The method of solution of equations with coefficients that contain measurement errors, using artificial neural network

Konrad Zajkowski

Abstract This paper presents an algorithm for solving N-equations of N-unknowns. This algorithm allows to determine the solution in a situation where coefficients $A_i$ in equations are burdened with measurement errors. For some values of $A_i$ (where $i = 1, \ldots, N$), there is no inverse function of input equations. In this case, it is impossible to determine the solution of equations of classical methods.

Keywords Artificial neural network · Measurement errors · Induction motor model · Parameter identification

1 Introduction

Mathematical models that describe electric dependencies in the receiver tested are built from discrete components. For a full description of such a model, it is required to identify the parameters $x_i$ of these elements (see Eq. 1). Most frequently, this identification is carried out indirectly through the measurements of electrical quantities $A_i$ on the object tested [5, 8]. The parameters sought are determined from the mathematical relations (1) that describe the object.

$$
\begin{align*}
A_1 &= f_1(y_1, y_2, \ldots, y_i, \ldots, y_N) \\
A_2 &= f_2(y_1, y_2, \ldots, y_i, \ldots, y_N) \\
& \vdots \\
A_i &= f_i(y_1, y_2, \ldots, y_i, \ldots, y_N) \\
& \vdots \\
A_N &= f_N(y_1, y_2, \ldots, y_i, \ldots, y_N)
\end{align*}
$$

where $f_i$ certain functions depending on the model, $A_i$ the values measured, $y_i$ parameters that describe the model.

The classic method to solve Eq. (1) consists in determining inverse functions (2). Measurement inaccuracies that are contained in $A_i$ are transferred to parameters $y_i$ to be determined.

$$
\begin{align*}
y_1 &= g_1(A_1, A_2, \ldots, A_N) \\
y_2 &= g_2(A_1, A_2, \ldots, A_N) \\
& \vdots \\
y_N &= g_N(A_1, A_2, \ldots, A_N)
\end{align*}
$$

In some cases, the determination of Eq. (2) may not be possible [14, 16]. This means that for adopted coefficients $A_i$, there are no inverse functions $g_i$. Eq. (2) are determined for the values of environment $A_i$ and which contain measurement errors. In this case, approximate solutions are sought which satisfy Relation (3).

$$
|A_i - f_i(y_1, y_2, \ldots, y_N)| \approx 0 \quad (3)
$$

The solution will be close to coefficients $A_i$.

2 An example of a model for identification

The analysis covered a single phase on an induction motor. The purpose of the analysis is to determine current–voltage dependences on the terminals of one motor phase. These relationships can be determined from the model that consists of serially connected elements: $R_s$, $L_s$, $L_e$, $\varphi$, $\alpha$ that are being sought represent many of the phenomena that occur in the motor and the system that is driven. For example, the inertia of the rotor and the system driven will affect $\varphi$, and the angular velocity will exert an influence on mutual inductions, which are described with $L_s$. When searching for

K. Zajkowski
Division of Electrotechnics and Electronics, Technical University of Koszalin, 15-17 Racławicka St., 75-620 Koszalin, Poland
e-mail: konrad.zajkowski@tu.koszalin.pl

© The Author(s) 2012. This article is published with open access at Springerlink.com
In the field of complex numbers, the following can be written:

\[ E_s = E_s e^{\phi_{s}}, \quad U = U e^{\phi_{s}}. \]  

For one mesh, the voltage equation is as follows:

\[ L_s (R_s + R + jX_s) + E_s - U = 0, \]  

where \( X_s = \omega L_s \).

Next, by transforming (7), we determine current \( I_a \):

\[ I_a = \frac{U - E_s}{R_s + R + jX_s}. \]

Voltmeter \( V \) measures the difference in the supply voltage and in the voltage drop across internal resistance \( R \). Thus, in the field of complex numbers, there will be the following:

\[ U_V = U - I_a R. \]  

From Eqs. (8) and (9), one can obtain the following:

\[ U_V = \frac{1}{R_s + R + jX_s} \left[ U(R_s + jX_s) + E_s R \right]. \]

Knowing that the forces and the current are equal, respectively:

\[ P_w = \text{Re}(U I_a'), \quad Q_w = \text{Im}(U I_a'), \quad I_a' = \frac{U^* - E_s^*}{R_s + R - jX_s}. \]

