Load resistance calculation based on an invariant input-output property of unstable communication line using a neural network with irregular steps of training data

Alexander Penin (alexandr.penin@iien.utm.md)
Technical university of Moldova https://orcid.org/0000-0002-3127-2979
Anatolie Sidorenko
Technical university of Moldova https://orcid.org/0000-0001-7433-4140

Research Article

Keywords: Resistive sensor, Wire line, Two-port, Projective transformations, Neural network, Relative error

Posted Date: November 28th, 2023

DOI: https://doi.org/10.21203/rs.3.rs-3649896/v1

Load resistance calculation based on an invariant input-output property of unstable communication line using a neural network with irregular steps of training data

Alexander Penin [1*], https://orcid.org/0000-0002-3127-2979
Anatolie Sidorenko [1,2,3], https://orcid.org/0000-0001-7433-4140
[1]Institute of Electronic Engineering and Nanotechnologies "D. Ghitsu" of Technical University of Moldova
[2]Department of Microelectronics and Biomedical Engineering of Technical University of Moldova
[3]Skobeltsyn Institute of Nuclear Physics, Moscow State University, Moscow, Russia
*/ ✉, E-mail address: alexandr.penin@iien.utm.md

Abstract
The calculation of a resistive sensor is considered as the load of an unstable communication line using a neural network. In the corresponding approximation or regression problem, the feedforward neural network is trained using training data and additional control data. Such data are calculated from a mathematical model of the communication line in the form of a resistive two-port with some type of change step (regular or irregular) of the load and line parameters. The training data is traditionally divided into training, validation and test sets. It was established that in the training epochs, the neural network reveals this internal pattern (inherent in the used step of change) in these three sets. Therefore, when training the network and then applying the additional control data, small errors are obtained. But for the additional control data with a different type of step, the errors appear. The use of mixed training data by combining data with diverse type of change step eliminates said internal regularity and the neural network shows the capability to generalization and small errors by presented numerical experiments. To quantify the quality of the trained network, a special index is introduced as a repeatability of the specified relative error in percent for multiple retraining.

Keywords Resistive sensor, Wire line, Two-port, Projective transformations, Neural network, Relative error.

1 Introduction

In approximation or regression problems, a feedforward neural network is trained using input and target data [1, 2]. These data for a technical device with a sensor of physical quantities (considering the environment influence) are calculated either according to the mathematical model of this sensor or are generated through experimental measurements [3-9]. Let us pay attention to the choice of a step or interval for changing parameters of these data. Usually or by default, the values of the changing parameters are set with some constant or regular step [10-14].

According to the well-known practice of solving the overfitting problem, data are usually split into training - 70%, validation - 15%, and test - 15% sets. As a result of training, using MATLAB Fit Data package, high regression or correlation coefficient and small mean squared error values for these data sets are achieved. But, these papers do not provide results for other (extended or control) data sets to ensure that there is no overfitting.

The similar data generation with a regular step was used to calculate a resistive sensor as the load of an unstable communication wire line [15]. Subsequent simulations for an extended or additional control data were to confirm the calculation accuracy of this trained network. But, as it turned out, there were large variations in relative error values for individual data.

The researches have shown that it is all about the training and control data generation. On the one hand, a given step of changing (not necessarily the regular) parameters is present in the training, validation and test sets. When training, the neural network reveals this internal pattern in these three sets. Therefore, the small mean squared errors are obtained.

On the other hand, if the control data uses the same type of change step, small relative error values for the entire data set are also obtained. But, if the control data with a different type of step, then relative errors immediately appear. Therefore, the quality of such trained network becomes questionable.

In the present paper, the training data generation is carried out by combining data with both regular and irregular steps (moreover, of different types) of changing parameters. Therefore, in the divided three sets, this internal pattern is excluded and the network shows the capability to generalization by presented numerical experiments. To quantify the quality of the trained network, a special index is introduced as a repeatability of the specified relative error in percent for multiple retraining. This index allows to find a compromise between the size of training data, the accuracy obtained, the number of neurons and provide purposeful and fast network training.

2 Prerequisites for the use of a neural network to calculate the load of an unstable wire line

Let us consider a two–port circuit as the model of unstable wire supply line for a power load or communication line for a resistive sensor. Parameters of this two–port are determined by known measurement methods [16, 17]. In turn, the load or sensor values are calculated from the measured current at the line input and the input–output

relationships of the two–port. But, the need to redefine line parameters takes time and complicates this calculation method.

On the other hand, these known input–output relationships are considered as projective transformations (mapping or conformity of points, lines, etc.) in the sense of projective geometry [18, 19]. The projective transformations preserve an invariant as the cross–ratio (double proportion) of four samples (values) of variable resistance and the corresponding values of the input currents. The three samples are the specified load base (reference) values, and the fourth sample is the running load value. This cross–ratio is independent of the line parameters being changed. Therefore, the running load value can be calculated from the measured samples of the input current and the load base values. It is important to note that the communication line parameters are not explicitly calculated. The structure of the cross–ratio expression attracts attention; errors in the measuring currents are mutually reduced [20, 21].

In turn, the load base values may not be known exactly, which limits the direct calculation of the running or measuring load from the input current. The use of a neural network removes this restriction [15]. As mentioned above, the task solved is an example of approximation or regression problems [1, 2]. To generate data, a set of the possible load base values, the measuring load and a variable resistance of the two–port itself was used. The corresponding calculated set of input currents forms the input vector data of four components, and the measuring load values are the target data.

