Neural Inverse Space Mapping EM-Optimization

J.W. Bandler, M.A. Ismail, J.E. Rayas-Sánchez and Q.J. Zhang

McMaster University, Hamilton, ON, Canada L8S 4K1, www.sos.mcmaster.ca

***Abstract* — For the first time, we present Neural Inverse Space Mapping (NISM) optimization for EM-based design of microwave structures. The inverse of the mapping from the fine to the coarse model parameter spaces is exploited for the first time in a Space Mapping algorithm. NISM optimization does not require: up-front EM simulations, multipoint parameter extraction or frequency mapping. The inverse of the mapping is approximated by a neural network whose generalization performance is controlled through a network growing strategy. We contrast our new algorithm with Neural Space Mapping (NSM) optimization.**

I. INTRODUCTION

An elegant new algorithm for EM-based design of microwave circuits is presented for the first time: Neural Inverse Space Mapping (NISM) optimization. This is the first Space Mapping (SM) algorithm that explicitly makes use of the inverse of the mapping from the fine to the coarse model parameter spaces.

NISM follows an aggressive formulation by not requiring a number of up-front fine model evaluations to start building the mapping. An innovative yet simple procedure for parameter extraction avoids the need of multipoint matching and frequency mappings. A neural network whose generalization performance is controlled through a network growing strategy approximates the inverse of the mapping at each iteration. NISM step simply evaluates the current neural network at the optimal coarse solution. This step is equivalent to a quasi-Newton step while the inverse mapping remains essentially linear.

We compare our new algorithm with Neural Space Mapping (NSM) optimization [1-2] by solving the same microwave design problem: an HTS microstrip filter.

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada under Grants OGP0007239 and STR234854-00, and through the Micronet Network of Centres of Excellence. M.A. Ismail is supported by a Nortel Networks Ontario Graduate Scholarship in Science and Technology. J.E. Rayas-Sánchez is funded by CONACYT (Consejo Nacional de Ciencia y Tecnología, Mexico), as well as by ITESO (Instituto Tecnológico y de Estudios Superiores de Occidente, Mexico).

J.W. Bandler is also with Bandler Corporation, P.O. Box 8083, Dundas, Ontario, Canada L9H 5E7.

Q.J. Zhang is with the Department of Electronics, Carleton University, 1125 Colonel By Drive, Ottawa, Canada K1S 5B6.

II. NEURAL INVERSE SPACE MAPPING (NISM)

A. Notation

Let the vectors $\boldsymbol{x}_c$ and $\boldsymbol{x}_f$ represent the design parameters of the coarse and fine models, respectively ($\boldsymbol{x}_c$, $\boldsymbol{x}_f \in \Re^n$). We denote the optimizable fine model responses at point $\boldsymbol{x}_f$ and frequency ω by $\boldsymbol{R}_f(\boldsymbol{x}_f, \omega) \in \Re^r$ where r is the number of responses to be optimized. The vector $\boldsymbol{R}_f(\boldsymbol{x}_f) \in \Re^m$ denotes the fine model responses at the F_p simulation frequencies, where $m = rF_p$. Similarly, $\boldsymbol{R}_c(\boldsymbol{x}_c) \in \Re^m$ denotes the corresponding coarse model responses to be optimized.

Additionally, we denote the characterizing fine model responses at point $\boldsymbol{x}_f \in \Re^n$ and frequency ω by $\boldsymbol{R}_{fs}(\boldsymbol{x}_f, \omega) \in \Re^R$, which includes the real and imaginary parts of all the available characterizing responses in the model (considering symmetry). For example, for a 2-port reciprocal network they include $\mathrm{Re}\{S_{11}\}$, $\mathrm{Im}\{S_{11}\}$, $\mathrm{Re}\{S_{21}\}$ and $\mathrm{Im}\{S_{21}\}$, and $R = 4$. The vector $\boldsymbol{R}_{fs}(\boldsymbol{x}_f) \in \Re^M$ denotes the characterizing responses at all the F_p frequency points, where $M = RF_p$. Similarly, $\boldsymbol{R}_{cs}(\boldsymbol{x}_c) \in \Re^M$ denotes the corresponding characterizing coarse model responses.

