NEURAL NETWORKS IN ANTENNAS AND MICROWAVES: A PRACTICAL APPROACH

Zbyněk RAIDA
Dept. of Radio Electronics
Brno University of Technology
Purkyňova 118, 612 00 Brno
Czech Republic

Abstract

Neural networks are electronic systems which can be trained to remember behavior of a modeled structure in given operational points, and which can be used to approximate behavior of the structure out of the training points. These approximation abilities of neural nets are demonstrated on modeling a frequency-selective surface, a microstrip transmission line and a microstrip dipole. Attention is turned to the accuracy and to the efficiency of neural models. The association of neural models and genetic algorithms, which can provide a global design tool, is discussed.

Keywords

Neural networks, genetic algorithms, planar transmission lines, frequency selective surfaces, microstrip antennas, modeling, optimization

1. Introduction

An artificial neural network (ANN) is an electronic system of a hardware or software nature, which is built in accordance with the human brain. Therefore, an ANN consists of many simple non-linear functional blocks of a few types, which are called **neurons**. Neurons are organized into layers, which are mutually connected by highly parallel **synaptic weights**. The ANN exhibits a **learning ability**: synaptic weights can be strengthened or weakened during the learning process, and by that way, information can be stored in the neural network [1], [2].

Due to the non-linearity of neurons, the ANN is able to solve even such types of problems that are unsolvable by linear systems. Due to the massive parallelism, the ANN exhibits a very high operational speed (when multi-processor systems or hardware implementation are elected). Due to the learning ability, the ANN can behave as adaptive systems, which automatically react on changes in its surrounding. Also, due to the presence of a few types of functional blocks in the structure only, the ANN is suitable for hardware implementation (VLSI circuits) or software one (object-oriented approach) [1], [2].

ANNs have been intensively exploited since the eighties in electrical engineering, when sufficient computational power of processors and sufficient capacity of computer memories were at their disposal. ANNs have been applied in pattern recognition systems, and have been exploited for input-output mapping, for system identification, for adaptive prediction, etc.

Dealing with the antenna applications, ANNs have been used as adaptive controllers in adaptive antenna arrays [3], have been applied in direction-finding arrays [4] and have been exploited for modeling and optimization.

Concentrating on neural modeling of antennas and microwave structures, ANNs have been applied to the calculation of resonant frequencies of microstrip antennas [5], to the computation of complex resonant frequencies of microstrip resonators [6], to the modeling of microwave circuits [7], [8], to the reverse modeling of microwave devices [9], to the calculation of effective dielectric constants of microstrip lines [10], etc. Moreover, neural networks have been applied to the optimization of microwave structures and antennas [11], [12].

Exploitation of neural network techniques in electromagnetics is even described in a few monographs. In [13], ANNs are shown being applied in RF and mobile communication techniques, in radar and remote sensing, in scattering, antennas, and computational electromagnetics. In [14], ANNs are applied to modeling interconnects and active devices, for circuit analysis and optimization, etc.

Moreover, matlab users can obtain a neural network toolbox, which is ready for the immediate exploitation of ANNs for modeling, optimization, etc. [15].

In the paper, the neural modeling of a selected frequency-selective surface, of a selected transmission line and of a selected microwave antenna is discussed in Section 2. These structures are modeled using numerical methods first. In the second step, obtained numerical results are exploited as **teachers**, which can train neural nets. Finally, neural models are in detail compared with numerical ones.

In Section 2, an original description of the influence of a number of training patterns and their position in the modeling space to the model accuracy is presented.

Section 3 deals with the exploitation of ANNs for the optimization of the above three structures. The presented approach combines neural models and genetic algorithms in order to reveal regions *suspected* of containing a global minimum. The revealed regions might be further examined

using Newton's method in order to find the global minimum as accurately as possible.

The conclusion is a detailed discussion of the results obtained during the neural modeling and optimization of a selected frequency-selective surface (FSS), of a selected transmission line (TL), and of a selected microstrip antenna (MA), when artificial neural networks are used. Generalized conclusions should answer the question when ANNs can help us and how; what is the most efficient way of building a neural model; when replacing a numerical model by a neural one gives sense and when not; etc.

2. Neural modeling of EM structures

When a neural model of an EM is going to be developed, a proper architecture of an ANN, a proper type of neurons, and a proper training algorithm shall be chosen.

Dealing with the architecture of ANNs, we are going to concentrate on the feed-forward structures because feed-forward ANN statically map input patterns to output ones.

Dealing with learning, back-propagation ANNs, driven by quasi-Newton algorithm (Levenberg-Marquardt) or by the Bayesian regularization that are implemented in the neural network toolbox of matlab, are the most suitable.

Dealing with the types of neurons, back-propagation ANNs require adaptive-non-linear neurons, which modify setting of weights and biases in order to minimize the learning error of an ANN [1].

Attention is now turned to building neural models of selected EM structures, which are as accurate as possible and whose preparation takes as short a time as possible.

2.1 Frequency-selective surface

The modeled frequency-selective surface (FSS) is depicted in Fig. 1. The FSS consists of equidistantly distributed identical rectangular elements, which are assumed to be perfectly electrically conductive (PEC). The conductive rectangles are positioned in the center of a discrete cell of the infinite plane of the same electrical parameters as the surrounding. The height of the conductive element is fixed at a = 11 mm, and even the height of the cell is assumed to be constant A = 12 mm. The width of the conductive element is changed within the interval $b \in$ <1 mm, 7 mm>, cell width can intervene between $B \in$ <10 mm, 22 mm>.

The described FSS is numerically modeled by the spectral-domain method of moments [24] utilizing harmonic basis and weighting functions. As a result, frequency f_2 of the first maximum of the reflection coefficient module of the Floquet mode (0,0), and frequencies f_1 and f_3 for 3-dB decrease of reflection coefficient module ($f_1 < f_2 < f_3$) are obtained. The analysis is performed for the perpendicular incidence of linearly polarized EM wave, whose electric intensity is oriented in the direction of axis x (see Fig. 1).

The neural model of the FSS consists of 2 input neurons (doublets [b, B] form the input patterns), and the output layer of 3 neurons (respective triplets [f_1, f_2, f_3] are the desired responses). Since output quantities (f_1, f_2, f_3) are positive numbers, output neurons should contain unipolar sigmoid as the non-linear activation function (i.e., opposite type of non-linearity is used at the output neurons instead of at the hidden ones).

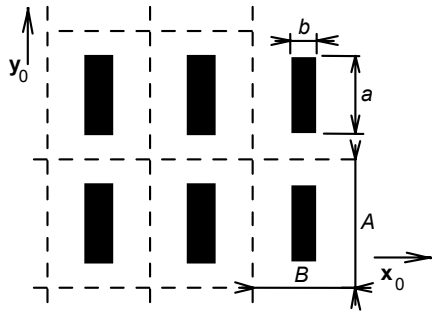

Fig. 1 Frequency-Selective Surface (FSS) consisting of perfectly electrically conductive (PEC) rectangles. Rectangles are assumed to be equidistantly placed on an infinite plane of the same electrical parameters as the surrounding.

Before the training of an ANN is started, the number of training patterns, and their position in the training space, and the number of hidden neurons is determined.

In discussing the training patterns, two contradictory requests are stated: the building process should consume as short a time as possible (i.e., number of training patterns should be minimized), and the developed neural model is to be as accurate as possible (i.e., the number of training patterns should be high). Therefore, some compromise has to be found in order to get a relatively accurate model which can be quickly developed.

Therefore, the input space of the ANN is sampled with a constant sampling step first. The sampling step is relatively long in order to obtain an initial notion about the behavior of the structure with the minimal effort. Second, the sampling is refined in order to reach a desired accuracy.

In discussing the number of hidden neurons, initial architecture has to be estimated. Then, Bayesian regularization is run, and the number of hidden neurons is changed until the number of efficiently used parameters does not intervene between 60 % and 90 %.

