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.

Moreover, 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.

In 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 \( a = 11 \, \text{mm} \), and even the height of the cell is assumed to be constant \( A = 12 \, \text{mm} \). The width of the conductive element is changed within the interval \( b \in <1 \, \text{mm}, 7 \, \text{mm}> \), cell width can intervene between \( B \in <10 \, \text{mm}, 22 \, \text{mm}> \).

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).

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

$$c(b,B) = \frac{100}{3} \sum_{i} \left( \frac{f_n(b,B) - f_i(b,B)}{f_i(b,B)} \right) \%$$

(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 $f_i$ 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 $B > 16 \text{ mm}$ might be considered as sufficiently small, for $B < 16 \text{ mm}$ 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 ($B < 16 \text{ mm}$ for all $b$) is re-sampled with smaller discretization step ($\Delta B = 1.5 \text{ mm}$). Therefore, 35 training patterns ($5 \times 7$) 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 $10^{-7}$ within 606 iteration cycles (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 $f_n = f_s(b,B)$, $n = 1, 2, 3$, is very steep in the area of the highest error (i.e. $\Delta f_n$ is high for 2 neighboring learning patterns). If sampling in this area is refined, then $\Delta f_n$ is reduced for neighboring patterns. Therefore, we can practically conclude that $\Delta f_n$ should be similar for all the neighboring patterns in the training set.

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

$$\delta f_n(a) = 100 \frac{f_i(b_i,B) - f_n(b_i,B)}{f_i(b_i,B) + f_n(b_i,B)} \%$$

(2 a, b)

$$\delta f_n(a) = 100 \frac{f_i(b_i,B) - f_n(b_i,B)}{f_i(b_i,B) + f_n(b_i,B)} \%$$

$n = 1, 2, 3$

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: $p_1 = [4 \text{ mm}, 11.5 \text{ mm}], p_2 = [4 \text{ mm}, 14.5 \text{ mm}], p_3 = [4 \text{ mm}, 17.5 \text{ mm}], p_4 = [4 \text{ mm}, 20.5 \text{ mm}], p_5 = [4 \text{ mm}, 23.5 \text{ mm}], p_6 = [4 \text{ mm}, 26.5 \text{ mm}], p_7 = [4 \text{ mm}, 29.5 \text{ mm}], p_8 = [4 \text{ mm}, 32.5 \text{ mm}].$
\[ p_1 = [4 \text{ mm}, 17.5 \text{ mm}], \quad p_2 = [7 \text{ mm}, 11.5 \text{ mm}], \quad p_3 = [7 \text{ mm}, 14.5 \text{ mm}], \quad p_4 = [7 \text{ mm}, 17.5 \text{ mm}], \quad p_5 = [5.5 \text{ mm}, 10 \text{ mm}], \quad \text{and} \quad p_6 = [5.5 \text{ mm}, 13 \text{ mm}]. \]

\[
\begin{array}{cccccc}
A & 10 - 13 & 13 - 16 & 16 - 19 & 19 - 22 \\
1.0 & 9.9 & 9.4 & 8.5 & 8.4 \\
4.0 & 14.7 & 10.6 & 10.1 & 9.3 \\
7.0 & 29.5 & 18.9 & 13.5 & 9.9 \\
\end{array}
\]

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. A lower 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 \( A = B = 12.7 \text{ mm} \). At the bottom of the shielding waveguide, a lossless dielectric substrate of the dielectric constant \( \varepsilon_1 \in <1.0, 5.0> \) and of the height \( h = 1.27 \text{ mm} \) is placed. At the center of the substrate, a PEC microstrip of a negligible thickness \( t \approx 0 \) and of the fixed width \( w = 1.27 \text{ mm} \) is placed. The microstrip can be covered by another dielectric layer of a dielectric constant \( \varepsilon_2 \in <1.0, 5.0> \) and of height \( h = 1.27 \text{ mm} \). 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 \( f_1 = 20 \text{ GHz} \) \((\beta_1)\) and on \( f_2 = 30 \text{ GHz} \) \((\beta_2)\) are obtained.