3 Input-output invariant of a two-port

Let us give the necessary information from the interpretation of circuit equations as a projective transformation to calculate the load by measured input currents. To do this, we consider a two–port in Fig. 1. The equation I0(RL) has the known fractionally linear view

$$I0 = V0*(RL + r1 + r10)/ (RL*r0 + RL*r10 + r0*r1 + r0*r10 + r1*r10) \quad (1)$$

Here and further the used values are represented in the MATLAB format. The corresponding replacement of the values designations will be understandable without explanations.

We consider the expression (1) as a projective transformation in the sense of projective geometry. Therefore, for this projective transformation, an input-output invariant is performed in the form of cross ratio for four load values $R_L^{SC}, R_L, R_L^{REF}, R_L^{OC}$ and of the corresponding input currents $I_0^{SC}, I_0, I_0^{REF}, I_0^{OC}$

$$\frac{R_L - R_L^{SC}}{R_L - R_L^{OC}} \div \frac{R_L^{REF} - R_L^{SC}}{R_L^{REF} - R_L^{OC}} = \frac{I_0 - I_0^{SC}}{I_0 - I_0^{OC}} \div \frac{I_0^{REF} - I_0^{SC}}{I_0^{REF} - I_0^{OC}}. \quad (2)$$

The three values $R_L^{SC}, R_L^{REF}, R_L^{OC}$ are the given base values. Let it be some minimum R_L^{SC}, maximum R_L^{OC}, and middle R_L^{REF} value. In turn, the values R_L and I_0 are the running ones. The invariant expression (2) do not explicitly contains two–port's parameters. Therefore, it is possible to calculate the load resistor R_L from the measured input currents $I_0^{SC}, I_0, I_0^{REF}, I_0^{OC}$ of this two–port with variable or unstable parameters. But it follows, that possible deviations of both the declared values of the base values $R_L^{SC}, R_L^{REF}, R_L^{OC}$ and measurement errors in the input currents lead to errors in the calculation of the load.

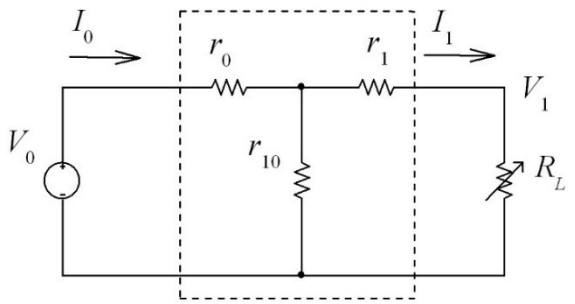

Fig.1 Two-port with a load resistor RL

4 Preparing the training data

Consider the specific example of two–port in Fig. 1. Let us take the following dimensionless parameter values V0=12, r0=4, r1=1, r10=9. In turn, let the load value RL change over a sufficiently large range 5…17. Then, the plot I0(RL) has the view in Fig. 2. Such a monotonous dependency in approximation or regression problems does not require a large amount of samples. This amount will be then determined.

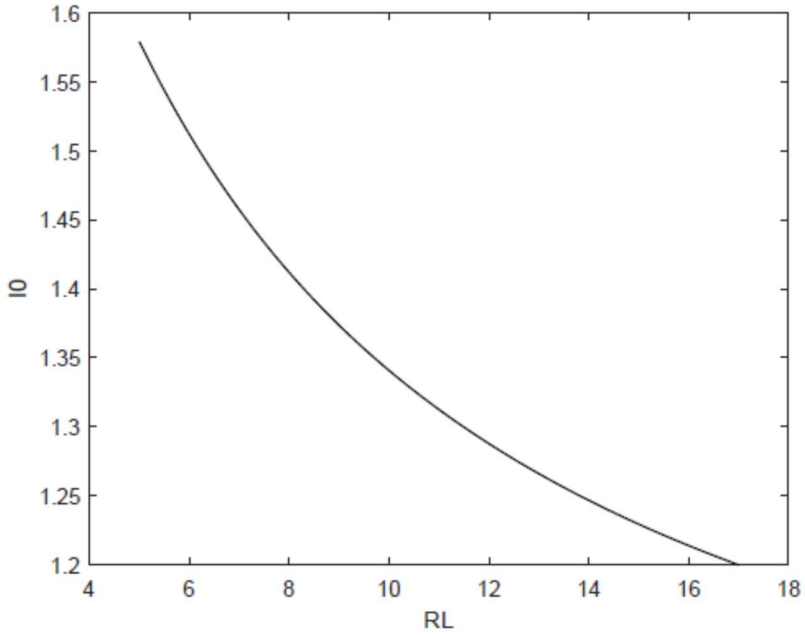

Fig. 2 Plot I0(RL) as a monotone function

As shown above, the training data are given by a set of possible values of both the base resistances R_L^{SC}, R_L^{REF}, R_L^{OC}, the load R_L, and a variable parameter of the two–port itself. Let such a variable parameter be the transverse resistance r10. The total amount of samples is formed due to the mutual search of the base resistances with the load and resistance r10. The corresponding calculated set of input currents I_0^{SC}, I_0, I_0^{REF}, I_0^{OC} forms the input vector of four components, and the load values R_L are the target vector.