Initially, both $b \in$ <1 mm, 7 mm> and $B \in$ <10 mm, 22 mm> are changed with the step $\Delta b = \Delta B$ = 3.0 mm, and the output responses [f_1, f_2, f_3] are computed for all the combinations of [b, B]. I.e., 3 × 5 = 15 analyses have to be performed. The complete training set is stored in the Excel file **fss.xls**[1].

[1] All the Excel files and all the m-files, which are described in the paper, can be downloaded via the web site **http://www.fee.vutbr.cz/UREL/present/ann/ann.html**.

The m-files were developed using the neural network toolbox of matlab 5.3.

Using Bayesian regularization (`fss_br_30mm.m`), a proper structure of an ANN is estimated: if ANN contains 2 hidden layers consisting of 5 neurons each, then 60 % parameters is efficiently used (after 500 iteration steps). When the proposed ANN is trained using the Levenberg-Marquardt algorithm (`fss_lm_30mm.m`), the training error reaches the level 10^{-7} within 98 iteration steps (the best result from 5 performed training processes).

The accuracy of the neural model (`fss_lm_3.mat`) over the training area is tested comparing results of the numerical analysis and respective simulation results of the ANN. For every input pattern, relative errors are computed and averaged. The result is called the cumulative error

$$c(b,B)=\frac{100}{3}\sum_{n=1}^{3}\frac{\left|\tilde{f}_n(b,B)-f_n(b,B)\right|}{f_n(b,B)} \quad [\%] \qquad (1)$$

Here, b is the width of the metallic element and B denotes the width of the cell, f_n is the frequency obtained by the numerical analysis and $\tilde{f}_n$ is the frequency produced by the neural model (n = 1, 3 are associated with the 3-dB decrease of module of reflection coefficient, n = 2 corresponds with its maximum).

The cumulative error of the model `fss_lm_3.mat` is depicted in Fig. 2A. It is obvious that points corresponding with training patterns exhibit negligible error (e.g. points [b, B] = [1,10], [4,10], [7,10]). Whereas the cumulative error for B > 16 mm might be considered as sufficiently small, for B < 16 mm the error is very high.

In order to increase accuracy of the model, the part of the input space corresponding with an unacceptably high error (B < 16 mm for all b) is re-sampled with smaller discretization step ($\Delta b = \Delta B$ = 1.5 mm). Therefore, 35 training patterns (5 × 7) have to be prepared in this case.

The new ANN consists of 3 hidden layers containing 6, 3 and 6 neurons. Performing the Levenberg-Marquardt training (`fss_lm_15mm.m`), the training error reaches the level 10^{-6} within 606 iteration cycles[2] (the best result from 5 performed processes). Observing the accuracy of the new model (`fss_lm_15c.mat`) in Fig. 2B, very low error is reached except for the area near the point [b, B] = [7,10]. Even finer re-sampling of the surroundings of this point can again reduce the error in the respective area.

In practical neural modeling, distribution of approximation error is unknown because the modeled structure is analyzed for the training patterns only. Therefore, a different criterion for pattern refinement has to be found.

Observing training patterns in **fss.xls**, the approximated function $f_n = f_n(b, B)$, n = 1, 2, 3, is very steep in the area of the highest error (i.e. Δf_n is high for 2 neighboring learning patterns). If sampling in this area is refined, then Δf_n is reduced for neighboring patterns. Therefore, we can practically conclude that Δf_n should be similar for all the neighboring patterns in the training set.

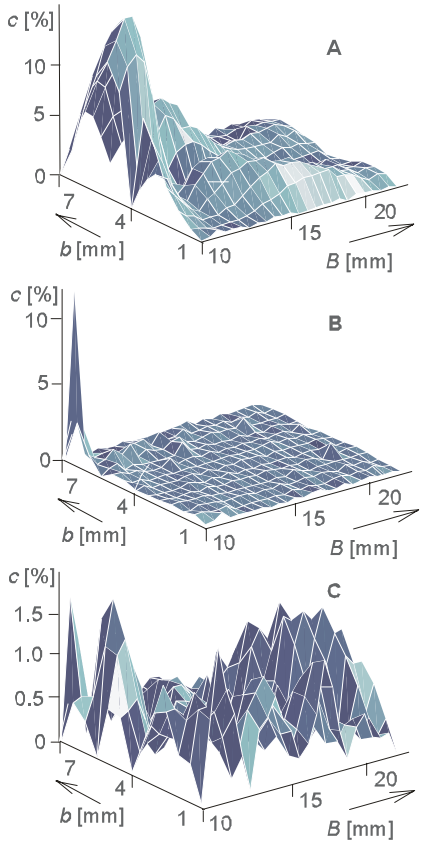

Fig. 2 Cumulative error of neural model of FSS: A) constant sampling $\Delta b = \Delta B$ = 3.0 mm, B) finer one $\Delta b = \Delta B$ = 1.5 mm in $B \in$ <10.0 mm, 16.0 mm> and $b \in$ <1.0 mm, 7.0 mm>, C) initial sampling refined to reduce relative variation of f_2.

Let us verify the above conclusion. In **fss.xls**, we compute the relative variation of the n-th approximated (output) quantity with respect to the i-th input one

$$\delta f_n^{(b)} = 100\,\frac{f_n(b_i,B)-f_n(b_{i+1},B)}{\frac{1}{2}\left[f_n(b_i,B)+f_n(b_{i+1},B)\right]} \quad [\%] \qquad (2\text{ a, b})$$

$$\delta f_n^{(B)} = 100\,\frac{f_n(b,B_i)-f_n(b,B_{i+1})}{\frac{1}{2}\left[f_n(b,B_i)+f_n(b,B_{i+1})\right]} \quad [\%] \qquad n=1,2,3$$

where i is an index of the respective input parameter in the training set. If the relative variation exceeds a prescribed level, then a new training pattern is inserted between two already existing ones.

In our case, we require the relative variation to be lower than 10 % for the central frequency f_2. This condition is not met for pairs [b, B_i - B_{i+1}] and [b_i - b_{i+1}, B] indicated in Tab. 1. Therefore, we have to insert 8 new patterns into training set: p_1 = [4 mm, 11.5 mm], p_2 = [4 mm, 14.5 mm],

[2] The training error is rapidly decreasing down to level 10^{-6}. Then, the minimization process exhibits very poor convergence, and the level 10^{-7} is not reached even within 5000 steps. From the economical point of view, reducing demands on the training error is the best solution.

p_3 = [4 mm, 17.5 mm], p_4 = [7 mm, 11.5 mm], p_5 = [7 mm, 14.5 mm], p_6 = [7 mm, 17.5 mm], p_7 = [5.5 mm, 10 mm], and p_8 = [5.5 mm, 13 mm].

A	10 - 13	13 - 16	16 - 19	19 - 22
1.0	9.9	9.4	8.5	8.4
4.0	14.7	10.6	10.1	9.3
7.0	29.5	18.9	13.5	9.9

B	10	13	16	19	22
1 - 4	8.4	3.7	2.4	0.9	0.0
4 - 7	30.1	15.3	6.9	3.5	2.9

Tab. 1 Percentage variation of the frequency f_2: A) for neighboring widths of cells (first row), B) for neighboring widths of elements (first column). Unacceptable variations highlighted.

A new training set containing 15 + 8 = 23 patterns is used to learn the ANN (three hidden layers consisting of 5-3-5 neurons, training error lower than 10^{-6} within 530 iteration steps, the best result from 5 performed training processes considered). The cumulative error of the neural model (`fss_lm_xe.mat`) is depicted in Fig. 2C. The error is lower than 1.5 % all over the output space, and even the number of training patterns is lower (23 versus 35). Moreover, no information about the error distribution over the input space is desired. Electing for a lower admissible error than 10 %, the number of training patterns have to be increased on one hand, and the approximation error can be reduced on the other hand.

In the following paragraph, the described procedure of building neural models is applied to a transmission line.

