Radial-Basis-Function Neural Network Optimization of Microwave Systems

by

Ethan K. Murphy

A Master's Project

Submitted to the Faculty

of

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Master of Science

in

Industrial Mathematics

by

____

December 2002

APPROVED

____

Dr. Vadim V. Yakovlev, Project Advisor

____

Dr. Homer Walker, Department Head

Abstract

An original approach in microwave optimization, namely, a neural network procedure combined with the full-wave 3D electromagnetic simulator *QuickWave-3D* implemented a conformal FDTD method, is presented. The radial-basis-function network is trained by simulated frequency characteristics of *S*-parameters and geometric data of the corresponding system. High accuracy and computational efficiency of the procedure is illustrated for a waveguide bend, waveguide T-junction with a post, and a slotted waveguide as a radiating element.

Acknowledgements

- Vadim Yakovlev
- My mother and father
- My brother who is lost in Africa with the Peace Corp
- Veronika Mechenova

List of Figures

Fig 2.1. *S*-parameter representation of a 2-port system.

Fig 2.2. Conventional frequency characteristic of the magnitude of S_{11}.

Fig 2.3. Model of a single neuron.

Fig 3.1. Layers of Neural Network

Fig 3.2. Linear function.

Fig 3.3. Hyperbolic Tangent function.

Fig 3.4. Radial basis neuron.

Fig 3.5. The Gaussian radial basis function.

Fig 3.6. Conventional frequency characteristic of $|S_{11}|$ in the constrained optimization problem

Fig 4.1. Outline of the RBF-NN used in the algorithm.

Fig 4.2: Uniform grid of samples from the database for a 90^o waveguide

bend (see Chapter 6.1).

Fig 5.1: Flow chart of the algorithm for optimization of the reflection coefficient $|S_{11}|$

Fig 5.2. Principle steps in `Databasegetter` and `Matrixgetter`

Fig 6.1: Geometry of Project A

Fig 6.2. Geometry of Project B.

Fig 6.3. Geometry of Project C

Fig 6.4. RBF NN mean square error for varying scaling parameter $1 \le r \le 20$ (a) and $1.75 \le r \le 4.25$ (b) for Projects A (1), B (2), and C (3).

Fig 6.5. *QW3D*-generated values of $|S_{11}|$ for Project A

Fig 6.6. RBF NN results for $p = 3$ (Project A)

Fig 6.7. RBF NN results for $p = 4$ (Project A)

Fig 6.8. RBF NN results for $p = 5$ (Project A)

Fig 6.9. Absolute value of error in Project A for $p = 5$

Fig 6.10. Optimized $|S_{11}|$ frequency characteristics for Project A.

Fig 6.11. Optimized $|S_{11}|$ frequency characteristics for Project B.

Fig 6.12: Optimized $|S_{11}|$ frequency characteristics for Project C.

Fig 6.13. Optimized frequency characteristics of $|S_{11}|$ in Project A for Trial 1. Thin curves: Methods A (marked by ∇), C (◊), and D (♦). Thick curves: NN technique for $p = 4$ and $p = 5$.

Fig 6.14. Optimized frequency characteristics of $|S_{11}|$ in Project A for Trial 2. Thin curves: Methods A (marked by ∇), and B (◊). Thick curves: NN technique for $p = 4$ and $p = 5$.

Fig 6.15. Optimized frequency characteristics of $|S_{11}|$ in Project A for Trial 3. Thin curves: Methods A (marked by ∇), B (◊), C (♦), and D (∨). Thick curves: NN technique for $p = 4$ and $p = 5$.

Fig 6.16. Optimized frequency characteristics of $|S_{11}|$ in Project A for Trial 4. Thin curves: Methods A (marked by ∇), B (◊), C (♦), and D (∨). Thick curves: NN technique for $p = 4$ and $p = 5$.

Contents

Chapter 1. Introduction **1**

Chapter 2. Background **5**

2.1 Electromagnetic Issues 5

2.2 Basics of Neural Networks 7

2.3 Neural Networks in Microwave Modeling 9

Chapter 3. Analysis **12**

3.1 Statement of the Problem 12

3.2 Feedforward MLP NN 14

3.3 RBF NN 17

3.4 Optimization Method 20

Chapter 4. Neural Model **23**

Chapter 5. Implementation **26**

5.1 Overview 26

5.2 Creation of the Database 27

5.3 Construction and Training of Radial Basis Network 29

5.2 Optimization and Comparison 31

Chapter 6. Illustrations **32**

6.1 MW Systems 32

6.2 Scaling 36

6.3 Accuracy 36

6.4 Optimization 41

6.5 Comparison with *QW3D* Optimizers 46

Chapter 7. Conclusions 50

Appendix 51

Bibliography 67

Chapter 1

Introduction

The modern trends towards production-oriented design and reduced time-to-market in the microwave (MW) industry require instruments assisting in accurate and fast design. Efforts to lower the cost and reduce the weight/volume of the circuits have caused a keen interest of electronic and microwave engineers in new efficient computer-aided design (CAD) tools.

Recent extraordinary growth of productivity and capabilities of computer hardware has made comprehensive, fast, accurate, and reliable numerical modeling of microwave circuits possible. Today, a number of pieces of modeling software allow one to get valuable data about the characteristics of the system prior to constructing a physical prototype. For example, *Microwave Studio* (*MWS*) [1], the electromagnetic (EM) code based on Finite Integration Method, and *Quick-Wave-3D*$^{\text{TM}}$ (*QW3D*) [2], the conformal FDTD 3D EM simulator, have been recently identified among the most efficient and proficient full-wave simulators in the market [3, 4].

However, a simple application of highly sophisticated computational tools for *analysis* of MW systems may not bring many direct recommendations for design implementation. Practical problems may be associated with specific *optimization goals*, which cannot be addressed with the use of the general tools in the software packages. This dictates the necessity of development of efficient optimization techniques for microwave modeling. Efficient computational procedures linked with advanced EM solvers should become powerful and flexible CAD tools revolutionizing the design of MW systems.

Several approaches based on the space mapping technique [5, 6] and a few other methods [7-9] form a group of modern advanced approaches to MW optimization. The techniques are applicable to a variety of microwave devices and demonstrate good performance in a number of practical situations. However, the extremely fast development of the MW industry encourages further research in this area towards resolution of many issues in accuracy, reliability, and computational resources.

One of the most important questions here comes up from the following. Some optimization techniques may work particularly well if joined with universal modeling software generating results of analysis of the MW structure. With the simulators applicable to a majority of systems and components in the microwave industry, such combinations could be highly universal instruments in the automated design. The emerging feasibility and practicality of inclusion of resourceful full-wave numerical simulators in optimization and automated design of MW structure time has been recently emphasized in [10].

In the meantime, so far, examples of optimization with involvement of full-wave modeling software are limited to just a few cases: e.g., *em*^{TM} by Sonnet Software [10] was used with the space mapping optimizations [5, 6, 10, 12], specifically to handle circuits containing complex subcircuits or components whose simulation requires significant computational effort. The approach proposed in [13] operates in connection with *HP-MDS*^{TM} [14]. It seems that many researchers are not concerned with generalization of their optimization schemes, but rather deal with the detailed physics-EM models and empirical approaches (see, e.g., [15, 16]). This can be explained by the fact that the inclusion of full-wave simulators in optimization and automated design has been traditionally considered unfeasible given the high cost in corresponding computational effort.

Meanwhile, it appears that with the current progress in computer hardware, packages like *MWS* and *QW3D* having expanded capabilities and characterized by high accuracy deserve a careful look at them as analysis tools backing an efficient MW optimization. Even if the "built-in" optimization options available in these simulators may appear to be general-purpose and slowly converging procedures characterized by heavy demand on computer resources, this still does not mean that, in case of a really efficient accompanying optimization technique, a truly competent solution cannot be obtained.

So far, the only known attempt to connect the advanced full-wave simulator with an efficient optimization procedure was made in [17], where the technique based on response surface methodology (RSM) and the Sequential Quadratic Programming (SQP) method for constrained optimization was implemented to run with *QW3D*. The concept behind this approach was generated by a condition of efficient operation of microwave heating systems: the method intentionally ignores possible resonance's of the response surface near the operating frequency. Being efficient for this particular class of MW devices, this approach has a drawback for others, which are frequently highly nonlinear, so that a quadratic function, as used in [17], could not always approximate a hypersurface with sufficient accuracy. The computer implementation of this method is still characterized by a notable CPU time.

To overcome the stated shortcomings, for the first time, the present paper proposes an efficient and simple optimization technique based on artificial neural networks (NN) made as a computational supplement for *QW3D*. We show that, given the resources of today's computers, such an approach can be reasonably productive and serve as a competent optimization tool in designing of various MW systems.

Chapter 2

Background

2.1 Electromagnetic Issues

Development of specialized efficient optimization algorithms for *QW3D* implemented the 3D conformal FDTD method requires dealing with many issues in numerical mathematics, programming, and computing. This project is focused on the relevant aspects in computational mathematics, but it needs some basic concept of microwave circuit analysis.

The concept of a scattering (*S*) matrix is one of the fundamental concepts of electromagnetics. It may be very convenient in analysis of characteristics of many electronic and communication devices as well as microwave circuits. Many important characteristics of MW system could be successfully described with the use of *S*-matrix terminology. Although it is applicable to any number of ports, in the illustration below we show a 2-port system for which two *S*-parameters can be introduced as follows:

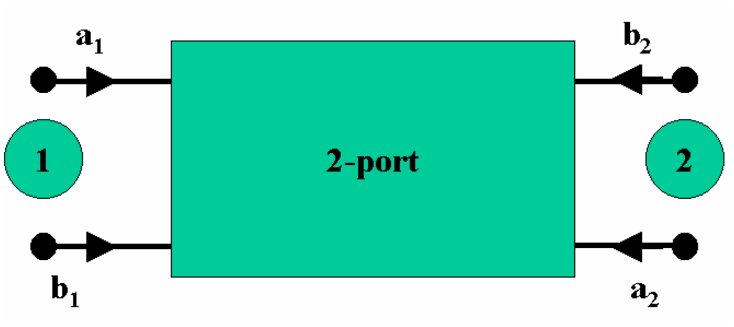


Figure 2.1. *S*-parameter representation of a 2-port system

$$s_{11} = \left.\frac{b_1}{a_1}\right|_{a2=0} \tag{2.1a}$$

$$s_{21} = \left.\frac{b_2}{a_1}\right|_{a2=0} \tag{2.1b}$$

where

$$a_n = \frac{V_n^+}{\sqrt{Z_o}}, \qquad b_n = \frac{V_n^-}{\sqrt{Z_o}} \tag{2.2}$$

Referring to Fig. 2.1, S_{11} is called the reflective coefficient. In accordance with (2.1a), if b_1 is equal to a_1, then the energy going in comes out, so $S_{11} = 1$, or, in other words, there would be 100%-reflection. S_{21} is the transmission coefficient describing the transmission of a field passing through the system and leaving it at Port 2.

As we see from an illustrative graph in Fig. 2.2, for every frequency, there is a distinct S_{11} coefficient. An important idea throughout the present study was that the microwave systems typically operate in some frequency ranges. Therefore, we are interested in the neighborhood of

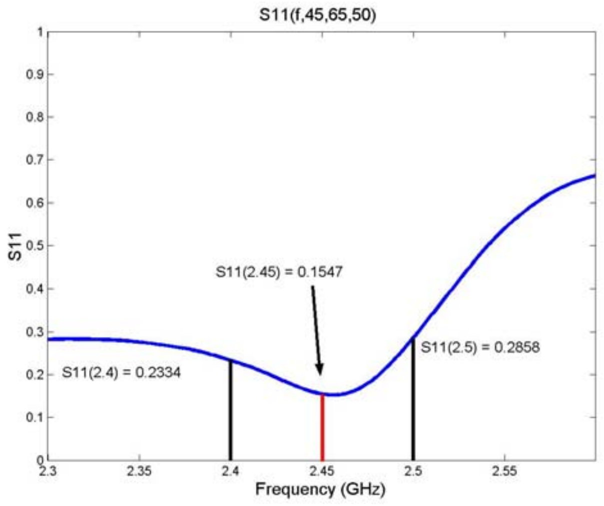


Figure 2.2. Conventional frequency characteristic of the magnitude of S_{11}.

an operating frequency f_0. This is illustrated by Fig. 2.2 showing f_0 at 2.45 GHz and the adjacent frequency range between $f_1 = 2.4$ and $f_2 = 2.5$ GHz, provided that $f_0 \in (f_1, f_2)$.

2.2 Basics of Neural Networks

As mentioned earlier this project utilizes NNs. An artificial neural network is a massively distributed parallel processor that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain since knowledge is acquired by the network through a learning process, and inter-neuron connection strengths are used to store the knowledge.

