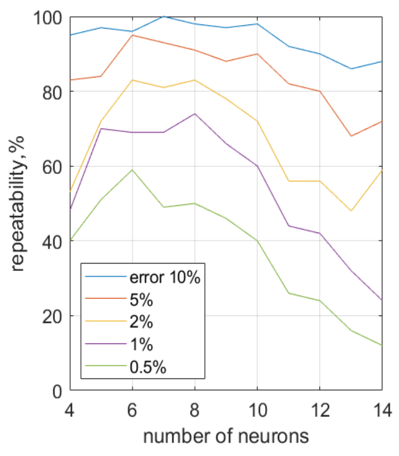
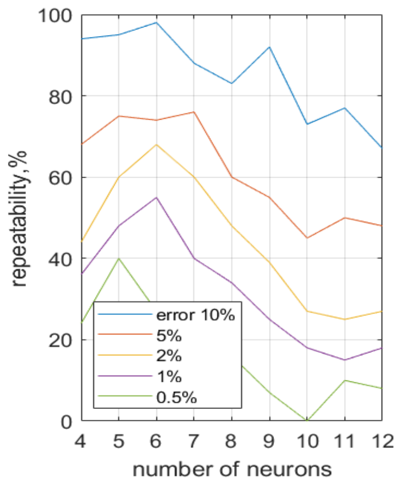


Fig. 6: Feedforward neural network with one hidden layer, two inputs, and outputs



(a)



(b)

Fig. 7: Repeatability of specified errors: (a) 81 samples; (b) 49 samples

So, with an increase in the number of neurons from 7 ones, the repeatability decreases monotonously, and for 81 samples this decrease manifests itself to a lesser extent.

This suggests that 49 samples are not enough. However, with a further increase in the number of neurons (even one neuron), the oscillations of repeatability are observed. And so, the optimal number of neurons depends on the given relative error and the size of the training data sets. Thus, for a 1% error for 81 samples and 5 to 10 neurons are obtained and 5 to 7 neurons are obtained for 49 samples. Obviously, the repeatability value increases for large errors of more than 5%, and the range of the optimal number of neurons expands.

In addition, an interesting and useful result is obtained. As we saw, the load interaction reduces the amounts of samples and its range for each load. A kind of cumulative effect of load interaction is manifested. The amount of samples for each of the two loads varies from 7 to 9 due to the second power. For comparison, in the case of one load, the amount of samples varied from 16 to 25 for comparable repeatability.

Attempts to train a network with two hidden layers produce dramatically worse results.

5 Calculation of Tree Loads

5.1 Preparing the Training Data and Expanded Control Data

For specificity, let us consider a six-port in Figure 8. To simplify calculations, we enter intermediate loads L_1, L_2, L_3, where:

$$L_1 = r_a(r_{1a} + R_{L1})/(r_a + r_{1a} + R_{L1}),$$
$$L_2 = r_b(r_{2b} + R_{L2})/(r_b + r_{2b} + R_{L2}),$$
$$L_3 = r_c(r_{3c} + R_{L3})/(r_c + r_{3c} + R_{L3}).$$

We must obtain the expressions $I_4(L_1, L_2, L_3)$, $I_5(L_1, L_2, L_3)$, $I_6(L_1, L_2, L_3)$, using the loop current method. Then, the MATLAB Script has the following view:

```
r4a=5.0; r5b=2.5; r6c=7.5; rab=4.0; rbc=6.0;
rac=12.0; V4=5.0; V5=5.0; V6=5.0;
syms I4 I5 I6 Iab Ibc Iac L1 L2 L3
eq1=(r4a+L1)*I4-L1*Iab-V4;
eq2=(r5b+L2)*I5+L2*Iab-L2*Ibc-V5;
eq3=(r6c+L3)*I6+L3*Ibc-V6;
eq4=(L1+rab+L2)*Iab-rab*Iac-L1*I4+L2*I5-
    L2*Ibc;
eq5=(L2+rbc+L3)*Ibc-L2*Iab-L2*I5-
    rbc*Iac+L3*I6;
eq6=(rab+rac+rbc)*Iac-rab*Iab-rbc*Ibc;
eqns = [eq1,eq2,eq3,eq4,eq5,eq6];
S=solve(eqns,I4,I5,I6,Iab,Ibc,Iac);
I4=S.I4; I5=S.I5; I6=S.I6;
```

The expressions for input currents are user-defined functions.

