Automated Modeling of Microwave Structures by Enhanced Neural Networks

Petr ŠMÍD, Zbyněk RAIDA

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

xsmidp01@stud.feec.vutbr.cz, raida@feec.vutbr.cz

Abstract. *The paper describes the methodology of the automated creation of neural models of microwave structures. During the creation process, artificial neural networks are trained using the combination of the particle swarm optimization and the quasi-Newton method to avoid critical training problems of the conventional neural nets.*

In the paper, neural networks are used to approximate the behavior of a planar microwave filter (moment method, Zeland IE3D). In order to evaluate the efficiency of neural modeling, global optimizations are performed using numerical models and neural ones. Both approaches are compared from the viewpoint of CPU-time demands and the accuracy. Considering conclusions, methodological recommendations for including neural networks to the microwave design are formulated.

Keywords

Feed-forward neural network, recurrent neural network, particle swarm optimization.

1. Introduction

The design of electromagnetic (EM) structures is usually based on exploiting their numerical models. Numerical models request high computational power. Their evaluation has to be repeated many times in the optimization cycle: each update of the optimized parameters has to be followed by a new analysis. Replacing a numeric model of the designed EM structure by a neural one is one of ways to reduce CPU-time demands [1].

At the present, two basic approaches are used to optimize neural models of EM structures: conventional neural optimization and several new methods were described in detail in [2]. A conventional approach exploiting feed-forward neural networks (NN) and recurrent ones (trained by gradient algorithms) is used for an automated development of EM models in most cases. Unfortunately, this technology exhibits several imperfections:

- The training has to be repeated several times in order to reveal a global minimum by local (gradient) optimization techniques. If the local algorithm is replaced by a global one (genetic, particle-swarm [3], etc.), the training consumes enormous CPU power.
- Developed neural models can suffer from over-training: the training process minimizes errors in training patterns, but interlaying points exhibit a relatively high error.

In the paper, possible solutions of the above-listed problems are proposed and applied to modeling EM structures:

- Global training algorithms and local one are suitably combined in order to obtain a robust and efficient training procedure.
- A methodology of the automated creation of neural models is proposed in order to eliminate the problems of over-training, an inappropriate architecture, and related difficulties.

The proposed approaches are tested and verified by modeling a canonical electromagnetic structure (a planar low-pass filter described in Section 2). In the second part of the paper, the developed neural model is used in conjunction with an optimization procedure in order to improve efficiency of the filter design.

2. Filter Description

A planar filter is chosen to be modeled (a three-pole low-pass stepped impedance filter with Chebychev response; Fig. 2). The cut-off frequency is f_c = 1 GHz, the pass-band ripple equals to 0.1 dB, and the source/load impedance is Z_0 = 50 Ω. The filter is etched on the substrate with the dielectric constant ε_r = 10.8 and height h = 1.27 mm.

The filter consists of five microstrip lines. The first segment and the last one are of the characteristic impedance Z_0 = 50 Ω at the cut-off frequency. The second segment and the fourth one are inductive lines of the characteristic impedance Z_L = 93 Ω. The third segment line is capacitive (the characteristic impedance Z_C = 24 Ω).

The width of the input/output microstrip line equals to w_0 = 1.1 mm to get Z_0 = 50 Ω at the cut-off frequency. Ini-

tially, we set the dimensions of the reactive segments of the filter to the following values:

- The length and the width of the inductive segments are l_{L0} = 9.81 mm, and w_{L0} = 0.20 mm, respectively.
- The length and the width of the capacitive segment is l_{C0} = 7.11 mm, and w_{C0} = 4.00 mm.

Around the point [l_{L0}, l_{C0}], we wish to approximate the dependence of the filter forward gain s_{21}(l_L, l_C, f) for the fixed widths of inductive segments w_L = 0.20 mm and the capacitive one w_C = 4.00 mm, changing the frequency from the cut-off one f_c = 1.0 GHz to f_{max} = 5.0 GHz. The dependence is approximated by a feed-forward neural network.

3. Neural Model

The filter is modeled by a feed-forward artificial neural network [1], [4]. Artificial neural networks (ANN) are electronic systems (software or hardware ones) which structure is similar to a human brain. Feed-forward ANN consists of a set of neurons placed in several layers. Synaptic weights between neurons are set during the learning process [1], [4].

Feed-forward ANN statically maps the input patterns into output ones. The output response is formed by multiplying and summing input signals and by processing the result by a non-linear activation function.

