

Circuit Model Parameter Extraction and Optimization for Microwave Filters

by

Dan Busuioc

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada 2002

©Dan Busuioc 2002

I hereby declare that I am the sole author of this thesis.

I authorize the University of Waterloo to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Dan Busuioc

Abstract

This thesis presents a method for parameter extraction of circuit elements from microwave filters. This diagnosis method can be applied to a sufficiently large number of filters and it can also be used in conjunction with a neural network model for filter design, greatly reducing development time. This thesis is an introduction of parameter extraction and circuit modelling through use of neural networks. It also presents an implementation of the proposed method as well as numerical results and validation data. Detailed implementation code is presented in the appendix.

Acknowledgments

I would like to thank my supervisor Professor Safavi-Naeini for his guidance, kindness and support. Many thanks to Mr. Amir Borji, without whom very little of this work would have been possible.

Finally I would like to extend my thanks to Professor Rifaat Mansour and Professor Anthony Vannelli for reading this thesis and providing me with valuable insight.

Dedication

To my father, Constantin, for your guidance, support, and love. Thank you for all the help you provided me with throughout my life.

Contents

1	Introduction	1
1.1	Overview	1
1.2	Our proposal	2
1.3	Outline	4
2	Background	5
2.1	Introduction	5
2.2	History	5
2.3	Current approach	7
3	Proposed Implementation	9
3.1	Introduction	9
3.2	Proposed subcomponents	9
3.3	Circuit function	10

3.4	File Parser	17
3.5	User Data File Parser	17
3.6	Measurement File Parser	19
3.7	Objective Function	19
3.8	Optimization Routine	21
3.9	Software Engineering Considerations	21
4	Numerical Results and Validation	23
4.1	Introduction	23
4.2	Circuit-designed 3-pole filter	24
4.3	EM Simulated 3-pole Filter	29
4.4	Industry Standard 6-pole filter	32
5	Neural Network Parameter Estimation	36
5.1	Introduction	36
5.2	Neural Networks	37
5.3	Model discussion	38
5.4	Numerical results	39
6	Conclusion and Future Work	43
6.1	Conclusion	43
6.2	Future Work	44

6.2.1	Engine Improvement	44
6.2.2	New Model Development	44
6.2.3	Model Inclusion	45
7	Appendix A	46
7.1	Parser	46
7.2	Code	46
7.3	Circuit function	55
7.4	Code	55
7.5	Objective function	59
7.6	Code	60
7.7	Optimization routine	62
7.8	Code	62

List of Figures

1.1	3 pole circuit model with in/out loops	3
2.1	Current methodologies for parameter extraction	8
3.1	3D view of combline filter structure	10
3.2	3 pole circuit model with in/out loops	11
3.3	Typical user input file	17
4.1	3 pole filter - design response	25
4.2	Coupling values between cavity resonators	25
4.3	3 Pole filter – circuit model	27
4.4	Optimization results for 3 pole filter	28
4.5	S-parameter comparison between ideal and extracted filters . .	29
4.6	3D view of 3-pole simulated filter	30
4.7	3 pole filter - EM simulation response	31
4.8	Optimization results for 3 pole EM simulated filter	31

4.9	S-parameter comparison between EM and extracted filters . . .	32
4.10	Industry 6-pole filter - measured response	33
4.11	6-pole filter physical layout	34
4.12	6-pole filter extraction results	35
5.1	Multilayer perceptron for input/output loop modelling	37
5.2	Generating the training and validation data for ANN	38
5.3	Generating the training and validation data for ANN	39
5.4	Mutual inductance obtained from full-wave solution and fast EM based model	40
5.5	Self inductance of the loop obtained from two methods	40
5.6	Parasitic capacitance obtained from two methods	41
5.7	Resonant frequency of the basic resonator as the excitation loops are introduced	41

Notations

- j always denotes the purely imaginary number $(0, 1)$.
- $jw = j\omega$ and they are used alternately in this text.

Table of Abbreviations

ADS	Advanced Design System software by Agilent Technologies
ANN	Artificial Neural Network
HFSS	High Frequency Software Simulator by Ansoft
SQP	Sequential Quadratic Programming

Chapter 1

Introduction

1.1 Overview

High performance RF/microwave filters are among the most critical components in the present and next generation wireless systems and their design optimization is a challenging task for successful design and operation of the entire system [1].

Hybrid optimization approaches are currently the most effective methods for analysis, design, and diagnosis of complex microwave circuits, such as complex branching filter structures widely used in wireless infrastructure.

Current research in the area of hybrid methods starts by defining specific sets of circuit models.

In the second step in the conventional hybrid approaches is taken by finding a mapping between the *coarse model* (such as the circuit model) parameter

space and the *fine model*. This is done by such techniques as Space Mapping or Artificial Neural Networks (ANNs) [2, 3, 4].

In this study, our main goal is to develop a general parameter extraction algorithm for RF/microwave structures[1, 5]. We also show how we can extend this method for design model generation through the use of neural networks.

1.2 Our proposal

In this thesis we talk about a way to extract the circuit parameters (R, L, C) from a measured or simulated filter response. We only shortly present the method of using a neural network to create a model from such results.