4.1 Case of the regular step of changing parameters

Let the resistance value R10 vary over a sufficiently large range of 7... 11 in increments or regular step of 0.666, giving 7 samples, according to Table 1. The range of changes for the base resistance rOC, rSC, rREF is adopted narrower, but also with 7 samples. In turn, the measurement load or sensor resistance rL itself changes in a wide range 5... 17.

It is now necessary to calculate the input vector or input current vector (as column vector) with the following base components I0OC, I0REF, I0SC and measuring component I0MES

$$I0\_49 = [I0OC;\ I0REF;\ I0SC;\ I0MES]. \quad (3)$$

Table 1 Training samples with the regular step of changes

Amount of samples	Range and increment of changes	Input vector	Target vector
7*7=49	R10=7:0.666:11 rL=5:2:17 rOC=22:0.333:24 rSC=2.2:10.0333:2.4 rREF=17.2:0.266:18.8	I0_49	t_49
5*5=25	R10=7:1:11 rL=5:3:17 rOC=22:0.5:24 rSC=2.2:0.05:2.4 rREF=17.2:0.4:18.8	I0_25	t_25
6*6=36	R10=7:0.8:11 rL=5:2.4:17 rOC=22:0.4:24 rSC=2.2:0.04:2.4 rREF=17.2:0.32:18.8	I0_36	t_36

Next, we calculate all the samples for the component I0OC. To do this, in the expression (1), we substitute by turn all the samples rOC for the next sample R10. For these calculations, it is convenient to organize a Script file "train_49". The following code or statement uses the *for* loop with the values rOC, the nested *for* loop with R10, and the user-defined function (1)

```
                    % train_49
            % input current OC samples
                  I0OC=zeros(0);
                    for RL=rOC
                    for r10=R10
        I0oc =in_curr_0(RL,r10); I0OC=[I0OC I0oc];
                    end
                    end
            % input current REF samples
                    . . .
             % input current SC samples
                    . . .
          % input current MES samples and target
                I0MES=zeros(0); t=zeros(0);
                    for RL=rL
                    for r10=R10
      I0mes =in_curr_0(RL,r10); I0MES=[I0MES I0mes];
                    t=[t RL];
                    end
                    end
        I0_49= [I0OC;I0REF;I0SC;I0MES]; t_49 =t;

            function I0 =in_curr_0(RL,r10)
                  V0=12;r0=4;r1=1;
     det=r0.*RL + r10.*RL + r0*r1 + (r0+r1).*r10;
          I0=(V0.*RL + V0*r1 + V0.*r10)./det;
                    end
```

(4)

Thus, the mutual search of the rOC, R10 values yields 49 combinations of I0OC current that are output from these loops as the vector. Similarly, we get 49 combinations for the rest components of the current and the target vector. Finally, the input current vector I0_49 according to (3) comprises four rows of 49 samples. The target vector t_49 includes one row of 49 samples too.

Similarly, the training data are obtained for the initial 5 and 6 samples, according to the Table 1.

4.2 Case of the irregular step of changing parameters

Let us introduce a clock variable with the constant step of changing, as x=0:6. Next, we use, for example, exponential expressions, so that the values are in the change area for the regular step according to the Table 2.

Table 2 Training samples with the irregular step of changes

Clock variable	Exponential expressions	Input vector	Target vector
x=0:6	R10=0.19*(x.^1.7) +7 rL=0.68*(x.^1.6) +5 rOC=0.16*x.^1.4+22 rSC=0.01947*x.^1.3+2.2 rREF=0.085*x.^1.25+17.2	I0_ir_49	t_ir_49
x=0:4	R10=0.3789*(x.^1.7) +7 rL=1.3058*(x.^1.6) +5 rOC=0.2872*x.^1.4+22 rSC=0.033*x.^1.3+2.2 rREF=0.2828*x.^1.25+17.2	I0_ir_25	t_ir_25
x=0:5	R10=0.2593*(x.^1.7) +7 rL=0.9138*(x.^1.6) +5 rOC=0.2101*x.^1.4+22 rSC=0.0247*x.^1.3+2.2 rREF=0.107*x.^1.25+17.2	I0_ir_36	t_ir_36

For the above specific example of two–port in Fig.1, the plot rL(x) in Fig. 3 clearly shows the irregular step of rL value change. For the comparison, the dash line straight y=2*x+5 demonstrates the regular step.

In turn, the plot I0ir_49(x) in Fig. 4 shows the obtained rough linearization of the two-port nonlinear characteristic. As will be shown below, it is not about a linearization, the linearization itself has no effect. This approach thus differs from the polynomial regression [23].

Similar to the expressions (3), (4), we calculate training data I0_ir_49, t_ir_49, I0_ir_25, t_ir_25, and I0_ir_36, t_ir_36 correspondently.