2.2 Transmission line

The modeled microstrip transmission line (TL) is depicted in Fig. 3. The TL is assumed to be longitudinally homogeneous. TL is shielded by a rectangular waveguide of PEC walls at fixed dimensions $A = B = 12.7$ mm. At the bottom of the shielding waveguide, a lossless dielectric substrate of the dielectric constant $\varepsilon_{r1} \in$ <1.0, 5.0> and of the height $h = 1.27$ mm is placed. At the center of the substrate, a PEC microstrip of a negligible thickness $t \approx 0$ and of the fixed width $w = 1.27$ mm is placed. The microstrip can be covered by another dielectric layer of a dielectric constant $\varepsilon_{r2} \in$ <1.0, 5.0> and of height $h = 1.27$ mm. Above the second layer, a vacuum is assumed.

The described transmission line is numerically modeled by a finite-element method exploiting hybrid nodal-edge finite elements [30]. As a result, propagation constants of the dominant mode on frequency $f_1 = 20$ GHz (β_1) and on $f_2 = 30$ GHz (β_2) are obtained.

The neural model of the TL consists of 2 inputs, because doublets [ε_1, ε_2] form the input patterns. The output layer again contains 2 neurons because respective doublets of propagation constants [β_1, β_2] form the desired output responses. Since the propagation constants are positive numbers, a unipolar sigmoid is used as the activation function in the output layer.

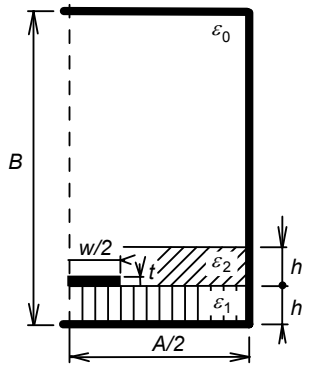

Fig. 3 Microstrip transmission line on the substrate (ε_{r1}, h), which might be covered by another dielectric layer (ε_{r1}, h). Longitudinal homogeneity is assumed. Losses in dielectrics and metal are neglected.

Constructing the neural model of the TL, both the dielectric constant of a substrate $\varepsilon_{r1} \in$ <1.0, 5.0 > and the dielectric constant of the second layer $\varepsilon_{r2} \in$ <1.0, 5.0> are changed with a discretization step $\Delta = 2$ in the initial stage. The desired output responses [β_1, β_2] are computed for all the combinations of [ε_{r1},ε_{r2}]. Hence, $3 \times 3 = 9$ numerical analyses have to be done. The resultant training set can be found in the Excel file **tl.xls**.

As described in paragraph 2.1, the initial training set is tested from point of view of relative variations among output patterns. In Tab. 2, relative variation is computed for propagation constants at $f = 20$ GHz. For 30 GHz, results are similar. If variation is required to be lower than 10 %, then training set shall be completed by additional 8 patterns p_1 = [2.0, 1.0], p_2 = [4.0, 1.0], p_3 = [1.0, 2.0], and p_4 = [3.0, 2.0], p_5 = [5.0, 2.0], p_6 = [1.0, 4.0], p_7 = [3.0, 4.0], and p_8 = [5.0, 4.0]. In the brackets, 1st value is associated with ε_{r2}, and 2nd one with ε_{r1}.

When above patterns are included into the training set, new testing of relative variations is performed. As a result, other two patterns p_9 = [2.0, 2.0], and p_{10} = [4.0, 2.0] are included into the training set.

A	1 - 3	3 - 5
1.0	23.6	19.8
3.0	8.6	8.1
5.0	5.0	4.4

B	1.0	3.0	5.0
1 - 3	44.3	29.6	18.1
3 - 5	24.6	21.1	17.4

Tab. 2 Percentage variation of propagation constant on 20 GHz: A) neighboring dielectric constants of cover (first row), B) neighboring dielectric constants of substrate (first column). Unacceptable variations are highlighted.

In total, the training set contains 9 + 8 + 2 = 19 patterns. Exploiting our experience with building the neural model of a FSS, we initially used an ANN consisting of 5 neurons in each of the 2 hidden layers. The Bayesian regularization tells us that 60 % of parameters is efficiently used (296 iteration cycles, desired error 10^{-7}). Since the result seems to be all right, we run the same learning using the Levenberg-Marquardt algorithm. Within 141 cycles, the network is trained (`tl_lm_55a.mat`). Verifying accuracy of the neural model (Fig. 4A), the cumulative error up to 4 % can be observed.

Let us try to interpret the relatively high cumulative error corresponding to non-training patterns as an over-training of the ANN. If the ANN is over-trained then the approximation at the output of the ANN oscillates among training patterns. Therefore, the training error is very small but the approximation error is relatively high.

In order to solve the above problem, the number of hidden neurons is reduced to 4 in each of the hidden layers (70 % of efficiently used parameters). If the maximal training error is set to 10^{-7}, the learning process is finished within 1531 cycles and the value of the maximal cumulative error is about 1 % (`tl_lm_44a.mat`). If the desired training error is reduced to 10^{-6}, then the training is over within 872 cycles and the cumulative error is lower than 0.6 % (`tl_lm_44b.mat`) as depicted in Fig. 4B.

If the number of hidden neurons is further reduced to 3 in each hidden layer (80 % of efficiently used parameters) then the ANN is trained within 150 cycles, both for the desired error 10^{-6}, 10^{-5} (`tl_lm_33a.mat`, `tl_lm_33b.mat`). In both cases, cumulative error again reaches the value 1 %. The result is caused by the fact that ANN contains an insufficient number of free parameters to be trained well.

Keeping the above results in mind, we can postulate the validity of following conclusions:

- The number of efficiently used parameters should be within the interval <65 %, 75 %>.
- Training should be finished within a *reasonable* number of iteration steps (below 1000 in our case).
- The value of the desired training error has to be selected such way so that both reasonable number of learning cycles and the quality training are reached (10^{-6} or 10^{-7} in our case).

The above practical conclusions can be verified on the final neural model of FSS (Fig. 2C, 5-3-5 hidden neurons, the training error lower than 10^{-6}, 530 iteration steps). Although the number of efficiently used parameters is about 60 %, over-training is eliminated here using bottleneck (a narrow central layer consisting of 3 neurons).

The processes of building neural models of the FSS and the TL are similar: approximated unipolar output quantities monotonously change when changing continuous input parameters. In the next paragraph, a different situation appears: output quantity (input impedance of a microstrip dipole, Fig. 5) is bipolar (reactance can be both positive and negative), and it is not of a monotonous nature (impedance characteristics of a microstrip dipole exhibits a resonance). Moreover, two input parameters can be changed continuously (length of the dipole, width of the dipole) and two can acquire discrete values only (height of a substrate, dielectric constant of a substrate). Therefore, the developed procedure of building neural models has to be modified.

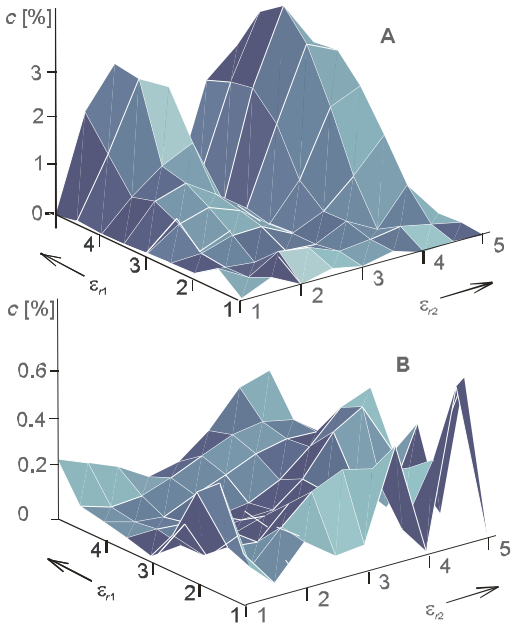

Fig. 4 Cumulative error of neural model of TL: A) 5 neurons in each of 2 hidden layers, desired training error 10^{-7}, B) 4 neurons in each of 2 hidden layers, desired training error 10^{-6}.