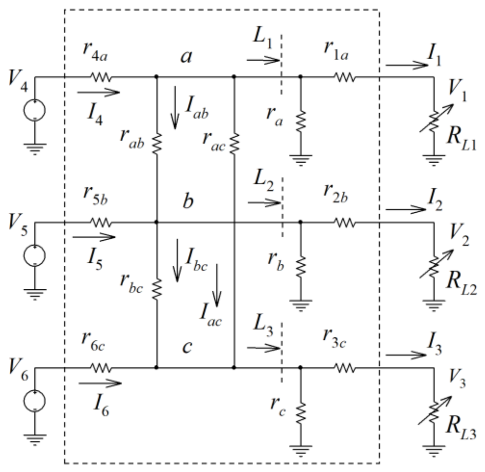


Fig. 8: Six-port with three load resistors

Now, we calculate the training data sets with the different amounts of samples according to Table 6.

Table 6. Training data

Total amount		512=8^3	343=7^3
Range and increment of load changes	rL1	3:2.5:20.5	3.0:2.9:20.4
	rL2	2.0:2.5:19.5	2.0:2.9:19.4
	rL3	3.5:3:24.5	3.5:3.5:24.5
Input vector		I0_512	I0_343
Target vector		t_512	t_343

The amount of load samples is the same, and the total amount of samples is formed due to the mutual search for each load. Therefore, the calculation code uses the *for* loop to the rL1 value, the first nested *for* loop to the rL2, the second nested *for* loop to the rL3, and the user-defined functions.

In turn, the calculated two control data are shown in Table 7.

Table 7. Control data

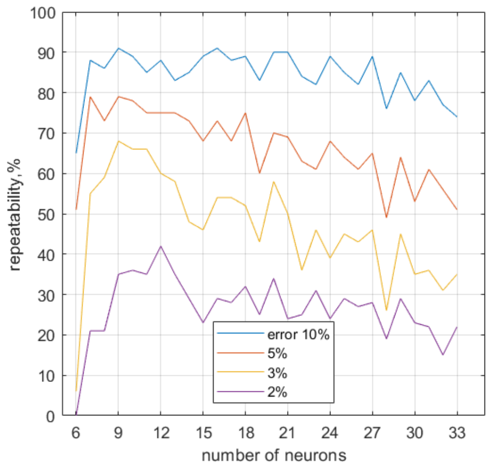
Total amount		4913=17^3	9261=21^3
Range and increment of load changes	rL1	4:1:20	4:0.8:20
	rL2	3:1:19	3:0.8:19
	rL3	4:1.25:24	4:1:24
Input vector		I0_4913	I0_9261
Target vector		t_4913	t_9261

5.2 Neural Network Training and Evaluation

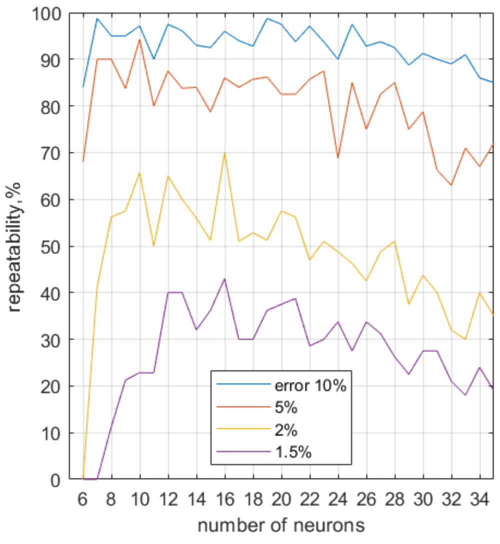
We are conducting similar experiments for the given samples with different numbers of neurons for one, two, and three hidden layers. At first, the neural network with one hidden layer is used, similar to Figure 6, but with three inputs and outputs. The initial number 7 of the hidden layer neurons corresponds to the formula 2n+1, where n=3 defines the input vector dimension. The obtained results are presented by the family of plots in Figure 9. So, the control data can be limited to 4913 samples also.

With an increase in the number of neurons, the repeatability decreases to a lesser extent compared to two loads due to the manifestations of the cumulative effect to a greater extent. Note that the amount of samples for each of the three loads varies slightly (from 6 to 8) due to the third power. Besides, the oscillations in repeatability values are observed with the increase of only one subsequent neuron.