We obtain the following equations:

\[
\begin{align*}
I_a &= \left| \frac{U - E_s}{R_s + R + jX_s} \right| \\
U_V &= \frac{1}{R_s + R + jX_s} \left| U(R_s + jX_s) + E_s R \right| \\
P_w &= \text{Re} \left\{ \frac{\left| U(R_s + jX_s) + E_s R \right|}{(R_s + R)^2 + X_s^2} \right\} \\
Q_w &= \text{Im} \left\{ \frac{\left| U(R_s + jX_s) + E_s R \right|}{(R_s + R)^2 + X_s^2} \right\}
\end{align*}
\]

Equation (11) is consistent with (1). The coefficients of the model of the receiver that are obtained from the above equations are not determinable for all the input parameters \((U_V, I_a, P_w, Q_w)\). There are those areas that result from measurement inaccuracies where the system of Eq. (11) has no solutions.

It was found that these coefficients cannot be determined using the Newton’s interpolation algorithm \([15, 16]\). There are no functions that are inverse to Eq. (11), either.
Process time constant $1/\alpha$ that is being sought, and which is mainly related to the inertia of the rotor and the system driven, can be determined experimentally by observing the course of voltage versus time at the motor terminals immediately after commutation.

In [13], the authors proved that amplitude $E_S$ can be equal to amplitude $U$. In this paper, it was also observed that frequency $f_S$ is similar to the frequency of the mains voltage. It was also noted that phase shift $\varphi_{ei}$ is equal to 0.

In this model, it is assumed that the frequencies of both sources are identical. This assumption does not substantially affect the results of further simulations.

3 Construction of an artificial neural network

Coefficients $E_m$, $R_S$, and $L_S$ can be determined from Eq. (11) using a neural network. The neural network input parameters $x_1 = U_v$ [V], $x_2 = I_a$ [A], $x_3 = P_w$ [W], $x_4 = Q_w$ [VAr] contain measurement errors. Due to the nature of the adopted activation function [1–3], the output neuron of the output layer must be within range $y \in (0, 1)$. The initial values were as follows: $y_1 = R_s$ [kΩ], $y_2 = L_s$ [H], $y_3 = E_s$ [kV].

Training of the network must be for those learning vectors $\sigma = \{x_1, \ldots, x_{N_0} \}, \{d_1^{(k)}, \ldots, d_{N_I}^{(k)} \}$ that do not contain any measurement errors. Learning vectors are constructed from Eqs. (1) or (11) for random values $y_1$, $y_2$, $y_3$ that lie within the set of permissible changes, and which is limited with values $a$ and $b$ [4, 12].

The test vector is built according to Fig. 3 for values $y_i$ that are not contained within the training set.

Further values $U_v$, $I_a$, $P_w$ and $Q_w$ are determined from Eq. (11).

After tests of several neural networks, a decision was made to build a neural network with topology (Fig. 5), with one hidden layer. The weights of neurons are determined by back propagation.

Individual neurons in the network are structured according to Fig. 6.

In the network being built, the following indications were accepted:

- $t$ iteration step, $t = 1, 2, \ldots$
- $y_i^{(k)}(t)$ $i$th output of the neuron $N_i^{(k)}$
- $y_i^{(L)}(t)$ $i$th output of the network
- $k$ network layer, $k = 1, \ldots, L$
- $L$ network output layer, the number of network layers
- $i$ neuron number in layer, $i = 1, \ldots, N_k$
- $x_j^{(k)}(t)$ input signal in the $k$th layer
- $j$ number of the input signal in the $k$th layer, $j = 1, \ldots, N_{k-1}$
- $N_0$ number of inputs to the network
- $N_k$ number of neurons in the $k$th layer
- $N_L$ number of neurons in the last layer
- $s_i^{(k)}(t)$ neuron membrane potential $N_i^{(k)}$ in the $k$th layer

Fig. 4 The workbook that builds learning vectors

The neural network was built in a VBA environment in EXCEL.