The neural model of the TL consists of 2 inputs, because doublets \([\varepsilon_1, \varepsilon_2]\) form the input patterns. The output layer again contains 2 neurons because respective doublets of propagation constants \([\beta_1, \beta_2]\) 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 \((\varepsilon_1, h)\), which might be covered by another dielectric layer \((\varepsilon_2, h)\). Longitudinal homogeneity is assumed. Losses in dielectrics and metal are neglected.](image)

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 \( f = 20 \text{ GHz} \). 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 \( p_1 = [2.0, 1.0], \quad p_2 = [4.0, 1.0], \quad p_3 = [1.0, 2.0], \quad p_4 = [3.0, 2.0], \quad p_5 = [5.0, 2.0], \quad p_6 = [1.0, 4.0], \quad p_7 = [3.0, 4.0], \quad p_8 = [5.0, 4.0]. \) In the brackets, 1st value is associated with \( \varepsilon_2, \) and 2nd one with \( \varepsilon_1. \)

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.

\[
\begin{array}{ccc}
A & 1 - 3 & 3 - 5 \\
1.0 & 23.6 & 19.8 \\
3.0 & 8.6 & 8.1 \\
5.0 & 5.0 & 4.4 \\
\end{array}
\]

\[
\begin{array}{ccc}
B & 1.0 & 3.0 & 5.0 \\
1 - 3 & 44.3 & 29.6 & 18.1 \\
3 - 5 & 24.6 & 21.1 & 17.4 \\
\end{array}
\]

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 $A \in <1.0 \text{ mm}, 4.0 \text{ mm}>$ and the width $B \in <0.05 \text{ mm}, 0.10 \text{ millimeters}>$ 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 $\varepsilon_r = [1.0, 1.6, 2.0]$ and of a height $h = [1.0 \text{ mm}, 1.5 \text{ mm}]$. Losses, both in the dielectrics and in the metal, are neglected. The antenna is assumed to operate on the frequency $f = 30 \text{ GHz}$.

Fig. 5 Microstrip dipole on the dielectric substrate of the dielectric constant $\varepsilon_r$ 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 $Z_{in}$ = $R_{in}$ + $jX_{in}$ on the frequency $f$ = 30 GHz.

The neural model of an MA consists of 4 inputs, because quadruplets $[A, B, \varepsilon, h]$ form the input patterns. Output layer contains 2 neurons because respective doubles $[R_{in}, X_{in}]$ 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 $\Delta A$ and $\Delta B$ have to be determined because the discretization of $h$ and $\varepsilon$ 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 doubles $[h, \varepsilon]$. The described training set is at one's disposal in the file ma.xls.

A proper choice of the discretization steps $\Delta A$ and $\Delta B$ 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

$$\rho_{\text{TL}} = \min \left\{ f_i(h, B) \right\} f_i(h, B) f_i(h, B) f_i(h, B) \right\} f_i(h, B) f_i(h, B) f_i(h, B) = 3 \cdot 10^{-1}$$

$$\rho_{\text{FSS}} = \min \left\{ R_{\text{in}}(A, B) X_{\text{in}}(A, B) \right\} R_{\text{in}}(A, B) X_{\text{in}}(A, B) = 4 \cdot 10^{-4}$$

Considering both non-monotonous nature and high dynamics of approximated quantities, the initial sampling step is set as very short: $\Delta A = 0.25$ mm, $\Delta B = 12.5$ mm. Then, training set has $N_h \times N_B \times N_{\varepsilon} \times N_r = 13 \times 5 \times 3 \times 390$ 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

$$\delta f^{(A)}_n = 100 \left[ \frac{f^*_n(A, B) - f_n(A, B)}{\max f^*_n(A, B), f_n(A, B)} \right] \quad n = 1, 2 \quad (3a)$$

where $f^*_n = R_{in}$ is input resistance and $f_n = X_{in}$ 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 $\delta R_{in}^{(A)} < 17\%$; 49 \%, and relative variations of the input reactance intervene within $\delta X_{in}^{(A)} < 12\%$; 197 \%. The highest values of $\delta R_{in}^{(A)}$ are dominantly at $A < 1.00$ mm, 1.25 mm for all $B$. Other hand, the highest values of $\delta X_{in}^{(A)}$ is found at $A < 2.50$ mm, 2.75 mm and at $A < 3.00$ mm, 3.25 mm for all $B$.

