Application of Neural Networks: Enhancing Efficiency of Microwave Design

Petr Šmíd, Zbyněk Raida

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.**

Neural models are required being wideband. In order to meet this requirement, feed-forward neural networks and recurrent ones are used for modelling, and their properties are in detail mutually compared.

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

Keywords – **Feed-forward neural networks, recurrent neural networks, quasi-Newton methods, particle swarm optimization.**

I. INTRODUCTION

Modern communication services require wider and wider frequency bands for their operation. Since lower frequency bands are out of their capacity today, broadband services have to operate on higher microwave frequencies.

Designing broadband microwave communication systems, efficient modelling tools are required. These modelling tools have to be based on the numeric solution of Maxwell's equations. Using the frequency-domain approach (a harmonic steady state is assumed); a broadband microwave structure has to be analyzed on each harmonics from the examined frequency band separately in order to characterize the structure. Numerical analysis in the frequency domain is therefore rather time-consuming, which complicates their potential usage in complex design tools. Numerical models are therefore to be replaced by closed-form approximate formulae [1]–[3], or by neural models [4]–[6].

Neural networks are electronic systems, which are built according to a human brain: they contain a large number of the same non-linear building blocks (neurons), they are highly parallel, they are organized in layers, and they are able to learn [7], [8]. The structure of a typical neural network is depicted in Fig. 1.

Petr Šmíd and Zbyněk Raida are with the Faculty of Electrical Engineering and Communication, Brno University of Technology, Purkyňova 118, CZ-61200 Brno, the Czech Republic, E-mail: raida@feec.vutbr.cz, xsmidp01@stud.feec.vutbr.cz

Neurons in the network (circles in Fig. 1) perform a few basic mathematical operations only: they multiply incoming signals by variable coefficients (synaptic weights $w_{i,j}(n)$ and thresholds $b_i(n)$), they sum the products and evaluate a non-linear function for the sum of products [4]. Hence, the neuron operation is computationally efficient.

Since the neural network is able to learn, we can train it to behave a similar way as a numerical model. Therefore, a properly trained neural network can replace a computationally inefficient numerical model in the design tools.

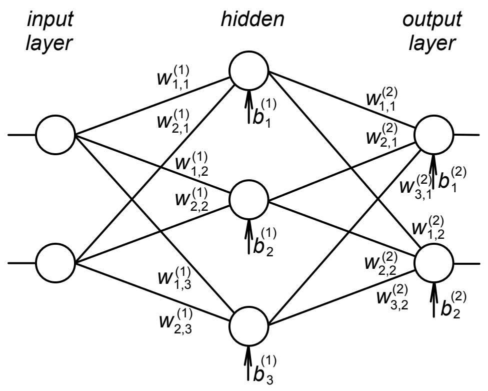


Fig. 1. An example of an artificial neural network. The symbol $w_{i,j}^{(n)}$ denotes a synaptic weight between the output of the ith neuron in the layer (n–1) and the input of jth neuron in the layer n. The symbol $b_i^{(n)}$ denotes a threshold of the ith neuron in the nth layer.

The training procedure consists of the following steps:

- Creating training patterns. Using a numerical method, the structure is analyzed for various dimensions and dielectric constants (various state variables). Combinations of state variables form the input patterns, corresponding computed parameters (impedances, scattering parameters, etc.) form the output targets. An input pattern and an output target give a training pattern together. All the training patterns form the training set.
- Building neural network. We create a neural network consisting of an estimated number of layers and of an estimated number of neurons in the layers (see Fig. 1). The number of neurons in the input layer is given by the number of state variables; the number of neurons in the output layer is determined by the number of computed parameters. The number of hidden layers and hidden neurons has to be estimated. Synaptic weights and biases are set randomly.
- Training neural network. During the training, input patterns are successively introduced into the inputs of the neural network, and synaptic weights are changed to reach desired output responses. The training is finished when the network reacts properly to all input patterns from the training set.

- Verifying neural model. We introduce such patterns to the inputs of the neural model, which differ from the input patterns of the training set. Exploiting the numerical model, correctness of the response of the network is verified. If the response is incorrect, additional training patterns have to be prepared, and training is repeated over a larger training set.
- Using neural model. The trained neural network produces output responses with a sufficient accuracy both for training patterns and for interlaying input patterns. Therefore, the neural network can replace the numerical model.

The trained neural network provides a special approximation in a fact: the exact results of the numerical analysis which are hidden in the training patterns are used for neural computing approximate results, which correspond to input parameters differing from input patterns. That way, a computationally modest neural network can replace a numerical analysis for parameters differing from training patterns.

In Section II, the described methodology is illustrated by developing a neural model of a planar stepped-impedance three-pole low-pass filter exploiting a feed-forward neural network and a recurrent one. Various algorithms are applied to reach an accurate training. The training procedures are discussed from the viewpoint of CPU-time demands and reached accuracies.