The script associated with the button in Fig. 4 determines the random values:

$\begin{align*}
  d(1) &= \text{Int}((200 - 5) \times \text{Rnd}) + 5) / 1000 \quad \text{Rs [kΩ]} = 0.005 - 0.200 \\
  d(2) &= \text{Int}((500 - 5) \times \text{Rnd}) + 5) / 1000 \quad \text{Ls [H]} = 0.005 - 0.500 \\
  d(3) &= \text{Int}((U - 150) \times \text{Rnd}) + 150) / 1000 \quad \text{Es [kV]} = 0.150 - U
\end{align*}$
The input neuron for \( k = 1 \) layer is equal to network inputs \( x_j(t) = x_j^{(1)}(t) \). Each layer has one input \( x_0^{(k)}(t) = 1 \). Other inputs are the outputs of the previous layer.

\[
x_j^k(t) = \begin{cases} x_j(t) & \text{for } k = 1 \\ y_j^{k-1}(t) & \text{for } k = 2, \ldots, L \\ 1 & \text{for } j = 0, \quad k = 1, \ldots, L \end{cases}
\]  

(14)

The error at the output of the network for one learning vector \( \sigma \) is:

\[
Q(t) = \sum_{i=1}^{N_k} (e_i^k(t))^2 = \sum_{i=1}^{N_k} (d_i^k(t) - y_i^k(t))^2
\]

(15)

The weights of the individual neuron inputs are determined from the steepest descent rule:

\[
w(t + 1) = w(t) - \eta \cdot g(w(t))
\]

(16)

where \( g(w(t)) = \left[ \frac{\partial Q(t)}{\partial w_{i_1}}, \frac{\partial Q(t)}{\partial w_{i_2}}, \ldots, \frac{\partial Q(t)}{\partial w_{i_{N_k}}} \right]^T \) is the vector gradient.

From Eq. (16), for any weight in any layer, the following is obtained:

\[
w_{ij}^{(k)}(t + 1) = w_{ij}^{(k)}(t) - \eta \frac{\partial Q(t)}{\partial w_{ij}^{(k)}} = w_{ij}^{(k)}(t) + 2\eta \delta_i^{(k)}(t)x_j^{(k)}(t)
\]

(17)

Parameter \( \delta_i^{(k)}(t) \) is determined differently than for the output layer and the hidden layer:

\[
\begin{align*}
\delta_i^{(L)}(t) &= f(s_i^{(L)}(t)) - x_i^{(L)}(t) \\
\delta_i^{(k)}(t) &= f(s_i^{(k)}(t)) \cdot \sum_{m=1}^{N_k} \delta_m^{(k+1)}(t)w_{im}^{(k+1)}(t) & \text{for } k \neq L
\end{align*}
\]

(18)

where \( \delta_i^{(L)}(t) = d_i^{(L)}(t) - y_i^{(L)}(t) \).

Network training is carried out by an incremental updating of weights, that is, each time after the entry of a successive learning vector, responses are determined and the weights are modified. The simulation is continued until the total output error for entire epoch \( Q^*(t) \) is smaller than the accepted set \( Q_{\text{min}} \).

\[
Q^*(t) = \sum_{m=1}^{M} Q_m(t) \leq Q_{\text{min}}
\]

(19)

where \( M \) is the number of learning vectors in the epoch.

The neuron activation function was adopted as a continuous unipolar function of the signum type:

\[
f(s_i(t)) = \frac{1}{1 + e^{-\beta s_i(t)}}
\]

(20)

where \( \beta \) is the steepness factor.

With low values of coefficient \( \beta \), the function is usually mild. By increasing \( \beta \), the plot becomes steeper until the threshold course is obtained.
The derivative of the activation function is as follows:

\[
f'(s(t)) = \frac{\beta \cdot e^{-\beta s(t)}}{(1 + e^{-\beta s(t)})^2} = \beta \cdot f(s(t)) \cdot (1 - f(s(t)))
\]  

(21)

The calculation sheet in Fig. 7 allows an observation of the characteristic values of the network tested. Starting of the network training produces a script written in VBA that executes in a loop of a neural network algorithm according to (12) and (21) and the block diagram in Fig. 8.

The start of the algorithm is possible for the weights that are selected at random from range \((-1, 1)\) or the reading stored from the previous simulations (Fig. 9).

4 Learning of the network

The set of learning vectors that form one epoch consists of 200 elements. Owing to the ability to read and write data, it is possible to pause the simulation and to change its parameters during operation [4, 12].