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

$$c_r(A, B) = \frac{100}{6} \sum_{a = 1}^{3} \sum_{b = 1}^{3} \frac{R_{\text{in}}(A, B, h_{a, b}, \varepsilon_{a, b}) - R_{\text{in}}(A, B, h_{a, b}, \varepsilon_{a, b})}{R_{\text{in}}(A, B, h_{a, b}, \varepsilon_{a, b})}$$

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

$$c_x(A, B) = \frac{100}{6} \sum_{a = 1}^{3} \sum_{b = 1}^{3} \frac{X_{\text{in}}(A, B, h_{a, b}, \varepsilon_{a, b}) - X_{\text{in}}(A, B, h_{a, b}, \varepsilon_{a, b})}{X_{\text{in}}(A, B, h_{a, b}, \varepsilon_{a, b})}$$

as depicted in Fig. 6B. In (4), $A$ is the length of the microstrip dipole, $B$ is its width, $h = [1.0 \text{ mm}, 1.5 \text{ mm}]$ denotes the height of the substrate, and $\varepsilon = [1.0, 1.6, 2.0]$ is the dielectric constant of the substrate. In addition, $R_{in}$ is the input reactance of the MA provided by a neural model, and $R_{in}$ is the same quantity provided by a numerical analysis. Similarly, $X_{in}$ 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 $[3.125 \text{ mm}, 0.075 \text{ mm}]$ makes the rest of the...
figure unreadable. Suppressing this error, Fig. 6C is obtained. In this figure, the relative error of input reactance reaches up to 20% for \( A \in <2.50 \text{ mm}, 3.50 \text{ mm}> \). Therefore, our hypothesis is confirmed.

If very high approximation error of the neural model of the MA is required to be reduced, discretization steps \( \Delta A, \Delta B \) 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

\[
\begin{align*}
 r_m(A, B) &= \ln[r_m(A, B) + 1] \\
x_m(A, B) &= \ln[x_m(A, B) + 1065]
\end{align*}
\]

where \( r_m \) and \( x_m \) are the input resistance and the input reactance of the MA from the original training set, respectively. Symbols \( r_m \) and \( x_m \) denote input resistance and input reactance of MA from the transformed set. Constant 1065 is derived from \( \min\{X_m\} = -1063 \Omega \).

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 \( \rho_{\text{MA}} = 4 \cdot 10^{-4} \) to

\[
\rho_{\text{MA}}^{\text{LN}} = \frac{\min\{r_m(A, B), x_m(A, B)\}}{\max\{r_m(A, B), x_m(A, B)\}} \approx \frac{0.366}{7.38} = 5 \cdot 10^{-2}
\]

Further, we have to investigate the relative variations in the transformed training set (see \texttt{ma.xls}). Variations of the input resistance with respect to the antenna length \( \delta r_m^{(A)} \) 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 \( \delta r_m^{(B)} \) are lower than 2% in all cases. Variations of input reactance \( \delta x_m^{(A)} \) and \( \delta x_m^{(B)} \) are lower than 10%, except for singular cases for \( A \in <1.00 \text{ mm}, 1.50 \text{ mm}> \), \( B \in <0.050 \text{ mm}, 0.075 \text{ mm}> \). 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 \( r_{in} \)) 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

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

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

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

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 approximated quantities exhibit very high dynamics (more than \( 10^{-7} \) 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.

<table>
<thead>
<tr>
<th></th>
<th>analysis</th>
<th>repeated</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>fss</td>
<td>24.7 s</td>
<td>19</td>
<td>469.0 s</td>
</tr>
<tr>
<td>sl</td>
<td>5.6 s</td>
<td>4</td>
<td>22.4 s</td>
</tr>
<tr>
<td>ma</td>
<td>16.6 s</td>
<td>1</td>
<td>16.6 s</td>
</tr>
</tbody>
</table>

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.

Due to the non-monotinous nature of the input impedances, a neural model of the MA had to be trained on a training set consisting of 590 patterns. Since the numerical analysis of a single pattern took 16.6 seconds, 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-monotinous nature of the input impedances, 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.

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).

<table>
<thead>
<tr>
<th>patterns [min]</th>
<th>train [min]</th>
<th>total [min]</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSS</td>
<td>23.469.0</td>
<td>180</td>
</tr>
<tr>
<td>TL</td>
<td>19.22.4</td>
<td>7</td>
</tr>
<tr>
<td>MA-LM</td>
<td>390.16.6</td>
<td>108</td>
</tr>
<tr>
<td>MA-BR</td>
<td>390.16.6</td>
<td>75</td>
</tr>
</tbody>
</table>

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.

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.).