In Section III, the finished neural models are associated with global optimization techniques to verify approximation abilities and evaluate computational demands of neural nets. The obtained results are confronted with the optimization based on numeric modelling.

Section IV concludes the paper.

II. Neural Models of a Planar Filter

A. Filter Description

In order to examine different approaches to the neural modelling of microwave structures, a simple planar filter is assumed. We consider a stepped-impedance three-pole low-pass 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 $Z_0 = 50\ \Omega$. The filter is etched on the substrate with the dielectric constant $\varepsilon_r = 10.8$, and with the height h = 1.27 mm.

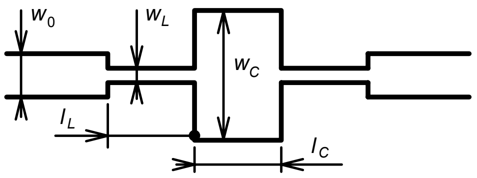


Fig. 2. A stepped-impedance three-pole low-pass filter with Chebychev response. The width of the inductive segment w_L, and the width of the capacitive segment w_C are state variables of the filter.

The width of the input/output microstrip line equals to w_0 = = 1.1 mm to get the characteristic impedance $Z_0 = 50\ \Omega$ on the cut-off frequency.

Initially, 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_L = = 9.81 mm, and w_{L0} = 0.20 mm, respectively.
- The length and the width of the capacitive segments are l_C = = 7.11 mm, and w_{C0} = 4.00 mm, respectively.

Around the point $[w_{L0}, w_{C0}]$, we wish to approximate the dependence of the filter forward gain $s_{21}(w_L, w_C, f)$ for the fixed lengths of inductive segments l_L = 9.81 mm and the capacitive one l_C = 7.11 mm, changing the frequency from the cut-off one f_c = 1.0 GHz to f_{max} = 5.0 GHz. First, the dependence is approximated by a feed-forward neural network, and second, we try to repeat the modelling procedure exploiting a recurrent neural network [5], [7], [8].

B. Feed-Forward Neural Model

Feed-forward neural networks are characteristic by a direct signal flow from the input layer to the output one without any feedback. Therefore, this type of networks simply statically maps input patterns to output targets.

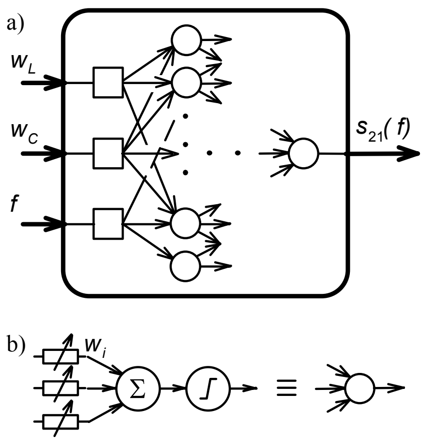


Fig. 3. a) Architecture of a feed-forward neural network for modelling a stepped-impedance three-pole low-pass filter with Chebychev response. Input layer consists of input neurons (squares). Other layers contain adaptive non-linear neurons (circles). b) Structure of an adaptive non-linear neuron.

The neural model is built to map the triplets $[w_L, w_C, f_n]$ to the filter forward gain on the frequencies f_n, i.e. $s_{21}(f_n)$. Therefore, the neural network consists of three input neurons (distributing nodes) and a single output one.

The initial (coarse) training set containing 60 training patterns consists of all the combinations of $w_L \in \{0.10, 0.30, 0.50\}$ mm, $w_C \in \{2.00, 4.00, 6.00, 8.00\}$ mm, and $f \in \{1.0, 2.0, 3.0, 4.0, 5.0\}$ GHz, which are completed by the corresponding values of the filter forward gain $s_{21}(w_L, w_C, f)$ computed by Zeland IE3D.

Next, we have to estimate a proper number of hidden layers of the network, and a proper number of neurons in those layers (i.e., we search for a proper size of the distributed memory to store information about the filter behaviour in the space $[w_L, w_C, f]$). The number of hidden layers, and the number of neurons in them, is usually estimated by Bayesian regularization [9]. In our case, the regularization proposes two hidden layers, which contain six neurons each, as the optimal network architecture. Using a larger network, an overtraining can appear (in between the training patterns, the neural approximation oscillate).

An initial training of the network is performed by applying the particle swarm optimization [10] in order to reveal global minima of the error function of the neural network (the difference between the actual network response, and the desired one – given by the training pattern). The training process is started with the following parameters: the number of particles is set to $M = 50$, absorbing walls are used on the boundaries of the training region, and the length of a single training process is limited by 150 iterations. The pre-training convergence for the initial feed-forward neural network is depicted in Fig. 4.