Inputs of the NN are given to the network and processed by simple processors (units, nodes, neurons) in parallel. Each processor holds a limited amount of memory. Unidirectional channels carry numerical data connecting these processors, in such a way that the NN can be viewed as a simple function mapping a set:

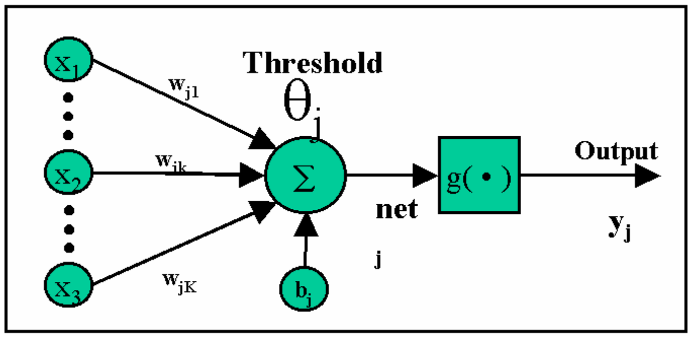


Figure 2.3. Model of a single neuron.

$$F(X) \rightarrow Y \tag{2.1}$$

A graphical representation of a neuron's model is shown in Fig. 2.3. There is an initial vector x given to the network which is multiplied by a weight matrix w_{jk} and added to a bias vector b_j, where j is the number of hidden neurons and k is the number of input neurons. The result of this is processed by g called a transfer function and finally arrives at the output of the layer.

Knowledge is programmed into the neural network by training runs. The neural network is trained by giving the NN a series of arguments x with known outputs y. Through a training algorithm weights and biases converge such that we have the following relation

$$F_n\left(x; w_{jk}^i, b_j^i\right) \rightarrow y \tag{2.2}$$

where n denotes the nth training iteration, and i represents the layer.

2.3 Neural Networks in Microwave Modeling

Neural networks are known as offering the ability of skillfully approximating highly nonlinear systems with a generally small amount of data and capable of competent solving problems of control and optimization. NNs were introduced into computational electromagnetics in the 1990s, and, since that time, their typical application has been associated with the networks representing (or directly imitating) the modeled devices and dealing with their physical/electrical characteristics. Sets of solution samples for these networks have been provided by physical/empirical models, or measured data. When developed appropriately, these models are convenient and accurate, but applicable only to the particular devices, so their usefulness is rather limited. When used with universal software, neural networks can be put in the background of an algorithm appropriately processing simulator's input/output data and generating the optimal solution for virtually any system to which the software is applicable.

QW3D is well suitable for building databases required for efficient operation of the NN-based procedures. This simulator is highly compatible with MATLAB, which seems to be a convenient environment for hosting NN algorithms. A master program could conveniently control operations of the entire computational structure.

Knowledge-based neural networks (KBNN) reduce EM simulator's involvement; two examples are [18] and [16]. KBNN are similar to that of an ordinary NN model except there is a layer or series of layers in which knowledge of the MW system is used. In [18], a detailed discussion shows how to design a model, which utilizes possible functions known on the boundary and throughout the space.

KBNN and NN differ in a subtle way. KBNN are more specified to a narrow model and rather small generalization while NN uses a universal approach. From the NN's point of view, there is only data coming in or out. This is in contrast to KBNN where the neural network is programmed in relations and/or formulas specified for the problem.

There are other types of neural networks, which are used to optimize a MW system. There are Space Mapping Based Neural Networks (SMNN) [12]. They efficiently use an EM simulator by creating a *coarse* NN and mapping it to a *fine* NN without having to create a large database for the fine NN. This mapping is produced by a third neural network that maps only the design variables.

Another approach to NN optimization is a neural network called Synthesis Neural Networks (SNN) [10]. The SNN is an approach, which is an inverse with respect to the mentioned above. In this technique, geometric parameters are the outputs of systems and the inputs are the *S*-parameters. It has been found that it is difficult to get an SNN converged due to the fact that multiple geometries may results in the same *S*-parameters. In [19], an algorithm using a combination of analysis and synthesis NNs to optimize a MW system was successfully implemented.

A major issue throughout all the papers reviewed in the course of this project is efficiency. Even though EM simulators give accurate and reliable results, the question is how can we use those simulators as little as possible and still have an accurate model. Several papers have referred to the time needed to simulate data for neural networks as a major drawback for NNs using EM simulators [7, 20]. It should be clearly noted that the methods like KBNN and SMNN are ingenious attempts at minimizing computing time. We have seen that with the rapid growth and efficiency of computers this is becoming less and less of a problem.

To summarize the introductory part, it should be emphasized that at the initial stage of the project we looked through much of the most recent works in microwave modeling, and have found that optimization using NNs is still a field of research with much room for growth. The approach used in this project includes creation of a database, development of a neural network, and operations towards getting an optimal solution. We attempt to show that it is now feasible to use a straightforward approach combining a universal full-wave simulator with an efficient optimization technique and maintain efficiency and accuracy of computation. This work addresses a universal approach to MW optimization with the goal to be able to expand the range of the microwave systems to which our method is applicable.

Chapter 3

Analysis

3.1 Statement of the Problem

Let

$$\vec{X} = [f, x_1, x_2, ..., x_m]^T \tag{3.1}$$

be a vector containing *m* (geometrical) parameters of a given device. In (3.1), *f* is frequency. We extract *f* from *X* in the following manner:

$$\vec{Y} = \vec{S}_{ij}(f_q) = [Y_q, q = 1,2,...,n]^T \ , \tag{3.2}$$

where *Y* is a vector containing the response of the device under consideration (e.g., *S*-parameters of a *p*-port device, *i*, *j* = 1, …, *p*). In the reality, the EM problem is:

$$\vec{Y} = F(\vec{X}) \tag{3.3}$$

Equation (3.3) can be modeled by training a NN through a set of sample pairs

$$\left\{\left(\vec{X}_k, \vec{Z}_k\right), k = 1, 2, ..., D\right\} \tag{3.4}$$

where X_k, Z_k, are m- and n- dimensional vectors representing the kth sample of X and Z respectively, and D is the number of samples of X and Z. Thus we can view Z_k as the following:

$$\vec{Z}_k = EMsim(\vec{X}_k) \approx \vec{Y}_k = F(\vec{X}_k) \quad \text{for} \quad k = 1, 2, \ldots, D \tag{3.5}$$

where *EMsim* denotes an operator which means the sample of *S*-parameters Z_k is generated by numerical simulations given the geometrical parameters X_k. The NN model for (3.3) is

$$\vec{Y} = G(\vec{X}, \vec{W}, \vec{b}) \tag{3.6}$$

where W and b are the parameters of the NN model (weight and bias vectors), and X and Y are the input and output of the neural model.

Definition of W and b and how Y is computed in (3.6) determines the structure of the NN. Equation (3.6) represents the original problem of (3.3) when the neural model is *trained* by data in (3.4).

The training problem is described as a determination of W and b such that the mean square error between the NN output Y and the desired output Z is minimized:

$$E(\vec{W}, \vec{b}) = \frac{1}{D} \sum_{k=1}^{D} \left(G(\vec{X}_k, W_l, b_l) - \vec{Z}_k \right)^2 \tag{3.7}$$

Once trained, the NN model can be used for predicting the output values of (3.3):

$$\vec{Y} \approx G(\vec{X}, \vec{W}_{OPT}, \vec{b}_{OPT}) \tag{3.8}$$

3.2 Feedforward MLP NN

For the class of MW optimization problems addressed in this project, we consider implementation with two neural network structures. The first of these was a feedforward Multilayer Perceptron (MLP) NN with training according to Levenberg-Marquardt optimization. The second was a Radial Basis Function (RBF) network. We start with a review of basic ideas of the MLP approach.

The first layer has weights coming from the input. Each following layer has a weight coming from the previous layer. The last layer is the network output. Each layer has biases imposed upon it.

In many typical problems, a two-layer MLP is used. This means that the input layer is layer zero followed by a hidden layer of neurons (layer one), and the network output is layer two. Fig. 3.1 shows this simple structure. We use the hyperbolic tangent as the first transfer function and a linear function as the second one.

The linear function defined as

$$pl(x) = x \, . \tag{3.9}$$

is illustration in Fig 3.2. The hyperbolic tangent function is presented as:

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{\left(e^{x} - e^{-x}\right)}{\left(e^{x} + e^{-x}\right)} \tag{3.10}$$

The graph of (3.10) is shown in Fig. 3.3.

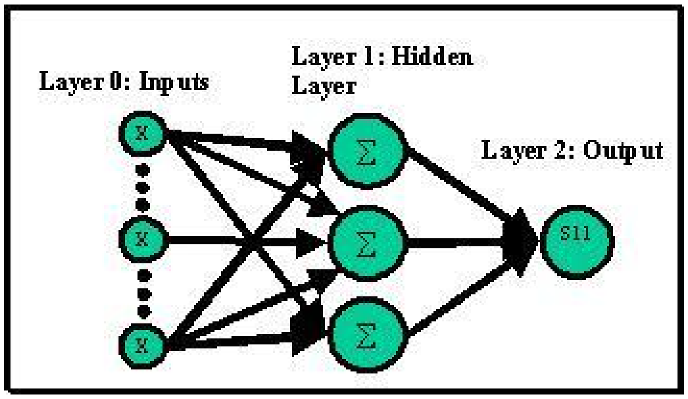


Figure 3.1: Layers of a neural network

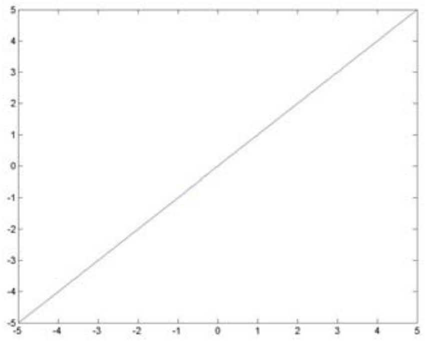


Figure 3.2: Linear function

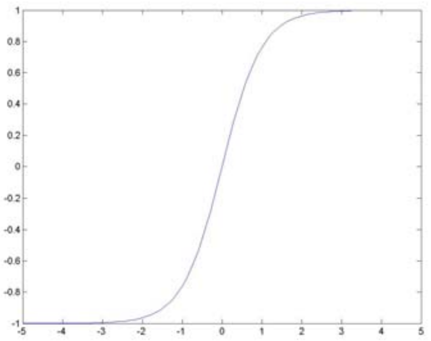


Figure 3.3: Hyperbolic tangent function

The neural network can be described in the following single equation:

$$pl\left(w^2{}_{nh}\left(\tanh\left(w^1{}_{hm}A_{md} + b^1{}_h\right)\right)_{hd} + b^2{}_n\right) = S_{11}(f)_{nd} \tag{3.11}$$

A simpler way of looking at (3.11) is its interpretation at one layer at a time:

$$\tanh\left(w^1{}_{hm}A_{md} + b^1{}_h\right) = B_{hd} \tag{3.12}$$

$$pl\left(w^2{}_{nh}B_{hs} + b^2{}_n\right) = S_{11}(f)_{ns} \tag{3.13}$$

Therefore, in our NN, we have *m* inputs, *d* samples, *h* hidden neurons, and *n* outputs. In (3.12) and (3.13), $w^1{}_{hm}$ and $w^2{}_{nh}$ represent weight matrices for the 0th and 1st layer of the NN. There are also biases for each layer $b^1{}_h$ and $b^2{}_n$.

The function representing the neural network as (3.11) can be expressed in combination with (3.8) as follows:

$$\vec{Y} \approx G(\vec{X}, \vec{W}_{OPT}, \vec{b}_{OPT}) = pl\left(w^2{}_{nh}\left(\tanh\left(w^1{}_{hm}A_{md} + b^1{}_h\right)\right)_{hd} + b^2{}_n\right) \tag{3.14}$$

Training a Neural Network

As stated above, for NN training, we use the Levenberg-Marquardt method, which has a similar form to that of Newton's Method. The Hessian is approximated by the following form:

$$H = J^T J \tag{3.15}$$

where *J* is the Jacobian matrix. The gradient is computed as

$$g = J^T \overline{E} \tag{3.16}$$

where $\overline{E}$ is a vector of the network error. The Jacobian is computed by the general backpropogation technique. The Levenberg-Marquardt algorithm uses the following iterative steps for updates:

$$x_{k+1} = x_k - \left[J^T J + \mu I\right]^{-1} J^T \overline{E} \tag{3.17}$$

When the scalar μ is zero, we reduce the algorithm to regular Newton's Method. When μ is large, the method becomes gradient descent method with a small step size. Since Newton's Method converges quicker near an error minimum, the goal of this algorithm is to decrease μ, so that it converges to Newton's Method.

It has been shown in literature, specifically in [21], that the Levenberg-Marquardt algorithm is much more efficient than the conjugate gradient algorithm and the variable learning rate algorithm. Therefore it is frequently used in implementations for the MLP NN.

3.3 RBF NN

Radial basis function neural networks have similar capabilities to that of MLP NNs. The difference is that the RBF approaches the problem as a function approximation problem [22].