B. Flow Diagram: An Overview

Fig. 1 shows a flow diagram for NISM optimization. We start by performing regular minimax optimization on the coarse model to find the optimal coarse solution $\boldsymbol{x}_c^*$ that yields the desired response. The characterizing fine model responses $\boldsymbol{R}_{fs}$ at the optimal coarse solution $\boldsymbol{x}_c^*$ are then calculated.

We realize parameter extraction, which consists of finding the coarse model parameters that makes the characterizing coarse responses $\boldsymbol{R}_{cs}$ as close as possible to the previously calculated $\boldsymbol{R}_{fs}$.

We continue by training the simplest neural network $\boldsymbol{N}$ that implements the inverse of the mapping from the fine to the coarse parameter space at the available points.

The new point in the fine model parameter space is then calculated by simply evaluating the neural network at the optimal coarse solution. If the maximum relative change in the fine model parameters is smaller than a previously defined amount we finish, otherwise we calculate the characterizing fine model responses at the new point and continue with the algorithm.

C. Parameter Extraction

The parameter extraction procedure at the ith NISM iteration is formulated as the following optimization problem

$$\boldsymbol{x}_c^{(i)} = \arg\min_{\boldsymbol{x}_c} U_{PE}(\boldsymbol{x}_c) \tag{1a}$$

$$U_{PE}(\boldsymbol{x}_c) = \left\| \boldsymbol{e}(\boldsymbol{x}_c) \right\|_2^2 \tag{1b}$$

$$\boldsymbol{e}(\boldsymbol{x}_c) = \boldsymbol{R}_{fs}(\boldsymbol{x}_f^{(i)}) - \boldsymbol{R}_{cs}(\boldsymbol{x}_c) \tag{1c}$$

We solve (1) using the Levenberg-Marquardt algorithm for nonlinear curve fitting available in the Matlab™ Optimization Toolbox [3].

We normally use $\boldsymbol{x}_c^*$ as the starting point for solving (1). This might not be a good starting point when an extremely severe matching problem is being solved, one that has some poor local minimum around $\boldsymbol{x}_c^*$. If the algorithm is trapped in a poor local minimum, we change the starting point for (1) by taking a small random perturbation $\Delta\boldsymbol{x}$ around $\boldsymbol{x}_c^*$ until we find an acceptable local minimum, i.e., until we obtain a good matching between models.

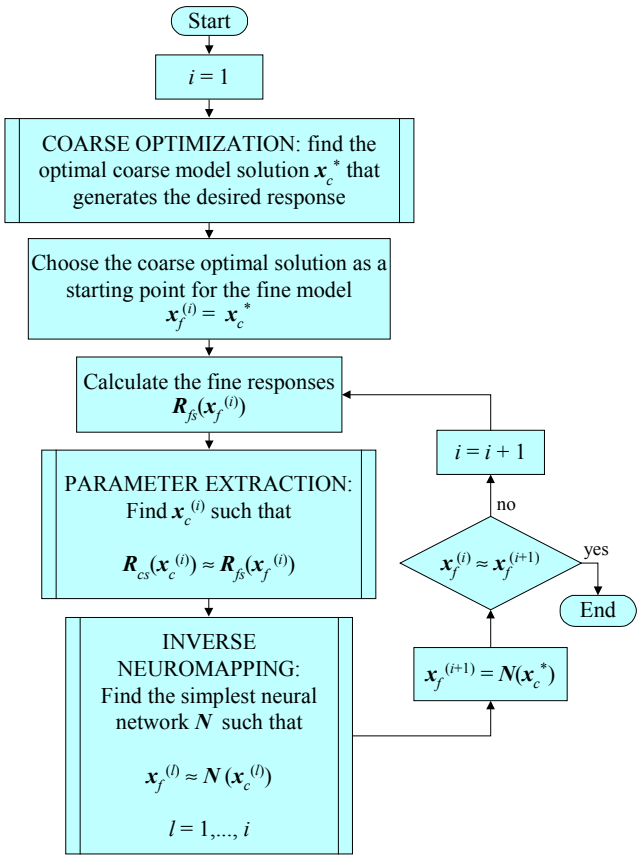


Fig. 1. Flow diagram of Neural Inverse Space Mapping (NISM) optimization.