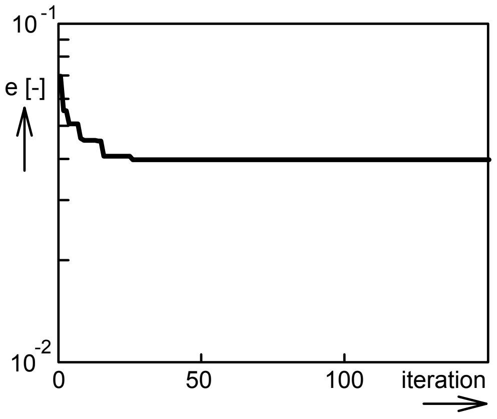


Fig. 4. Particle swarm optimization pre-training convergence for the initial feed-forward neural network configuration.

The global training moves the state vector of the neural network (synaptic weights and thresholds) in a vicinity of a global minimum of the error function. Using the particle swarm optimization for the global training, the neural model reaches the value of the mean squared error over all training patterns equal to $e = 3.95 \cdot 10^{-2}$, and the pre-training process takes $t = 5.30$ s of the CPU time when using the regular PC equipped by AMD Athlon 2700+, 512 MB RAM, Windows XP, and MATLAB 6.1.

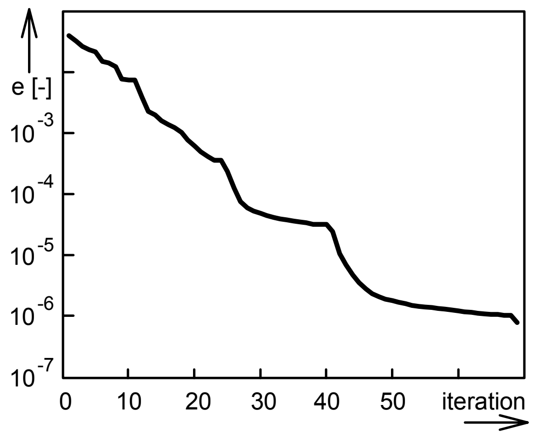


Fig. 5. Levenberg-Marquardt refinement training convergence for the initial feed-forward neural network configuration.

In the following step, training of the neural model is refined applying Levenberg-Marquardt method [11]. I.e., a local optimization method is used to move the state vector from the point in the vicinity of the global optimum to the global optimum. Time response of the training error is shown in Fig. 5. After 68 iterations, the training error equals to $e = 8.00 \cdot\cdot 10^{-7}$ (the network refinement is stopped when the error drops below the prescribed limit $e_{max} = 10^{-6}$). The refinement training consumes 2.06 s of the CPU time.

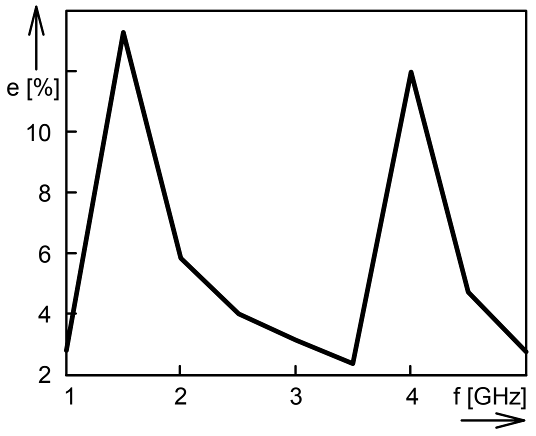


Fig. 6. The maximum approximation error of the neural model for $w_L = 0.20$ mm, and $w_C = 4.00$ mm (the worst case) for the initial feed-forward neural network configuration.

In order to evaluate the model accuracy, a set of testing patterns is compiled. The testing set enriches the training one by the interlaying patterns in order to test the approximation abilities of the neural model outside the training points also. The testing set contains 315 patterns.

The worst case, when the model error reaches the highest values, appears for the width of the inductive segment $w_L =$ $= 0.20$ mm, and for the width of the capacitive segment $w_C =$ $= 4.00$ mm. In Fig. 6, the error is depicted as a frequency dependent quantity.

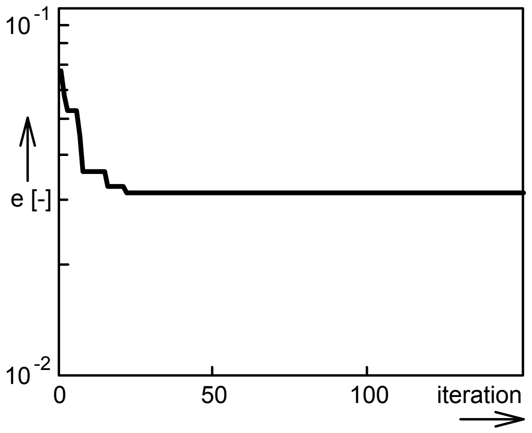


Fig. 7. Particle swarm optimization pre-training convergence for the refined feed-forward neural network configuration.