Reading of the stored data allows a continuation of the previously stopped simulation. The window in Fig. 9 retrieves the values from the appropriate data sheet (Fig. 10).

The output values of the neurons (Fig. 8a) are determined by analyzing the neurons in layers starting from the input layer; the output layer comes last.
For all the layers, the steepness factor $b$ (20) of activation function $f_x$ was assumed as equal to 0.1. The determination of value $d_{i}^{(k)}(t)$ (Fig. 8b) shall be in accordance with Formula (18). This determination takes place starting from the output layer; the input layer comes last.

For all the layers, the steepness factor $\beta$ (20) of activation function $f_x$ was assumed as equal to 0.1. The determination of value $d_{i}^{(k)}(t)$ (Fig. 8b) shall be in accordance with Formula (18). This determination takes place starting from the output layer; the input layer comes last.

For all the layers, the steepness factor $\beta$ (20) of activation function $f_x$ was assumed as equal to 0.1. The determination of value $d_{i}^{(k)}(t)$ (Fig. 8b) shall be in accordance with Formula (18). This determination takes place starting from the output layer; the input layer comes last.

For all the layers, the steepness factor $\beta$ (20) of activation function $f_x$ was assumed as equal to 0.1. The determination of value $d_{i}^{(k)}(t)$ (Fig. 8b) shall be in accordance with Formula (18). This determination takes place starting from the output layer; the input layer comes last.

For all the layers, the steepness factor $\beta$ (20) of activation function $f_x$ was assumed as equal to 0.1. The determination of value $d_{i}^{(k)}(t)$ (Fig. 8b) shall be in accordance with Formula (18). This determination takes place starting from the output layer; the input layer comes last.

For all the layers, the steepness factor $\beta$ (20) of activation function $f_x$ was assumed as equal to 0.1. The determination of value $d_{i}^{(k)}(t)$ (Fig. 8b) shall be in accordance with Formula (18). This determination takes place starting from the output layer; the input layer comes last.

For all the layers, the steepness factor $\beta$ (20) of activation function $f_x$ was assumed as equal to 0.1. The determination of value $d_{i}^{(k)}(t)$ (Fig. 8b) shall be in accordance with Formula (18). This determination takes place starting from the output layer; the input layer comes last.

For all the layers, the steepness factor $\beta$ (20) of activation function $f_x$ was assumed as equal to 0.1. The determination of value $d_{i}^{(k)}(t)$ (Fig. 8b) shall be in accordance with Formula (18). This determination takes place starting from the output layer; the input layer comes last.

For all the layers, the steepness factor $\beta$ (20) of activation function $f_x$ was assumed as equal to 0.1. The determination of value $d_{i}^{(k)}(t)$ (Fig. 8b) shall be in accordance with Formula (18). This determination takes place starting from the output layer; the input layer comes last.

For all the layers, the steepness factor $\beta$ (20) of activation function $f_x$ was assumed as equal to 0.1. The determination of value $d_{i}^{(k)}(t)$ (Fig. 8b) shall be in accordance with Formula (18). This determination takes place starting from the output layer; the input layer comes last.

For all the layers, the steepness factor $\beta$ (20) of activation function $f_x$ was assumed as equal to 0.1. The determination of value $d_{i}^{(k)}(t)$ (Fig. 8b) shall be in accordance with Formula (18). This determination takes place starting from the output layer; the input layer comes last.

For all the layers, the steepness factor $\beta$ (20) of activation function $f_x$ was assumed as equal to 0.1. The determination of value $d_{i}^{(k)}(t)$ (Fig. 8b) shall be in accordance with Formula (18). This determination takes place starting from the output layer; the input layer comes last.

For all the layers, the steepness factor $\beta$ (20) of activation function $f_x$ was assumed as equal to 0.1. The determination of value $d_{i}^{(k)}(t)$ (Fig. 8b) shall be in accordance with Formula (18). This determination takes place starting from the output layer; the input layer comes last.

For all the layers, the steepness factor $\beta$ (20) of activation function $f_x$ was assumed as equal to 0.1. The determination of value $d_{i}^{(k)}(t)$ (Fig. 8b) shall be in accordance with Formula (18). This determination takes place starting from the output layer; the input layer comes last.