The maximum perturbation $\Delta_{\max}$ is obtained from the maximum absolute sensitivity of the parameter extraction objective function at $\boldsymbol{x}_c^*$ as follows

$$\Delta_{\max} = \frac{\delta_{PE}}{\left\| \nabla U_{PE}(\boldsymbol{x}_c^*) \right\|_\infty} \tag{2}$$

Let $\boldsymbol{rand} \in \Re^n$ be a vector whose elements take random values between 0 and +1 every time it is evaluated. The values of the elements of $\Delta\boldsymbol{x}$ are calculated as

$$\Delta x_k = \Delta_{\max}(2rand_k - 1), \quad k = 1,\ldots,n \tag{3}$$

A value of δ_{PE} = 0.03 is used in our implementation. Many other values of δ_{PE} could be used in (3), since we use it only to escape from a poor local minimum.

A similar strategy for statistical parameter extraction was proposed in [4], where an exploration region is first created by predefining a fixed number of starting points around $\boldsymbol{x}_c^*$.

The algorithm for realizing parameter extraction is stated as follows

algorithm: Parameter Extraction
begin solve (1) using $\boldsymbol{x}_c^*$ as starting point while $\|\boldsymbol{e}(\boldsymbol{x}_c^{(i)})\|_\infty > \varepsilon_{PE}$ calculate $\Delta\boldsymbol{x}$ using (2) and (3) solve (1) using $\boldsymbol{x}_c^* + \Delta\boldsymbol{x}$ as starting point end

A value of ε_{PE} = 0.15 is used in our implementation, assuming that all the response values are normalized.

D. Inverse Neuromapping

When training the neural network $\boldsymbol{N}$ that implements the inverse mapping we solve the following optimization problem

$$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} U_N(\boldsymbol{w}) \tag{4a}$$

$$U_N(\boldsymbol{w}) = \left\| [\cdots \quad \boldsymbol{e}_l^T \quad \cdots]^T \right\|_2^2 \tag{4b}$$

$$\boldsymbol{e}_l = \boldsymbol{x}_f^{(l)} - \boldsymbol{N}(\boldsymbol{x}_c^{(l)}, \boldsymbol{w}), \quad l = 1,\ldots,i \tag{4c}$$

where i is the current NISM iteration and vector $\boldsymbol{w}$ contains the internal parameters (weights, bias, etc.) of the neural network $\boldsymbol{N}$. The starting point $\boldsymbol{w}^{(0)}$ for solving (4) is a unit mapping, i.e., $\boldsymbol{N}(\boldsymbol{x}_c^{(l)}, \boldsymbol{w}^{(0)}) = \boldsymbol{x}_c^{(l)}$, for $l = 1,\ldots, i$. We use the Scaled Conjugate Gradient (SCG) algorithm available in the Matlab™ Neural Network Toolbox [5] for solving (4).

To control the generalization performance of the neural

network $\boldsymbol{N}$, we follow a network growing strategy [6], in which case we start with a small perceptron to match the initial points and then add more neurons only when we are unable to meet a small error.

We initially assume a 2-layer perceptron given by

$$\boldsymbol{N}(\boldsymbol{x}_c, \boldsymbol{w}) = \boldsymbol{x}_f = \boldsymbol{W}^o \boldsymbol{x}_c + \boldsymbol{b}^o \tag{5}$$

where $\boldsymbol{W}^o \in \Re^{n \times n}$ is the matrix of output weighting factors, $\boldsymbol{b}^o \in \Re^n$ is the vector of output bias elements, and vector $\boldsymbol{w}$ contains $\boldsymbol{b}^o$ and the columns of $\boldsymbol{W}^o$. The starting point is obtained by making $\boldsymbol{W}^o = \boldsymbol{I}$ and $\boldsymbol{b}^o = \boldsymbol{0}$.