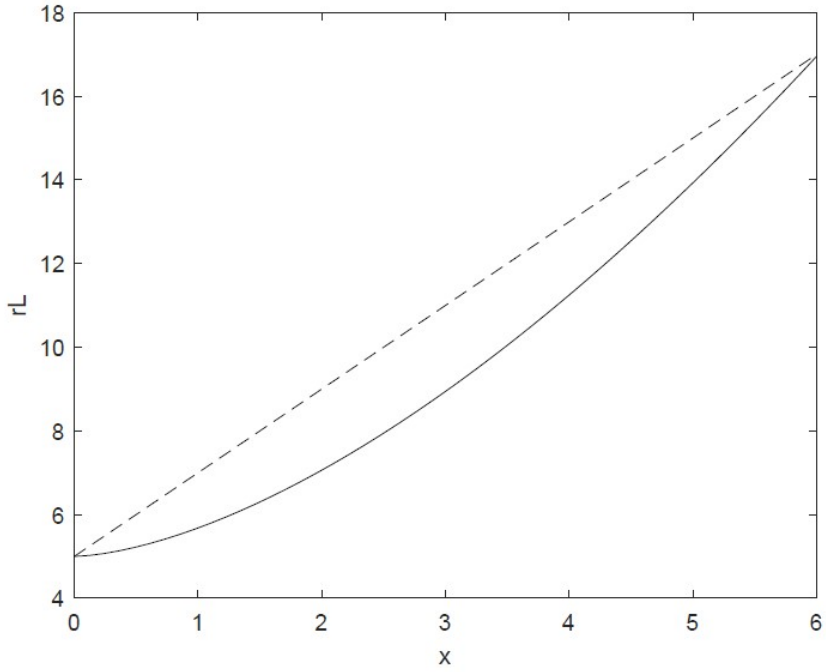

Fig. 3 Irregular step of the RL value change;
the dash line straight demonstrates the regular step

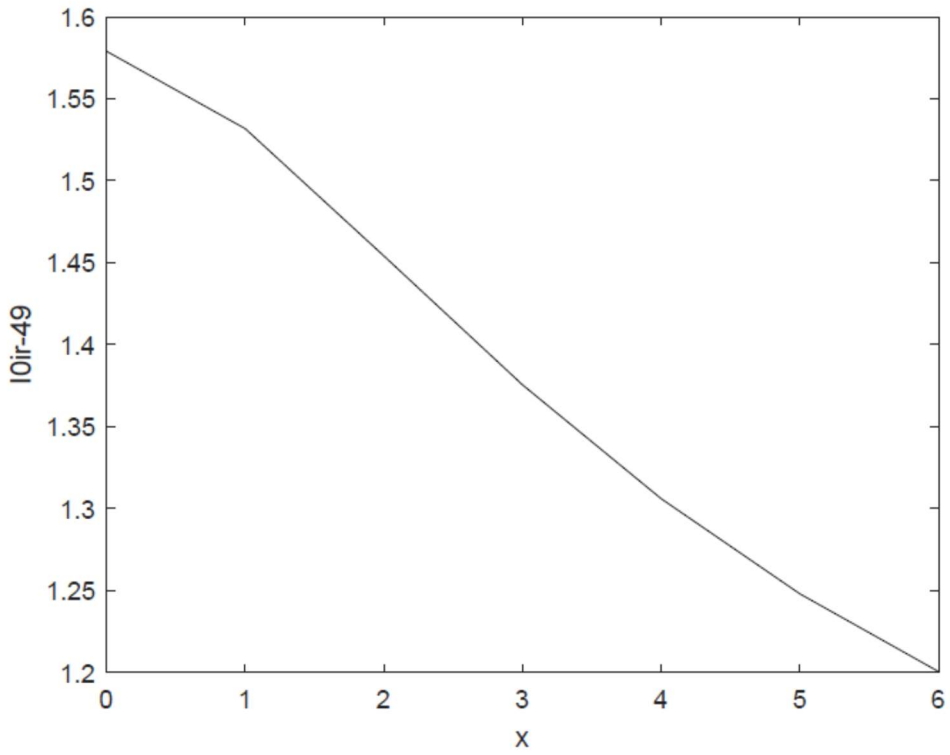

Fig. 4 Rough linearization of the two-port characteristic

4.3 Case of the inverse step of changing parameters

We organize another type of step change according to Table 3.

Table 3 Training samples with the inverse step of changes

Clock value	Exponential expressions	Input vector	Target vector
x=0:6	R10=1.141*(x.^0.7) +7 rL=4.095*(x.^0.6) +5 rOC=0.976*x.^0.4+22 rSC=0.1168*x.^0.3+2.2 rREF=1.0223*x.^0.25+17.2	I0_inv_49	t_inv_49
x=0:4	R10=1.5157*(x.^0.7) +7 rL=5.2233*(x.^0.6) +5 rOC=1.1487*x.^0.4+22 rSC=0.132*x.^0.3+2.2 rREF=1.1314*x.^0.25+17.2	I0_inv_25	t_inv_25
x=0:5	R10=1.2965*(x.^0.7) +7 rL=4.5688*(x.^0.6) +5 rOC=1.0506*x.^0.4+22 rSC=0.1234*x.^0.3+2.2 rREF=1.07*x.^0.25+17.2	I0_inv_36	t_inv_36

The plot rL(x) in Fig. 5 clearly shows the inverse step of rL value change.