For all the layers, the steepness factor $\beta$ (20) of activation function $f_x$ was assumed as equal to 0.1. The determination of value $d_{i}^{(k)}(t)$ (Fig. 8b) shall be in accordance with Formula (18). This determination takes place starting from the output layer; the input layer comes last.

For all the layers, the steepness factor $\beta$ (20) of activation function $f_x$ was assumed as equal to 0.1. The determination of value $d_{i}^{(k)}(t)$ (Fig. 8b) shall be in accordance with Formula (18). This determination takes place starting from the output layer; the input layer comes last.

For all the layers, the steepness factor $\beta$ (20) of activation function $f_x$ was assumed as equal to 0.1. The determination of value $d_{i}^{(k)}(t)$ (Fig. 8b) shall be in accordance with Formula (18). This determination takes place starting from the output layer; the input layer comes last.

For all the layers, the steepness factor $\beta$ (20) of activation function $f_x$ was assumed as equal to 0.1. The determination of value $d_{i}^{(k)}(t)$ (Fig. 8b) shall be in accordance with Formula (18). This determination takes place starting from the output layer; the input layer comes last.

For all the layers, the steepness factor $\beta$ (20) of activation function $f_x$ was assumed as equal to 0.1. The determination of value $d_{i}^{(k)}(t)$ (Fig. 8b) shall be in accordance with Formula (18). This determination takes place starting from the output layer; the input layer comes last.

For all the layers, the steepness factor $\beta$ (20) of activation function $f_x$ was assumed as equal to 0.1. The determination of value $d_{i}^{(k)}(t)$ (Fig. 8b) shall be in accordance with Formula (18). This determination takes place starting from the output layer; the input layer comes last.

For all the layers, the steepness factor $\beta$ (20) of activation function $f_x$ was assumed as equal to 0.1. The determination of value $d_{i}^{(k)}(t)$ (Fig. 8b) shall be in accordance with Formula (18). This determination takes place starting from the output layer; the input layer comes last.

For all the layers, the steepness factor $\beta$ (20) of activation function $f_x$ was assumed as equal to 0.1. The determination of value $d_{i}^{(k)}(t)$ (Fig. 8b) shall be in accordance with Formula (18). This determination takes place starting from the output layer; the input layer comes last.
Correction of the values of weights (Fig. 8c) is carried out according to Relation (17).

Network learning factor $\eta$ from Formula (17) was adopted on the first stage of the simulation as being constant and equal to 0.1. After an analysis of ca. 70,000 epochs, the value of target function $Q^*(t)$, which was calculated in accordance with Formula (19), began to oscillate on the level of 1.42. A decrease in $Q^*(t)$ occurred only after a reduction in network learning rate $\eta$. The correct procedure for the network training should provide for an ability to change this ratio during the analysis (Fig. 11).

Oscillations around the optimal solution are manifested with a momentary increase in the value of $Q^*(t)$. $Q'(t) \geq Q^*(t-1)$  \hspace{1cm} (22)

Once the required value of $Q^*(t)$ from Eq. (19) has been reached, the network test is performed (Fig. 12).

5 Network test

The network test consists in determining the values of $U_V$, $I_P$, $P_W$, and $Q_W$ from Relation (11). These values are then substituted into the neural network input, whose solution is $R_S$, $L_S$, and $E_S$. The window in Fig. 12 also allows a determination of the network’s solution for a selected set of weights.

Table 1 illustrates the network test for randomly selected values of $R_S$, $L_S$, and $E_S$.

The relative error for all the output neurons for the randomly adopted input vectors is:

$$\delta_1 = \frac{y_1 - R_S}{R_S}, \quad \delta_2 = \frac{y_2 - L_S}{L_S}, \quad \delta_3 = \frac{y_3 - E_S}{E_S}$$  \hspace{1cm} (23)

The total network error for the accepted values of $R_S$, $L_S$, and $E_S$ are:

$$\delta = \left| \frac{y_1 - R_S}{R_S} \right| + \left| \frac{y_2 - L_S}{L_S} \right| + \left| \frac{y_3 - E_S}{E_S} \right|.$$  \hspace{1cm} (24)

The percentage error made by the network is determined from the largest error (the top bar in the chart in Fig. 13), and it is equal to 26.8 %.