Since the approximation error is unacceptably high (up to 14 %), a refined model is going to be trained. Therefore, a new training set has to be prepared, which contains more training patterns (441 compared to 60 in the initial training set). The width of inductive segments $w_L \in$ <0.05, 0.65> mm is sampled now with the step Δw_L = 0.10 mm (compared to Δw_L = 0.20 mm in the initial training), and the width of capacitive segments $w_C \in$ <2.00, 8.00> mm is sampled now with the step Δw_C = 1.00 mm (compared to Δw_C = 2.00 mm in the initial training). The frequency $f \in$ <1.0, 5.0> GHz is sampled with the step Δf = 0.5 GHz (compared to Δf = 1.0 GHz in the initial training).

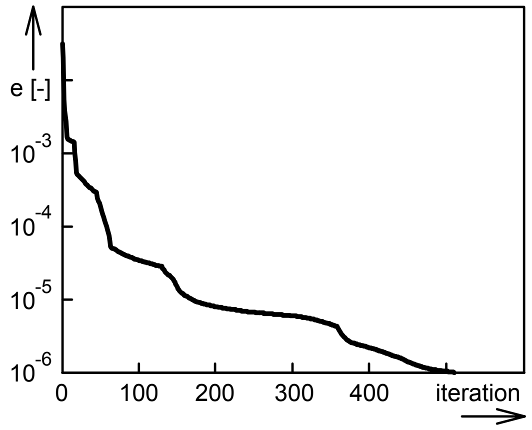


Fig. 8. Levenberg-Marquardt refinement training convergence for the refined feed-forward neural network configuration

Applying Bayesian regularization, the optimal number of hidden neurons is estimated to ten in two hidden layers each.

The pre-training error after 150 iterations is $e = 3.14 \cdot 10^{-2}$ with the CPU time consumption t = 19.1 s (Fig. 7). Training of the neural model is stopped when the training error drops bellow 10^{-6} after 509 iterations, which takes 54.5 s (Fig. 8).

Error of the new neural model over the refined testing set reaches maximum values at w_L = 0.02 mm and w_C = 8.00 mm. The highest error is lower than 3 % (Fig. 9).

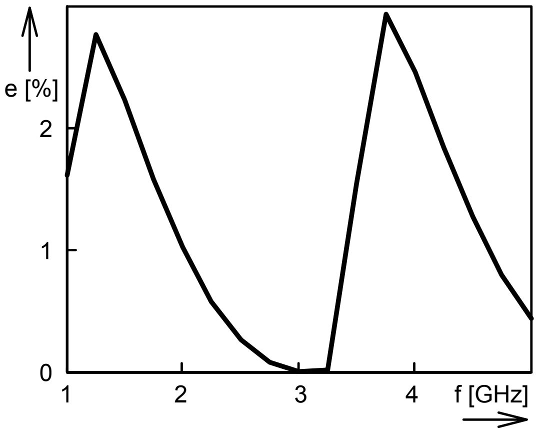


Fig. 9. The maximum approximation error of the neural model for w_L = 0.05 mm, and w_C = 8.00 mm (the worst case) for the refined feed-forward neural network configuration.

We can conclude that the refined neural model of the stepped-impedance three-pole low-pass filter can fully replace the numeric one – the maximum error below 3 % is comparable to the accuracy of numerical models and measurements.

C. Recurrent Neural Model

Recurrent neural networks are characteristic by feedbacks in their structure (see Fig. 10). Due to these feedbacks, the recurrent neural network is able to map an input sequence into an output one. In case of modelling a stepped-impedance three-pole low-pass filter with Chebychev response, doublets of widths w_L and w_C can play the role of input patterns, and a sequence of filter forward gains on prescribed frequencies s_{21} is a target sequence.

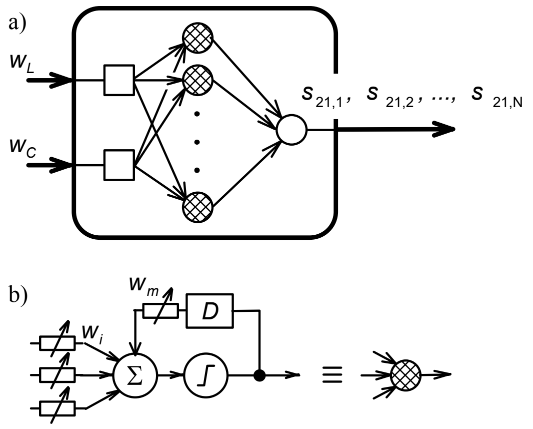


Fig. 10. a) Architecture of a recurrent neural network for modelling a stepped-impedance three-pole low-pass filter with Chebychev response. The input layer consists of input neurons (squares). In a hidden layer, adaptive non-linear feedback neurons are used (hatched circles). The output layer consists of usual adaptive non-linear neurons. b) Structure of an adaptive non-linear feedback neuron (D denotes a delay block).

Structure of the recurrent model of the filter is depicted in Fig. 10. The input layer consists of two neurons (the input for frequency was removed); the output layer is identical with the output of the feed-forward neural network. Dealing with hidden layers, recurrent networks contain a single hidden layer of feedback neurons.