If a 2-layer perceptron is not sufficient to make the learning error $U_N(\boldsymbol{w}^*)$ small enough, then we use a 3-layer perceptron with h hidden neurons given by

$$\boldsymbol{N}(\boldsymbol{x}_c, \boldsymbol{w}) = \boldsymbol{W}^o \boldsymbol{\Phi}(\boldsymbol{x}_c) + \boldsymbol{b}^o \tag{6a}$$

$$\boldsymbol{\Phi}(\boldsymbol{x}_c) = [\varphi(s_1) \quad \varphi(s_2) \quad \ldots \quad \varphi(s_h)]^T \tag{6b}$$

$$\boldsymbol{s} = \boldsymbol{W}^h \boldsymbol{x}_c + \boldsymbol{b}^h \tag{6c}$$

where $\boldsymbol{W}^o \in \Re^{n \times h}$, $\boldsymbol{b}^o \in \Re^n$, $\boldsymbol{\Phi}(\boldsymbol{x}_c) \in \Re^h$ is the vector of hidden signals, $\boldsymbol{s} \in \Re^h$ is the vector of activation potentials, $\boldsymbol{W}^h \in \Re^{h \times n}$ is the matrix of hidden weighting factors, $\boldsymbol{b}^h \in \Re^h$ is the vector of hidden bias elements and h is the number of hidden neurons. In this work we use hyperbolic tangents as nonlinear activation functions, i.e., $\varphi(\cdot) = \tanh(\cdot)$. Vector $\boldsymbol{w}$ contains $\boldsymbol{b}^o$, $\boldsymbol{b}^h$, the columns of $\boldsymbol{W}^o$ and the columns of $\boldsymbol{W}^h$.

Our starting point for solving (4) using (6) is also a unit mapping, which is obtained by making $\boldsymbol{b}^o = \boldsymbol{0}$, $\boldsymbol{b}^h = \boldsymbol{0}$, $\boldsymbol{W}^h = 0.1[\boldsymbol{I}\ \boldsymbol{0}]^T$ and $\boldsymbol{W}^o = 10[\boldsymbol{I}\ \boldsymbol{0}]$, assuming that the training data has been scaled between −1 and +1. Notice that we consider $h \geq n$.

The algorithm for finding the simplest inverse neuromapping is stated as follows

algorithm: Inverse Neuromapping
begin
solve (4) using (5)
$h = n$
while $U_N(\boldsymbol{w}^*) > \varepsilon_L$
solve (4) using (6)
$h = h+1$
end

In our implementation we use $\varepsilon_L = 1 \times 10^{-4}$. Notice that the algorithm for finding the inverse neuromapping uses a 2-layer perceptron during at least the first n+1 NISM iterations, since the points $(\boldsymbol{x}_c^{(i)}, \boldsymbol{x}_f^{(i)})$ can be mapped with a linear mapping for $i = 1 \ldots n+1$. A 3-layer perceptron is needed only when we exceed n+1 NISM iterations and the mapping is significantly nonlinear.

E. Nature of the NISM step

We can prove that the NISM step, $\boldsymbol{x}_f^{(i+1)} = \boldsymbol{N}(\boldsymbol{x}_c^*)$, is equivalent to a quasi-Newton step while the inverse mapping built during NISM optimization remains linear, i.e., while a 2-layer perceptron is enough to approximate the inverse mapping. We can also prove that the NISM step gradually departs from a quasi-Newton step as the amount of nonlinearity needed in the inverse mapping increases. Both proofs are omitted in this paper due to the limitations in space.

III. EXAMPLE

We apply NISM optimization to a high-temperature superconducting (HTS) quarter-wave parallel coupled-line microstrip filter, and contrast our results with those obtained by using NSM optimization on the same problem [1-2].