6 Conclusions

The large error value is shown for the input values that occur least frequently in the training set. An improved performance is possible by enlarging the training set or by reducing the range of acceptable changes of the values being sought.
Owing to the method presented of the selection of the electrical model parameters from the values that are measured on the receiver, it is not required to build any complex physical and electrical dependences. The engineering method of voltage, current, and power measurement allows one to determine the parameters of the model for constant electrical and mechanical conditions in the engine. The method presented is particularly useful in situations where measurement errors make it impossible to solve Eq. (2).

Building of a network with the use of the VBA environment is relatively simple. It requires the knowledge of the language basics. An important advantage of this approach is the ability to build its own networks of any topology. The design loop iteration depends largely on how one defines those variables that describe the network.

### Table 1 Verification of the network error

| Random values | Network input | Network output | Relative error |
|---------------|---------------|----------------|---------------|
| $R_5$ [Ω] | $L_5$ [H] | $E_5$ [V] | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $y_1$ | $y_2$ | $y_3$ | $y_4$ | $Q$ [W] | $P$ [VA] | $R_s$ [Ω] | $L_s$ [H] | $E_s$ [V] | $\delta_1$ | $\delta_2$ | $\delta_3$ | $\delta$ |
| 20 | 0.2 | 180 | 229.95 | 0.7576 | 53.21 | 165.88 | 19.721 | 0.210 | 168.626 | -0.014 | 0.050 | -0.063 | 0.127 |
| 151 | 0.315 | 193 | 229.97 | 0.2048 | 39.395 | 25.7897 | 147.083 | 0.354 | 192.849 | -0.026 | 0.124 | -0.001 | 0.151 |
| 150 | 0.09 | 197 | 229.96 | 0.2159 | 48.794 | 9.18694 | 152.107 | 0.093 | 197.019 | 0.014 | 0.033 | 0.0001 | 0.048 |
| 147 | 0.3 | 197 | 229.97 | 0.1888 | 36.563 | 23.415 | 147.413 | 0.352 | 192.976 | 0.003 | 0.173 | -0.020 | 0.197 |
| 47 | 0.46 | 193 | 229.99 | 0.2434 | 17.368 | 53.211 | 41.373 | 0.409 | 195.946 | -0.120 | -0.111 | 0.015 | 0.246 |
| 150 | 0.09 | 193 | 229.95 | 0.2421 | 54.707 | 10.301 | 151.590 | 0.092 | 196.866 | 0.011 | 0.022 | 0.020 | 0.053 |
| 45 | 0.448 | 190 | 229.98 | 0.2706 | 19.015 | 59.256 | 41.182 | 0.408 | 195.771 | -0.085 | -0.089 | 0.030 | 0.205 |
| 145 | 0.4 | 190 | 229.97 | 0.2083 | 36.218 | 31.353 | 145.773 | 0.357 | 192.810 | 0.005 | -0.108 | 0.015 | 0.128 |
| 50 | 0.03 | 160 | 229.73 | 1.3705 | 309.422 | 58.163 | 58.641 | 0.026 | 159.829 | 0.173 | -0.133 | -0.001 | 0.307 |
| 150 | 0.1 | 200 | 229.96 | 0.1955 | 44.0056 | 9.20584 | 152.394 | 0.093 | 197.103 | 0.016 | -0.070 | -0.014 | 0.100 |
| 140 | 0.317 | 189 | 229.96 | 0.2384 | 44.6931 | 31.755 | 146.133 | 0.354 | 192.544 | 0.044 | 0.117 | 0.019 | 0.179 |
| 40 | 0.317 | 195 | 229.98 | 0.3259 | 28.036 | 69.507 | 40.416 | 0.402 | 195.029 | 0.010 | 0.268 | 0.0002 | 0.279 |
| 135 | 0.4 | 195 | 229.97 | 0.1896 | 31.937 | 29.691 | 143.751 | 0.358 | 192.972 | 0.065 | -0.105 | -0.010 | 0.180 |
| 145 | 0.3 | 190 | 229.96 | 0.2311 | 44.568 | 28.936 | 146.443 | 0.353 | 192.636 | 0.010 | 0.177 | 0.014 | 0.201 |
| 140 | 0.4 | 190 | 229.97 | 0.2125 | 36.378 | 32.615 | 145.157 | 0.356 | 192.790 | 0.037 | -0.110 | 0.015 | 0.162 |

Fig. 12 The window for testing and results analysis

![Fig. 12](image1)

![Fig. 13](image2)

Fig. 13 Mistake made by the network
In the present solution, the individual variables occupy adjacent bytes of the memory. A sample definition of the variable holding the weights of neurons is:

```
Public w(1 To Lweights)As Single
```

where Lweights is the number of weights of all neurons. This solution facilitates the construction of a loop program, but special attention is to be paid to assigning the weight number with the neuron number.