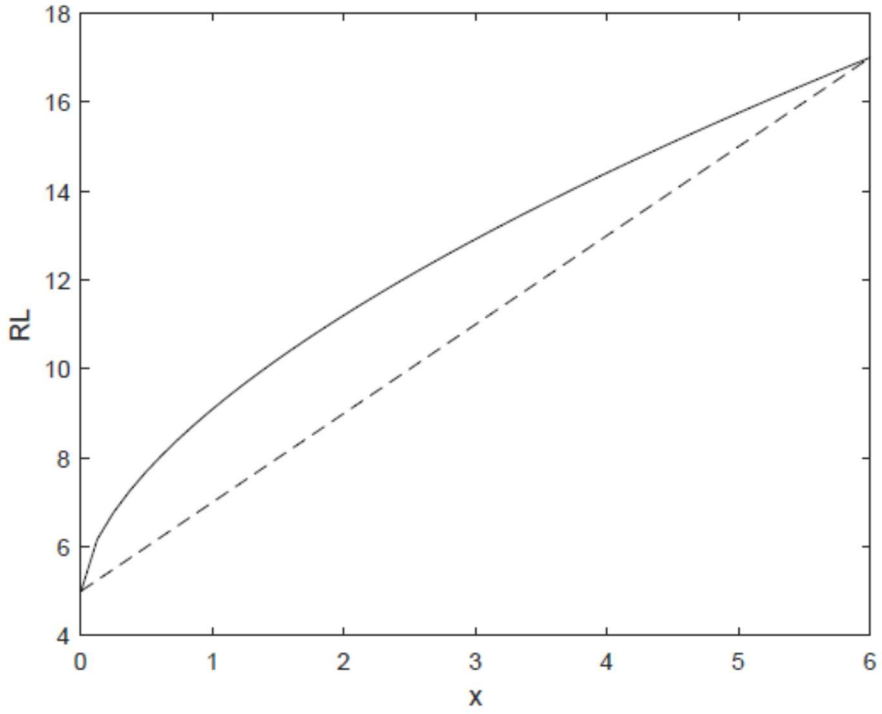

Fig. 5 Invers step of the RL value change;
the dash line straight demonstrates the regular step

In turn, the plot I0_inv_49(x) in Fig. 6 shows the obtained of the two–port nonlinear characteristic. This characteristic is more nonlinear than the original characteristic with regular step.

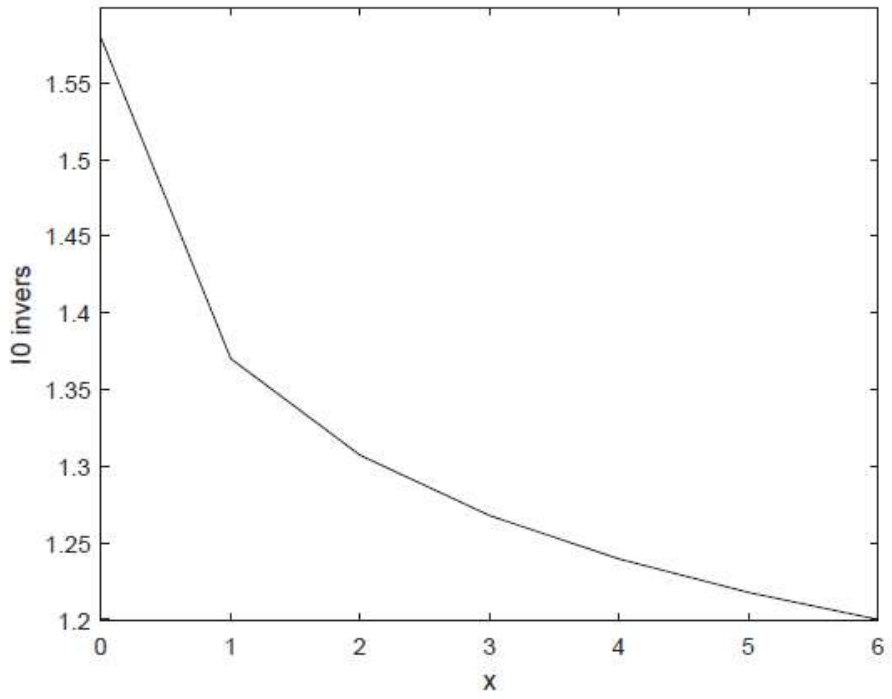

Fig. 6 Invers characteristic of the two-port

Similar to the expressions (3), (4), we calculate all the training data

5 Control data

To check the neural network during or after training, we will prepare several control data, which values should be within the range of changing parameters for training, but not coincide with them. This is control data of different sizes to determine the required size and be sure of the accuracy of the trained network or its capability to generalization. We also use the regular and irregular step of changing parameters.

5.1 Control data with the regular step of changing parameters

Initial parameters for the calculation of control data are presented in Table 4.

Table 4 Control samples with the regular step of changes

Amount of samples	Range and increment of changes	Input vector	Target vector
27*27=729	R10=7:0.1538:11 rL=5:0.461:17 rOC=22:0.0769:24 rSC=2.2:0.00769:2.4 rREF=17.2:0.0307:18	I0ts_729	tts_729
121*121=14641	R10=7:0.0333:11 rL=5:0.1:17 rOC=22:0.01666:24 rSC=2.2:0.001666:2.4 rREF=17.2:0.00666:18	I0ts_14641	tts_14641
4*4=16	-	I0ts_16	tts_16
9*9=81	-	I0ts_81	tts_81

Similar to the expressions (3), (4), we calculate all the control data

5.2 Control data with the irregular step of changing parameters

Initial parameters for the calculation of control data are presented in Table 5.

Table 5 Control samples with the irregular step of changes

Clock value	Exponential expressions	Input vector	Target vector
x=0:26	R10=0.0157*(x.^1.7) +7 rL=0.0653*(x.^1.6) +5 rOC=0.0209*x.^1.4+22 rSC=0.0029*x.^1.3+2.2 rREF=0.0136*x.^1.25+17.2	I0ts_ir_729	tts_ir_729
x=0:120	R10=0.001168*(x.^1.7) +7 rL=0.005655*(x.^1.6) +5 rOC=0.00245*x^1.4+22 rSC=0.000396*x.^1.3+2.2 rREF=0.002014*x.^1.25+17.2	I0ts_ir_14641	tts_ir_14641

Similar to the expressions (3), (4), we calculate all the control data.