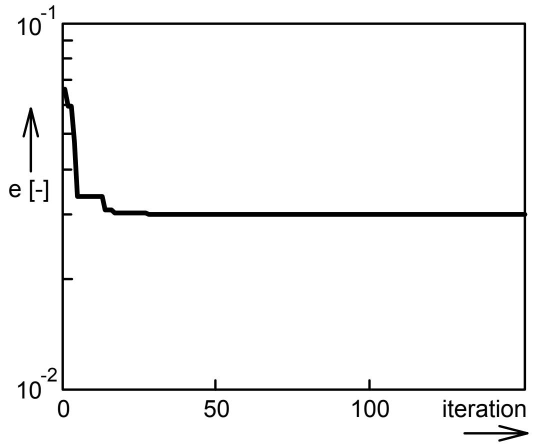


Fig. 11. Particle swarm optimization pre-training convergence for the initial recurrent neural network configuration.

The neural model reacts on an input doublet recurrently, and provides a sequence of values of the filter forward gain s_{21} (a series of forward gains on the frequencies of interest). This is the most important difference between the feed-forward neural model and the recurrent one. Otherwise, the methodology of building a recurrent neural model of the filter stays unchanged:

1. A coarse training set for the initial training of the recurrent network is similar compared to the training set used for training the feed-forward model. We only complete the input patterns by zeros in time to obtain input sequences, and we divide a single training iteration to several periods. In a period, the network is trained using 5 training patterns (5 frequencies) corresponding to a single doublet [w_L, w_C]. In this sense, we can consider the training set consisting of 60 patterns (identical with the coarse training set of the feed-forward neural network).

2. Respecting the size of the training set, the number of hidden neurons in the recurrent layer has to be estimated in order to prepare large enough distributed memory for saving the information. Applying Bayesian regularization, six neurons in the recurrent layer seems to be optimal.

3. In order to move the state vector of the network towards the global minimum of the error function, the particle swarm pre-training is applied using the same parameters like in case of the feed-forward neural model. After pre-training, the mean squared error reaches the value $e = 0.030$ in 68.90 seconds. The time course of the pre-training process is depicted in Fig. 11.

4. When the state vector of the neural network is in the vicinity of the global minimum of the error function, Levenberg-Marquardt algorithm is applied to move the state vector to the global minimum.

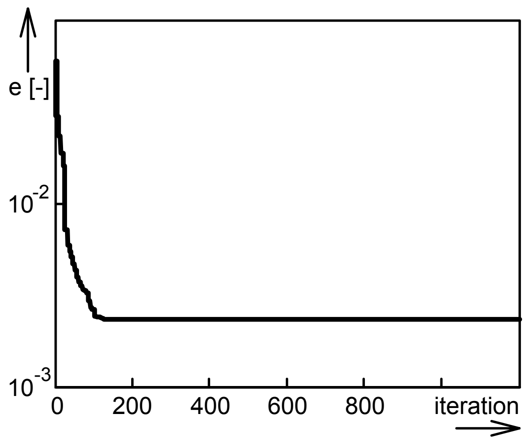


Fig. 12. Levenberg-Marquardt refinement training convergence for the initial recurrent neural network configuration.

Unfortunately, in case of the recurrent neural model, Levenberg-Marquardt algorithm was not able to reach the prescribed level of the error function $e_{pr} = 10^{-6}$, and stopped after 1200 iterations on the level $e = 2.30 \cdot 10^{-3}$. The time passed was $t = 26.09$ s. Time response of the error during training is depicted in Fig. 12.

The percentage error of the recurrent neural model over the testing set reaches the maximum value about 48 % for the worst case $w_L = 0.50$ mm and $w_C = 6.00$ mm (see Fig. 13).

Considering the feed-forward model trained over the initial, coarse training set, the maximum approximation error was more than three-times lower (14 %), pre-training consumed less than 10 % of the pre-training time of the recurrent model (5.30 s versus 68.90 s), and also the refinement consumed less than 10 % of the refinement time of the recurrent model (2.06 seconds versus 26.09 ones).

Now, we are going to investigate whether abilities of the recurrent network can be improved using a denser training set.

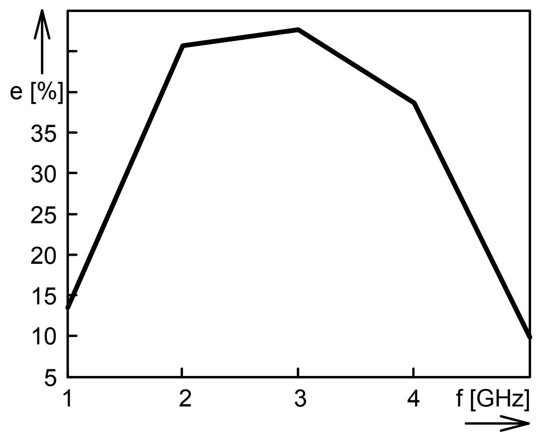


Fig. 13. The maximum approximation error of the neural model for $w_L = 0.50$ mm, and $w_C = 6.00$ mm (the worst case) for the initial recurrent neural network configuration.

