ARTIFICIAL NEURAL NETWORK FOR TESTING ANALOG CIRCUITS

Janusz A. Starzyk and Mohamed A. El–Gamal

Department of Electrical and Computer Engineering
Ohio University
Athens, Ohio 45701–2979

Abstract

This paper describes a new method for testing of analog circuits with the aid of neural networks. It takes an advantage of the high parallel information processing capabilities of these networks. The testing problem is efficiently formulated in a neural network context. A back propagation neural network and a functional link net have been trained to synthesize the complicated mapping from the circuit measurements space to the circuit elements space. Testing example is presented to demonstrate the proposed method.

I. Introduction

Fault diagnosis of analog circuits has become an increasingly important issue in circuit design, fabrication and maintenance. One can distinguish two main approaches when dealing with analog testing and diagnosis. The parameter identification approach where all circuit elements are identified and compared to their nominal values. Elements that are outside their tolerances are considered faulty. The second approach is often referred to as the fault location approach where only faulty elements are identified under the assumption that there are few of them and the rest are fault free . By a fault, we mean any change of an element value from its nominal value outside the tolerance limits that can cause the failure of the circuit performance.

Neural networks and their applications have been in the focus of interest in recent years. One of the most popular neural network models is the back propagation multilayered neural network [1]. The reason is its ability to synthesize complicated input–output mappings through learning from examples. Back propagation neural network have been used in a number of applications such as pattern recognition and classification [2], and power system security assessment [3]. The back propagation algorithm suffers from a very slow convergence since it is relied on the method of steepest descent. In an effort to speed up the learning rate Pao introduced the functional link net [4] that improves the input representation by using higher order terms.

In this paper a new method that uses neural networks to aid fault diagnosis of analog circuits is proposed. The objective is to build a neural network that can identify the element values of the circuit under test if they are within their tolerances. If the circuit is faulty this neural network is used to locate the faults by recognizing the faulty elements and indicating the direction of their changes. In other words one would like to have a trained neural network that can be used to determine the status of the circuit under test. If it is fault free, actual values of circuit elements are identified by the trained neural network. When the circuit is faulty such neural network determines the faulty elements by approximating their actual values. These approximated values are primarily used to indicate that these elements are faulty and that their values have increased or decreased out of tolerance bounds.

Results reported here describe the procedure to construct this neural network using back propagation model. We also show that the functional link net can be used efficiently to build such network. A selection technique that determines the most important nonlinear terms to be included in a functional link net is also introduced.

The paper is organized as follows. In Section II we present the formulation of the new proposed method in a neural network context. An efficient approach for selection of higher order terms to be used in a functional link net is described in Section III. Illustrative example which shows results of using back propagation neural network and functional link net is given in Section IV.

II. Problem Formulation

The basic steps used in the development of an artificial neural network for analog testing are explained in this section. We start with the features selected to represent the diagnosis (testing) problem. The construction of the training set that is presented to the neural network during learning is then described and the learning algorithm is briefly outlined.

Feature Selection

The selection of the features that are going to be used in the input–output training data is an important step towards applying this neural network approach. The importance of such selection stems from the fact that these features should contain enough information to adequately represent the diagnosis problem. It seems natural in this case, since we are concerned with diagnosis rather than analysis, to select circuit measurements as inputs and the corresponding circuit element values as outputs.

In this research the short circuit transfer admittance functions were selected as inputs and associated circuit element values as outputs. This selection provides a set of independent circuit functions for DC circuits, which we have used as test examples. The proposed approach however is still applicable to general circuits with active elements. Other circuit functions, like nodal voltages, could have been selected without changing the final results.

Mathematical relations between transfer admittances and circuit element values can be represented as follows. Assume that a linear circuit under test has b elements and n nodes, n_a of them accessible for measurement and excitation and n_i inaccessible. Elements that are incident with accessible nodes only are called free elements and those incident with at least one inaccessible node are called fundamental elements. The transfer admittance between any two accessible nodes is of the form [5],

$$Y_{ij} = y_{ij} + \frac{\sum_k g_k}{Q}, \quad i \neq j \qquad (1)$$

where y_{ij} is the admittance of the free element directly connecting the two accessible nodes i and j, Q is the sum of different terms, each consisting of a different product of n_i fundamental element admittances. In the numerator, every g_k is a product of (n_i+1) element admittances.

Training Set Design

The construction of the training set used in the learning phase of a neural network, resembles the formulation of fault diagnosis equations in other conventional testing techniques. Methodology adopted in this construction is explained next.