2.3 Microstrip antenna

The modeled microstrip antenna (MA) is depicted in Fig. 5. An MA consists of a microstrip dipole of the length $A \in$ <1.0 mm, 4.0 mm> and the width $B \in$ <0.05 mm, 0.10 millimeters> that is supplied by a symmetric transmission line. The metallic ground plane plays the role of the planar reflector. The MA can be fabricated from dielectric substrates of a dielectric constant ε_r = [1.0, 1.6, 2.0] and of a height h = [1.0 mm, 1.5 mm]. Losses, both in the dielectrics and in the metal, are neglected. The antenna is assumed to operate on the frequency f = 30 GHz.

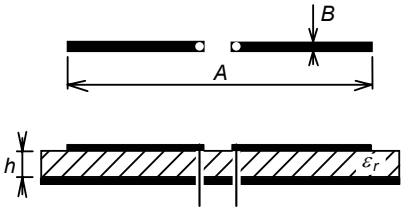

Fig. 5 Microstrip dipole on the dielectric substrate of the dielectric constant ε_r and of the height h. Both the dipole and the reflector (the ground plane) are perfectly conductive. No losses in dielectrics are assumed.

The described antenna is numerically modeled by the method of moments [27] - [29] using a piece-wise constant basis functions and Dirac weighting. The analysis results are created by the value of the input impedance of the antenna $Z_{in} = R_{in} + j\,X_{in}$ on the frequency $f = 30$ GHz.

The neural model of an MA consists of 4 inputs, because quadruplets [A, B, ε_r, h] form the input patterns. Output layer contains 2 neurons because respective doublets [R_{in}, X_{in}] form output responses. Since input reactance of antenna X_{in} can be both positive and negative, output neurons should contain bipolar sigmoid as the non-linearity.

Dealing with proper discretization of the input space, only ΔA and ΔB have to be determined because the discretization of h and ε_r is prescribed. Although the dimension of the input space is 4, we operate on the two-dimensional input spaces [A, B] organized into relatively independent planes, which are associated with doublets [h, ε_r]. The described training set is at one's disposal in the file **ma.xls**.

A proper choice of the discretization steps ΔA and ΔB should differ from the above described procedure.

Whereas the output quantities of neural models of the FSS and the TL are positive and change monotonously, the input impedance of an MA exhibits a non-monotonous behavior due to the resonance of the antenna and the input reactance is of a bipolar nature.

Whereas the dynamics of the output quantities (the ratio of the lowest output value and the highest one) of the FSS and the TL is relatively low

$$\rho_{\mathrm{FSS}} = \frac{\min\{f_1(b,B);\, f_2(b,B);\, f_3(b,B)\}}{\max\{f_1(b,B);\, f_2(b,B);\, f_3(b,B)\}} = \frac{f_1(7.0{,}10.0)}{f_3(7.0{,}10.0)} = 3\cdot 10^{-1}$$

$$\rho_{\mathrm{TL}} = \frac{\min\{\beta_1(\varepsilon_{r1},\varepsilon_{r2});\, \beta_2(\varepsilon_{r1},\varepsilon_{r2})\}}{\max\{\beta_1(\varepsilon_{r1},\varepsilon_{r2});\, \beta_2(\varepsilon_{r1},\varepsilon_{r2})\}} = \frac{\beta_1(1.0{,}1.0)}{\beta_2(5.0{,}5.0)} = 3\cdot 10^{-1}$$

the output data of the MA are of a very high dynamics

$$\rho_{\mathrm{MA}} = \frac{\min\{R_{in}(A,B);\, |X_{in}(A,B)|\}}{\max\{R_{in}(A,B);\, |X_{in}(A,B)|\}} = \frac{R_{in}(1.00{,}0.10)}{|X_{in}(1.00{,}0.05)|} = 4\cdot 10^{-4}$$

Considering both non-monotonous nature and high dynamics of approximated quantities, the initial sampling step is set as very short: ΔA = 0.25 mm, ΔB = 12.5 μm. Then, training set has $N_A \times N_B \times N_h \times N_\varepsilon = 13 \times 5 \times 2 \times 3 = 390$ training patterns as shown in **ma.xls**.

For the described discretization, the relative variation among training patterns is computed. Due to bipolar nature of the input reactance of the MA, the denominator of (2) might approach zero, and the relative variation is very high, although the output quantity does not change dramatically between the respective sampling points. In order to eliminate this phenomenon, we modify relations (2) for the MA the following way

$$\delta f_n^{(A)} = 100\,\frac{|f_n(A_i,B) - f_n(A_{i+1},B)|}{\max\{|f_n(A_i,B)|,\, |f_n(A_{i+1},B)|\}} \qquad (3a)$$

$$\delta f_n^{(B)} = 100\,\frac{|f_n(A,B_i) - f_n(A,B_{i+1})|}{\max\{|f_n(A,B_i)|,\, |f_n(A,B_{i+1})|\}} \quad n = 1,2 \qquad (3b)$$

where $f_1 \equiv R_{in}$ is input resistance and $f_2 \equiv X_{in}$ is input reactance of the MA. In addition, A denotes the length of the MA, B is the width of the MA, and i is an index of a respective input parameter in the training set.

In evaluating training set, variations of the input resistance with respect to dipole length A are $\delta R_{in}^{(A)} \in$ <17 %; 49 %>, and relative variations of the input reactance intervene within $\delta X_{in}^{(A)} \in$ <12 %; 197 %>. The highest values of $\delta R_{in}^{(A)}$ are dominantly at $A \in$ <1.00 mm, 1.25 mm> for all B. Other hand, the highest values of $\delta X_{in}^{(A)}$ is found at $A \in$ <2.50 mm, 2.75 mm> and at $A \in$ <3.00 mm, 3.25 mm> for all B.

Dealing with relative variations with respect to B, variations of input resistance, $\delta R_{in}^{(B)} \in$ <0.01 %; 9.5 %>, are in all the cases lower than 10 %, and variations of input reactance, $\delta X_{in}^{(B)} \in$ <0.02 %; 79.6 %>, exceeds 10 % dominantly for A = 2.50 mm and A = 3.00 mm for all B.

Considering the location of the highest relative variations, a high approximation error of the input resistance can be expected for A < 1.50 mm, and a high error of input reactance can be supposed at $A \in$ <2.50 mm, 3.50 mm>.

In order to verify the validity of our expectations, a neural model of the MA consisting of 17-7-17 hidden neurons is developed (`ma_lm_17_7_17a.mat`). An ANN is trained within 133 iteration cycles with the training error lower than 10^{-7}. The neural model exhibits the cumulative error of the input resistance

$$c_R(A,B) = \frac{100}{6}\sum_{m=1}^{2}\sum_{n=1}^{3}\frac{|\tilde{R}_{in}(A,B,h_m,\varepsilon_{r,n}) - R_{in}(A,B,h_m,\varepsilon_{r,n})|}{R_{in}(A,B,h_m,\varepsilon_{r,n})} \qquad (4a)$$

as depicted in Fig. 6A, and the cumulative error of input reactance

$$c_X(A,B) = \frac{100}{6}\sum_{m=1}^{2}\sum_{n=1}^{3}\frac{|\tilde{X}_{in}(A,B,h_m,\varepsilon_{r,n}) - X_{in}(A,B,h_m,\varepsilon_{r,n})|}{X_{in}(A,B,h_m,\varepsilon_{r,n})} \qquad (4b)$$

as depicted in Fig. 6B. In (4), A is the length of the microstrip dipole, B is its width, h = [1.0 mm, 1.5 mm] denotes the height of the substrate, and ε_r = [1.0, 1.6, 2.0] is the dielectric constant of the substrate. In addition, $\tilde{R}_{in}$ is the input resistance of the MA provided by a neural model, and R_{in} is the same quantity provided by a numerical analysis. Similarly, $\tilde{X}_{in}$ and X_{in} denote the input reactance.