During the training process, ANN is learned to behave the same way as the numeric model of the filter: considering dimensions and permittivities of the analyzed structure as input parameters, the neural model provides input impedances, directivity patterns, gains and other parameters of the structure as the output patterns.

Training the ANN is a time-demanding process. Nevertheless, when a trained ANN is used, we can expect lower time demands compared to numerical models. Since summation, multiplication and evaluating non-linear activation function are the only three mathematical operations needed to compose the output; the modeling with learned ANN is very efficient and faster than numerical methods.

The training process is based on the minimization of the highly non-linear error function of the ANN. Exploiting standard gradient algorithms, the training is usually stopped in a local minimum located in the near distance from the starting point. Therefore, the training should be started with a global algorithm, and later on switched to a gradient method.

In order to create a neural model of the filter, we need to prepare training patterns. First, the set of input patterns is obtained by combining all the values of all the state parameters (l_L, l_C, f). Next, the target patters s_{21}(l_L, l_C, f) are computed using a numerical method (Zeland IE3D). The analysis is repeated for all the input parameters (all the triplets [l_L, l_C, f]). Finally, the training patterns are obtained by integrating input patterns and corresponding output ones.

The neural network consists of three neurons in the input layer (three state parameters), several neurons in hidden layer(s) and one output neuron (one target parameter s_{21}). The number of hidden layers and neurons in them can be estimated using Bayesian regularization e.g., or, if the ANN contains only one hidden layer, according to [5]

$$n_1 = \text{integer}\left(\frac{n_0}{2} + \sqrt{R}\right). \tag{1}$$

In (1), n_1 denotes the number of neurons in the hidden layer, n_0 is the number if input neurons (input distributing nodes) [1], [4] and R denotes the number of training patterns. First, synoptic weights of the ANN are set randomly.

Next, the ANN has to be trained. The input patterns are repeatedly introduced to the input neurons and the synoptic weights and biases are changed to minimize the difference between ANN output and target patterns.

The neural network has to be trained to be able to approximate the target parameters according to the input ones. In order to obtain such a model, the training process consists of the following steps:

1. The equidistant training patterns are extracted from the file of pre-computed patterns. The patterns are pre-computed by Zeland IE3D.
2. The number of hidden neurons is computed according to the number of training patterns (1).
3. If more hidden neurons than the number of neurons in the previous run of the algorithm are necessary, or if the algorithm runs for the first time, a new ANN of random synoptic weights is built. Otherwise, we have to go to the step 8.
4. Pre-training the ANN using PSO produces several ANN. The best neural networks are selected (several ANN with the lowest error over the training patterns) and stored for the next training.
5. The selected neural networks are trained with the gradient Levenberg-Marquardt (LM) method. Training is stopped if one of the following conditions can occur:
   - the error over the training patterns drops below a prescribed tolerance;
   - the error decreases below the prescribed dropping (avoiding “long training” without getting better the error);
   - the training process reaches the maximum number of iterations;
   - the maximum training time is reached.
6. We test the trained ANN using testing patterns, which differ from training ones. First, the set of testing patterns consists of equidistant interlaying patterns (with respect to training ones) over the state space. The testing patterns are obtained using the numerical model as well.

During testing, two values are highlighted. First, the percentage error at each testing pattern is computed. Second, the number of training patterns exhibiting a higher percentage error than the desired tolerance (so called *high error patterns*) is stored.

7. The best ANN is selected according to the lowest number of high-error patterns. If every ANN has no high-error pattern, the best ANN is selected according to the lowest testing error.
8. If there is no need to increase the number of hidden neurons (step 3), and if the algorithm does not run for the first time, the current ANN is trained by LM.
9. The number of high error patterns is computed.
10. All the points, which exhibit a higher error than the desired tolerance, are incorporated into the set of the training patterns. If the high error patterns occur in more runs of the algorithm, this step can cause duplicating the patterns in the training set. The duplicated patterns have to be deleted. An index of the refinement (see the next step) has to be changed.
11. The surroundings of each added training pattern are sampled with a half-length step compared to the step in the previous refinement in the surroundings of a pattern. The length of the sampling step is derived from the refinement index of a pattern.

    If the testing error exceeds the maximum allowed error on a pattern, the pattern is incorporated into the training patterns (step 10), and the refinement index is increased by one. Each state parameter in the surroundings of the pattern is sampled with the step given by

$$k = k_{ei} / 2^{r_v} \quad , \tag{2}$$

    where k_{ei} denotes the original equidistant sampling step, i denotes the direction in the state space and r_v signs the refinement index of an added pattern.