The structure of a radial basis neuron is illustrated in Fig 3.4. The procedure starts with a vector of inputs. Then the distance between the inputs and the vector of weights is calculated, multiplied by the vector *b* and sent to the radial function. This can be expressed as a function

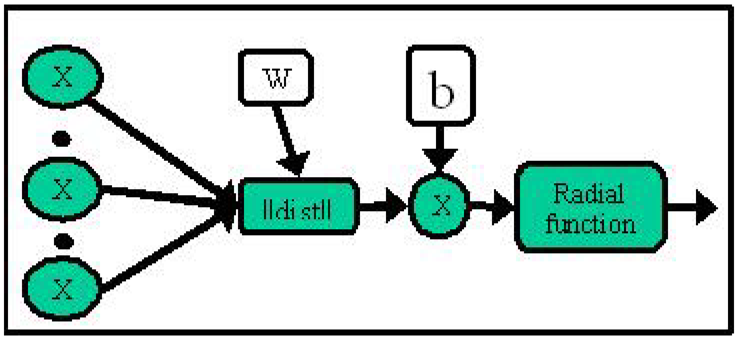


Figure 3.4: Radial basis neuron

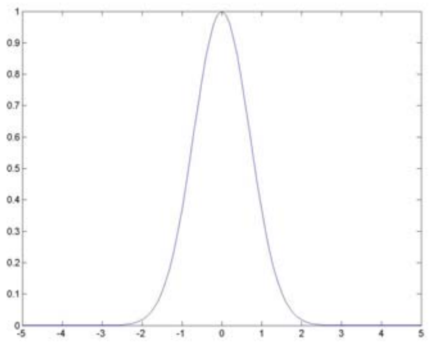


Figure 3.5: The Gaussian radial basis function

$$a = radbas(\| w - p \| b) \qquad (3.18)$$

The commonly used radial basis activation functions [22] are the multiquadratic function and the Gaussian function given by

$$radbas(a) = e^{-a^2} , \qquad (3.19)$$

and illustrated in Fig 3.5.

The architecture of the entire RBF network consists of two layers. The first layer is a hidden layer of radial basis neurons and the second layer is a linear layer – the same as used in the second layer of the feedforward MLP NN.

The motivation with a radial basis network is quite simple. The closer an input is to a weight, the closer that value is to zero. Thus going through the RBF that node's result will be close to one. Therefore, it will have a larger affect on the network.

There are two types of radial basis networks are used for testing. The first of these is a zero training error network. Given *m* inputs, *m* radial basis neurons are created. Therefore, there is no error for the network training because each neuron correctly detects each input. The drawback of this approach is that there is a large number of inputs/neurons.

The second approach is as follows. Initially, the radial basis layer has no neurons. The following steps are repeated until the network's mean squared error (MSE) falls below a specified goal:

1. The network is simulated.
2. The input vector with the greatest error is found.
3. A radial basis neuron is added with weights equal to that vector.
4. The linear layer weights are redesigned to minimize error.

Once the MSE is below a certain limit, the network is said to be trained and we proceed to minimize the RBF NN.

Resulted from the analysis of data, the RBF approach has been chosen for implementation in the computational procedure developed in this project. Several trials were tested with the zero training error approach, but it has been found that this method was insufficient for our needs because of a large number of inputs, which may not always properly define the network. For this reason, the second type of RBF training has been implemented in the computational procedure.

3.4 Optimization Method

We consider the optimization problem as follows: find a configuration of the structure such that a magnitude of an *S*-parameter under consideration is less or larger than the assigned level (S_0) in the frequency range (f_1, f_2) around the operating frequency f_0 ($f_1 < f_0 < f_2$). $|S_{mn}|$ is a multivariable function of frequency f and system parameters $\mathbf{X} = [X_1\ X_2\ \ldots\ X_m]^{\mathrm{T}}$ which becomes an objective function of the optimal design, and S_0 and (f_1, f_2) are interpreted as the relevant constraints. A representation of this for a specific *S*-parameter, S_{11}, is shown in Fig 3.6.

Least Squares Method

After a NN is created and trained, it can be defined as function *G* represented by (3.8). Least squares minimization technique can be used to determine its minimum. The algorithm implements a subspace trust region method and is based on the interior-reflective Newton method

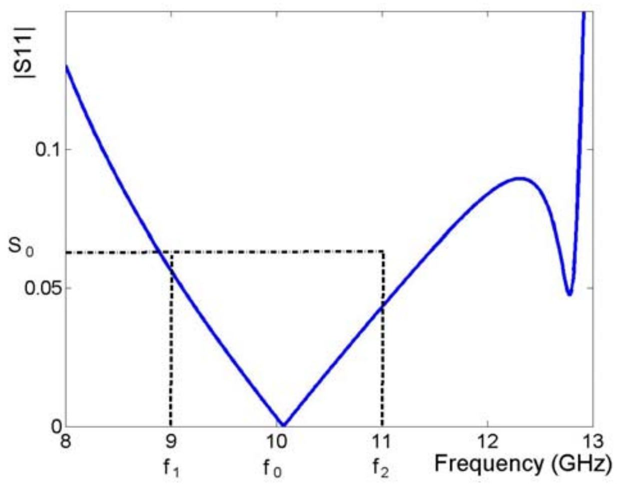


Figure 3.6. Conventional frequency characteristic of $|S_{11}|$ in the constrained optimization problem

described in [23, 24]. Particularly, it is shown in [24] that this technique is globally and quadratically convergent.

We consider the following problem:

$$\min_{x \in R^n} \frac{1}{2}\|G(x)\|_2^2 = \frac{1}{2}\sum_i G_i(x)^2, \quad l \le x \le u \tag{3.20}$$

where $l \in \{R \cup (-\infty)\}^n$, $u \in \{R \cup (\infty)\}^n$, $l < u$ and $G : R^n \to R^m$. This algorithm is an iterative procedure where $s_k = x_{k+1} - x_k$ is an approximate solution to a quadratic subproblem:

$$\min_{s \in R^n} \left\{ \psi(s) \equiv g_k^T s + \frac{1}{2} s^T B_k s : \left\|\overline{D}_k s\right\| \le \Delta_k \right\} \tag{3.21}$$

With g_k defined as $g_k \equiv \nabla G(x_k)$, B_k is a symmetric approximation to the Hessian matrix $\nabla^2 G(x_k)$, $\overline{D}_k$ is a scaling matrix, and Δ_k is a positive scalar representing the trust region size.

The basic idea of the algorithm above is to approximate the function $G(x)$ with a simpler function $\psi(s)$ which basically reflects the behavior of $G(x)$ in a neighborhood Δ_k around the point x. A trial step s is computed to minimize the function over Δ_k. Therefore, the current point is updated as $(x + s)$ if $G(x + s) < G(x)$; otherwise, the current point is unchanged and the trust region is shrunk. The method iterates and quadratically approaches a minimum value.

The algorithm returns a minimum corresponding to the geometrical parameters of size m. This method does not necessarily return a global minimum. Therefore, multiple guesses were used throughout the domain to increase the probability of finding the global minimum.

Getting local optimal solutions in our analysis does not seem to be a drawback. For a majority of applied MW devices, it is enough to fulfill the goals of the constrained optimization problem formulated in the beginning of this section without guaranteeing that the obtained solution corresponds to a global minimum.

Chapter 4

Neural Model

After a series of experiments with various structures of standard feedforward neural networks, we have constructed the Radial Basis Function network with the Gaussian activation function. A suitability of the RBF NN for our problem is conditioned by their capability of faster than multiplayer perceptron (MLP) learning and low sensitivity to the order in which training data are presented [22].

The chosen basic NN structure shown in Fig. 4.1 possesses m inputs in accordance with the number of the system parameters to be optimized and one output associated with the value of $S_{ij}(f_k)$ obtained from the EM solver. The entire network consists of n distinct NNs corresponding to a particular frequency; n here is determined by the number of approximating points in (f_1, f_2). For many practical scenarios in MW optimization, we do not expect n to be a large number, so the choice of the RBF network suited, compared to MLP, for problems with smaller number of inputs [22], appears to be particularly reasonable.

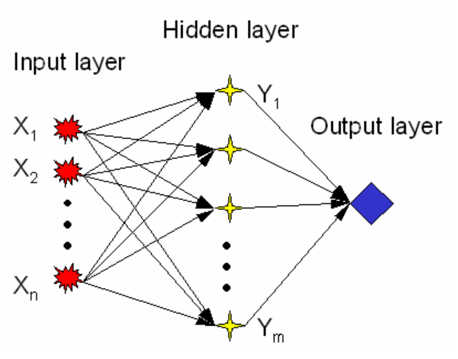


Figure 4.1: Outline of the RBF NN used in the algorithm.

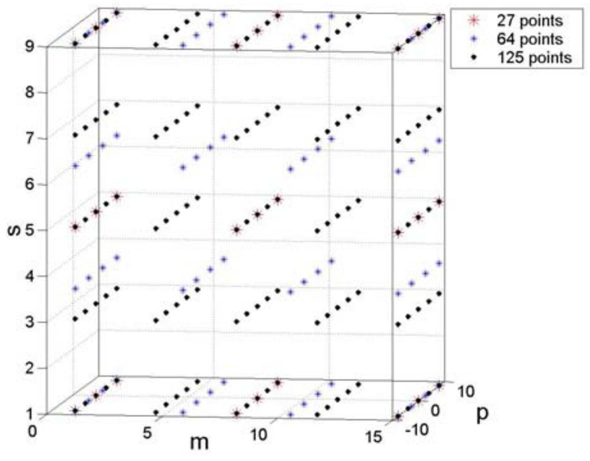


Figure 4.2: Uniform grid of samples from the database for a 90^o waveguide bend (see Chapter 6.1).

Frequency characteristics of *S*-parameters obtained in FDTD simulations performed by *QW3D* compose the network database. In order to have the optimization procedure suitable for a variety of MW systems, we use uniform grid sampling giving no preference to any particular subregions of the input space. An illustration of this is shown in Fig. 4.2.

In MW applications, scaling is regarded highly valuable operation since the order of magnitude of input parameter values can be very different [25], so making the problem better conditioned for training and thus helping the network with learning process, we apply linear scaling of data samples on the input parameters from the database in accordance with the following formula:

$$D(\bar{x}) = \bar{x}_{\min} + \frac{x - x_{\min}}{x_{\max} - x_{\min}}\left(\bar{x}_{\max} - \bar{x}_{\min}\right) \tag{4.1}$$

Chapter 5

Implementation

5.1 Overview

The algorithm has been implemented in MATLAB 6 R12 environment. The master program controls operations of *QW3D*'s Editor and Simulator, manages processing and transferring data, communicates with appropriate procedures from the MATLAB Neural Network and Optimization Toolboxes, and conducts required computations. The project consists of five basic steps: specification of input parameters, database creation, neural network construction and training, minimization of the NN function approximation (3.8), and choice of the best geometry and corresponding frequency characteristic of $|S_{ij}|$. A general layout of the algorithm can be seen in Fig. 5.1. The following description of the algorithm is given for $i = j = 1$.

The first step is implemented in the script `rad_method` (see Appendix, part A), which loads the input data for a specific project (e.g., `ant_input` presented in Appendix, part D). This

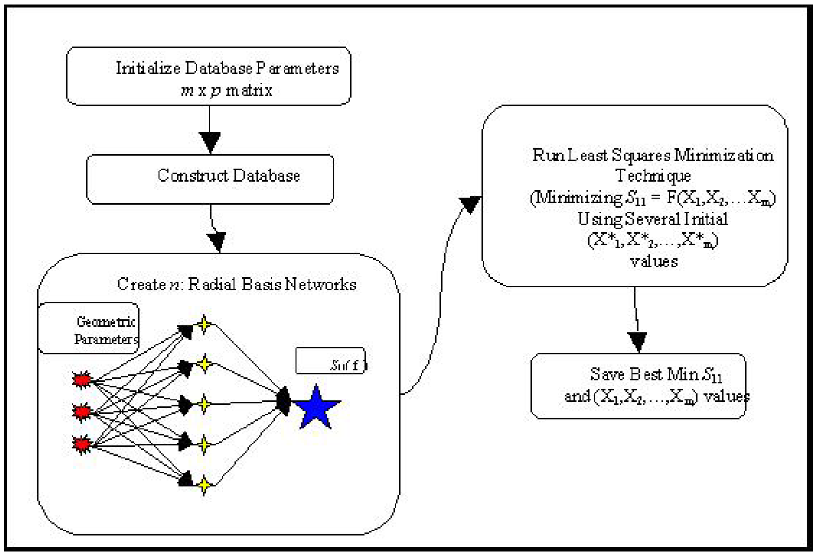

Initialize Database Parameters
m x p matrix
Construct Database
Create n: Radial Basis Networks
Geometric Parameters
S_{11}(f)
Run Least Squares Minimization Technique
(Minimizing S_{11} = F(X_1,X_2,…X_m)
Using Several Initial
(X^*_1,X^*_2,…,X^*_m)
values
Save Best Min S_{11}
and (X_1,X_2,…,X_m) values

Figure 5.1: Flow chart of the algorithm for optimization of the reflection coefficient $|S_{11}|$

input data holds information about frequencies range and the matrix of points to be taken for the database.

5.2 Creation of the Database

After initial parameters have been chosen, the database is built by calling `Databasegetter` (see Appendix, part B). The latter starts by creating a list of points by calling the script `paramaker` (Appendix, part H) made from the matrix of bounds of the given parameters. Throughout the implementation of this project we choose equal number of points for each variable. Although the program is written in such a way it can accept any number of geometric variables, for each example illustrated in this project, there were only three variables used.

Therefore, the parameter matrix is of the form 3 by *p*, where *p* is the number of points for each parameter, and the list of the database points is of size 3 by p^3.