As we can see, the optimal number of neurons depends on the given error and the size of the training data. Thus, for 2% of the error, from 8 to 14 neurons are obtained for 343 samples in Figure 9(a), and from 11 to 22 ones for 512 are obtained for 1.5% of the error in Figure 9(b). The repeatability is 35%. Note the significant oscillation in repeatability in the region of the optimal number of neurons for 512 samples. This suggests that there are not enough neurons.



(a)



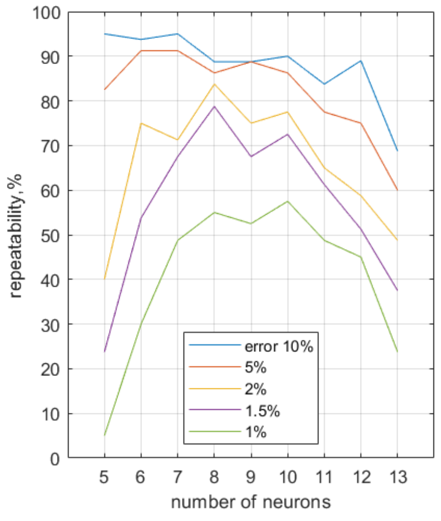
(b)

Fig. 9: Repeatability of specified errors for one hidden layer: (a) 343 samples; (b) 512 samples

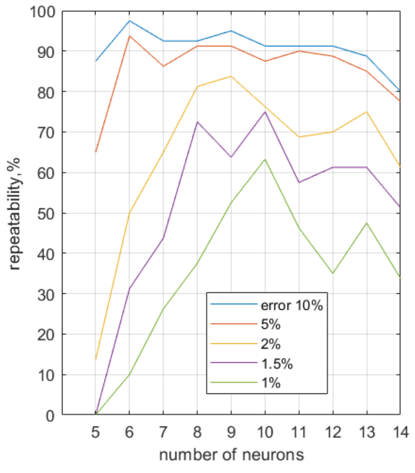
A further increase in the number of neurons leads to a more monotonous dependence. On the other hand, for 343 samples, an increase in the number of neurons also leads to significant oscillations in repeatability. This may indicate that the size of the training data is insufficient.

In a general case, when the given error increases, the optimal number of neurons shifts to the region of smaller values. In this case, the repeatability value is also increased. It is obvious that for large errors of more than 5%, the repeatability value is even more increased, and the range of the optimal number of neurons expands.

Let us consider the range of the number of neurons (for example, more than 30 ones in Figure 9), when the number of neurons significantly exceeds the optimal number. On the one hand, the repeatability value decreases. On the other hand, using such an excess number of neurons is useful in practice to improve the reliability of the neural network.



(a)



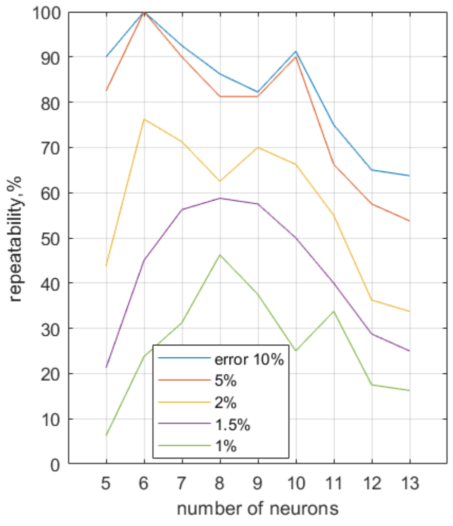
(b)

Fig. 10: Repeatability of specified errors for 512 samples and number of neurons for each layer: (a) three layers; (b) two layers