CH2868-8/90/0000-1851$1.00 © 1990 IEEE

We simulate the circuit with different combinations of element values at the boundaries of their tolerances, the case with all element values at their nominal is also simulated. In each simulation the transfer admittances are computed. These admittances are used as inputs with the corresponding circuit element values as outputs to form a training set. A schematic representation when the output consists of two elements is shown in Fig. 1. The coordinates of the black dots are the element values that might be used in the simulations. In general, if the circuit has b elements we will have a hypercube. Corners of this hypercube represent different combinations of elements used in the simulations with values at their tolerance bounds. The center of this hypercube represents the nominal case.

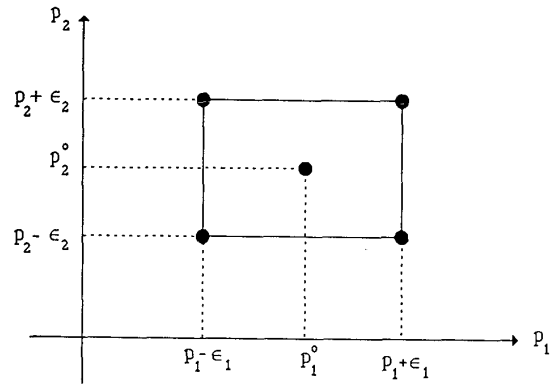

Fig. 1 Schematic representaion of the circuit element values used in the simulatiom.

The idea behind this choice of circuit element values is to present extreme scenarios (elements at their tolerance bounds) to a neural network in the learning process. Consequently, this choice provides a distinction between fault free and faulty situations. For a fault free circuit the element values are inside the range which a trained neural network had encountered during learning. However, for a faulty circuit the element values are outside the range used during learning phase. Therefore, in the testing stage identifying the fault free cases could be viewed as interpolations and evaluating faulty cases as extrapolations.

In fact, our experience as well as others [3] is that the ability of a neural network to perform interpolation is more robust and accurate than to perform extrapolation. This makes our choice of the training set consistent with the objective set up for the diagnosis. The trained neural network will perform interpolation to estimate fault free circuit element values. In faulty cases the trained neural network will perform a limited extrapolation that determines the faulty elements and indicates whether their values have increased or decreased outside the tolerance bounds with a rough approximation of their actual values.

The choice of input dimension (i.e., number of input nodes, which is the number of measurements in this application) is also important. It should be large enough to allow the neural network to synthesize the input–output mapping. If the input dimension is small with information that is inadequate to define the problem, the neural network can not achieve the necessary categorization. At the same time input dimensions should contain the most independent information that is essential for successful characterization without too much redundancy that might unnecessary increase the size of the problem. This point is going to be discussed in the next section.

Another key factor that can significantly improve the learning rate is to appropriately map the training data to the interval (0,1) before being fed to a neural network in a way that is helpful in element categorization. The idea behind this normalization procedure is to map the training data at each input and output node to the interval [A,B] where $0< A < B < 1$. At testing stage, values greater or less than those encountered in the training data are mapped to the intervals (B,1) and (0,A) respectively. This mapping, particularly for output nodes, ensures that circuit elements which are fault free will have values in the interval [A,B] and faulty ones will occupy the remaining of the (0,1) interval. The choice of B is arbitrary and is decided upon before training, A is dependent on the value of B and is equal to $1 - B$. The mathematical details of the normalization are not given here due to space limitations.

Back Propagation Algorithm

A typical multilayered neural network used in this research is shown in Fig. 2. It consists of an input layer, one hidden layer and an output layer. Each layer consists of a number of nodes. Every node in a layer is connected to all nodes in the succeeding layer in a strictly feedforward manner. These connections are called connection weights or simply weights.

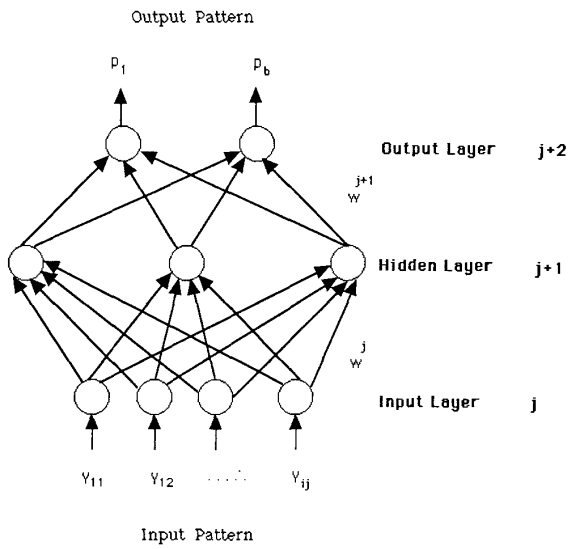

Fig. 2 Back propagation network used in the testing example.