Then `Databasegetter` forms a loop of the parameter list. For each specified three points $\left(x_1^i, x_2^i, x_3^i\right)$, we first call the *QuickWave-3D*'s Editor and modify the project so that we will actually simulate the correctly specified microwave system. Following the modification of the system and saving the project file, we analyze the Tasker file, *.ta3, which specifies the operating frequency f_0, the number of iterations, and the name of data file to be saved.

The next step is to call the Simulator. The latter takes the Tasker file and runs the project for the specified number of iterations. Each project converges at a different speed, so the user needs to decide the correct number of iterations to use; this is normally made by a simple inspection. After the said number of iterations is reached, the Simulator saves the results of S_{11} into a file in the project directory.

After the Simulator has computed all of the samples, another Matlab script called `matrixgetter` (see Appendix, part C) is called. `Matrixgetter` assembles all of the information into a convenient format.

`Matrixgetter` opens one file at a time and takes the second column of the file containing the S_{11} data obtained from the EM simulator for a number of points in a specific frequency range. After data is extracted from each file, all the data is saved into a *.mat file. The mat-file consists of two matrices and one vector, `vars` (size 3 by *S*), `f` (size *n*), and `S11` (size *S* by *n*).

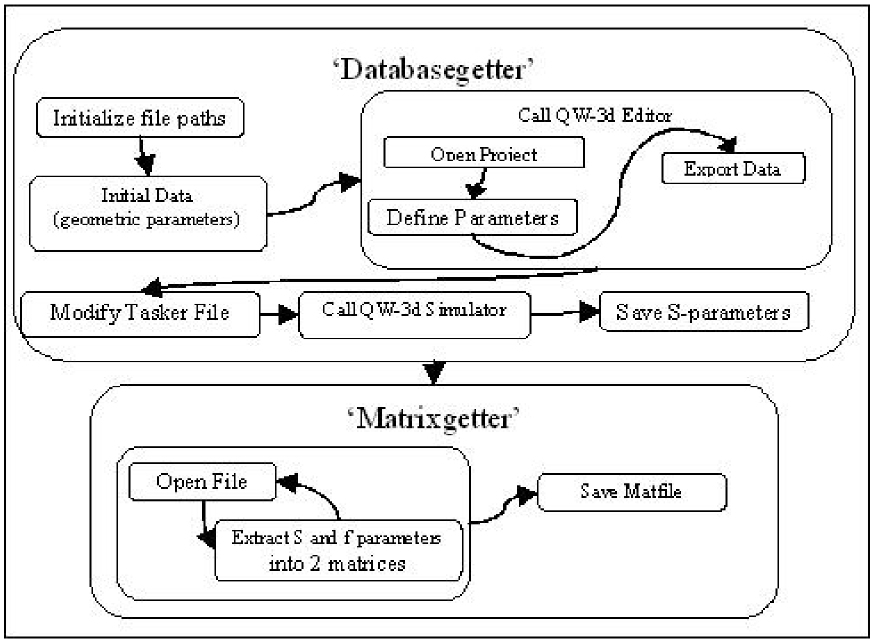


Figure 5.2. Principle steps in `Databasegetter` and `Matrixgetter`

At this point, we have a database consisting of $p^3 = S$ points and having a simple format to be used in the following steps. Fig. 5.2 presents a flowchart of the operations performed in scripts `Databasegetter` and `Matrixgetter`:

5.3 Construction and Training of Radial Basis Network

The next step in the program is constructing the neural network. The first step in this process is scaling, specifically linear scaling in accordance with (4.1). Through the analysis it was seen that only certain intervals of inputs into RBF NN converge. To specify the optimal interval at which the data are given to the network, a series of computational experiments is required. The Matlab script called `scalar` (see Appendix, part F) is responsible for scaling.

The implemented procedure of scaling can be described as follows. Given a matrix, or a vector, and the corresponding minimum and maximum values of both its inputs and outputs, the function returns the scaled matrix or vector. That is, for matrix `params` (geometric parameters) with corresponding input and output maxima and minima, the function `scalar` returns a matrix `scaled_params` (input to the RBF NN). This can be numerically illustrated as:

$$\texttt{params}: \begin{bmatrix} 30 & 50 & 70 & 90 \\ 30 & 43.33 & 56.66 & 70 \\ 30 & 60 & 90 & 120 \end{bmatrix};$$

$$\text{inputs:} \begin{bmatrix} \max: & 90 & 70 & 120 \\ \min: & 30 & 30 & 30 \end{bmatrix}; \ \text{output:} \begin{bmatrix} \max: & 3 & 3 & 3 \\ \min: & -3 & -3 & -3 \end{bmatrix};$$

$$\texttt{scaled\_params}: \begin{bmatrix} -3 & -1 & 1 & 3 \\ -3 & -1 & 1 & 3 \\ -3 & -1 & 1 & 3 \end{bmatrix}.$$

Once the matrices `vars` and `params` have been scaled, they are used in `rad_method`. This script organizes the entire process; it also creates and trains the neural network. The first step is to extract the specified frequencies corresponding to the S_{11} values from the matrix `S11`.

The matrix `ouputs` is extracted from `S11` with the dimensions *n* by *S*, where *S* is the number of samples. The *n* rows of this matrix correspond to the *n* chosen frequencies. The matrix `inputs` is given by the scaled `vars` matrix. `Inputs` has dimension *m* by *S*, in which *m* represents the number of geometric variables. In the examples considered in this project, $n = m = 3$.

5.2 Optimization and Comparison

When the network is trained, we take several initial guesses and minimize function *G* with the MATLAB's least square method's algorithm `lsqnonlin`. Due to the fact that our minimization technique does not guarantee the global minimum, we choose two values for each of *m* geometric parameter. The collection of these values is, therefore, equivalent to 2^m guesses. The results from the minimization procedure yield possible optimized geometric values. Numerically, the output data generally have many decimal places. Since in engineering practice MW systems are normally described in millimeters, we introduce the script `rounder` (see Appendix, part E), which rounds off to a specified decimal place so that the results are more meaningful.

Then these geometric values are passed to the `opttest` script (Appendix, part G). This one runs *QW3D* and tests the value. If the results of simulation are the minimum of the other optimized guesses, then we save the geometric and *S*-parameters, and have our solution.

Chapter 6

Illustrations

The described computational procedure has been applied to optimize geometrical parameters of several MW components – from a waveguide structure through an antenna to MW heating devices. In this Chapter, we present the detailed results for three particular constructions.

6.1 Microwave Systems

In this section, we present the geometric shapes and constrained parameters of the microwave devices considered in this project. In each case, minimization of $|S_{11}|$ being a function of frequency and three geometric parameters was the goal.

Project A: 90° Waveguide Bend

The first scenario, a 90° 23 x 11.5 mm waveguide non-smooth bend, is the simplest example that we used, so, from the computational point of view, simulation of the model of this project was quickest.

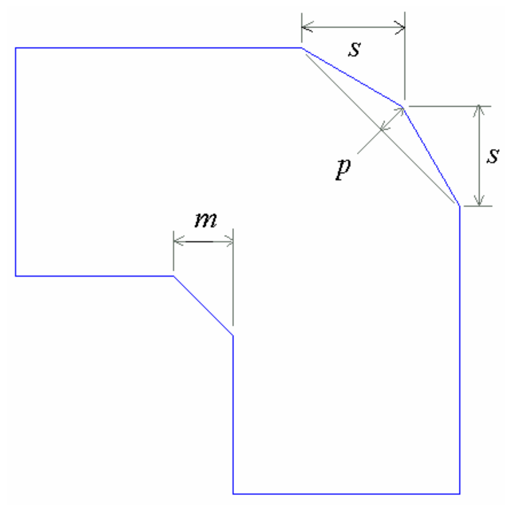


Figure 6.1: Geometry of Project A

As one can see in Fig. 6.1, the waveguide redirects the path of the EM field from the top left exiting in the bottom right. In the optimization procedure we minimize $|S_{11}|$, i.e., the reflections generated by the non-regular cross-sections along the direction of propagation. The minimized S_{11} results in the maximized the transmission of the field through the waveguide bend.

Table 6.1 contains the geometric variables of the bend and the corresponding ranges. As for the frequency range, it was chosen to be $f \in (9, 12 \text{ GHz})$. We assume that $n = 3$, i.e., we minimize $S_{11}(f)$ at the points f_1, f_0, f_2. The operating frequency f_0 is 10.5 GHz.

Table 6.1: Variables of Project A (Fig. 6.1)

Variable	Range
s	$1 \leq s \leq 15$ mm
p	$-8 \leq p \leq 8$ mm
m	$1 \leq m \leq 9$ mm

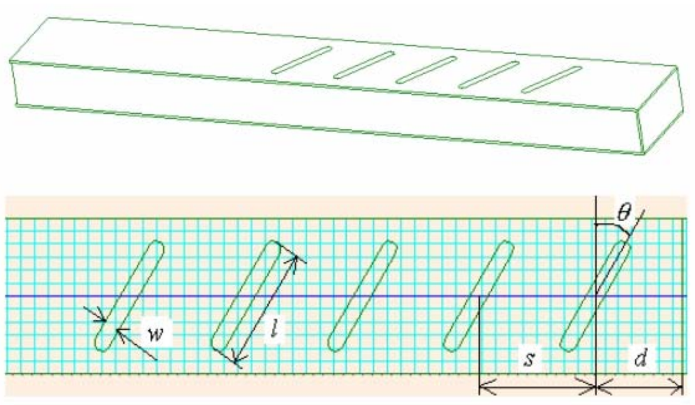


Figure 6.2. Geometry of Project B.

Project B: Slotted Waveguide

This project is a waveguide-fed slot-antenna array. It has been used as resonant and traveling-wave antennas in many ground-based and airborne radar systems for many years [26]. It is made up of five narrow inclined slots. The full description of the structure could be given by 5 parameters shown in Fig. 6.2. We consider this antenna to be based on the rectangular waveguide WR430 (86 x 43 mm) and assume that the configuration of each slot is not changed (w = 8 mm, l = 65 mm)

The operating frequency for this project is f_0 = 2.45 GHz, and we optimize the S_{11} characteristic in the interval (f_1, f_2) = (2.4, 2.5 GHz).

Table 6.2: Variable of Project B (Fig. 6.2)

Variable	Range
θ	$20° \le \theta \le 90°$
s	$30 \le s \le 70$ mm
d	$30 \le d \le 120$ mm

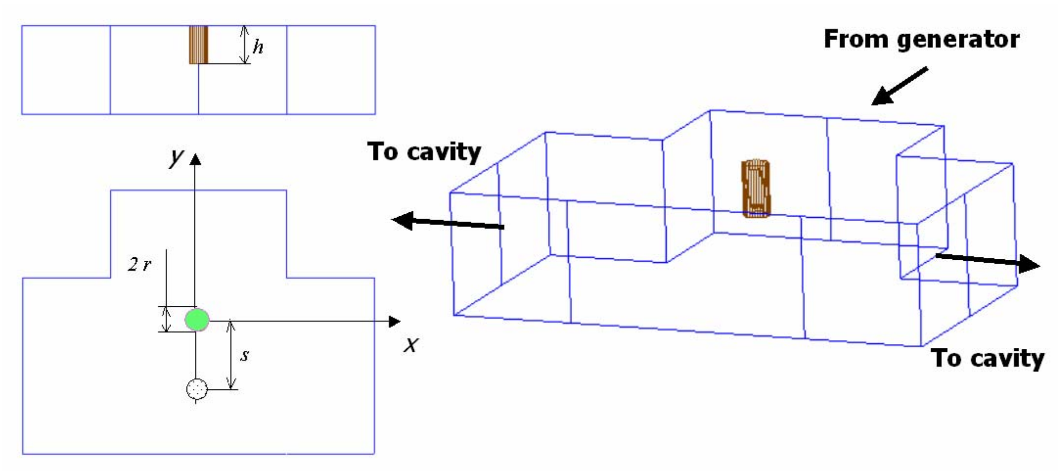


Figure 6.3. Geometry of Project C

Project C: Waveguide T-junction with a Post

This is a typical junction of rectangular waveguides in which a post plays the role of the matching element [27]. It is located along the central line of the input waveguide. We analyze a junction of the waveguides WR75 (19.05 x 9.53mm). The considered construction is characterized by three geometric parameters outlined in Fig. 6.3.

The operating frequency for this project is $f_0 = 12.5$ GHz, and $(f_1, f_2) = (11, 14$ GHz).