A denser training set was identical with the set for training the feed-forward model: 441 combinations of the widths of inductive segments $w_L \in <0.05, 0.65>$ mm sampled with the step $\Delta w_L = 0.10$ mm, and the widths of capacitive segments $w_C \in <2.00, 8.00>$ mm sampled with the step $\Delta w_C = 1.00$ mm. The frequency $f \in <1.0, 5.0>$ GHz was sampled with the step $\Delta f = 0.5$ GHz (compared to $\Delta f = 1.0$ GHz in the initial training set).

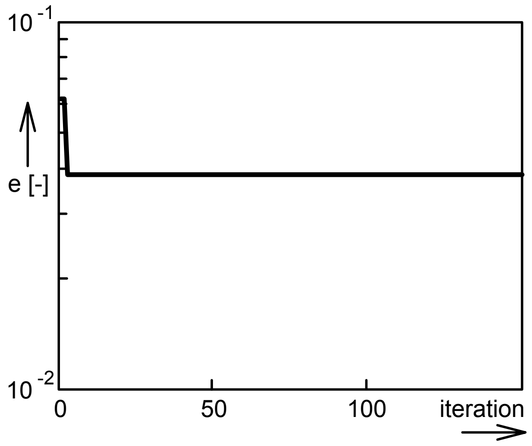


Fig. 14. Particle swarm optimization pre-training convergence for the refined recurrent neural network configuration.

A larger neural network contains 11 hidden neurons in a single layer. The model pre-training error after $t = 465$ s equals to $e = 3.84 \cdot 10^{-2}$ (Fig. 14). The training process is stopped after 1200 iterations ($t = 272.0$ s) without satisfactory results. The training error is $e = 1.59 \cdot 10^{-2}$ (Fig. 15).

The error of the refined recurrent model over the enriched testing set is still high, especially for w_L = 0.05 mm and w_C = = 5.50 mm. The highest value of the percentage error reaches up to 55 % at the frequency f = 5.0 GHz (see Fig. 16).

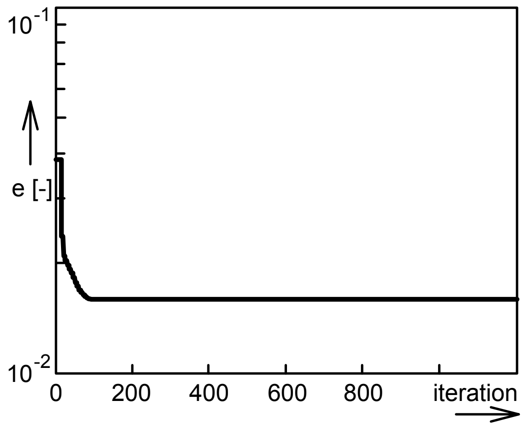


Fig. 15. Levenberg-Marquardt refinement training convergence for the refined recurrent neural network configuration.

Obviously, the enrichment of the training set caused further degradation of the recurrent neural model of the filter.

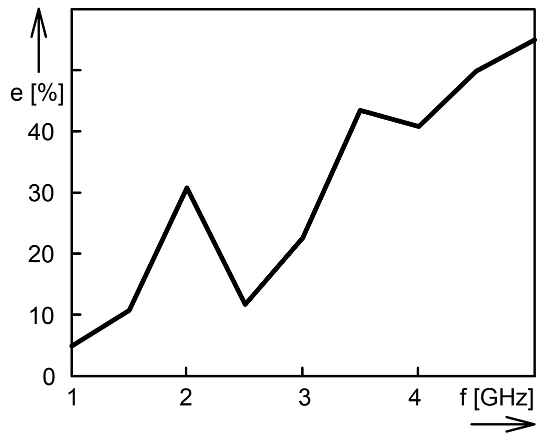


Fig. 16. The maximum approximation error of the neural model for w_L = 0.05 mm, and w_C = 8.00 mm (the worst case) for the refined recurrent neural network configuration.

We can therefore conclude that recurrent neural networks are not suitable for modelling electromagnetic structures in a relatively narrow band of frequencies.

III. OPTIMIZATION

In Section II, we discussed CPU-time demands of building neural models of microwave structures. Now, we are interested in CPU-time demands of the neural nets in the role of a model. The neural network is therefore associated with an optimization routine, and is used for the design of a filter.

The optimization is asked to find an inductive segment width w_L and a capacitive segment width w_C so that the filter can meet target values of the forward gain at f_1 = 1 GHz: $s_{21}(f_1)$ = –0.1 dB, at f_2 = 2 GHz: $s_{21}(f_2)$ = –10 dB, and at f_3 = = 3 GHz: $s_{21}(f_3)$ = –15 dB.

The objective function is minimized by particle swarm method with M = 30 particles and absorbing walls on the boundaries. Due to the random initial values of the inductive segment width w_L and the capacitive segment width w_C, each training process is run five times, and the learning error is averaged. Therefore, the length of optimization is fixed (80 iterations).