The input data is fed to the nodes of the input layer which acts as a buffer. The outputs of these nodes are then fed to the hidden layer. The input to each node in the hidden layer is the summation of these outputs, each multiplied by its associated connection weight. These inputs, after passing through a sigmoid nonlinearity are multiplied by their associated weights and used as inputs to the nodes in the output layer.

Continuing in the forward path, these inputs to the output layer are in turn passed through sigmoid nonlinearity to produce the network outputs. Starting from a random set of initial weights, a neural network produces its own output pattern which is compared with the desired output pattern. Consequently, a least squares error function is constructed. This error function is minimized by adjusting the weights between different nodes to produce the desired output using the method of steepest descent. The training data is represented many times to the network until the convergence of connection weights is achieved. The details of this algorithm and the scheme to update the weights between different layers during iterations are presented in [1].

III. Functional link net and selection of higher order terms

The functional link net was introduced in [4] in order to increase the slow convergence often encountered in back propagation networks. It is essentially a feedforward network that uses the back propagation algorithm. The basic principle of this net is to expand the input data representation by increasing the dimensionality of the input space. This can be done by the usage of additional input data that incorporates higher order effects. The expanded input data is then used in the training instead of the actual input.

In the so called tensor model, the additional input data added as extra input nodes are obtained by multiplying each component of the input pattern by the entire pattern vector. In other words if the input pattern is denoted by the set

$$\{x_i:\ 1 \le i \le n\} \qquad (2)$$

then the input pattern in the tensor model that includes second order terms is described as

$$\{x_i, x_i x_j:\ 1 \le i \le n,\ i \le j\} \qquad (3)$$

and we can continue this way to include other terms of higher orders in the input pattern.

This improved representation significantly increases the number of components in the input pattern and might lead to an unacceptable increase in the complexity of the problem. Consequently, there is an absolute need to find a way to select from additional components only those that contribute the most to the improvement of the input representation.

In order to achieve this selection we propose the following approach. To facilitate our analysis, let us represent the expanded input data for different patterns in a training matrix P with t rows and s columns. The rows of P represent training patterns and its columns represent input nodes (components) that describe each pattern which includes original inputs and their second order nonlinear combinations. We suggest to use the QR factorization algorithm with pivoting [6], to choose and order the most important training patterns and also to decide upon the most independent measurements to be selected and ordered from the set of original measurements and their second order combinations.

We start by selecting the most independent patterns from the training set. This is accomplished by performing the QR factorization on the matrix P^T. We denote the matrix that has these selected patterns only by P'^T.

In order to select the most independent measurements we run a QR process on P' and measurements corresponding to the selected columns only are chosen to represent each input pattern. Accordingly, we denote the new training matrix after QR selection by P_r. The effect of incorporating higher order combinations of measurements (e.g., third order combinations) can be determined in a similar way. The QR algorithm can then be used again to determine which of these new combinations may improve the input patterns representation. The simulation results showed that this selection algorithm is effective in reducing both the number of training patterns and the nonlinear measurement combinations to only those which are contributing the most to the learning task.

IV. Test Example

Consider the resistive circuit shown in Fig. 3 which was previously studied in [5]. Nodes 1, 2, 3 and 4 are accessible while nodes 5 and 6 are inaccessible. The nominal values of elements $G_i = 1$, $i = 1, 2, 3$ and $G_i = 2$, $i = 4, 5, 6$ with tolerances $\epsilon_i = \pm 10\%$, $i = 1,2, \ldots, 6$. The six transfer admittances are given by

$$Y_{12} = \frac{G_1 G_2 G_3}{Q} \quad , \quad Y_{13} = \frac{G_1 G_4 (G_2+G_3+G_5)}{Q}$$

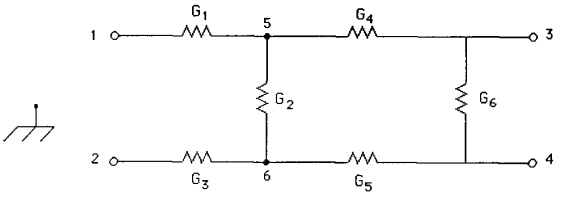

Fig. 3 Resistive circuit example.

$$Y_{14} = \frac{G_1 G_2 G_5}{Q} \quad , \quad Y_{23} = \frac{G_2 G_3 G_4}{Q}$$

$$Y_{24} = \frac{G_3 G_5 (G_1+G_2+G_4)}{Q} \quad , \quad Y_{34} = G_6 + \frac{G_2 G_4 G_5}{Q}$$

where $Q = (G_1+G_4)(G_2+G_3+G_5) + G_2(G_3+G_5)$.