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-monotonal 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-

<table>
<thead>
<tr>
<th>$c_E = 1$ [min]</th>
<th>$c_E = 5$ [min]</th>
<th>analyses</th>
</tr>
</thead>
<tbody>
<tr>
<td>FSS 181</td>
<td>908</td>
<td>120</td>
</tr>
<tr>
<td>TL 8</td>
<td>40</td>
<td>110</td>
</tr>
<tr>
<td>MA (LM) 138</td>
<td>690</td>
<td>2500</td>
</tr>
<tr>
<td>MA (BR) 183</td>
<td>915</td>
<td>3300</td>
</tr>
</tbody>
</table>

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).

\(^5\) The first value corresponds with the Levenberg-Marquardt training, the second one with the Bayesian regularization.
Algorithm 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 \( f_1 \) of Floquet mode \((0,0)\) is maximal at the frequency: \( f_2 = 12.0 \text{ GHz} \), and its 3-dB decrease appears at \( f_1 = 9.0 \text{ GHz} \) and \( f_2 = 15.0 \text{ GHz} \). The optimization is stopped when the value of the cost function is lower than \( e_{\text{FSS}} \leq 0.050 \text{ GHz}^2 \).

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 \( b \in (3.27 \text{ mm}, 4.04 \text{ mm}) \), and \( B \in (16.38 \text{ mm}, 16.65 \text{ mm}) \). A numerical analysis of the FSS with optimal widths \( b_{\text{opt}} \), and \( B_{\text{opt}} \) (columns numeric analysis of Tab. 6) confirms vicinity to desired frequency properties.

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 \( \varepsilon_1 \) and of dielectric cover layer \( \varepsilon_2 \) so that phase constant of the dominant mode is \( \beta_1 = 800 \text{ m}^{-1} \) on 20 GHz and is equal to \( \beta_2 = 1200 \text{ m}^{-1} \) on 30 GHz. The optimization is stopped when the value of the cost function is lower than \( e_{\text{TL}} \leq 25 \text{ m}^2 \).

<table>
<thead>
<tr>
<th>steps [-]</th>
<th>cost [\text{m}^2]</th>
<th>( \varepsilon_1 ) [-]</th>
<th>( \varepsilon_2 ) [-]</th>
<th>( \beta_1 ) [\text{m}^{-1}]</th>
<th>( \beta_2 ) [\text{m}^{-1}]</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>16</td>
<td>25.0</td>
<td>3.52</td>
<td>3.77</td>
<td>796.9</td>
</tr>
<tr>
<td>#2</td>
<td>7</td>
<td>18.4</td>
<td>3.91</td>
<td>3.68</td>
<td>799.0</td>
</tr>
<tr>
<td>#3</td>
<td>22</td>
<td>25.0</td>
<td>4.44</td>
<td>3.50</td>
<td>798.3</td>
</tr>
<tr>
<td>#4</td>
<td>7</td>
<td>18.9</td>
<td>4.38</td>
<td>3.54</td>
<td>799.9</td>
</tr>
<tr>
<td>#5</td>
<td>24</td>
<td>19.2</td>
<td>3.83</td>
<td>3.70</td>
<td>798.7</td>
</tr>
</tbody>
</table>

Tab. 7 Genetic optimization of a TL using the neural model. Desired phase constant on 20 GHz: \( \beta_1 = 800 \text{ m}^{-1} \), on 30 GHz: \( \beta_2 = 1200 \text{ m}^{-1} \). Stopping value of cost: \( e_{\text{TL}} \leq 25 \text{ m}^2 \).

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 \( \varepsilon_1 \in (3.50; 3.77) \), and \( \varepsilon_2 \in (3.52; 4.44) \). A numerical analysis of the TL with optimal dielectric constants \( \varepsilon_{\text{opt}} \) (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 \( \varepsilon \), and the height of the substrate \( h \) so that the input impedance on 30 GHz is equal to \( Z_{\text{in}} = (25 + j0) \Omega \). 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 \( e_{\text{MA}} \leq 0.001 \Omega^2 \).

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 \( A \in (3.17 \text{ mm}, 3.20 \text{ mm}) \), \( B \in (0.055 \text{ mm}, 0.093 \text{ mm}) \), \( \varepsilon = 1.6 \), and \( h = 1.5 \text{ mm} \). 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 \( \Omega \). Nevertheless, the results can be considered sufficiently close to the optimum so that the local optimization routine can be applied.
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 $10^{-3}$). 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