Table 6.3: Variable of Project C (Fig. 6.3)

Variable	Range
r	$0.5 \leq r \leq 1.5$mm
h	$4 \leq h \leq 8$mm
s	$-6 \leq s \leq 6$mm

6.2 Scaling

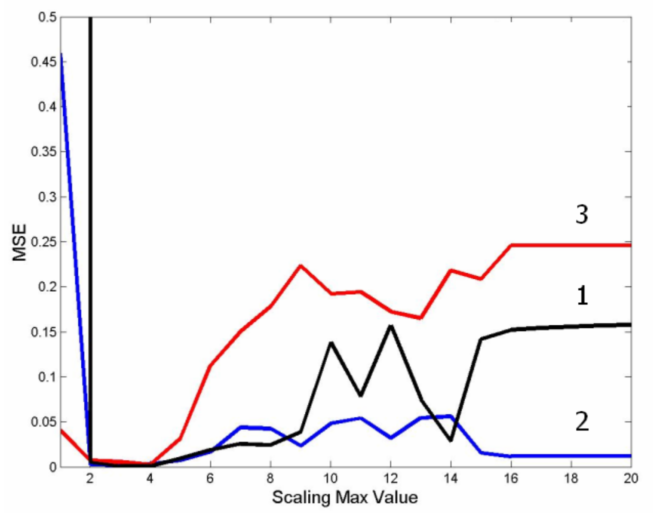
Scaling was implemented in order to find a optimal range for RBF NN inputs for each of the analyzed projects. For Projects A to C, scaling was implemented for varying ranges [-*r*, *r*] with scaling parameter *r* ranging from 1 to 20 with inspection of the generated mean square error (MSE). From Fig. 6.4, a, it is seen that MSE of the RBF NN has a minimum in the interval (2, 3) for all three projects.

We also ran another test in which we reduced our search range to the interval of [1.75, 3.25] taking again 20 points of testing. Fig. 6.4, b represents the results from this test. We found that the minimum is not clearly defined in this interval. Thus, when optimizing configurations of the systems in Project A to C, an optimal scaling interval was chosen to be $r = 3$.

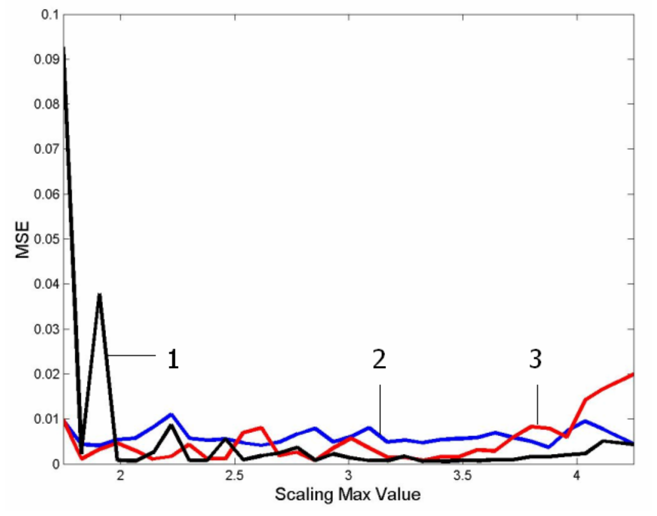
6.3 Accuracy

The accuracy of the presented approach is illustrated by the results obtained for Project A.

Fig. 6.5 is the graph of the S_{11} parameter computed by *QW3D* assuming that two geometric parameters *m* and *s* vary and one is held constant ($p = 0$). Frequency is also constant at the operating frequency of $f_0 = 10.5$ GHz. We have taken a 30 by 30 point area meaning 900 runs of *QW3D*. We view this surface as exact, and intend to compare it with the outputs of the RBF NN.



(a)



(b)

Figure 6.4. RBF NN mean square error for varying scaling parameter $1 \leq r \leq 20$ (a) and $1.75 \leq r \leq 4.25$ (b) for Projects A (1), B (2), and C (3).

The surface in Fig 6.6 shows that with p = 3 (27 point database), our method does not converge well as compared to the exact solution (Fig. 6.5). The result generated by the 64 point

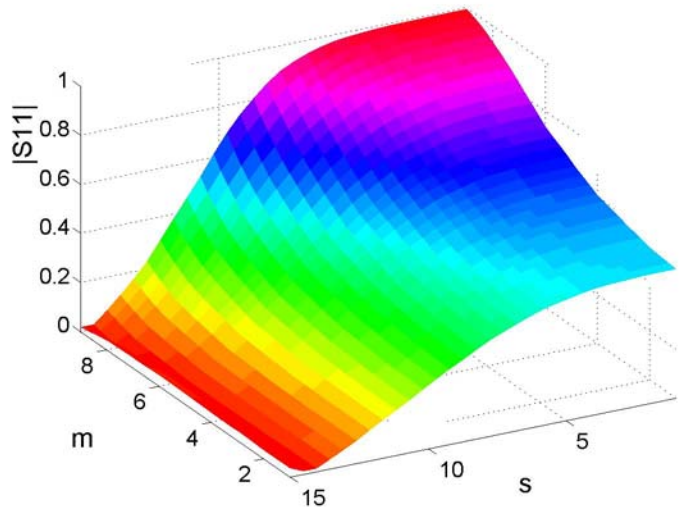


Figure 6.5. *QW3D*-generated values of $|S_{11}|$ for Project A

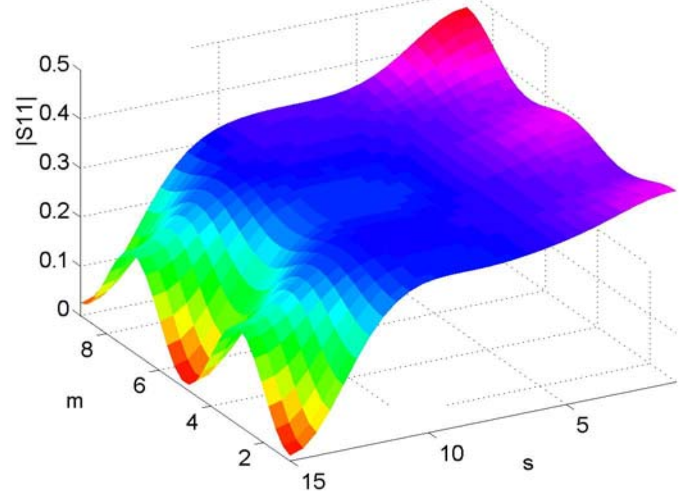


Figure 6.6. RBF NN results for p = 3 (Project A)

database (p = 4) and shown in Fig 6.7 resembles the graph in Fig. 6.5 moderately well. Fig. 6.8 corresponding to p = 5 (125 points database) seems to be the most accurate of the three.

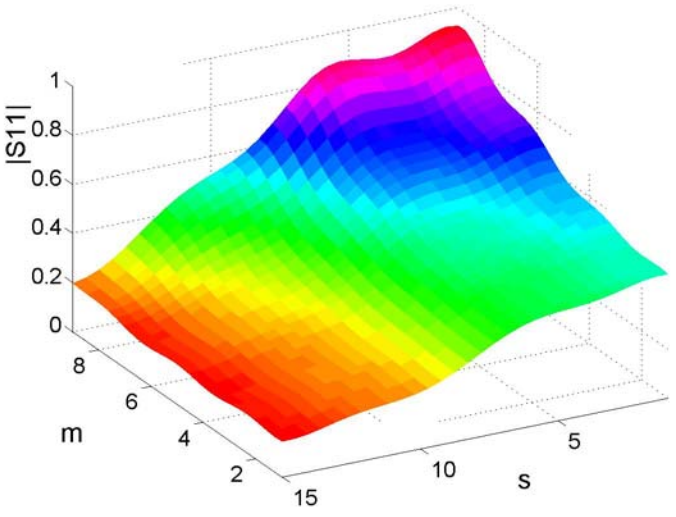


Figure 6.7. RBF NN results for p = 4 (Project A)

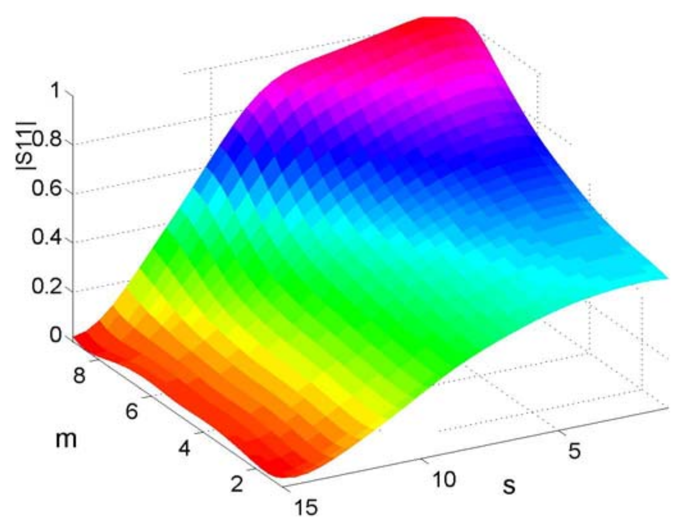


Figure 6.8. RBF NN results for p = 5 (Project A)

Fig. 6.9 represents the absolute value of the difference between the graphs in Figs. 6.5 and 6.8. We see that the maximum error is below 0.1 across the entire domain that the neural

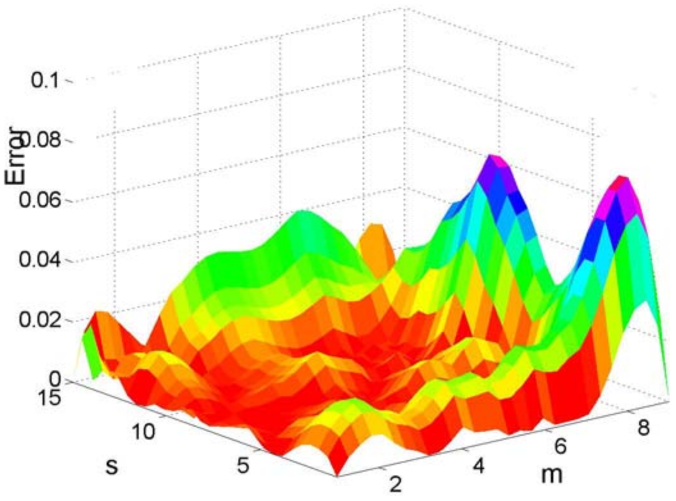


Figure 6.9. Absolute value of error in Project A for $p = 5$

network was approximated for. The mean squared error for the three cases presented above is shown in Table 6.4.

Quality of training of the used RBF NN was checked for the different number of training samples in the database. For Projects A to C, mean square error is quite low even for small number of training samples. The comparison of RBF-NN-generated results with the accurate *QW3D* simulation can be estimated as fairly acceptable.

Table 6.4. Testing Error (MSE) of the Developed RBF NN

	Database: # of samples

Project	**27**	**64**	**125**
A	< 0.001	0.005	0.016
B	0.005	0.002	0.003
C	0.001	0.002	0.001

6.4 Optimization

For each project, the optimal solutions have been obtained with the use of the databases of different size. The objective function was minimized in the frequency range, specified by f_1 and f_2 (their values are presented below in the graphs of Figs. 6.10-6.12), whereas the limiting value S_0 has not been explicitly assigned; the characteristic was rather forced to be within the range of (f_1, f_2) as small as possible.

Table 6.5 contains the results for the waveguide bend: the optimized values of $|S_{11}|$ for the three specified points in the frequency range. The operating frequency represents the midpoint of the interval. Three sets of the optimized geometric parameters of the waveguide bend (Project A) are presented in Table 6.6 for the databases of different sizes (27, 64, 125 samples). The *S*-parameters from Table 6.5 correspond to the values of *s*, *p*, and *m* in Table 6.6.

Table 6.5: Optimized Values of $|S_{11}|$ for Project A

***S*-parameters at**	$p = 3$	$p = 4$	$p = 5$
f_1 **= 9 GHz**	0.3171	0.1073	0.0594
f_0 **= 10.5 GHz**	0.3399	0.0191	0.0181
f_2 **= 12 GHz**	0.1335	0.0569	0.0809

Table 6.6: Optimized Geometry for Project A

Geometry	***p* = 3**	***p* = 4**	***p* = 5**
s	14.973	10.798	14.735
p	-7.967	-2.974	0.105
m	8.992	3.279	1.5

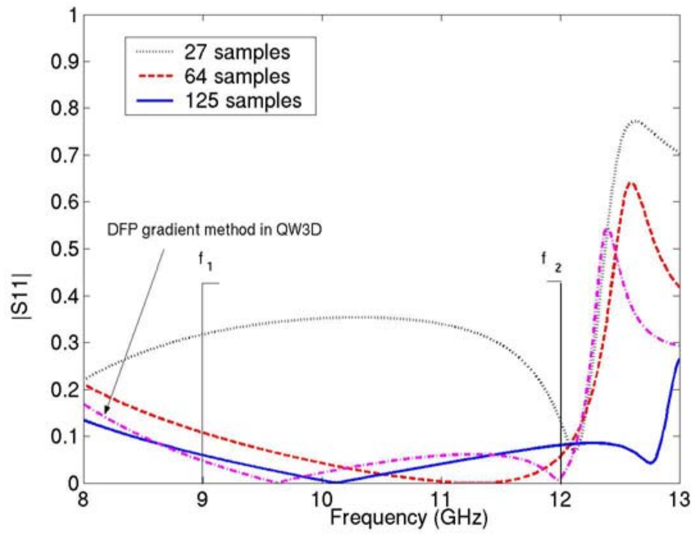


Figure 6.10. Optimized $|S_{11}|$ frequency characteristics for Project A.