We first train a back propagation neural network so that it can be used in the diagnosis of the circuit of Fig. 3 as explained in Section II. In other words this network is trained to identify the element values of the circuit of Fig. 3 if they are within their tolerances. Moreover, in the case of single or double fault, the trained network should locate the faulty elements and indicate the direction of the changes of their values.

The training set is generated by simulating the resistive circuit of Fig. 3 with nominal element values and then with different combinations of two resistors at a time at the bounds of their tolerances, while the rest are at their nominal values. For each simulation, the six transfer admittances shown above were calculated. Consequently, the training set consists of 61 six–dimensional patterns, the inputs are the transfer admittances and the desired outputs are the corresponding element values. The normalization mentioned in Section II was employed with A = 0.3 and B = 0.7.

The neural network used in the training phase is as shown in Fig. 2 with six units in the input layer, sixty units in the single hidden layer and six units in the output layer. Convergence of the connection weights was achieved after 500 sweeps of the training data. Initial weights were randomly selected in the interval [–0.1, 0.1]. In the testing phase, the trained neural network is tested for cases that had not been used in the training phase. The results are shown in Table 1. The percentage error in element estimation is calculated as the absolute value of the difference between the actual element value and the value estimated by a neural net divided by the actual element value. The entries in the column labeled the norm of error vector are the norms of the vector of percentage errors for the associated testing pattern. Cases 1–4 correspond to data presented in the training process. Cases 5–12 represent new data that the network had not seen before. Elements outside their tolerances in cases 5–12 are underlined. We can see from the table that the trained neural network was able to predict that the circuit is not faulty by responding with the element values within their tolerances as in case 6 where the table shows very small percentage error. In other cases, it was also able to predict the actual faulty elements even with the tolerances added to the nominal values of the fault–free elements. For the case (11) of large change in a faulty element, the trained network does not give the accurate value of the faulty element as indicated by the value of the percentage error but still locates the fault and recognizes the direction of the change with an approximation of the true value.

The same example is repeated with a flat (no hidden layer) functional link net. The training matrix consists of 61 rows representing training patterns and 27 columns represents measurements and their second order combinations. We have used the selection algorithm based on QR factorization as explained in Section III to select the most independent patterns and measurements. The selected matrix that was used

in training was reduced to 22 rows (patterns) and 22 columns (measurements and their second order combinations). The topology of the functional link net implemented in the training is shown in Fig. 4. Results presented in Table 1 proved that although a much smaller number of patterns were used in this case, the norm of the error vector is comparable to those calculated for the case of a trained back propagation network.

V. Conclusions

We have presented a neural network based method for analog testing. The input feature selection and the methodology used to construct a training set that can be effectively used in the learning phase of a neural network are explained. In particular, a back propagation model and a functional link net are trained to perform the diagnosis task. The motivation behind using these two models in analog testing is to utilize their ability to realize nonlinear input–output mappings from learning examples. This ability is needed in analog testing due to the nonlinear relationship between measurements and elements even for a linear circuit. Moreover, the generalization capabilities of a properly trained neural network is of important significance, since it can be used in the recall (testing) phase to diagnose much larger number of cases than those presented in a training set. Therefore, the presented approach can be viewed as a generalization of the popular fault dictionary approach [7], with fast access to the stored data and ability to approximate cases not stored in the dictionary.

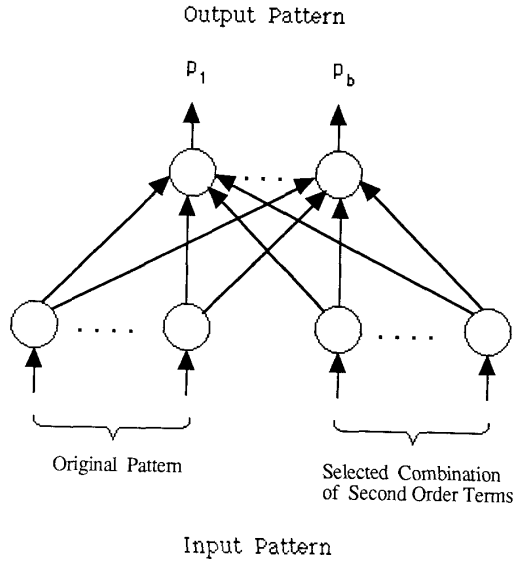

Fig. 4 Functional link net used in the testing example.