6 Neural network training and evaluation for 49 samples

The Fit Data with a Shallow Neural Network section provides a Script example [2]. So, we use this example as "Script_49". When training a network, the initial weights are randomly selected, so in order to get more correct results, we conduct a series of 10 experiments for the training data.

6.1 Training data with only one type

Let us consider the "Script _49" with the regular train data I0_49, t_49

```
% Solve an Input-Output Fitting
x = I0_49; t = t_49; %input and target data
% Training Function and Create a Fitting Network
trainFcn = 'trainlm'; hiddenLayerSize = 8;
net = fitnet (hiddenLayerSize, trainFcn);
% Division of Data for Training, Validation, Testing
net. divideParam.trainRatio = 70/100; net. divideParam.valRatio = 15/100;
net. divideParam.testRatio = 15/100; net. trainParam.epochs = 500;
% Train and test the Network
[net, tr] = train (net, x, t); y = net(x); e = gsubtract (t, y);
performance = perform (net, t, y), figure, plotperform(tr)
% Test the Network by the two control data sets
Rts_729=sim (net, I0ts_729); er_726=100*(tts_729 -Rts_729). /tts_729;
max_729=max(er_729), min_729=min(er_729)

Rts_ir_729=sim(net,I0ts_ir_729); er_ir_729=100*(tts_ir_729-
Rts_ir_729)./tts_ir_729;
max_ir_729=max(er_ir_729); min_ir_729=min(er_ir_729);
```

(5)

Apparently, only extreme (maximum and minimum) relative error values er_729 and er_ir_729 are output for the control regular and irregular data respectively due to the large number of samples.

We put 9 neurons for the hidden layer. This number corresponds to the well-known formula 2n+1, where n=4 defines the input vector dimension [24]. After ten retraining, we give the results of the I0ts_729 in the

.

Table 6 Regular train data I0_49. Number of experiments and relative errors values for the regular control data I0ts_729

No. of exper.	1	2	3	4	5	6	7	8	9	10
errors, %	0.410	0.661	0.082	1.213	0.052	0.526	0.278	0.222	0.088	0.368

As you can see from this table, there are 7 errors up to 0.5%, and the number of errors up to 1% is 9. Therefore, we may introduce a repeatability of the specified relative error or error repeatability, as special index in percent that show a tendency to change the training quality of such data according to Table 7.

Table 7 Regular train data I0_49. The error repeatability of 10 experiments for the regular control data I0ts_729

specified errors, %	0.5	1	2	5	10
error repeatability, %	70	90	100	100	100

Next, we do 10 experiments for 8 and 10 neurons. For all these experiments, we find errors for the control irregular data I0ts_ir_729 also. All the results are given in the Table 8.

As you can see, for regular control data, the repeatability of small errors (about 1-2%) is very high and corresponds to 80-100%. In turn, only large errors (more or equal to10%) appear for the irregular control data. Small errors, of the order of 2%, rarely occur, which corresponds to the repeatability value of 20%.

Table 8. Regular train data I0_49. The error repeatability for the regular and irregular control data

Number of neurons	Control date	regular I0ts_729	irregular I0ts_ir_729
	specified errors, %	0.5 1 2 5 10	0.5 1 2 5 10
8	error repeatability, %	70 80 80 90 100	0 0 10 30 80
9		70 90 100 100 100	0 0 20 70 100
10		50 80 80 100 100	0 0 20 70 100

To confirm such a conclusion, we will change the training data to irregular I0_ir_49, t_ir_49 in the training code (5). All the results are given in the Table 9.

Table 9 Irregular train data I0_ir_49. The error repeatability for the regular and irregular control data

Number of neurons	Control date	regular I0ts_729	irregular I0ts_ir_729
	specified errors, %	0.5 1 2 5 10	0.5 1 2 5 10
8	error repeatability, %	10 20 60 90 90	70 90 90 100 100
9		10 30 50 70 90	60 70 80 90 100
10		10 20 30 80 90	50 70 80 90 100

As you can see, already for the irregular control data, the repeatability of small errors (about 1-2%) is very high and corresponds to 70-90%. At the same time, for the regular control data, small errors (of 1-2%) appear less often, which corresponds to the repeatability value of 20-60%.

Thus, it is confirmed that if control date of a different type than the training data, then larger errors appear to a greater extent. That is, the network does not show the capability to generalization.

To further confirm such a result, experiments were conducted for regular training data on 121 and 256 samples. The result is the same. Also, experiments for regular control data of different sizes as I0ts_16, I0ts_81, and I0ts_14641 showed errors commensurable with the I0ts_729 size.

6.2 Improved the calculation accuracy and capability to generalization

An obvious way to generate training data is to combine different types of training data.

6.2.1 Combining regular and inverse training data

Let us use the combine input vector x= [I0_49, I0_inv_49] and target vector t= [t_49, t_inv_49] in the training code (5). All the results are given in the Table 10. As can be seen, commensurable errors are already occurring for the two types of control data compared to Table 8.