The convergence of optimization exploiting the feed-forward neural model of the planar filter is depicted in Fig. 17. There are three curves in the figure: the time course of criteria function of the best run of the optimization process (dotted line), the worst run (dashed line), and the average run (solid line). The best, the worst and the averaged cases are sorted according to the criteria function value after the last iteration.

After 80 iterations, the value of the criteria function stays constant on the level $e = 2.5 \cdot 10^{-3}$. This corresponds with the inductive segment width w_L = 0.10 mm and the capacitive segment one w_C = 3.91 mm. The optimal widths of segments are found in 0.062 s of the CPU time.

The optimization was asked to yield the filter forward gains s_{21} = {–0.1, –10.0, –15.0} dB on frequencies f = {1.0, 2.0, 3.0} GHz. Analyzing the optimized filter values of the s_{21} parameter, the reached values of the filter forward gain are s_{21} = = {–0.57, –9.67, –13.63} dB on respective frequencies.

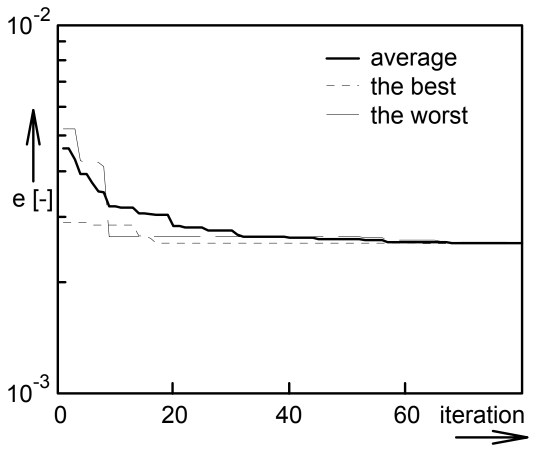


Fig. 17. Convergence of the particle swarm optimization of the low-pass planar filter exploiting feed-forward refined neural model.

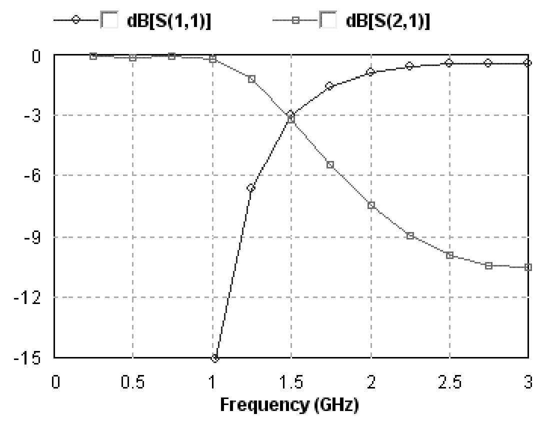
In order to verify the neural design, we use Zeland IE3D in order to perform the full-wave analysis of the original filter (w_L =0.20 mm, w_C = 4.00 mm), and the optimized one (w_L = = 0.10 mm, w_C = 3.91 mm).

The obtained filter characteristics are depicted in Fig. 18. Obviously, the neural design provides quite good results.

Due to the unsuccessful training of the recurrent model, the optimization based on that model will provide wrong results: we therefore do not use this model in conjunction with the particle swarm optimization.

In order to demonstrate the strength of the neural design, we repeat the whole optimization using Zeland IE3D to evaluate the criteria function in each iteration step of the optimization. The convergence of the full-wave design is depicted in Fig. 19, and the obtained optimal widths are w_L = 0.10 mm, w_C = 3.94 mm, compared to initial ones w_L =0.20 mm, w_C = 4.00 mm, and neural ones w_L =0.10 mm, w_C = 3.91 mm.

a)



b)

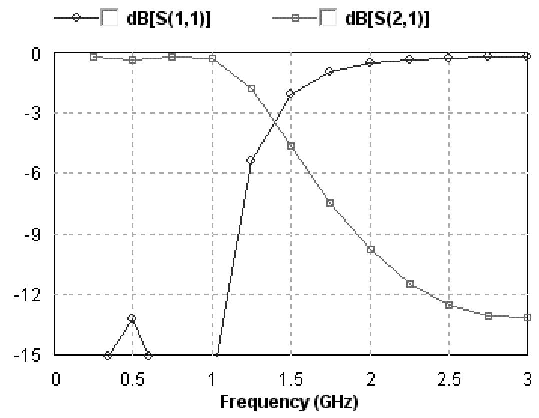


Fig. 18. Frequency response of the reflection coefficient at the filter input s_{11}, and the filter forward gain s_{21}. a) The original filter. b) The optimized filter (particle swarm optimization plus the refined feed-forward neural model). Computed by Zeland IE3D.

After 80 iterations, the value of the criteria function stays constant on the level $e = 1.6 \cdot 10^{-3}$. In case of the full-wave optimization, the CPU time consumptions were $t = 5\,597$ s.