L_1, L_2 and L_3 are the lengths of the parallel coupled-line sections and S_1, S_2 and S_3 are the gaps between the sections. The width W is the same for all the sections as well as for the input and output lines, of length L_0. A substrate with thickness H and dielectric constant ε_r is used.

The design parameters are $\boldsymbol{x}_f = [L_1\ L_2\ L_3\ S_1\ S_2\ S_3]^T$. The specifications as well as the values of L_0, H, W, ε_r, loss tangent and metalization are taken as in [1-2].

Sonnet's ***em***™ [7] is employed as the fine model, using a high-resolution grid. As the coarse model we use sections of OSA90/hope™ [8] built-in microstrip lines, two-coupled microstrip lines and open circuits connected by circuit theory over the same substrate.

We use the same optimal coarse model solution $\boldsymbol{x}_c^*$ as in [1-2]. The coarse and fine model responses at the optimal coarse solution are shown in Fig. 2.

After only 3 fine model simulations the optimal NISM solution was found, as shown in Fig. 3.

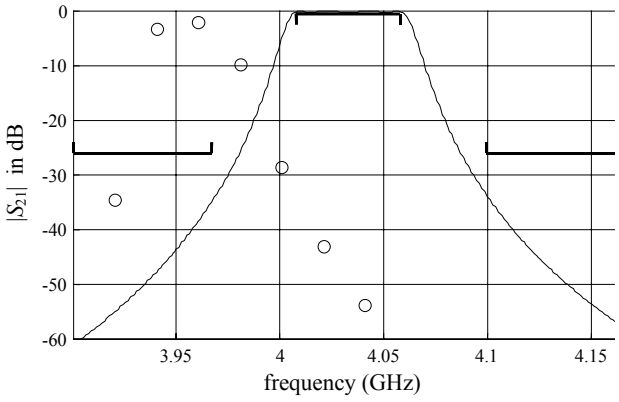
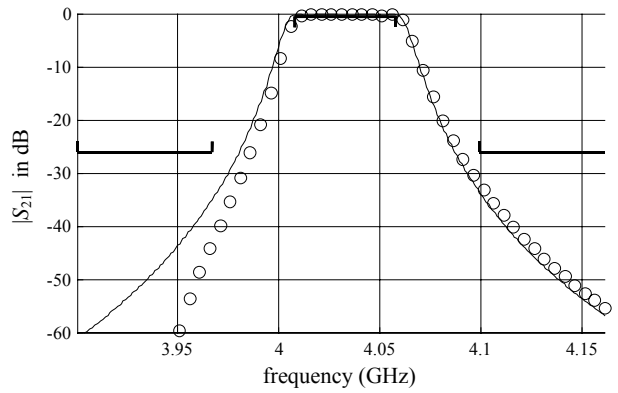
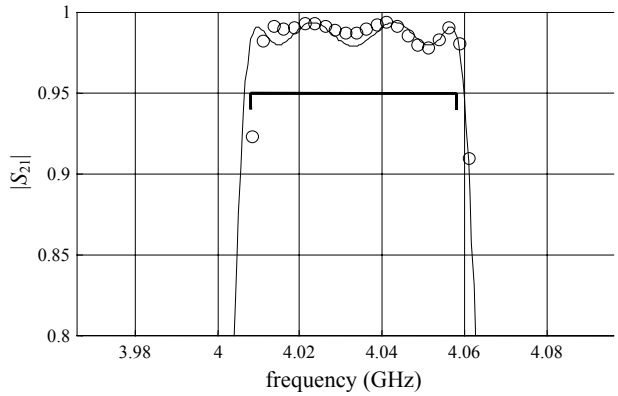


Fig. 2. Coarse model response (–) and fine model response (○) at $\boldsymbol{x}_c^*$.



(a)



(b)

Fig. 3. Coarse model response at $\boldsymbol{x}_c^*$ (–) and fine model response at $\boldsymbol{x}_f^{NISM}$ (○): (a) at all the frequencies of interest, (b) at the passband.