Fig. 6A confirms our hypothesis that the highest error of modeling input resistance is located at A < 1.50 mm. Dealing with input reactance, the relative error over 150 % at the position [3.125 mm, 0.075 mm][3] makes the rest of the

[3] The position of the highest approximation error of the input reactance X_{in} is identical with the position of the highest relative variance of X_{in} with respect to the dipole length A.

figure unreadable. Suppressing this error, Fig. 6C is obtained. In this figure, the relative error of input reactance reaches up to 20 % for $A \in$ <2.50 mm, 3.50 mm>. Therefore, our hypothesis is confirmed.

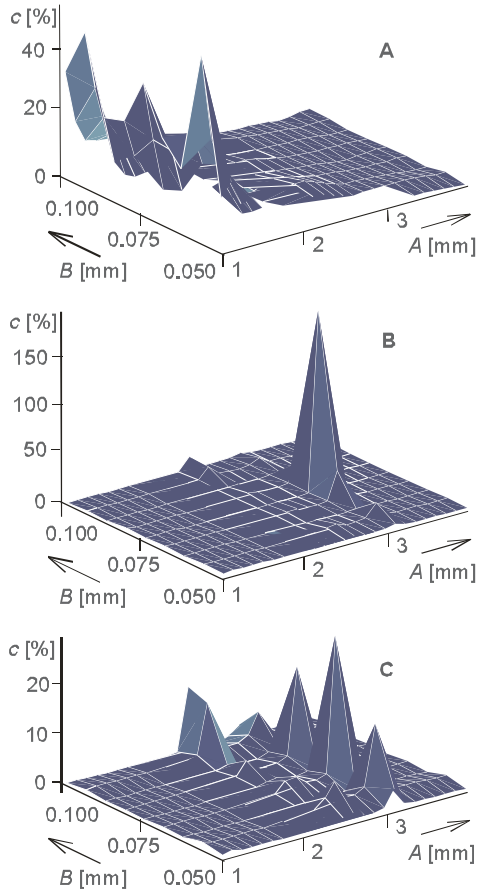

Fig. 6 Cumulative error of the neural model of MA: A) input resistance, B) input reactance, C) input reactance when error maximum eliminated. ANN consists of 17-7-17 neurons, the desired training error 10^{-7}.

If very high approximation error of the neural model of the MA is required to be reduced, discretization steps ΔA, ΔB have to be shortened in regions where high relative variations were revealed. Unfortunately, refinement of the training set significantly increases the number of training patterns (i.e., the number of numerical analyses which have to be performed), and consequently, the ANN has to contain more neurons (i.e., the training process consumes more CPU time).

Therefore, instead of refining the training set, we try to approach the problem of modeling the MA to the situation when modeling the FSS and the TL, which provided satisfactory results with minimal effort.

In the first step, we decrease the dynamics of output patterns by applying the natural logarithm to all of the output set. Since the input reactance of the MA might be negative, we add a constant to every reactance in order to get positive numbers higher than one. If even every input resistance of the MA is increased in order to be higher than one, then all the logarithms are positive. In our case

$$r_{in}(A,B) = \ln\left[R_{in}(A,B)+1\right] \quad (5a)$$

$$x_{in}(A,B) = \ln\left[X_{in}(A,B)+1065\right] \quad (5b)$$

where R_{in} and X_{in} are the input resistance and the input reactance of the MA from the original training set, respectively. Symbols r_{in} and x_{in} denote input resistance and input reactance of MA from the transformed set. Constant 1065 is derived from $\min\{X_{in}\}$ = -1063 Ω.

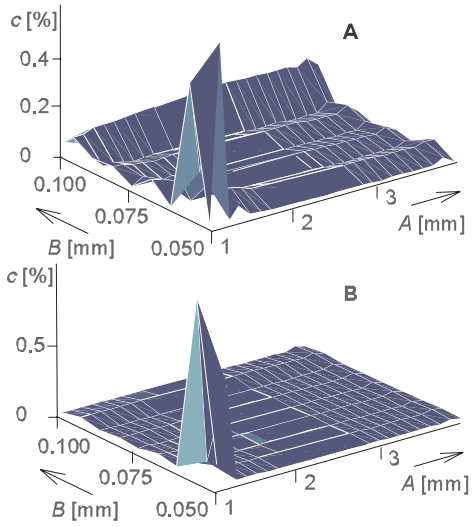

Fig. 7 Cumulative error of the neural model of MA exploiting logarithmic transform: A) input resistance, B) input reactance. The ANN consists of 17-8-17 neurons, the desired training error 10^{-6}, the Bayesian regularization applied.

The transformed data set contains positive numbers only, and therefore, the unipolar sigmoid can be used in the output layer of an ANN. Moreover, the dynamics of the original data set is reduced from the value $\rho_{\mathrm{MA}} = 4 \cdot 10^{-4}$ to

$$\rho_{\mathrm{MA}}^{\mathrm{LN}} = \frac{\min\{r_{in}(A,B), x_{in}(A,B)\}}{\max\{r_{in}(A,B), x_{in}(A,B)\}} = \frac{r_{in}(1.00,0.10)}{x_{in}(4.00,0.10)} = \frac{0.366}{7.38} = 5\cdot 10^{-2}$$

Further, we have to investigate the relative variations in the transformed training set (see **ma.xls**). Variations of the input resistance with respect to the antenna length $\delta r_{in}^{(A)}$ exceeds 30 % for small value of A, and it is about 10 % for the high value of A. Variations of input resistance with respect to the antenna width $\delta r_{in}^{(B)}$ are lower than 2 % in all cases. Variations of input reactance $\delta x_{in}^{(A)}$, and $\delta x_{in}^{(B)}$ are lower than 10 %, except for singular cases for $A \in$ <1.00 mm, 1.50 mm>, $B \in$ <0.050 mm, 0.075 mm>. Therefore, the neural model of the MA might be expected to exhibit the highest approximation error for small values of A and B.

In the first step of verifying the above hypothesis, the proper structure of hidden layers is estimated to be 17-8-17 neurons. The Bayesian regularization tells us that 87 % parameters of the ANN is efficiently used (500 steps, a training error lower than 10^{-6}).

The Bayesian regularization produces a neural model of the MA which is stored in `ma_br_17_8_17a.mat`. The approximation error of this model is depicted in Fig. 7. As shown, the highest approximation error (0.8 % for r_{in}, 1.3 % for x_{in}) is really associated with the smallest values of A and B as we had predicted.

In the second step, the same ANN is trained using the Levenberg-Marquardt algorithm. When testing results of the training, approximation oscillations are observed. Therefore, the number of neurons in hidden layers is consecutively reduced to 16-6-16 (further reduction increases the approximation error). Within 260 iteration steps, the neural model of the MA (`ma_lm_16_6_16b.mat`) is trained with the error lower than 10^{-7}. The approximation error is lower than 3 % both for r_{in} and for x_{in}.

In this case, the Bayesian training provides better results than the Levenberg-Marquardt algorithm. Higher CPU-time demand of the Bayesian training is the price we have to pay for a more accurate neural model of the MA.

Finally, we have to investigate the transformation of the approximation error (natural logarithm of input resistance and reactance) to the deviation of obtained input impedance from the numerical model. The highest approximation error is located in the area where A, B are small. In that region, $r_{in} < 1.15$, which converts the approximation error 0.8 % to

$$\delta_r = 100\,\frac{[\exp(1.008\cdot 1.15)-1]-[\exp(0.992\cdot 1.15)-1]}{[\exp(0.992\cdot 1.15)-1]} \approx 3\,\%$$

Similarly, deviation of the input reactance of the MA can reach in the region of the highest error where $x_{in} < 6.77$ and the approximation error is maximally 1.3 %, the value

$$\delta_x = 100\,\frac{[\exp(1.013\cdot 6.74)-1065]-[\exp(0.987\cdot 6.74)-1065]}{[\exp(0.987\cdot 6.74)-1065]} \approx 50\,\%$$

Therefore, the region of the highest error has to be resampled and a new ANN has to be trained.

In the rest of the training space, the approximation error is lower than 0.2 % which causes the highest error of r_{in}, x_{in} lower than 5 %.