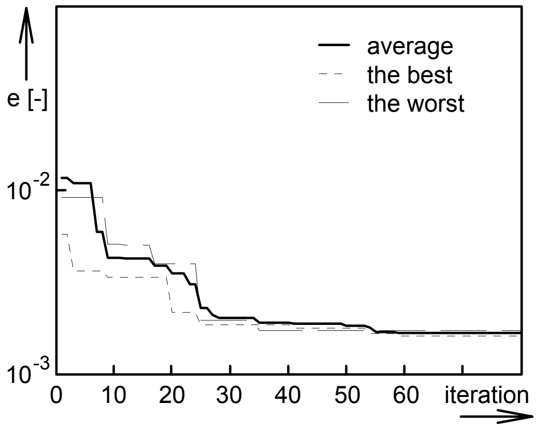


Fig. 19. Convergence of the particle swarm optimization of the low-pass planar filter exploiting full-wave numeric model (Zeland IE3D).

The optimization was asked to yield the filter forward gains s_{21} = {–0.1, –10.0, –15.0} dB on frequencies f = {1.0, 2.0, 3.0} GHz. Analyzing the optimized filter values of the s_{21} parameter, the reached values of the filter forward gain are s_{21} = = {–0.33, –9.81, –13.163} dB on respective frequencies. Obviously, the numeric result is quite close to the neural one.

We can therefore conclude that the refined feed-forward neural model of the filter can successfully replace the full-wave numeric model of the filter in the global optimization procedure.

Whereas the neural design consumed 0.062 s of the CPU time during the global optimization, the full-wave design required 5 597 s for the same purpose.

IV. Conclusions

Comparing the results, the feed-forward neural model trained by the particle swarm method and Levenberg-Marquardt algorithm is the fastest and sufficiently accurate approach to model a low-pass planar filter specified above. Using the neural model in the global optimization procedure, the CPU-time consumed by the numerical model of the filter is more than 90 thousand times demanding compared to the neural one.

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

The recurrent model was not able to be trained with the sufficient accuracy. We therefore concluded that such a type of neural networks is not suitable for this kind of modelling tasks.

TABLE I
Optimization Results

		original	feed-forw.	IE3D	
optimized variables	w_L [mm]	0.20	0.10	0.10	
	w_C [mm]	0.40	3.91	3.95	
time [s]	training	-	73.6	-	
	otimizing	-	0.062	5597	target
1 GHz	s_{21} [dB]	-0.23	-0.32	-0.33	-0.10
2 GHz	s_{21} [dB]	-7.47	-9.71	-9.83	-10.00
3 GHz	s_{21} [dB]	-10.51	-13.12	-13.17	-15.00

Acknowledgement

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] V. Akan and E. Yazgan, “Quasi-static solutions of multilayer elliptical, cylindrical coplanar striplines and multilayer coplanar striplines with finite dielectric dimensions – asymmetrical case”, *IEEE Transactions on Microwave Theory and Techniques*, vol. 53, no. 12, pp. 3681–3686, 2005.

[2] P. Pramanick, and P. Bhartia, “A new model for the apparent characteristic impedance of finned waveguide and finlines”, *IEEE Transactions on Microwave Theory and Techniques*, vol. 34, no. 12, pp. 1437–1441, 1986.

[3] J. Perini, “Periodically loaded transmission lines”, *IEEE Transactions on Microwave Theory and Techniques*, vol. 28, no. 9, pp. 1029–1031, 1980.

[4] M. Isaksson, D. Wisell, and D. Ronnow, “Wide-band dynamic modeling of power amplifiers using radial-basis function neural networks”, *IEEE Transactions on Microwave Theory and Techniques*, vol. 53, no. 11, pp. 3422–3428, 2005.

[5] Z. Raida, “Modeling EM structures in Neural network toolbox of Matlab”, *IEEE Antennas and Propagation Magazine*, vol. 44, no. 6, pp. 46–67, 2002.

[6] G. L. Creech, B. J. Paul, C. D. Lesniak, T. J. Jenkins, and M. C. Calcatera, “Artificial neural networks for fast and accurate EM-CAD of microwave circuits”, *IEEE Transactions on Microwave Theory and Techniques*, vol. 45, no. 5, pp. 794–802, 1997.

[7] S. Haykin, *Neural networks: A comprehensive foundation*, Englewood Cliffs, Macmillan Publishing Company, 1994.

[8] A. Cichocki, and R. Unbehauen, *Neural networks for optimization and signal processing*, Chichester, J. Wiley & Sons, 1994.

[9] H. Demuth and M. Beale, *Neural Network Toolbox for Use with Matlab: User's Guide* (version 4), Natick, The MathWorks Inc., 2000.

[10] J. Robinson, and Y. Rahmat-Samii, Particle swarm optimization in electromagnetics. *IEEE Transactions on Antennas and Propagation*, vol. 52, no. 2, pp. 397–407, 2004.

[11] P. E. Gill, W. Murray, M. H. Wright, *Practical Optimization*, London, Academic Press, 1981.