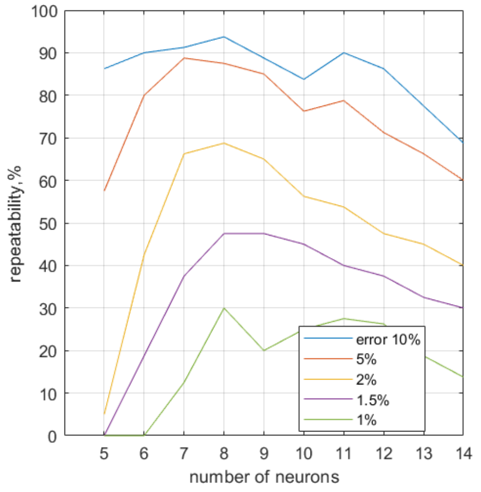
Next, the plots of 512 samples with two and three layers are presented in Figure 10. There are also oscillations in repeatability, but the plots are smoother for three layers. This suggests that two layers are not enough. The repeatability values for three and two layers are approximately the same. As we can see, the optimal number of neurons depends on the given error and the number of layers. So, for 1% of the error, it turns out from 7 to 11 neurons for three layers and from 9 to 11 for two layers. The repeatability is 50%.

When the specified error increases to 2%, the optimal number of neurons shifts to the region of smaller values, especially for two layers. In this case, the repeatability value is also increased to 70%.

Obviously, for large errors of more than 5%, the repeatability value increases to 100%. Therefore, the range of the optimal number of neurons expands.



(a)



(b)

Fig. 11: Repeatability of specified errors for 343 samples and number of neurons for each layer: (a) three layers; (b) two layers

In turn, the plots for 343 samples with three and two layers are presented in Figure 11. It can be noted that the three layers produce large repeatability values, especially for small values of specified errors. There are also oscillations in repeatability, but more smoothed plots appear already for two layers. This suggests that there are not enough data sets for the three layers. The changes in the optimal number of neurons are similar to the case with 512 samples.

The comparison of the characteristics with the 512 and 343 samples shows that the repeatability values for the 512 are markedly higher. This follows a possible compromise between the size of the training data sets, the accuracy obtained, and the number of neurons.

5 Conclusion

The solution of the direct and inverse problem of electric circuits theory is associated with the determination of multi-port parameters. Known methods for determining parameters using analytical expressions are rather laborious. A neural network is an alternative for the multi-port calculation without explicit determination of its parameters. The relation of MSE values for the Validation and Test curves on the plotperform plot provides preliminary information on the quality of the training. However, the Best Validation Performance value ambiguously defines the relative error for the expanded control data. The used repeatability index quantifies the training quality defines the criterion for performing multiple training, gives a practical compromise between the size of the training data, the accuracy obtained, the number of neurons, and the number of hidden layers, and provides purposeful network training. The interaction of two and three loads reduces the number of samples for each load, which is important in practice. This positive effect can be expected to occur for more loads. Although the results obtained are more demonstrative and qualitative, the identified patterns in the behavior of neural networks provide a basis for practical problems. Thus, an excessive number of neurons increases the reliability of neural networks.

References:

[1] K. Charles and N. Matthew, *Fundamentals of Electric Circuits*, McGraw–Hill Education, 2017.

[2] S. Bhattacharyya, L. Keel and D. Mohsenizadeh, *Linear Systems: A Measurement-based* Approach, Springer, 2014.

[3] V. Oliveira, R. Alzate and S. Bhattacharyya, A Measurement-based Approach with Accuracy Evaluation and its Applications to Circuit Analysis and Synthesis, *International Journal of Circuit Theory and Applications*, vol. 45, no. 12, 2017, pp. 1920–1941. https://doi.org/10.1002/cta.2315.

[4] K. Pereira, R. Alzate, V. Oliveira and S. Bhattacharyya, Modeling the Parametric Dependence in a Linear Circuit by Experimental Measurements, *Proceeding Series of the Brazilian Society of Computational and AppliedMathematics*, vol. 5, no. 1, 2017, pp. 010396-1–010396-7. https://doi.org/10.5540/03.2017.005.01.0396.

[5] Z. Sun, G. Pedretti, E. Ambrosi, A. Bricalli, W. Wang and D. Ielmini, Solving Matrix Equations in One Step with Cross–point Resistive Arrays, *Proceedings of the National Academy of Sciences*, vol. 116, no. 10, 2019, pp. 4123-4128. https://doi.org/10.1073/pnas.1815682116.

[6] H. Chen, M. Manry and H. Chandrasekaran, A Neural Network Training Algorithm Utilizing Multiple Sets of Linear Equations, *Neurocomputing*, vol. 25, no. 1–3, 1999, pp. 55–72. https://doi.org/10.1016/S0925-2312(98)00109-X.