Table 10 Combining regular and inverse train data I0_49, I0_inv_49. The error repeatability for the regular and irregular control data

Number of neurons	Control date	regular I0ts_729	irregular I0ts_ir_729
	specified errors, %	0.5 1 2 5 10	0.5 1 2 5 10
8	error repeatability, %	90 100 100 100 100	90 100 100 100 100
9		100 100 100 100 100	100 100 100 100 100
10		100 100 100 100 100	100 100 100 100 100

To further confirm such a result, experiments were conducted for larger size of control data I0ts_ir_14641. For smaller control data, the training data as I0_ir_49, was used. The results correspond to the Table 10. Thus, the network shows the capability to generalization due to these combined training data. Especially note that the training data do not contain the irregular data, but high repeatability was obtained for the irregular control data.

6.2.2 Combining irregular and inverse training data

Let us combine the input vector x = [I0_ir_49, I0_inv_49] and target vector t= [t_ir_49, t_inv_49]. All the obtained results correspond to the above case. Thus, entirely irregular training data produce good results for regular control data. The network also confirms the capability to generalization by these combine training data.

6.2.3 Combining regular, irregular, and inverse training data

We use the combine the input vector x = [I0_49, I0_ir_49, I0_inv_49] and target vector t= [t_49, t_ir_49, t_inv_49]. All the obtained results correspond to the above cases.

All these results showed that the accuracy of the sensor calculation (of 1 or 2%) is guaranteed for the combined training data with 49 samples. The type and size of the control data is not critical to assessing the quality of the trained network.

7 Neural network training and evaluation for 25 samples

In the considered case with 49 samples, according to Table 8, the network is obtained, as it were, with a high quality of training for regular control data. Now you need to consider the extreme case with fewer samples so that small errors rarely occur for the regular control data, which corresponds to the small error repeatability value also.

7.1 Training data with only one type

Such a case corresponds to 25 samples with regular training data I0_25, t_25. Using code (5), make Table 11, similar to Table 8.

Table 11 Regular train data I0_25. The commensurable error repeatability for the regular and irregular control data

Number of neurons	Control date	regular I0ts_729	irregular I0ts_ir_729
	specified errors, %	0.5 1 2 5 10	0.5 1 2 5 10
8	error repeatability, %	0 30 50 80 80	0 0 10 40 70
9		0 20 40 40 60	0 0 20 30 50
10		0 20 20 60 60	0 10 10 30 60

As you can see, small errors, of 1–2%, rarely occur, which corresponds to the repeatability value of 20–50% for regular control data. Also, the experiments for regular control data of different sizes as I0ts_16, I0ts_81, and I0ts_14641 showed the repeatability commensurable with the I0ts_729 size.

7.2 Improved the calculation accuracy and capability to generalization

Now let us try to increase the accuracy and capability to generalization by combining different types of training data.

7.2.1 Combining regular and inverse training data

Let us use the combine input vector x= [I0_25, I0_inv_25] and target vector t= [t_25, t_inv_25] in the training code (5). All the results are given in the Table 12. As can be seen, it is possible to significantly improve the quality of training. To further confirm such a result, experiments were conducted for larger size of control data I0ts_ir_14641. For smaller control data, the training data I0_ir_49 was used. The results correspond to the Table 12.

Table 12 Combining regular and inverse train data I0_25, I0_inv_25. The error repeatability for the regular and irregular control data

Number of neurons	Control date	regular I0ts_729	irregular I0ts_ir_729
	specified errors, %	0.5 1 2 5 10	0.5 1 2 5 10
8	error repeatability, %	10 60 90 100 100	10 60 90 100 100
9		40 40 50 100 100	40 40 40 100 100
10		30 70 80 90 90	40 60 80 90 90

7.2.2 Combining irregular and inverse training data

Let us combine the input vector x = [I0_ir_25, I0_inv_25] and target vector t= [t_ir_25, t_inv_25]. All the obtained results correspond to the above case.

7.2.3 Combining regular, irregular, and inverse training data

We use the combine the input vector x = [I0_25, I0_ir_25, I0_inv_25] and target vector t= [t_25, t_ir_25, t_inv_25]. All the obtained results are given in the Table 13.

Table 13 Combining regular, irregular, and inverse train data I0_25, I0_ir_25, I0_inv_25. The error repeatability for the regular and irregular control date

Number of neurons	Control date	regular I0ts_729					irregular I0ts_ir_729				
	specified errors, %	0.5	1	2	5	10	0.5	1	2	5	10
8	error repeatability, %	70	90	90	100	100	70	90	90	100	100
9		90	90	90	100	100	90	90	90	90	100
10		70	90	100	100	100	90	90	100	100	100

As can be seen, it is possible to significantly improve the quality of training. To further confirm such a result, experiments were conducted for control data I0ts_ir_14641 and as I0_ir_49.

And so, 25 samples are the minimum number of samples. In this case, it is possible to obtain the accuracy of the sensor calculation (of 2 or 5%) due to a multiple retraining. In turn, experiments with 36 samples give a noticeable increase in the quality of training. It is also possible to obtain the accuracy of the sensor calculation (of 1 or 2%) through multiple retraining. Therefore, you can find a compromise between the number of samples (from 25 to 49) and the time spent on training.

8 Conclusions

The traditional calculation of the load resistance of a two-port circuits with variable parameters leads to redefine these parameters in operating regime. All this leads to the complexity of the possible hardware implementation of the calculation unit and increases the calculation time.