In summarizing our experience with building a neural model of the MA, following conclusions can be done:

- If the ANN is asked to approximate non-monotonous bipolar quantities, then the discretization step has to be short. Attention has to be paid to the relative variations of approximated quantities (computed according to the modified relations in the case of bipolar output values) and to the dynamics of the approximated quantities.
- If the approximated quantities exhibit very high dynamics (more than 10^{-2} in our case), then the dynamics have to be properly reduced. As shown in our paper, exploring the natural logarithm for this purpose is not the best solution (the error 5% is much higher than in the case of neural models of the FSS and TL).
- If the approximated quantities exhibit high relative variations, then the discretization has to be refined in the respective area. The refinement is performed in the same way as described in paragraphs 2.1 and 2.2.
- If even the optimal architecture[4] of an ANN does not perform with satisfactory results when trained by the Levenberg-Marquardt algorithm, then the Bayesian regularization can be used to achieve better results.

Now, we are familiar with the techniques used for building neural models of electromagnetic systems. In the next paragraph, we are going to discuss in depth CPU-time demands of building neural models so that we can determine whether neural modeling provides more advantages or disadvantages.

2.4 CPU-time demands of neural modeling

In this section, we utilize our experience with developing neural models of the FSS, TL, and MA in order to evaluate CPU-time demands of this development. CPU-time demands consist of time necessary for building training patterns and of time used for training an ANN.

Time demands of numerical modeling of our structures are concentrated in Tab. 3. The total time used for computing a single training pattern is obtained by multiplying the time of a single analysis by the number of its executions. A single pattern of the FSS requires 19 executions (on average) because the maximum of the reflection coefficient and 3-dB decrease shall be numerically found. A single pattern of the TL needs 4 executions because the structure is analyzed in two frequencies, both for electric intensity and magnetic one (in order to minimize the error of the analysis). A single pattern of the MA is equivalent to the single analysis.

	analysis	repeated	total
fss	**24.7 s**	**19**	**469.0 s**
tl	**5.6 s**	**4**	**22.4 s**
ma	**16.6 s**	**1**	**16.6 s**

Tab. 3 CPU-time demands of the preparation of a single training pattern: **analysis**: time of a single numerical analysis of the structure, **repeated**: the number of single analysis repetitions needed for completing the training pattern, **total**: time necessary for building a single training pattern.

CPU-time demands of training are given in Tab. 4. In this Table, we concentrate on those neural models that were elected as optimal in previous paragraphs.

[4] We consider such an architecture of an ANN as optimal, which provides the lowest approximation error. Both increasing and decreasing the number of hidden neurons causes the increase of the approximation error.

A neural model of the FSS (`fss_lm_xe.mat`) was developed using a training set which consisted of 23 patterns (the initial training set of 15 patterns was completed by the additional 8 patterns in order to reduce very high relative variations). Since the numerical computation of a single pattern took approximately 469 seconds, building of the whole training set was finished within 180 minutes. Due to the small size of the respective ANN (5-3-5 hidden neurons), the training process was over within 1 minute using the Levenberg-Marquardt algorithm.

	patterns [min]		train [min]	total [min]
FSS	**23×469.0**	**180**	**1**	**181**
TL	**19× 22.4**	**7**	**1**	**8**
MA-LM	**390× 16.6**	**108**	**30**	**138**
MA-BR	**390× 16.6**	**108**	**75**	**183**

Tab. 4 Total CPU-time needed for developing a neural model of FSS (5-3-5 neurons, 530 epochs), TL (4-4 neurons, 872 epochs), and MA (16-6-16 neurons and 260 epochs for the Levenberg-Marquardt (LM) training, 17-8-17 neurons and 500 epochs for the Bayesian (BR) training): **patterns**: total time needed for computing whole training set, **train**: total duration of training, **total**: addition of patterns and learning.

A neural model of the TL (`tl_lm_44b.mat`) was based on a training set of 19 patterns (the initial training set consisting of 9 patterns was completed by 8 patterns in the 1st step and by additional 2 patterns in the 2nd step because the 1st step did not satisfactorily reduce the relative variations). The numerical computation of a single pattern was completed within approximately 22.4 seconds, and therefore, the whole training set was prepared within 7 minutes. The size of the respective ANN was again very small, and therefore, the training took only 1 minute when the Levenberg-Marquardt algorithm was used.

Due to the non-monotonous nature of the input impedance, a neural model of the MA had to be trained on a training set consisting of 390 patterns. Since the numerical analysis of a single pattern took 16.6 seconds, the whole training set was built within 108 minutes. A huge amount of the training data corresponded with relatively large size of the respective ANN (16-6-16 hidden neurons for Levenberg-Marquardt training, `ma_lm_16_6_16b.mat`, 17-8-17 neurons for Bayesian, `ma_br_17_8_17a.mat`). Therefore, the training time is much longer now (30 minutes, and 75 min., respectively) than in the previous cases.

Unfortunately, the time needed for the development of neural models does not consist only of CPU time. In addition, we have to consider the time used for refining training set, for testing approximation error, for optimizing the architecture of an ANN, etc. Moreover, training is performed on the multi-start basis because the random starting values of weights and biases lead to different results for the same training patterns and neural networks.

In our validation, we respect the above described additional time requirements by multiplying CPU-time from Tab. 4 by a coefficient c_E. The value of c_E strongly depends on the experience of the person developing a neural model, and on good fortune (sometimes, very good model is obtained from the 1st training, other times, we have to repeat training many times to get a good approximation).

	c_E = 1 [min]	c_E = 5 [min]	analyses
FSS	**181**	**908**	**120**
TL	**8**	**40**	**110**
MA (LM)	**138**	**690**	**2500**
MA (BR)	**183**	**915**	**3300**

Tab. 5 The number of numerical analyses equivalent to the CPU-time demands of building a neural model.

In Tab. 5, we selected $c_E = 5$ and computed the number of numerical analyses where CPU-time demands are equivalent to the time needed for building a neural model.

The good efficiency of the development of the neural model of the FSS is caused by the fact that the numerical analysis is relatively time-consuming with respect to the MA (469 seconds versus 16.6 seconds). Moreover, an accurate neural model of the FSS is based only on 23 training patterns (versus 390 patterns to the MA), and the training is finished within 1 minute (vs. 30 min./75 min.[5] for the MA).

The development of the neural model of the TL is efficient too. Although the CPU time of a single numerical analysis of the TL is comparable to the MA (22.4 versus 16.6 seconds), even less patterns than for FSS is needed (19 versus 23), and training takes the same time (1 min.).

Obviously, low efficiency of building a neural model of MA is caused by extensive training set, which is necessary due to the non-monotonous nature of approximated quantities, and consequently, by a CPU-time demanding training process. Moreover, these long-time periods are compared with short duration of single numerical analysis.

Nevertheless, the final answer, whether building a neural model makes any sense or not, can only give us an optimization. If an optimization of a given structure can be completed within a lower number of steps equivalent to building a neural model, then neural modeling does not make any sense, and vice versa.

In the following chapter, we use genetic algorithms in conjunction with neural models in order to optimize the FSS, TL, and MA. Then we compute the number of numerical analyses needed, and we compare it with Tab. 5.

3. Neural design

In this chapter, we are going to utilize neural models of the FSS, TL, and MA in conjunction with a genetic al-

[5] The first value corresponds with the Levenberg-Marquardt training, the second one with the Bayesian regularization.

gorithm in order to reveal regions, which are suspected of containing a global minimum of a cost function. The revealed localities can be efficiently examined using numerical models and local optimization techniques.

For all three structures of interest, the same genetic algorithm is used. As a selection strategy, population decimation is exploited. Every generation consists of 20 individuals, probability of cross-over is set to 90%, and probability of mutation equals to 10%. Continuous parameters are binary encoded using 8 bits.

Genetic optimization is stopped when a prescribed value of the cost function is reached (*a successful realization*) or when 500 iteration steps are passed (*an unsuccessful realization*). The optimization of every structure is performed over 5 successful realizations (unsuccessful realizations are not considered).