[7] L. Xiao, R. Li, Z. Tan, Z. Zhang, B. Liao, R. Chen, L. Jin and S. Li, Nonlinear Gradient Neural Network for Solving System of Linear Equations, *Information Processing Letters*, vol. 142, no. 2, 2019, pp. 35–40. https://doi.org/10.1016/j.ipl.2018.10.004.

[8] M. Beale, M. Hagan and H. Demuth, *Deep Learning Toolbox R2020a User's Guide,* The MathWorks, Inc., 2020, [Online]. https://ge0mlib.com/papers/Books/04_Deep_Learning_Toolbox_Users_Guide.pdf (Accessed Date: November 10, 2024).

[9] F. Chollet, Deep Learning with Python, 2nd ed. Manning Publications Co., 2021.

[10] J. Braun and M. Griebel, On a Constructive Proof of Kolmogorov's Superposition Theorem, *Constructive Approximation,* vol.30, 2009, pp. 653–675. https://doi.org/10.1007/s00365-009-9054-2.

[11] G. Cybenko, Approximation by Superposition of a Sigmoidal Function, *Mathematics of Control Signals and Systems*, vol. 2, no. 4, 1989, pp. 303–314.

[12] K. Funahashi, On the Approximate Realization of Continuous Mappings by Neural Networks, *Neural Networks*, vol. 2, no. 3, 1989, pp. 183–192.

[13] N. Guliyev and V. Ismailov, Approximation capability of two hidden layer feedforward neural networks with fixed weights. *Neurocomputing*, vol. 316, no. 11, 2018, pp.

262–269. https://doi.org/10.1016/j.neucom.2018.07.075.

[14] T. De Ryck, S. Lanthaler and S. Mishra, On the Approximation of Functions by Tanh Neural Networks, *Neural Networks*, vol. 143, 2021, pp. 732–750. https://doi.org/10.1016/j.neunet.2021.08.015

[15] E. Paluzo–Hidalgo, R. Gonzalez–Diaz and M. Gutiérrez–Naranjo, Two-Hidden–Layer Feedforward Networks are Universal Approximators: A constructive approach, *Neural Networks*, vol. 131, no. 11, 2020, pp. 29–36. https://doi.org/10.1016/j.neunet.2020.07.021.

[16] Z. Shen, H. Yang, and S. Zhang, Neural network approximation: Three hidden layers are enough, Neural Networks, vol. 141, no. 9, 2021, pp. 160–173. https://doi.org/10.1016/j.neunet.2021.04.011

[17] K. Suzuki, *Artificial Neural Networks-Industrial and Control Engineering Applications*, InTech, Rijeka, Croatia, 2011.

[18] A. Alanis, N. Arana-Daniel and C. Lopez-Franco, *Artificial Neural Networks for Engineering Applications*. Academic Press, Cambridge, 2019.

[19] D. Merkulov and I. Oseledets, Empirical Study of Extreme Overfitting Points of Neural Networks. *J. Commun. Technol. Electron.*, vol. 64, 2019, pp. 1527–1534. https://doi.org/10.1134/S1064226919120118.

[20] Z. Allen–Zhu, Y. Li and Y. Liang, *Learning and generalization in overparameterized neural networks, going beyond two layers*. arXiv:1811.04918, 2020. https://doi.org/10.48550/arXiv.1811.04918.

[21] A. Penin, A. Sidorenko, Load resistance calculation based on an invariant input-output property of unstable communication line using a neural network with irregular steps of training data. *Research Square Company*. https://doi.org/10.21203/rs.3.rs-3649896/v1.

[22] A. Penin, A., A. Sidorenko, Irregular step of changing for neural network data sets improves the accuracy of resistive sensors calculation. *IFMBE Proceedings of 6th International Conference on Nanotechnologies and Biomedical Engineering ICNBME-2023*, September 20-23, Chisinau, Moldova, pp. 150-159. https://doi.org/10.1007/978-3-031-42782-4_17.

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

The authors equally contributed in the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

The study was supported by the Grant RSF No. 22-79-10018 “Controlled kinetic inductance based on superconducting hybrid structures with magnetic materials” and by the Moldova State Program Project «Functional nanostructures and nanomaterials for industry and agriculture» no. 20.80009.5007.11.

Conflict of Interest

The authors have no conflicts of interest to declare.