The presented numerical experiments confirm the cumulative effect of applying a feedforward neural network with the two–port invariant properties in the load calculation or measurement.

Such an index as the repeatability of specified error, provides a quantitative assessment of the quality of training.

The training data with one type of parameter change step does not provide quality of training. Control data with the same change step type does not detect errors. The training data with different types of parameter step provides the required quality of the trained network.

The obtained results provide the basis for the study of different generation algorithms for the irregular step and the application for other regression tasks.

Acknowledgments

The study was supported by the Grant RSF No. 22-79-10018 “Controlled kinetic inductance based on superconducting hybrid structures with magnetic materials” (theory development, samples measurements, results evaluation), and by the Moldova State Program Project «Functional nanostructures and nanomaterials for industry and agriculture» no. 20.80009.5007.11 (samples fabrication and characterization).

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability statements

The authors declare that the data supporting the findings of this study are available within the paper

References

1. Chollet F (2021) Deep learning with Python. 2nd ed., NY, Manning Publications Co.
2. Function Approximation and Nonlinear Regression. https://www.mathworks.com/help/deeplearning/function-approximation-and-nonlinear-regression.html (accessed 10 November, 2023.)
3. Almalki M et al (2020) A Levenberg–Marquardt backpropagation neural network for the numerical treatment of squeezing flow with heat transfer model. IEEE Access 8: 227340-227348. https://doi.org/10.1109/ACCESS.2020.3044973
4. Aljohani J et al (2021) Backpropagation of Levenberg Marquardt artificial neural networks for wire coating analysis in the bath of Sisko fluid. Ain Shams Engineering Journal 12(4): 4133-4143. https://doi.org/10.1016/j.asej.2021.03.007
5. Rezk H, Hasaneen E (2015) A new MATLAB/Simulink model of triple-junction solar cell and MPPT based on artificial neural networks for photovoltaic energy systems. Ain Shams Engineering Journal 6(3): 873-881. https://doi.org/10.1016/j.asej.2015.03.001
6. Almassri A et al (2018) Self-calibration algorithm for a pressure sensor with a real-time approach based on an artificial neural network. Sensors 18(8): 2561. https://doi.org/10.3390/s18082561
7. Hamadouche Z et al (2021) Intelligent voltage regulator for distributed generation-based network. Indonesian Journal of Electrical Engineering and Computer Science 23(1): 98-109. http://doi.org/10.11591/ijeecs.v23.i1.pp98-109
8. Yang C, Huang H (2019) Determination of complex permittivity of low-loss materials from reference-plane invariant transmission/reflection measurements. IEEE Access 7:131865-131872. https://doi.org/10.1109/ACCESS.2019.2940723
9. Rivera J et al (2007) Self-calibration and optimal response in intelligent sensors design based on artificial neural networks. Sensors 7(8): 1509-1529. https://doi.org/10.3390/s7081509
10. Michelucci U (2018) Applied Deep Learning—A Case-Based Approach to Understanding Deep Neural Networks. Apress Media, LLC: New York, NY, USA.
11. Shahid A et al (2018) Least squares neural network-based wireless E-Nose system using an SnO2 sensor array. Sensors 18(5): 1446. https://doi.org/10.3390/s18051446
12. Kazemi N et al (2020) A temperature-compensated high-resolution microwave sensor using artificial neural network. IEEE Microwave and Wireless Components Letters 30(9): 919-922. https://doi.org/10.1109/LMWC.2020.3012388
13. Patra J et al (2010) Development of Laguerre neural-network-based intelligent sensors for wireless sensor networks. IEEE Transactions on instrumentation and measurement 60(3): 725-734. https://doi.org/10.1109/TIM.2010.2082390
14. Islam T et al (2015) Temperature effect on capacitive humidity sensors and its compensation using artificial neural networks. Sensors & Transducers 191(8): 126. https://www.proquest.com/scholarly-journals/temperature-effect-on-capacitive-humidity-sensors/docview/1729740249/se-2
15. Penin A (2022) Neural network based calculation of load resistances taking into account the multiport input-to-output ratio invariant properties. Elektrichestvo 4: 47-58. https://doi.org/10.24160/0013-5380-2022-4-47-58
16. Sadiku M, Alexander C (2009) Fundamentals of electric circuits. New York: McGraw-Hill.
17. Bhattacharyya S et al (2014) Linear Systems: A measurement based approach. Springer, India.
18. Penin A (2020) Analysis of electrical circuits with variable load regime parameters: Projective geometry method. 3rd ed. Springer International Publishing, Cham, Switzerland.
19. Ayres F (1967) Schaum's Outline Series theory and problems of projective geometry. New York, McGraw–Hill.
20. Mazin V (1983) Error of measurement in the compound-ratio method. Measurement Techniques 26(8): 628–629.
21. Tcybulskii O (2013) Projective properties of wide-range measurements. Measurement Techniques 56(1): 37-40. https://doi.org/10.1007/s11018-013-0155-8
22. Riaza R (2019) Circuit theory in projective space and homogeneous circuit models. IEEE Transactions on Circuits and Systems I: Regular Papers 66(2): 463-476. https://doi.org/10.1109/TCSI.2018.2868277
23. Raschka S (2015) Python machine learning. Birmingham, Packt Publishing Ltd, UK.
24. Suzuki K (Ed.) (2011) Artificial Neural Networks-Industrial and Control Engineering Applications. InTech, Rijeka, Croatia.