An alternative is to build one’s own variable (using the opportunity to build one’s own type of variables) that represents the neuron, and then group all the parameters that describe the type of the neuron in this variable. This approach will make the program more transparent, but there are problems in the construction of iterative loops. This will make the source code longer and will require more CPU load.

Open Access This article is distributed under the terms of the Creative Commons Attribution License which permits any use, distribution, and reproduction in any medium, provided the original author(s) and the source are credited.

References

1. Duer S (2009) Artificial neural network-based technique for operation process control of a technical object. Def Sci J 59(3):305–313. http://publications.drdo.gov.in/ojs/index.php/dsj
2. Duer S (2010) Diagnostic system with an artificial neural network in diagnostics of an analogue technical object. Neural Comput Appl 19(1):55–60
3. Duer S (2010) Diagnostic system for the diagnosis of a reparable technical object, with the use of an artificial neural network of RBF type. Neural Comput Appl 19(5):691–700
4. Duer S, Duer R (2010) Diagnostic system with an artificial neural network which determines a diagnostic information for the servicing of a reparable technical object. Neural Comput Appl 19(5):755–766
5. Duer S (2010) Investigation of the operation process of a reparable technical object in an expert servicing system with an artificial neural network. Neural Comput Appl 19(5):767–774
6. Duer S (2011) Qualitative evaluation of the regeneration process of a technical object in a maintenance system with an artificial neural network. Neural Comput Appl 20(5):741–752
7. Duer S (2010) Expert knowledge base to support the maintenance of a radar system. Def Sci J 60(5):531–540. http://publications.drdo.gov.in/ojs/index.php/dsj
8. Duer S (2011) Modelling of the operation process of reparable technical objects with the use information from an artificial neural network. Expert Syst Appl 38:5867–5878. http://dx.doi.org/10.1016/j.eswa.2010.11.036
9. Duer S (2011) Assessment of the quality of decisions worked out by an artificial neural network which diagnoses a technical object. Neural Comput Appl. doi:10.1007/s00521-011-0725-0. http://www.springerlink.com/openurl.asp?genre=article&id
10. Duer S (2011) Examination of the reliability of a technical object after its regeneration in a maintenance system with an artificial neural network. Neural Comput Appl. doi:10.1007/s00521-011-0723-2. http://www.springerlink.com/openurl.asp?genre=article&id
11. Duer S (2011) Applications of an artificial intelligence for servicing of a technical object. Neural Comput Appl. doi:10.1007/s00521-011-0788-y
12. Duer S (2012) Artificial neural network in the control process of object’s states basis for organization of a servicing system of a technical objects. Neural Comput Appl 21(1):153–160
13. Gacek Z (1999) Technika wysokich napieć. Izolacja wysokonapięciowa w elektroenergetyce. Przepiecia i ochrona przed przepięciami. Skrypt uczelniany nr 2137 wyd. III. Wydawnictwo Politechniki Śląskiej, Gliwice (in polish)
14. Zajkowski K (2004) Algorytm do rozwiązywania układów równań z danymi obarczonymi błędami pomiarowymi. In: Conference on computer applications in electrical engineering, T.II, IEP Politechniki Poznańskiej, pp 499–502 (in polish)
15. Zajkowski K (2009) Analysis of overvoltages on inductive system with varistor and capacitor. Poznan University of Technology, Academic Journals Issue 59 2009, Electrical Engineering, Publication by Poznan University of Technology, pp 87–97
16. Zajkowski K (2011) (monograph) Analiza stanu nieustalonego w obwodach rezystancyjno-indukcyjnych w aspekcie minimalizacji przepięć komutacyjnych. Wydawnictwo Uczelniane Politechniki Koszalińskiej, Koszalin (in polish)