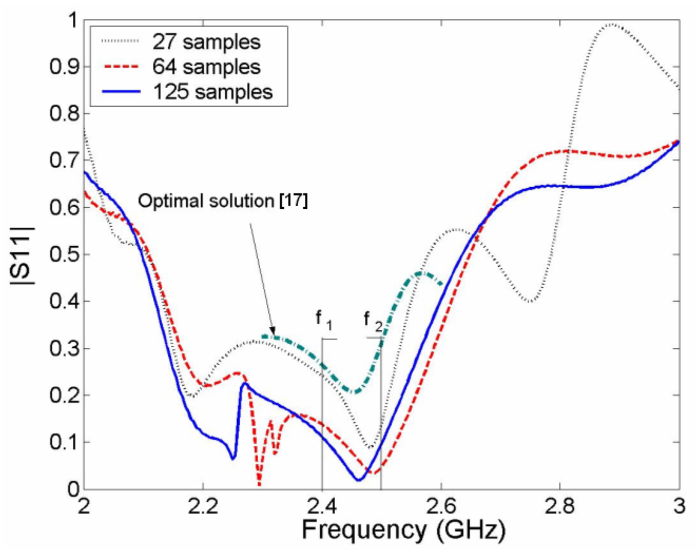


Figure 6.11. Optimized $|S_{11}|$ frequency characteristics for Project B.

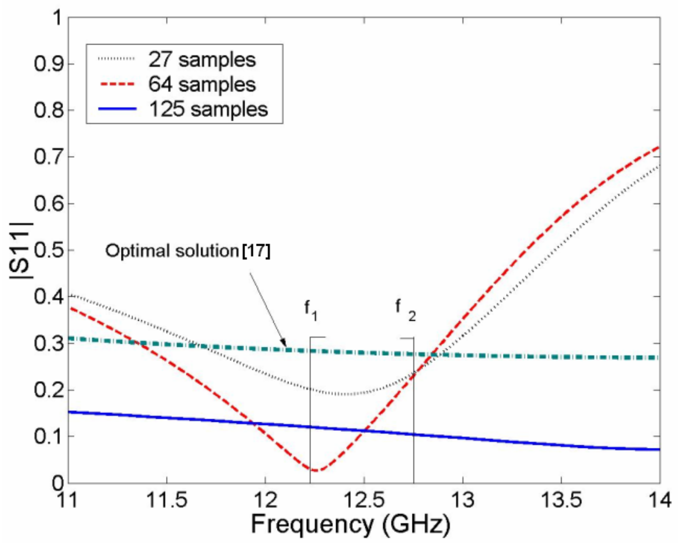


Figure 6.12: Optimized $|S_{11}|$ frequency characteristics for Project C.

Table 6.7: Optimized Values of $|S_{11}|$ for Project B

***S*-parameters**	***p* = 3**	***p* = 4**	***p* = 5**

f_1 = 2.4 GHz	0.043	0.1446	0.0696
f_0 = 2.45 GHz	0.0424	0.0744	0.0294
f_2 = 2.5 GHz	0.0419	0.0709	0.0231

Table 6.8: Optimized Geometry for Project B

Geometry	***p*** **= 3t**	***p*** **= 4**	***p*** **= 5**
θ= 90-input	34.484	18.362	42.983
s	69.754	70	40.268
d	30.322	44.938	55.091

Table 6.7 contains the optimized values of $|S_{11}|$ in the slotted antenna (Project B) for the three specified points in the frequency range with f_0 at the midpoint of the interval. Three sets of the optimized geometric parameters are presented in Table 6.8 for the databases of different sizes

Table 6.9: Optimized values of $|S_{11}|$ for Project C

S-parameters	***p*** **= 3**	***p*** **= 4**	***p*** **= 5**
f_1= 12.45 GHz	0.191165	0.088163	0.113607
f_0 = 12.5 GHz	0.193691	0.110458	0.112232
f_2 = 12.55 GHz	0.198225	0.133406	0.110781

Table 6.10: Optimized Geometry for Project C

Geometry	***p*** **= 3t**	***p*** **= 4**	***p*** **= 5**
r	0.500762	0.815124	0.505496
h	4.000077	4.011209	7.980402
s	5.999768	5.969846	-2.36053

(27, 64, 125 samples). The S-parameters from Table 6.7 correspond to the values of θ, s, and d in Table 6.8.

In Table 6.9, we present the optimized values of $|S_{11}|$ in the T-junction with a metal post (Project C) for the three specified points in the frequency range (f_0 is at the midpoint of the interval). Three sets of the optimized geometric parameters are given in Table 6.10 for the databases of different sizes (27, 64, 125 samples). The S-parameters from Table 6.9 correspond to the values of r, h, and s in Table 6.10.

To evaluate the accuracy of performance of the developed RBF NN approach, we have generated optimized results by alternative techniques; corresponding graphs are shown in Figs. 6.10 to 6.12. The waveguide bend (Project A) was optimized by one of the *QW-Optimizers* implementing the Davidon-Flethcer-Powell (DFP) method. The optimal solution for the slotted antenna (Project B) and the T-junction (Project C) were obtained by the RSM-SQP method [17]. From the curves presented in these figures, we can see that the RBF NN procedure gives either equally good, or better results.

For example, for the slotted antenna, with $S_0 < 0.3$, the optimal geometry suggested by [17] is represented by the parameters $\theta = 27°$, s = 56mm, and d = 118mm which correspond to $|S_{11}|$ = 0.283 at f_0 = 2.45 GHz. Our procedure give different geometric configurations which yield the values of $|S_{11}|$ equal to 0.042, 0.074, and 0.029 for 27-, 64-, and 125-point databases respectively.

For the waveguide junction, with the constraint of $S_0 < 0.33$, the RSM-SQP-method in [17] gives the set r = 1.5mm, h = 8mm, s = -6mm, which corresponds to $|S_{11}|$ = 0.2062 at f_0 = 12.5

GHz. With the use of three different databases, the RBF NN generates the values of the reflection coefficient 0.194, 0.11, and 0.112.

Therefore, optimizing Projects B and C, the presented procedure has shown superior results in comparison with the algorithm described in [17]. Even trained with the 27-sample database, the Radial Basis network has generated significantly improved solutions. Computation benefits of our approach over [17] are illustrated in Table 6.11.

Table 6.11. Optimization Time (min.) by RFB-NN and [17]

Project, # of FDTD cells, RAM (*QW3D*)	Time (P III 1.0 GHz)				Time (P III, 750 MHz) in [17]
	Database: # of samples			Optimiza-tion	
	27	64	125		
B: 13,600 cells, 1 MB	10	25	51	< 5	70
C: 102,000 cells, 10 MB	91	208	412	< 56	660

Table 6.12. Comparison of *QW3D* Optimizers to the RBF NN Optimization (Project A)

Method	**$\|S_{11}\|$ at f_0 = 10.5 GHz**			
Trials	**1**	**2**	**3**	**4**
***QW3D*: Powell Non-Gradient Method (A)**	0.0323	0.0331	0.0202	0.025
***QW3D*: DFP Gradient Method (B)**	**MF**	0.0302	0.0442	0.0453
***QW3D*: Controlled Random Search (C)**	0.0372	**MF**	0.0184	0.055
***QW3D*: Evolutionary Strategy (D)**	0.0457	**MF**	0.0087	0.0983

RBF NN: *p* = 4, or 64-point database	0.0191	0.0191	0.0191	0.0191
RBF NN: *p* = 5, or 125-point database	0.0181	0.0181	0.0181	0.0181

6.5 Comparison with QW3D Optimizers

For Project A, the RBF NN procedure supported by the 64- and 125-sample databases was compared with the optimization modules in the *QW3D* package; the results of this comparison are shown in the present section.

Table 6.12 showing the values of the reflection coefficient at the operating frequency includes the references to the five optimization techniques involved the comparison. The results obtained with the *QW3D* optimizers strongly depend on the initial guess. The method may even

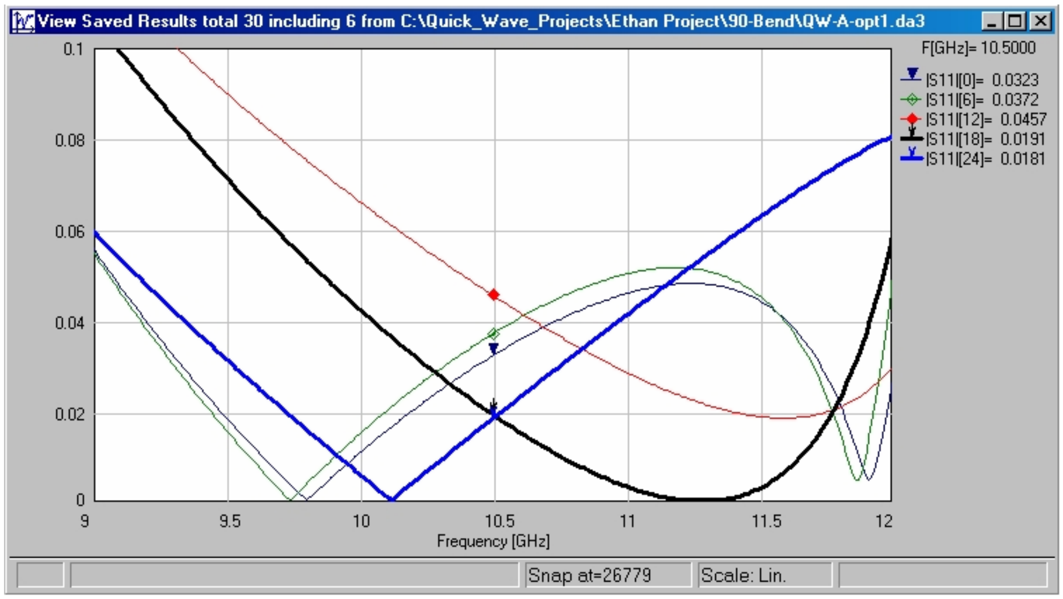


Figure 6.13. Optimized frequency characteristics of $|S_{11}|$ in Project A for Trial 1. Thin curves: Methods A (marked by ∇), C (◊), and D (♦). Thick curves: NN technique for $p = 4$ and $p = 5$.

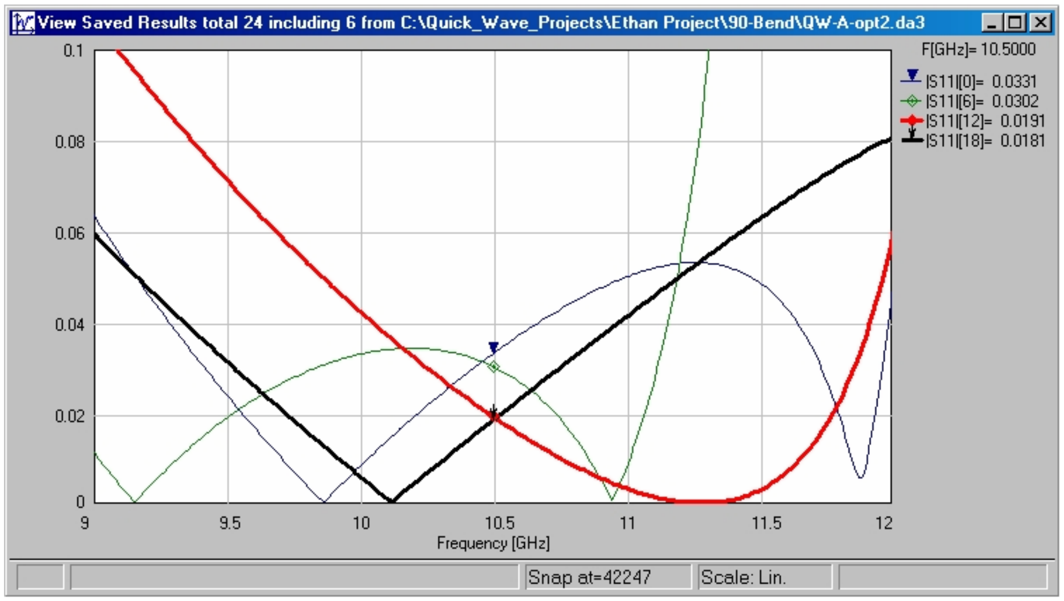


Figure 6.14. Optimized frequency characteristics of $|S_{11}|$ in Project A for Trial 2. Thin curves: Methods A (marked by ∇), and B (◊). Thick curves: NN technique for $p = 4$ and $p = 5$.