    In order to build new additional testing input patterns, the refined state parameters (in the surrondings of a pattern) are combined the same way as the original patterns at the beginning. If the refining is performed arround two or more neighbouring patterns, some testing patterns could be duplicated. These patterns have to be removed.
12. The target response for each new testing pattern in the file of patterns is found. All the new patterns, that are not included in the existing file of patterns, have to be computed by IE3D.
13. The newly computed patterns have to be incorporated into the file of patterns.
14. If the best neural model still exhibits an unacceptably high error, return to the step 2.

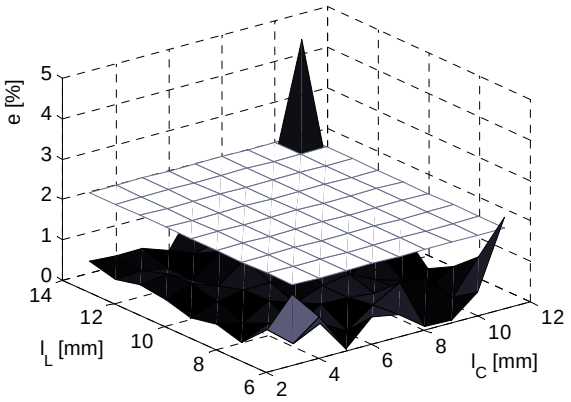

Fig. 1. The maximum error of the filter model at the frequency f = 1 GHz and the 2 % error level plane.

The state variables lay in following intervals:

- $l_L \in$ <6.0 mm, 14.0 mm>;
- $l_C \in$ <3.0 mm, 11.0 mm>;
- $f \in$ <1.0 GHz, 3.0 GHz>.

The initial (coarse) equidistant training set has contained 125 patterns, and the initial equidistant testing set has included 729 patterns. The maximum desired error of the model has equaled to 2 %. The maximum error (in equidistant testing parameters) after the first iteration has been 4.42 % (see Fig. 1).

The time response of the training error during the particle swarm optimization (PSO) pre-training in the fourth refined iteration (the fourth iteration of the above algorithm) is depicted in Fig. 2. The pre-training error reaches e = 0.027 after 100 iterations and gives 3 pre-trained ANN.

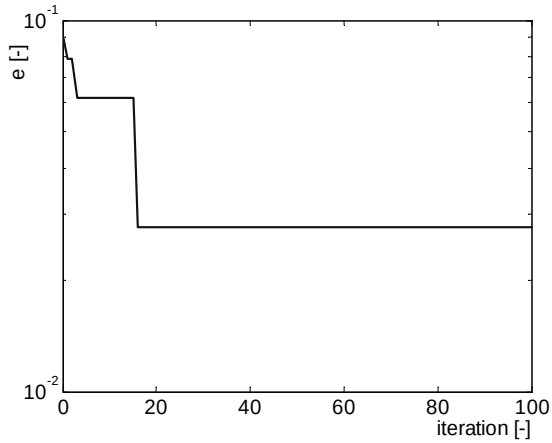

Fig. 2. The time response of the PSO pre-training in the 4th refining iteration.

These three ANN were trained using LM algorithm (Fig. 3). The best ANN reaches the training error $e = 1.3 \cdot 10^{-5}$ after 365 iterations.

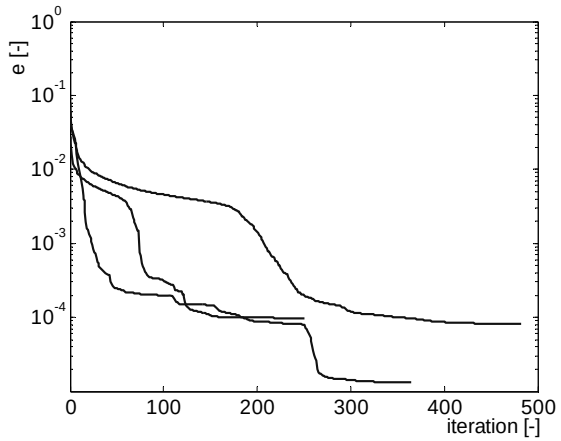

Fig. 3. The time response of LM training in the 4th refining iteration.