FSS is optimized using the model `fss_lm_xe.mat`. During optimization, the width of a conductive element b, and the width of a discretization cell B is searched so that the module of reflection coefficient of Floquet mode (0,0) is maximal at the frequency: f_2 = 12.0 GHz, and its 3-dB decrease appears at f_1 = 9.0 GHz, and f_3 = 15.0 GHz. The optimization is stopped when the value of the cost function is lower than $e_{\mathrm{FSS}} \le 0.050$ GHz^2.

Successful realizations of the optimization process are listed in Tab. 6. The results indicate a potential global minimum of the cost function in the region $b \in \langle 3.27$ mm, 4.04 mm$\rangle$, and $B \in \langle 16.38$ mm, 16.65 mm$\rangle$. A numerical analysis of the FSS with optimal widths b_{opt}, and B_{opt} (columns numeric analysis of Tab. 6) confirms vicinity to desired frequency properties.

		optimum			numeric analysis		
	steps [-]	cost GHz^2	b [mm]	B [mm]	f_1 [GHz]	f_2 [GHz]	f_3 [GHz]
#1	10	0.050	3.27	16.38	9.03	12.20	14.97
#2	17	0.047	3.58	16.47	8.97	12.12	14.97
#3	1	0.044	4.04	16.65	8.97	12.12	14.97
#4	6	0.049	3.51	16.56	8.97	12.12	14.88
#5	4	0.044	3.86	16.61	8.97	12.12	14.94

Tab. 6 Genetic optimization of an FSS using the neural model. Desired frequency course of the module of reflection coefficient: f_1 = 9.0 GHz, f_2 = 12.0 GHz, and f_3 = 15.0 GHz. Stopping value of the cost function: $e_{\mathrm{FSS}} \le 0.050$ GHz^2.

The average number of iteration steps of the genetic optimization is approximately equal to 8 generations (see column cost of Tab. 6). Since every generation consists of 20 individuals, 160 triplets $[f_1, f_2, f_3]$ is computed. On the contrary, CPU-time demands of development of a neural model are equivalent to computing 120 triplets (Tab. 5).

We can therefore conclude that the neural model of the FSS replaces the numerical one in an effectual way.

TL is optimized using the model `tl_lm_44b.mat`. The optimization is aimed to estimate such dielectric constants of substrate ε_{r1} and of dielectric cover layer ε_{r2} so that phase constant of the dominant mode is β_1 = 800 m^{-1} on 20 GHz and is equal to β_2 = 1200 m^{-1} on 30 GHz. The optimization is stopped when the value of the cost function is lower than $e_{\mathrm{TL}} \le 25$ m^{-2}.

		optimal			numeric anal.	
	steps [-]	cost [m^{-2}]	ε_2 [-]	ε_1 [-]	β_1 [m^{-1}]	β_2 [m^{-1}]
#1	16	25.0	3.52	3.77	796.9	1201.1
#2	7	18.4	3.91	3.68	799.0	1203.2
#3	22	25.0	4.44	3.50	798.3	1203.9
#4	7	18.9	4.38	3.54	799.9	1205.7
#5	24	19.2	3.83	3.70	798.7	1202.8

Tab. 7 Genetic optimization of a TL using the neural model. Desired phase constant on 20 GHz: β_1 = 800 m^{-1}, on 30 GHz: β_2 = 1200 m^{-1}. Stopping value of cost: $e_{\mathrm{TL}} \le 25$ m^{-2}.

Successful realizations of the optimization process are listed in Tab. 7. The results show that a potential global minimum of the cost function can be located in the region $\varepsilon_{r1} \in \langle 3.50, 3.77\rangle$, and $\varepsilon_{r2} \in \langle 3.52, 4.44\rangle$. A numerical analysis of the TL with optimal dielectric constants ε_{1opt}, and ε_{2opt} (columns numeric analysis of Tab. 7) confirms vicinity to the desired dispersion characteristics.

The average number of iteration steps of the genetic optimization is equal approximately to 15 generations (see column cost of Tab. 7). Since every generation consists of 20 individuals, 300 doublets $[\varepsilon_1, \varepsilon_2]$ is computed. On the contrary, CPU-time demands of development of a neural model are equivalent to computing 110 doublets (Tab. 5).

We can therefore conclude that the neural model of TL replaces the numerical one in an effectual way again.

The MA is optimized using the logarithmic neural model `ma_br_17_8_17a.mat`. The optimization procedure is asked to estimate the length of the dipole A, the width of the dipole B, the dielectric constant of the substrate ε, and the height of the substrate h so that the input impedance on 30 GHz is equal to $Z_{in} = (25 + j0)\ \Omega$. In the genetic optimization, dielectric constant is binary coded using 2 bits (three possible values), and the height of the substrate is binary coded using 1 bit (two possible values). The optimization is stopped when the value of the cost function is lower than $e_{\mathrm{MA}} \le 0.001\ \Omega^2$.

Successful realizations of the optimization process are listed in Tab. 8. The results show that a potential global minimum of the cost can be in $A \in \langle 3.17$ mm, 3.20 mm$\rangle$, $B \in \langle 0.055$ mm, 0.093 mm$\rangle$, $\varepsilon = 1.6$, and $h = 1.5$ mm. A numerical analysis of the MA with optimal parameters (columns of **numeric analysis** of Tab. 8) shows that the input resistance is very close to the desired value, and the input reactance deviates desired value for 20 Ω. Nevertheless, the results can be considered sufficiently close to the optimum so that the local optimization routine can be applied.

	optimal						numeric a.	
	step [-]	cost $m\Omega^2$	A [mm]	B [mm]	h [mm]	ε [mm]	R_{in} [Ω]	X_{in} [Ω]
#1	5	0.44	3.20	0.060	1.5	1.6	25.1	+21.4
#2	7	0.35	3.17	0.093	1.5	1.6	25.1	+20.5
#3	6	0.73	3.17	0.071	1.5	1.6	24.4	+16.7
#4	4	0.47	3.19	0.073	1.5	1.6	25.2	+21.6
#5	4	0.36	3.20	0.055	1.5	1.6	25.0	+20.4

Tab. 8 Genetic optimization of an MA using the neural model. Desired input impedance on 30 GHz: Z_{in} = 25 Ω, X_{in} = 0 Ω,. Stopping value of the cost function: $e_{MA} \leq 0.001\ \Omega^2$.

The average number of iteration steps of the genetic optimization is equal approximately to 5 generations (see column **cost** of Tab. 14). Since every generation consists of 20 individuals, 100 quadruples [A, B, ε, h] is computed. Other hand, CPU-time demands of the development of a neural model are equivalent to compute 3300 quadruplets.

We can therefore conclude that building the neural model of the MA does not make any sense in our situation.

Finally, our experience can be summarized in the following items:

- Neural models that are intended to approximate monotonous quantities, can be developed efficiently (a small number of training patterns, rapid training, good accuracy), and therefore, they can successfully replace numeric models of EM structures.
- If neural models are asked to approximate quantities on non-monotonous, oscillatory-like nature, then their development is rather laborious, and their building for a single use is inefficient. On the other hand, these models can be included with CAD tools instead of approximate mathematical models, and then, their development makes sense.

In the last chapter, we are going to briefly summarize all the conclusions from the paper and to end by making a few comments on the topic.

4. Conclusions

In the paper, we have discussed the exploitation of artificial neural networks in relation to the modeling of EM structures. We used an ANN in the role of a transformer, which statically maps physical parameters of modeled structures (dimension, permittivity, permeability, etc.), to the technical parameters (reflection coefficient, phase constant, input impedance). For this purpose, feed-forward neural networks can be used.

ANN can contain neurons called ADALINE (back-propagation networks completed by local training routines). As an activation function, we use a tangential sigmoid (hidden layers, bipolar output quantities) or a logarithmic one (unipolar output quantities).

In our situation, the best results are obtained using professionally programmed ANNs from the neural network toolbox of matlab. As training algorithms, we exploit local training routines by reordering training patterns and with multi-starting in order to avoid convergence at the local minimum of the error surface.