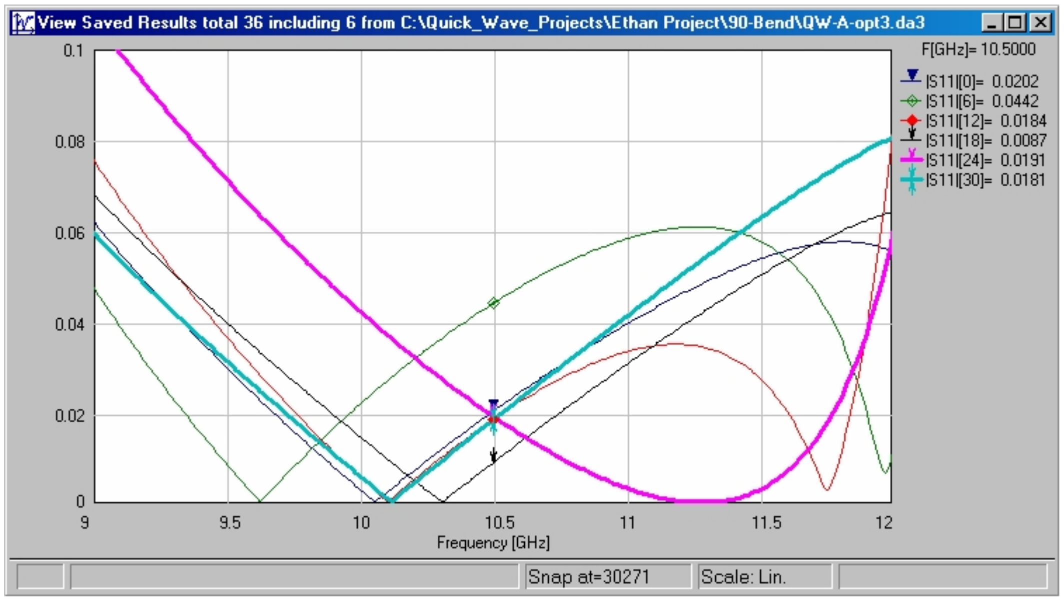


Figure 6.15. Optimized frequency characteristics of $|S_{11}|$ in Project A for Trial 3. Thin curves: Methods A (marked by ∇), B (◊), C (♦), and D (∨). Thick curves: NN technique for $p = 4$ and $p = 5$.

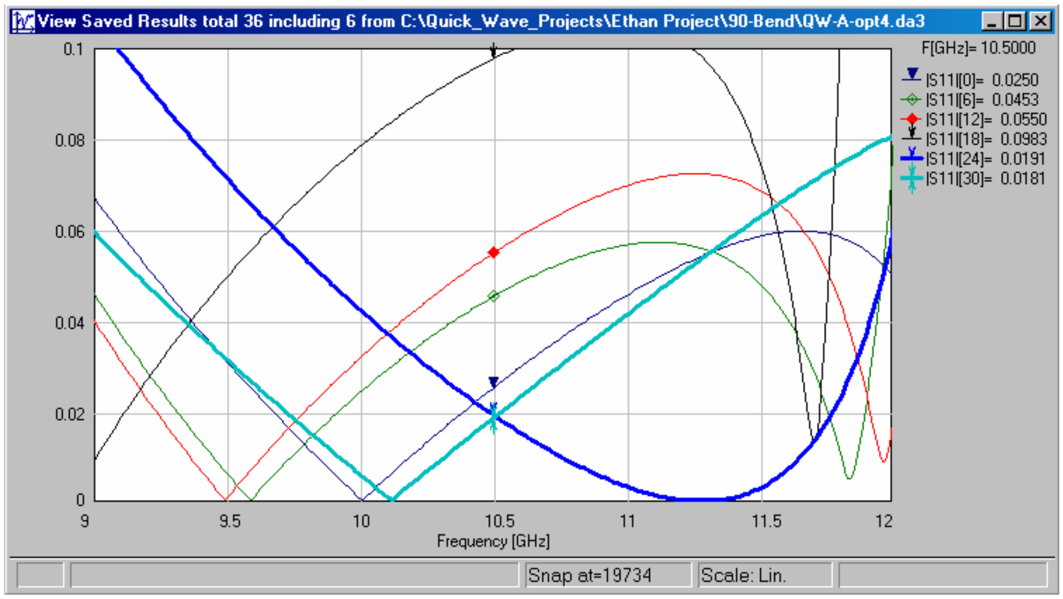


Figure 6.16. Optimized frequency characteristics of $|S_{11}|$ in Project A for Trial 4. Thin curves: Methods A (marked by ∇), B (◊), C (♦), and D (∨). Thick curves: NN technique for $p = 4$ and $p = 5$.

fail if this guess is not quite successful; in Table 6.12, MF (*method fails*) stands for such a failure. For this reason, we show the results of 4 trials. It is seen that the RBF NN solution is independent on the initial guess.

The optimized solutions generated by the five methods are presented in Fig. 6.13-6.16. Each of these four graphs represents the distinct initial guess of these trials.

Analysis of this comparison reveals the following observations. Among *QW3D* Optimizers, the Powell Non-Gradient method looks most reliable. The other three techniques failed in 1 out of 4 (25%) initial guesses. The Controlled Random Search and Evolutionary Strategy have particular difficulties with some initial guess. The DFP Gradient Method finds the optimal solution by placing one variable (*s*) on the endpoint of its interval and not changing another variable (*p*) at all. Therefore, we conclude that our optimization technique looks more reliable and stable since it does not depend on any of these circumstances. The RBF NN procedure with 64 and 125 samples has generated smaller values of $|S_{11}|$ at f_0 compared with almost all methods and trials.

Chapter 7

Conclusion

In the present project, we propose an original, efficient and simple optimization technique based on artificial neural networks and developed as a computational supplement for *QuickWave-3D*. We have shown that, given the resources of today's computers, such an approach can be reasonably productive and serve a competent optimization tool in designing of various MW systems. Our approach is characterized by high accuracy and efficiency. In addition, when compared with the optimization routines in the *QuickWave-3D* package*,* this approach achieves the improved results. This work contributes to the development of efficient and universal CAD instruments suitable for optimization of various microwave systems and components.

With computer hardware becoming faster and faster, methods such as this will become much more practical. Therefore, we suppose that this method can only be improved in the coming years.

Appendix

This appendix supplies the core MATLAB scripts used to run the algorithm.

`Rad_method` is the main script. It first calls input data; `ant_input` is an example of an input data script for Project B. Following this, `rad_method` calls `Databasegetter`. `Databasegetter` is a script that communicates with *QuickWave-3D*'s Editor and Simulator in order to create a database.

`Databasegetter`'s first step is to call `paramaker`. `Paramaker` creates a list of points that will be used for the database from the input data. Upon this `Databasegetter` communicates with *QW3D* Editor to update the project.

The *QW3D* projects are completely parameterized *.udo files, created in such a way that from Matlab the project can be modified to any desired geometric configuration. Several projects of this sort were developed in [17]; in this work, we used the udo-scripts from [17] to perform computations for Projects A to C in Chapter 6.

Then *QW3D* Simulator is called to run the project. A data file is saved into the project folder and this process is repeated a specified number of times determined by the input data.

`Matrixgetter` is then called by `rad_method`. `Matrixgetter` opens each data file and extracts the pertinent information. In the examples illustrated above we extract $|S_{11}|$ values for varying frequencies.

Then `rad_method` calls `scalar`. `Scalar` is a script that scales the input data (i.e., geometric variables) of the NN into normal intervals. After that `rad_method` creates and trains a RBF NN from the database information.

The RBF NN is then minimized with a least squares technique (MATLAB's `lsqnonlin` procedure) using a series of guesses because of predetermined limitation of the method. Then the function `rounder` is used to round off the results of the minimized RBF NN (i.e., geometric parameters). The geometric parameters are sent to `opttest`. `Opttest` tests the validity of the supposed minimum by generating results from *QW3D*. Finally, the best optimized solution is saved and the algorithm is completed.

The presented MATLAB scripts are given below in the following subsections:

A. `Rad_method`

B. `Databasegetter`

C. `Matrixgetter`

D. `Ant_input`

E. `Rounder`

F. `Scalar`

G. `Opttest`

H. `Paramaker`

A: Rad_method.m

```
function [bestmin, mins11, mse, e, neural_ans, suppose, bestopts,
f , net] = rad_method(project_name,div,fdiv,scalexmax)

%radial basis method
tout =cputime;

global s11 f vars keyf params


%load project%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
loading = [project_name,'_input'];
eval(loading)
%load project%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%Database%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
params = rounder(params);
project_name
Databasegetter(params , project_name);
matrixgetter(params , project_name);
loading = ['load ',project_name,'data'];
eval(loading);

ex = cputime-tout
saving = ['save ',project_name,'_database',num2str(div)];
eval(saving)

%Database%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

tout = cputime;

xmax = params(:,div);
xmin = params(:,1);

%Set upper and lower bounds for Least Squares
Minimization%%%%%%%%%%%%%
vars=vars';
scalexmin = -scalexmax;
params = scalar(params , scalexmax , scalexmin , xmax , xmin)
vars = scalar(vars , scalexmax , scalexmin , xmax , xmin);
lb = params(:,1);
ub = params(:,div);

for l = 1:length(params(:,1))
    A(l,:) = linspace(params(l,1) , params(l,div), 5);
end
middle = length(keyf)/2;      middle = round(middle);
```

```
%Least Squares Minimization Options%%%%%%%%%%%%%%%%%%%%%%%

qo=1;
mins11 = ones(length(keyf),1);
%Neural Network%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Extract Key Frequencies
x_t = ['['];
for pt = keyf
    pt
    [ m, ind ] = min( abs(f - pt) );
    x_t = [x_t, ' s11( :,',num2str(ind),')'];
end
x_t = [x_t,']'];

%Extract Key Frequencies

%Inputs and Targets%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

inputs = vars;
targets = eval(x_t)';

%Strings to create NN%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i=1:length(keyf)
    net = ['net', num2str(i) , ' = newrb(inputs,targets(',
num2str(i) ,',:));'];
    eval(net)

    siming = ['train(', num2str(i) ,',:) = sim(net', num2str(i)
,', inputs );'];
    eval(siming)
end

%Error: Mean Squared error%%%%%%%%%%%%%%%%%%%%%%%%
for j=1:length(keyf)
    ies = ['ie(',num2str(j),') = sum( ( targets(:,',num2str(j),')
- train(:,',num2str(j),') ).^2 )/length( train );'];
    eval(ies)
end
% for j=1:3
%     ie(j) = sum( ( targets(:,j) - train(:,j) ).^2 )/length(
train );
% end

e = sum(ie)/length(ie)

%Error: Mean Squared error%%%%%%%%%%%%%%%%%%%%%

%Create fun function
fid = fopen( 'fun.m', 'wt');
x = ['function y = fun(x) '] ;
```

```
globals = ['global '];funs = [' '];
for j=1:length(keyf)
    globals = [globals, ' net',num2str(j),' '];
    funs = [funs, ' y(',num2str(j),') = sim(net',num2str(j),',
x); '];
end
fprintf(fid,'%s',x);
fprintf(fid,'\n %s',globals);
fprintf(fid,'\n %s',funs);
fclose( fid );
clear fid globals funs x j

%Neural Network%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%TESTING THE Neural Network%%%%%%%%%%%%%%%%%%%%%%%%%%

x = paramaker([A(:,2) A(:,4)])';
for qo = 1:length(x(1,:))
    mini(:,qo) = lsqnonlin('fun',x(:,qo),lb,ub);
    realmini(:,qo) = scalar(mini(:,qo), xmax, xmin , scalexmax ,
scalexmin );
    %Optimization test%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    realmini(:,qo) = rounder(realmini(:,qo));
    [suppose(qo,:),opts11] = opttest(realmini(:,qo),
project_name, div, fdiv);


    %check with Neural Net
    neural_ans(qo,:) = fun(mini(:,qo));
    error(qo,:) = abs(neural_ans(qo,:) - suppose(qo,:));

    %Compare with Previous Optimal Solution
    if (suppose(qo,middle) <= mins11(middle)) &
(suppose(qo,middle) > 0.0001)
        mins11 = suppose(qo,:)';
        mins11
        nnmins11 = neural_ans(qo,:)';
        bestopts11 = opts11;
        bestmin = realmini(:,qo);
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%
    %Optimization test%%%%%%%%%%%%%%%%%%%%%%%%%%%
end

%TESTING THE Neural Network%%%%%%%%%%%%%%%%%%%%%%%%

%MSE OF Testing Procedure%%%%%%%%%%%%%%%%%%%%%%%

for j=1:3
```

```
    init_mse(j) = sum( ( suppose(:,j) - neural_ans(:,j) ).^2
)/length( neural_ans );
end
mse = sum(init_mse)/length(init_mse) ;
%MSE OF Testing Procedure%%%%%%%%%%%%%%%%%%%%%%%%%%%

figure(1)
plot(f,bestopts11,'k')
axis([min(f) max(f) 0 1])

[filler,five] = opttest(veronica, project_name , div, fdiv);
figure(2)
plot(f,bestopts11,'b')
hold on
plot(f,five,'r')
hold off
axis([min(f) max(f) 0 1])
title(['Comparison: Veronicas solution = red, neural sol =
blue'])

ex = cputime-tout

saving = ['save
',project_name,'div',num2str(div),'fdiv',num2str(fdiv)];
eval(saving)
```