After 6 repetitions of the above algorithm, the filter model exhibits a sufficient precision $e_P < 2$ % (Tab. 1). The column *Refining* contains the number of the iteration, when the training (testing) patterns were refined. The numbers of training parameters and testing patterns are given in the columns N_{TR} and N_{TST}, respectively. The maximum percentage error of the neural model is given in column e_P, and the number of hidden neurons is show in the column N_H. The last column concentrates the values of time consumptions for refining the patterns and re-training the ANN.

refining	N_{TR}	N_{TST}	e_P	N_H	t
[-]	[-]	[-]	[%]	[-]	[s]
equidist.	125	729	4.4	13	26.0
1	159	729	3.2	14	19.9
2	188	782	4.5	14	53.4
3	210	883	3.3	15	122.2
4	211	883	2.2	16	38.1
5	285	1010	3.6	16	180.0
6	311	1505	1.5	18	62.3

Tab. 1. Training parameters and results for the modeled planar filter.

4. Filter Optimization

A sufficiently accurate neural model can be used as a computationally efficient substitution of a numerical model in optimization. The optimization is asked to find an inductive segment length l_L and a capacitive segment length l_C so that the filter can meet the target values of the forward gain, which are given in Tab. 2.

f_i [GHz]	0.5	1.0	1.5	2.0	2.5	3.0
s_{21i} [dB]	0.0	0.0	-3.0	-7.3	-10.0	-10.6

Tab. 2. The target filter gain.

The objective function is

$$e = \sqrt{\sum_{i=1}^{6} \left(s_{21i} - s_{21iM} \right)^2} \tag{2}$$

where s_{21i} denotes the target filter forward gain and s_{21iM} is the response of the current model. Eqn. (2) is minimized by PSO with $M = 20$ particles and absorbing walls at the boundaries. Due to the random initial values of the inductive segment length l_L and the capacitive segment length l_C, each training process is run five times, and the learning error is averaged. Therefore, the length of optimization is fixed (20 iterations).

The convergence of the optimization, which exploits the feed-forward neural model of the planar filter, is depicted in Fig. 4. There are three curves in the figure: the objective function time response of the best run (dotted line), the worst run (dashed line), and the average run (solid line). The best case, the worst one and the averaged one are sorted according to the objective function value after the last iteration.

After 20 iterations, the average value of the objective function is $e = 4.5 \cdot 10^{-3}$. This values corresponds to the inductive segment length $l_L = 9.89$ mm and the capacitive segment length $l_C = 6.85$ mm. The optimal lengths of segments have been computed in 0.069 s of the CPU time. The values of the error e correspond to the mean squared error of target filter gain over frequencies above (Tab. 2, equation. 2).

In order to compare the optimization results, the optimization of the filter is repeated using Zeland IE3D to evaluate the objective function (Fig. 5). The obtained optimal lengths (when using the numerical model to evaluate the objective function) are $l_L = 9.71$ mm, and $w_C = 6.91$ mm.

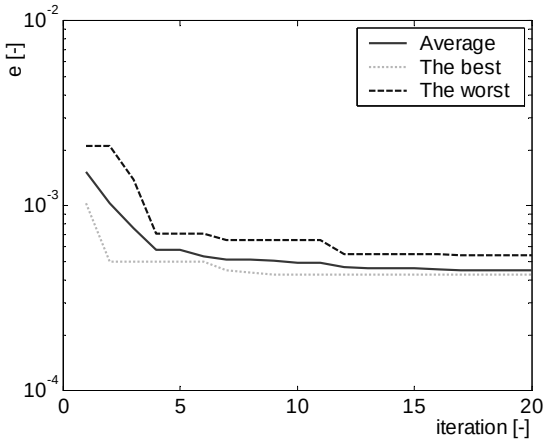

Fig. 4. Convergence of the PSO optimization of the low-pass filter. Objective function evaluated using a neural model.

Note that training and optimization methods were implemented in MATLAB 6.1 on a regular PC equipped by the processor Athlon XP 2700+, by 512 MB of RAM, and by Windows XP.

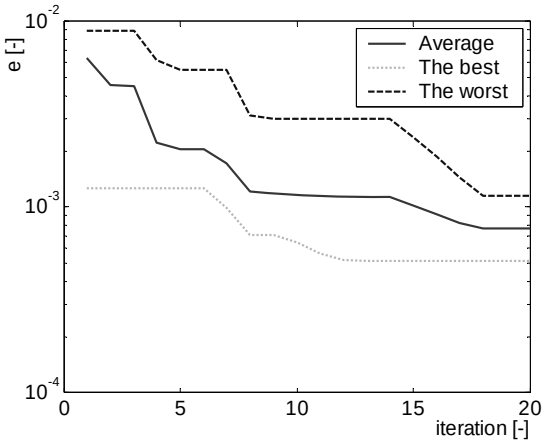

Fig. 5. Convergence of the PSO optimization of the low-pass filter. Objective function evaluated using Zeland IE3D.