Fig. 4 shows the results obtained using NSM optimization [2], where the optimal NSM solution was found after 14 fine model evaluations.

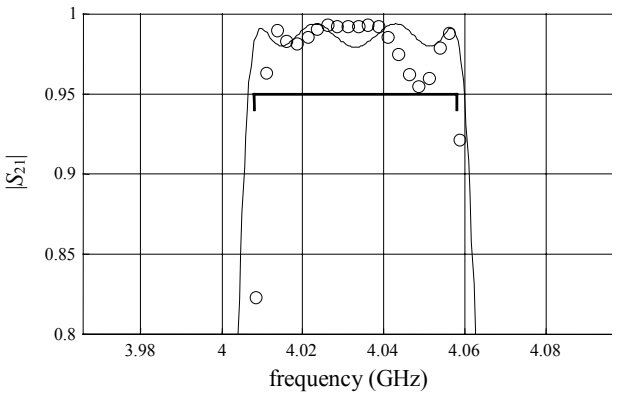


Fig. 4. Coarse model response at $\boldsymbol{x}_c^*$ (–) and fine mode response at $\boldsymbol{x}_f^{NSM}$ (○) at the passband.

It is seen that NISM optimization is not only more efficient in terms of the required fine model evaluations, but also yields a solution closer to the optimal solution of the original optimization problem (compare Figs. 3b and 4). We have arrived at similar conclusions in other microwave design problems where NISM and NSM were applied (2-section impedance transformer, band-stop microstrip filter with quarter-wave open stubs, etc.).

IV. Conclusions

We propose Neural Inverse Space Mapping (NISM) optimization for EM-based design of microwave structures. The inverse of the mapping is exploited for the first time in a space mapping algorithm. NISM optimization does not require: up-front EM simulations, multipoint parameter extraction nor frequency mapping. Our new algorithm exhibits superior performance than Neural Space Mapping (NSM) optimization.

Acknowledgement

The authors thank Dr. J.C. Rautio, President, Sonnet Software, Inc., Liverpool, NY, for making ***em***™ available.

References

[1] M.H. Bakr, J.W. Bandler, M.A. Ismail, J.E. Rayas-Sánchez and Q.J. Zhang, "Neural space mapping optimization of EM microwave structures," *IEEE MTT-S Int. Microwave Symp. Digest* (Boston, MA), 2000, pp. 879-882.

[2] M.H. Bakr, J.W. Bandler, M.A. Ismail, J.E. Rayas-Sánchez and Q.J. Zhang, "Neural space mapping optimization for EM-based design," *IEEE Trans. Microwave Theory Tech*., vol. 48, 2000, pp. 2307-2315.

[3] Matlab™ Optimization Toolbox, Version 2, The MathWorks, Inc., 3 Apple Hill Drive, Natick MA 01760-2098, 1999.

[4] J.W. Bandler, R.M. Biernacki, S.H. Chen and D. Omeragić, "Space mapping optimization of waveguide filters using finite element and mode-matching electromagnetic simulators," *IEEE MTT-S Int. Microwave Symp. Dig.* (Denver, CO), vol. II, 1997, pp. 635-638.

[5] Matlab™ Neural Network Toolbox, Version 3, The MathWorks, Inc., 3 Apple Hill Drive, Natick MA 01760-2098, 1998.

[6] S. Haykin, *Neural Networks: A Comprehensive Foundation*. New Jersey, MA: Prentice Hall, 1999.

[7] ***em***™ Version 4.0b, Sonnet Software, Inc., 1020 Seventh North Street, Suite 210, Liverpool, NY 13088, 1997.

[8] OSA90/hope™ Version 4.0, formerly Optimization Systems Associates Inc., P.O. Box 8083, Dundas, Ontario, Canada L9H 5E7, 1997, now Agilent EEsof EDA, 1400 Fountaingrove Parkway, Santa Rosa, CA 95403-1799.