B: Databasegetter.m

```
function Databasegetter(params , project_name)

t = cputime;

%Quickwave Paths%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%SimulatorPath = 'c:\Program
Files\QWED\QW_3D\v21Xmas\Local\Qw_sim\bin\ker1.exe';
SimulatorPath = 'C:\Program
Files\QWED\QW_3D\v22rev1\Local\Qw_sim\bin\ker1.exe';
%EditorPath = ['c:\Program
Files\QWED\QW_3D\v21Xmas\Local\Qw_edi\bin\zed.exe'];
EditorPath = ['C:\Program
Files\QWED\QW_3D\v22rev1\Local\Qw_edi\bin\zed.exe'];

Simulator  = SimulatorPath;
Editor     = EditorPath;

%Quickwave Paths%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Project Input Data%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
get_inputs = [project_name, '_input_data'];
eval(get_inputs);

%Project Input Data%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Create function to make databasepoints

num_vars = length(params(:,1));
fid = fopen( 'paramaker.m', 'wt');
dats=[' '];command = ['points(k,:) = ['];
x = ['function points = paramaker(params) '] ;    fprintf(fid,'%s
\n',x);
k = ['k = 1;  '];    fprintf(fid,'%s \n',k);
for j=1:num_vars
    fors = ['for i',num2str(j),'=params(',num2str(j),',:) '];
fprintf(fid,'\n %s',fors);
    dats = [dats,' i',num2str(j),' '];
end
command = [command,dats,']; k=k+1;'];   fprintf(fid,'\n
%s',command);
for j=1:num_vars
    ends = [' end '];   fprintf(fid,'\n %s',ends);
end
fclose( fid );
clear num_vars dats command x k j fors ends fid

%Create function to make databasepoints
points = paramaker(params);
```

```
for  i= 1:length(points(:,1))
    %Gives a vector of each of the parameters that quickwaves
will use
    idata = points(i,:);
    disp( idata);
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %construct the whole string needed to call the editor
    ModifyProject = [' -p"',Project,'" -i',' -m',' -e',' -q'];
    LoadProject = [' -p"',Project,'" -i',' -m',' -
o',num2str(Iter),' -i',' -e',' -q'];
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %construct the string needed to call the Simulator
    PerformSimulation = [ ' -t',' "', TaskerFile,'"'];
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %save parameters into a params file needed for run ant.udo
    f = fopen( ParsFile, 'wt');
    fprintf(f, '%f ', idata);
    fclose( f );
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %call Editor to modify the project
    run_qw1 =['!"',EditorPath,'"',ModifyProject];
    run_qw2 =['!"',EditorPath,'"',LoadProject];
    %Calls the editor to modify the project
    eval(run_qw1);
    eval(run_qw2);
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % modify Tasker file %%%%%%%%%%%%%%%%%%%%%%%%%%%%
    fid = fopen( TaskerFile, 'r+');
    il = 0;
    while ~feof(fid)
        il = il +1;
        data = fscanf( fid,'%c',1 );
        if ~isempty(data)
            TaskerCont(il) = data;
        end
        if il >13
            word = TaskerCont(il-12:il-1);
            if strcmp(word,'Save_Results') == 1
                point = il-1;
            end
        end

    end
    fclose( fid );
    presFile = ['A'];
    for j=1:length(idata)
        presFile = [presFile, 'A',num2str(idata(j))];
    end
    presFile = ifdec(presFile);
    presFile = [presFile,DatExt];

    TaskerCont = [ TaskerCont(1:point)];
    Filetype = ['QW_Pure'];
    fid = fopen( TaskerFile, 'w+');
    fprintf(fid,'%c',TaskerCont);
    fprintf(fid,'\n %s',presFile);
    fprintf(fid,'\n %s',Filetype);
    fclose( fid );
    % modify Tasker file %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    % call Simulator with the modified tasker file
    run_qws =['!"',SimulatorPath,'"',PerformSimulation]

    eval(run_qws);
end

e = cputime-t
```

C: Matrixgetter.m

```
function matrixgetter(params , project_name)

%first we have to set the initial data
%Gives the input for the specified project,

get_inputs = [project_name, '_input_data'];
eval(get_inputs);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


%Since the vector of frequencies is always the same we'll do it
above the loop

points = paramaker(params);
idata = points(1,:);
File = ['A'];
for j=1:length(idata)
    File = [File, 'A',num2str(idata(j))];
end
File = ifdec(File);
loading = ['load ',project_name,'/',File,DatExt];
eval(loading);
File = ifneg(File);
File = [project_name,'_',File];
x = [File,'(:,1)'];
f = eval(x);
f = f/10^9;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for  i= 1:length(points(:,1))

    %Gives a vector of each of the parameters that quickwaves
will use
    idata = points(i,:);
    %Loading and deleting each data file%%%%%%%%%%%%%%%%%%%%%%
    File = ['A'];
    for j=1:length(idata)
        File = [File, 'A',num2str(idata(j))];
    end
    File = ifdec(File);
    loading = ['load ',project_name,'/',File,DatExt];
    eval(loading);
    deleting = ['delete ',project_name,'/',File,DatExt];
    eval(deleting);
    %Loading and deleting each data file%%%%%%%%%%%%%%%%%%%%%%

    %Getting vectors of the s11 parameter%%%%%%%%%%%%%%%%%%%%%
    File = ifneg(File);
    File = [project_name,'_',File];
    y = [File,'(:,2)'];
    s11vector = eval(y);
    %Getting vectors of the s11 parameter%%%%%%%%%%%%%%%%%%%%

    %now we will make our vectors:
    %first is a matrix called variables; entailing the three
    %from the file name: Theta(t),s, and d in that order
    %first column theta, second s, and third d
    for l = 1:length(idata)
        vars(i,l) = idata(l);
    end

    %second is s parameters; each row represents the s parameters
    %for the first row of the variables matrix
    s11(i,:)  = s11vector';

    clearing = ['clear ',File];
    eval(clearing);
end

%save workspace%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
saving = ['save ',project_name,'data s11 f vars'];
eval(saving)
%save workspace%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

D: Ant_input.m

(Sample Input File)

```
%ANT%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%params row 1: Theta; row 2: slot; row 3: distance
keyf = [linspace(2.4, 2.5, fdiv)];

project_name = ['ant'];

params = [linspace(30 , 90, div); linspace(30 , 70,
div);linspace(30 , 120, div)];

%veronica = [ 52.45; 51.17; 33.62];
veronica = [63; 56; 118];
%ANT%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

E: Rounder.m

```
function y = rounder(x)

trunc = 100;

x = trunc*x;
x = round(x);
y = x/trunc;
```

F: Scalar.m

```
function x = scalar(y , xmax , xmin , ymax , ymin)

%This function will scale y into a the interval xmin xmax

if length(xmax) == 1 & length(xmin) == 1
    xmax = xmax*ones(length(y(:,1)),1);
    xmin = xmin*ones(length(y(:,1)),1);
end

if length(ymax) == 1 & length(ymin) == 1
    ymax = ymax*ones(length(y(:,1)),1);
    ymin = ymin*ones(length(y(:,1)),1);
end

for i = 1:length(y(:,1))
    for j = 1:length(y(1,:))
        x(i,j) = xmin(i) + (y(i,j) - ymin(i)) / (ymax(i) -
ymin(i)) * (xmax(i) - xmin(i));
    end
end
```

G: Opttest.m

```
function [opts11 , s11, f] = opttest(dum , project_name, div,
fdiv)
dum
Databasegetter(dum , project_name);
matrixgetter(dum , project_name)
loading = ['load ',project_name,'data'];
eval(loading);

get_inputs = [project_name, '_input'];
eval(get_inputs);

opts11 = ['['];
for pt = keyf
    [ m, ind ] = min( abs(f - pt) );
    opts11 = [opts11, ' s11(', num2str(ind),');'];
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

opts11 = [opts11,']'];
opts11 = eval(opts11)';

%plot(f,opts11)
```

H: Paramaker.m

```
function points = paramaker(params)
k = 1;

 for i1=params(1,:)
 for i2=params(2,:)
 for i3=params(3,:)
 points(k,:) = [  i1  i2  i3 ]; k=k+1;
  end
  end
  end
```

Bibliography

1. *Microwave Studio*^{TM}, CST, GmbH, Bad Nauheimer Str. 19, 64289 Darmstadt, Germany, http://www.cst.de/.
2. *QuickWave-3D*^{TM}, QWED, ul. Zwyciezcow 34/2, 03-938 Warsaw, Poland, http://www.qwed.com.pl/.
3. V.V. Yakovlev, "Commercial EM codes suitable for mode-ling of microwave heating - a comparative review," In: U. van Reinen, et al, Eds., *Scientific Computing in Electrical Engineering*, *Lecture Notes in Computational Sciences and Engineering*, vol. 18, Springer Verlag, pp. 87-96, 2001.
4. V.V. Yakovlev, Examination of contemporary electromagnetic software capable of modeling problems of microwave heating, In: *Advances in Microwave and High-Frequency Processing*, Springer Verlag, 2002, 13 p.
5. J.W. Bandler, N. Georgieva, et al., "A generalized space-mapping tableau approach to device modeling," *IEEE Trans. Microwave Theory Tech*., vol. MTT-49, no 1, pp. 67-79, Jan 2001.

6. J.W. Bandler, M.H. Bakr, et al., "A hybrid aggressive space mapping algorithm for EM optimization," *IEEE Trans. Microwave Theory Tech*., vol. MTT-47, no 12, pp. 2440-2440, Dec 1999.

7. J.W. Bandler, R.M., Biernacki, et al., "Yield-driven electromagnetic optimization via mutlilevel multidimensional models," *IEEE Trans. Microwave Theory Tech*., vol. MTT-41, no. 12, pp. 2269-2278, Dec 1993.

8. F. Alessandri, M. Mongiardo, and R. Sorrentino, "New efficient full wave optimization of microwave circuits by the adjoint network method," *IEEE Microwave Guided Wave Lett*., vol. 3, no 11, pp. 414-416, Nov 1993.

9. F. Alessandri, M. Dionigi, et al., "Rigorous and efficient fabrication-oriented CAD and optimization of complex waveguide networks," *IEEE Trans. Microwave Theory Tech*., vol. MTT-45, No 12, pp. 2366-2374, Dec 1997.

10. J.W. Bandler, J.E. Rayas-Sanchez, and Q.-J. Zhang, "Yield-driven electromagnetic optimization via space mapping-based neuromodels," *Int. J. RF and Microwave CAE*, vol. 12, no 1, pp.79-89, 2002.

11. *em*^{TM}, Sonnet Software, Inc., 1020 7th N. St., Suite 210, Liverpool, NY 13088, http://www.sonnetusa.com/.

12. M.H. Bakr, J.W. Bandler, et al., "Neural space-mapping optimization for EM-based design," *IEEE Trans. Microwave Theory Techn*., vol. MTT-48, No 12, pp. 2307-2315, Dec 2000.

13. P.M. Watson, and K.C. Gupta, "EM-ANN models for microstrip vias and interconnects in multilayer circuits", *IEEE Trans. Microwave Theory Techn.*, vol. MTT-44, no 12, pp.2495-2503, Dec 1996.
14. *HP-MDS*, Hewlett-Packard Co., Santa Rosa, CA, 1996.
15. J. Purviance, and M. Meehan, "CAD for statistical analysis and design of microwave circuits," *Int. J. RF and Microwave CAE*, vol. 1, pp. 59-76, 1991.
16. P.M. Watson, G.L. Creech, and K.C. Gupta, "Knowledge based EM-ANN models for the design of wide bandwidth CPW path/slot antennas," *1999 IEEE MTT S Int. Microwave Symp. Dig.*, pp. 2588-2591, June 1999.
17. V.A. Mechenova, *Method of Efficient Optimization of Microwave Systems*, M.S. Thesis, Worcester Polytechnic Institute, Worcester, MA, 2002.
18. F. Wang, and Q.-J. Zhang, "Knowledge-Based Neural Models for Microwave Design", *IEEE Transactions on Microwave Theory and Techniques*, vol. 45, no 12, Dec 1997.
19. P.M. Watson, C. Cho, and K.C. Gupta, "Electromagnetic-artificial neural network model for synthesis of physical dimensions for multiplayer asymmetric coupled transmission structures," *Int. J. RF and Microwave CAE*, vol. 9, pp. 175-186, 1999.
20. P. Burrascano, M. Dionigi, et al., "A neural network model for CAD optimization of microwave filters," *1998 IEEE MTT S Int. Microwave Symp. Dig.*, pp. 13-16, June 1998.
21. Hagan, M.T., Menhaj, M.B, "Training feedforward networks with the Marquardt algorithm," *IEEE Trans. Neural Networks*, vol. 5, no 6, pp. 989-993, Nov 1994.

22. F. Wang, V.K. Devabhaktuni, et al, "Neural network structures and training algorithms for RF and microwave applications," *Int. J. RF and Microwave CAE*, vol. 9, pp. 216-240, 1999.

23. Coleman, T.F, Lu, Y., "An interior trust region approach for nonlinear minimization Subject to Bounds," *SIAM Journal on Optimization*, Vol. 6, No. 2, pp. 418-445, May 1996

24. Coleman, T.F., Li, Y., "On the convergence of reflective Newton methods for large-scale nonlinear minimization subject to bounds", *Mathematical Programming*, Vol. 67, No. 2, pp. 189-224, 1994.

25. V.K. Devabhaktuni, M.C.E. Yagoub, et al, "Neural networks for microwave modeling: model development issues and nonlinear modeling techniques," *Int. J. RF and Microwave CAE*, vol. 11, No 1, pp. 4-21, 2001.

26. R.E. Collin, *Antennas and Radiowave Propagation*, McGraw-Hill Book Co., 1987.

27. K.-L. Wu, and H. Wang, "A rigorous modal analysis of H-plane waveguide T-junction loaded with a partial-height post for wide band applications," *IEEE Trans. Microwave Theory Techn.*, vol. MTT-49, no 5, pp. 893-901, June 2001.