The results of optimization and their verification are given in Tab. 3. The upper part of the table concentrates the optimized lengths of inductive and capacitive segments. The gain of the optimized models, and the verified gain of both the neural model and the numerical one, are given in the bottom part of the table.

Fig. 6 shows the frequency dependence of the forward gain and the reflection coefficient of the filter designed by the optimization using the neural model. The frequency characteristic was computed using Zeland IE3D.

5. Conclusions

The proposed algorithm trains the neural model of an EM structure automatically. The feed-forward neural model trained by the algorithm is the fast and sufficiently accurate approach to model low-pass planar filters. If the numerical model of the planar filter is replaced by the neural one in global optimization procedure, the CPU time demands decrease more than 32 thousand times.

		Neural model		Zeland IE3D		
optimized variables	l_L [mm]	9.79		9.71		
	l_C [mm]	6.85		6.91		
time [s]	training	501.9		-		
	optimizing	0.067		2194.0		
forward gain		model	verif.	model	verif.	target
0.5 GHz	s_{21} [dB]	-0.12	-0.18	-0.18	-0.18	0.00
1.0 GHz	s_{21} [dB]	-0.21	-0.20	-0.20	-0.20	0.00
1.5 GHz	s_{21} [dB]	-3.00	-3.00	-2.97	-2.92	-3.00
2.0 GHz	s_{21} [dB]	-7.26	-7.22	-7.22	-7.22	-7.30
2.5 GHz	s_{21} [dB]	-9.77	-9.75	-9.78	-9.75	-10.00
3.0 GHz	s_{21} [dB]	-10.55	-10.45	-10.48	-10.46	-10.50

Tab. 3. Optimization results and verification.

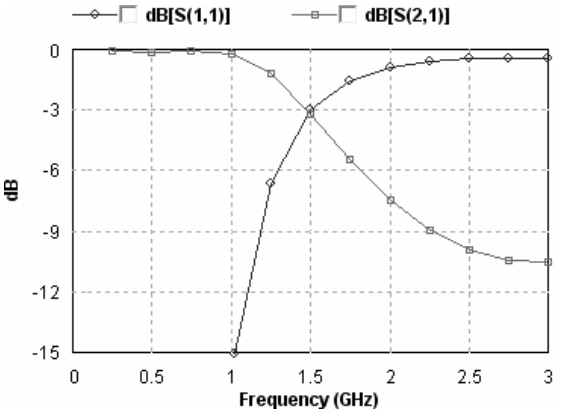

Fig. 6. Frequency response of the reflection coefficient at the filter input s_{11}, and the filter forward gain s_{21}. The filter was designed using a neural network. The scattering parameters were computed using Zeland IE3D.

Acknowledgements

The research was supported by the Czech Grant Agency under the projects no. 102/04/1079 and 102/03/H086, and by the Czech Ministry of Education under the research plan no. MSM 0021630513.

References

[1] RAIDA, Z. Modeling EM structures in the neural network Toolbox of MATLAB. *IEEE Antennas & Propagation Magazine.* 2002, vol. 44, no. 6, p. 46–67.

[2] RAYAS-SÁNCHEZ, J., E. EM-based optimization of microwave circuits using artificial neural networks: The state-of-the-art. *IEEE Transactions on Microwave Theory and Techniques*. 2004. no. 1, p. 402–434.

[3] RAHMAT-SAMII, Y., GIES, D., ROBINSON, J. Particle swarm optimization (PSO): A novel paradigm for antenna design. *The Radio Science Bulletin*. 2003, no. 305, p. 14–22.

[4] GUPTA, M. M., JIN, L., HOMMA, N. *Static and Dynamic Neural Networks from Fundamentals to Advanced Theory.* New Jersey: John Wiley & Sons, 2003.

[5] CHI D., D., SHIE-YUI, L. *Generalization for Multilayer Neural Network Bayesian Regularization or Early Stopping*. http://www.wrrc.dpri.kyoto-u.ac.jp/~aphw/APHW2004/proceedings/FWR/56-FWR-M185/56-FWR-M185%20(1).pdf

About Authors...

Petr ŠMÍD graduated at the Faculty of Electrical Engineering and Communication (FEEC), Brno University of Technology in 2003. He is interested in neural modeling of microwave structures.

Zbyněk RAIDA – for biography, see p. 83 of this issue.