REFERENCES

[1] D.E. Rumelhart, G.E. Hinton and R.J. Williams, "Learning internal representations by error propagation in parallel distributed processing", Explorations in the Microstructures of Cognition, Vol. I: Foundations, MIT Press, 1986,pp. 318–362.

[2] R.P. Gorman and T.J. Sejnowski, "Analysis of hidden units in a layered network trained to classify sonar targets", Neural Networks, vol. 1, pp. 75–89, 1988.

[3] D.J. Sobajic and Y.H. Pao, "Artificial neural net based dynamic security assessment for electric power systems", IEEE Trans. on Power Systems, vol. 4, 1989, pp. 220–228.

[4] Y.H. Pao, Adaptive Pattern Recognition and Neural Networks, Addison–Wesley,1989.

[5] N. Navid and A.N. Willson, Jr., "A theory and algorithm for analog circuit fault diagnosis", IEEE Trans. Circuits Syst., vol. CAS 26, 1979, pp. 440–457.

[6] S. J. Leon, Linear Algebra With Applications, Macmillan Publishing Co., New York, 1980.

[7] W. Hochwald and J.D. Bastian, " A dc approach for analog fault dictionary determination", IEEE Trans. Circuits Syst., vol. CAS 26, 1979, pp. 523–529.

TABLE 1

Comparison of Actual and Estimated Element Values of the Circuit of Fig. 3.

	Actual element values						Percentage error in element estimation by back propagation						Norm of error vector	Percentage error in element estimation by a functional link						Norm of error vector
Case	G_1	G_2	G_3	G_4	G_5	G_6	G_1	G_2	G_3	G_4	G_5	G_6		G_1	G_2	G_3	G_4	G_5	G_6	
1	1.0	1.0	1.1	2.0	2.0	1.8	0.09	0.21	0.53	0.00	0.41	0.17	0.73	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	1.0	1.0	0.9	2.2	2.0	2.0	0.39	0.95	0.35	0.03	0.49	0.20	1.21	0.28	0.45	0.08	0.99	0.69	0.18	1.33
3	1.0	0.9	1.0	2.0	2.2	2.0	0.16	0.07	0.24	0.50	0.01	0.07	0.59	0.02	0.05	0.15	0.37	0.93	0.04	1.01
4	1.1	1.0	1.0	1.8	2.0	2.0	0.67	1.33	0.33	1.82	0.22	0.20	2.39	0.31	0.21	0.30	1.01	0.59	0.07	1.27
5	1.2	1.0	1.0	2.0	1.7	2.0	2.07	1.95	2.32	1.05	3.82	0.78	5.47	0.20	0.68	0.02	0.93	1.22	0.15	1.70
6	0.95	1.03	1.05	2.1	2.05	1.95	0.97	0.91	0.60	0.41	0.34	0.06	1.55	0.11	0.07	0.22	0.20	0.37	0.00	0.49
7	1.05	0.95	1.03	2.3	1.95	1.7	3.06	2.03	1.57	3.16	1.49	0.29	5.32	0.16	0.03	0.07	0.28	1.04	0.05	1.09
8	1.2	0.8	1.03	1.95	2.05	2.0	2.21	2.67	1.55	0.82	0.09	0.61	3.93	1.99	0.57	0.68	0.92	1.19	0.17	2.66
9	1.02	0.98	1.0	2.06	2.3	1.6	0.31	0.18	1.54	0.68	2.54	0.1	3.08	0.71	0.87	0.52	0.55	4.31	0.23	4.36
10	1.3	1.0	1.0	2.0	2.0	2.4	0.41	2.80	0.57	2.86	0.96	3.81	5.65	0.54	0.43	0.75	0.78	0.13	0.09	1.29
11	1.0	1.0	1.0	2.0	4.0	2.0	1.72	2.00	4.27	2.14	25.7	2.04	26.3	0.60	5.79	0.43	3.79	18.7	1.46	19.9
12	1.2	1.03	0.95	2.1	1.6	2.0	3.61	0.28	0.56	1.41	1.30	0.56	4.18	0.00	0.55	0.13	0.06	3.18	0.05	3.22