First, a proper training set has to be prepared. In order to accomplish this aim, we equidistantly *sample* the input space (physical parameters) with the relatively long sampling step. For all *samples* of input parameters, we perform numerical analysis to obtain corresponding output responses (technical parameters). That way, initial set is built.

In order to achieve a good accuracy of the neural model, the training set has to be refined. Computing relative variations of the output responses, we add new patterns to the initial training set in order to reduce the relative variations below the prescribed level (e.g., 10 %). If the approximated quantities exhibit very high dynamics then the dynamics have to be properly reduced (a suitable transform should be used). As shown in our paper, exploring the natural logarithm for this purpose is not the best solution.

In the second step, a proper architecture of the ANN has to be estimated. According to the number of training patterns, the number of hidden layers and the number of their neurons are guessed. Then, the Bayesian regularization is used in order to estimate the number of efficiently used parameters, whose value should be from 70% to 90%. If the number of efficiently used parameters is not within this interval, architecture has to be modified.

In the third step, the ANN of the proposed architecture has to be trained using the Levenberg-Marquardt algorithm. Training should be finished within a reasonable number of iteration steps (200 to 1000) and with sufficiently low training error (from 10^{-5} to 10^{-7}). Since the training error describes deviation between the numeric and neural models in the sampling points, the quality of the neural model should be tested even for the inter-lying points (e.g., a certain number of randomly located *samples* is generated, and the error is evaluated for those samples). If significant deviations are revealed, then the ANN has to be re-trained with a lower number of neurons, with introduced bottleneck, or with the Levenberg-Marquardt procedure replaced by the Bayesian regularization (over-training is prevented).

By performing the above-described steps, an accurate model of an EM structure can be efficiently built.

Acknowledgment

Research published in the paper has been financed by grants 102/01/0571 and 102/01/0573 of the Czech Grant Agency, and by the research program MSM 262200011.

References

[1] HAYKIN, S. Neural Networks: A Comprehensive Foundation. Englewood Cliffs. Macmillan Publishing Company, 1994.

[2] CICHOCKI, A.,UNBEHAUEN, R. Neural Networks for Optimization and Signal Processing. Chichester: J. Wiley & Sons, 1994.

[3] CHANG, P. R., YANG, W. H., CHAN, K. K. A Neural Network Approach to MVDR Beamforming Problem. IEEE Transactions on Antennas and Propagation.1992, vol. 40, no. 3, p. 313 - 322.

[4] MAILLOUX, R. J., SOUTHALL, H. L. The Analogy Between the Butler Matrix and the Neural-Network Direction-Finding Array. IEEE Antennas and Prop. Magaz. 1997, vol. 39, no. 6, p. 27 - 32.

[5] SAGIROGLU, S., GUNEY, K. Calculation of Resonant Frequency for an Equilateral Triangular Microstrip Antenna with the Use of Artificial Neural Networks. Microwave and Optical Technology Letters. 1997, vol. 14, no. 2, p. 89 - 93.

[6] MISHRA, R. K., PATNIAK, A. Neurospectral Computation for Complex Resonant Frequency of Microstrip Resonators. IEEE Microw. and Guided Wave Lett. 1999, vol. 9, no. 9, p. 351 - 353.

[7] BANDLER, J. W., ISMAIL, M. A., RAYAS-SÁNCHEZ, J. E., ZHANG, Q.-J. Neuromodeling of Microwave Circuits Exploiting Space-Mapping Technology. IEEE Trans. on Microwave Theory and Techniques. 1999, vol. 47, no. 12, p. 2417 - 2427.

[8] WANG, S., WANG, F., DEVABHAKTUNI, V. K., ZHANG, Q.-J., A Hybrid Neural and Circuit-Based Model Structure for Microwave Modeling. In Proceedings of the 29th European Microwave Conference. Munich (Germany), 1999, p. 174 - 177.

[9] WU, S., VAI, M., PRASAD, S. Reverse Modeling of Microwave Devices Using a Neural Network Approach. In Proc. of the Asia Pacific Microwave Confer. New Delhi (India), 1996, p. 951 - 954.

[10] PATNIAK, A., PATRO, G. K., MISHRA, R. K., DASH, S. K. Effective Dielectric Constant of Microstrip Line Using Neural Network. In Proceedings of the Asia Pacific Microwave Conference. New Delhi (India), 1996, p. 955 - 957.

[11] GUPTA, K. C., WATSON, P. M. Applications of ANN Computing to Microwave Design. In Proceedings of the Asia Pacific Microwave Conference. New Delhi (India), 1996, p. 825 - 828.

[12] ANTONINI, G., ORLANDI, A. Gradient Evaluation for Neural-Networks-Based EM Optimization Procedures. IEEE Trans. on Microwave Theory and Tech. 2000, vol. 48, no. 5, p. 874 - 786.

[13] CHRISTODOULOU, C., GEORGIOPOULOS, M. Applications of Neural Networks in Electromagnetics. Norwood: Artech House, 2000.

[14] ZHANG Q. J., GUPTA, K. C. Neural Networks for RF and Microwave Design. Norwood: Artech House, 2000.

[15] DEMUTH, H., BEALE, M. Neural Network Toolbox for Use with Matlab: User's Guide. Natick: The MathWorks Inc., 2000.

[16] HAUPT, R. L. An Introduction to Genetic Algorithms for Electromagnetics. IEEE Antennas and Propagation Magazine. 1995, vol. 37, no. 2, p. 7 - 15.

[17] WEILE, D. S., MICHIELSSEN, E. Genetic Algorithm Optimization Applied to Electromagnetics: A Review. IEEE Trans. on Antennas and Propagation, AP-45, 3, March 1997, p. 343 - 353.

[18] JOHNSON, J. M., RAHMAT-SAMII, Y. Genetic Algorithms in Engineering Electromagnetics. IEEE Antennas and Propagation Magazine. 1997, vol. 39, no. 4, p. 7 - 21.

[19] ALTMAN, Z., MITTRA, R., BOAG, A. New Designs of Ultra Wide-Band Communication Antennas Using a Genetic Algorithm. IEEE Transactions on Antennas and Propagation. 1997, vol. 45, no. 10, p. 1494 - 1501.

[20] JONES, E. A., JOINES, W. T. Design of Yagi-Uda Antennas Using Genetic Algorithms. IEEE Transactions on Antennas and Propagation. 1997, vol. 45, no. 9, p. 1368 - 1392.

[21] YAN, K.-K., LU, Y. Sidelobe Reduction in Array-Pattern Synthesis Using Genetic Algorithm. IEEE Transactions on Antennas and Propagation. 1997, vol. 45, no. 7, p. 1117 to 1121.

[22] GEN, M. CHENG, R. Genetic Algorithms & Engineering Design. Chichester: John Wiley & Sons, 1997.

[23] HAUPT, R. L., HAUPT, S. E. Practical Genetic Algorithms. Chichester: John Wiley & Sons, 1998.

[24] SCOTT, C. Spectral Domain Method in Electromagnetics. Norwood: Artech House, 1989.

[25] HERTZ, J., KROGH, A., PALMER, R. G. Introduction to the Theory of Neural Computation. Reading: Addison Wesley, 1991.

[26] HAYKIN, S. Adaptive Filter Theory, 2nd edition. Englewood Cliffs: Prentice-Hall, 1991.

[27] MOSIG, J. R., GARDIOL, F. E. A dynamical radiation model for microstrip structures. In P. HAWKES, Advances in Electronics and Electron Physics. New York: Academic Press, 1982.

[28] MOSIG, J. R., GARDIOL, F. E. Analytical and numerical techniques in the Green's function treatment of microstrip antennas and scatterers. IEE Proc. H. 1982, vol. 130, no. 2, p. 172 - 182.

[29] MOSIG, J. R., GARDIOL, F. E. General integral equation formulation for microstrip antennas and scatterers. IEE Proceedings H. 1985, vol. 132, no. 7, 424 - 432.

[30] LEE, J. F. Finite element analysis of lossy dielectric waveguides. IEEE Transactions on Microwave Theory and Techniques. 1994, vol. 42, no. 6, p. 1025 - 1031.