The use of a neural network for specific parameters of the filter has been recently investigated for input and output loop couplings [6]. Such a development is time consuming and is beyond the scope of this thesis. However given enough parameter extractions we will show how such a model can be developed in Chapter 5.

For our method we propose a software implementation to perform parameter extraction through optimization. The requirements for this are:

- measurement or simulation of physical filter
- starting approximation for the circuit parameters to be extracted

The program starts with the circuit model (which in this case has been developed for combline-type filters, with input and output coupling loops) and

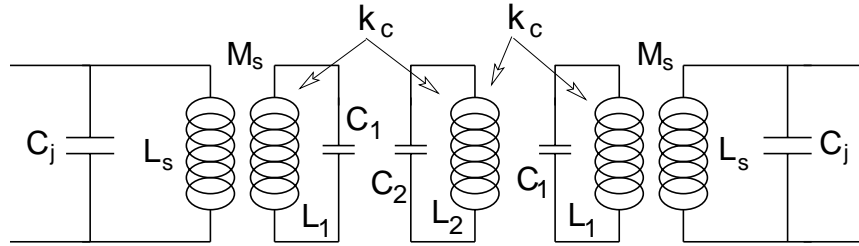


Figure 1.1: 3 pole circuit model with in/out loops

computes the output S parameters. The output S parameters are then compared to the measured S parameters. An error function is created through this process.

In figure 1.1 we show a typical 3-pole circuit model. Note that in this case the input and output loops are included. For the loops we consider an inductance L_s - the self-inductance of the loop, and a capacitance C_j in parallel, as can be seen in the figure.

We then use the NAG routine E04UCF to change the error function by changing the circuit parameter values towards values that minimize the error function. This is done through the sequential quadratic programming method (SQP) [7].

Upon a series of such extractions we can use the results as the training and validation data for a proposed artificial neural network (ANN). The details of how this can be accomplished, as well as a discussion of an actual problem are given in a later Chapter 5.

In the next chapter we develop the requirements for our proposed method.

1.3 Outline

In Chapter 2 we discuss the current methods for parameter extraction and what the proposed method does. In Chapter 3 we detail the proposed implementation and our goal, and we propose some implementation details. In Chapter 4 we show the validation of our method, including numerical results. In Chapter 5 we extend the discussion with the use of a generalized Artificial Neural Network for mapping and model derivation. We conclude in Chapter 6 and discuss the possibility and requirements of future work. The full implementation code is also given in the appendix.

Chapter 2

Background

2.1 Introduction

In this chapter we discuss the need for parameter extraction in the development of microwave filters. Further, we talk about the current methods for such work, and we introduce the basics of our proposed method.

2.2 History

Design and synthesis of different multiply-coupled resonator filter type structures has been the subject of extensive research [8, 5, 9, 10].

In the current design of such filters we are presented with the problem of having a fast, accurate method for the development and modelling of such filters. One requirement when designing such devices is the development of

an equivalent circuit model for the given filter. Once a model is developed, it can be used to extract circuit element values from measured or simulated physical filters. These values can be used as training and validation data for a neural network. Having a trained neural network reduces development time for the end user by providing a mapping between circuit element values and physical dimensions and vice-versa.

The development of equivalent circuit models are usually time intensive. There have been a few methods proposed [5, 11, 12]. Usually the user starts with the formal synthesis of the filter. In this process, the specifications (center frequency, bandwidth, etc.) are defined. Further, the user chooses a desired (physical) layout for the filter and computes the desired couplings for – in the case of cavity filters – the aperture windows between cavities.

Upon completion of this computation approximate models [1] can be used to translate circuit models into actual physical models. These physical models are then sent for manufacturing.

An important requirement has been to translate a physical implementation of a filter into a circuit equivalent model. This entitles the reverse engineering of a measured or simulated model. If the circuit elements from such a model are obtained, a neural network model can be created to aid in the design, tuning, diagnosis and modelling future filters.

Currently, there are very few approximate models that are available for translating circuit element values into their physical representation. These approximate models are valid for very limited cases.

It is our contention that using a large enough sample size for different fil-

ters will allow the creation of a model which could allow the development of microwave filters to be done much more rapidly and with a greater accuracy. Having such a model extraction can aid in the tuning and diagnosis of microwave filters.

2.3 Current approach

Currently there are two ways of mapping physical parameters into frequency response of the actual filter in the design and diagnosis process.

The first of such methods uses a space mapping technique. In this case the space of physical parameters and frequency are directly mapped into the response of the circuit (such as the S-parameters). This can be done through the use of multi-layer neural networks [2, 6]. A multilayer perceptron network (MLP) is one of the most popular types of neural networks. It is capable of approximating generic classes of functions including continuous and integrable functions [2].

The second of such methods – and the one we will investigate here – uses an intermediary circuit model. Physical parameters are mapped into resistor, capacitor, and inductor values of an equivalent circuit through the use of some parametric models. These models are either EM base analytical models or purely functional approximations like curve-fitting or artificial neural networks.

In Figure 2.1, X_c represents the vector of coarse model parameters. Input of this vector to the circuit model provides the response R_c of the filter. R_f

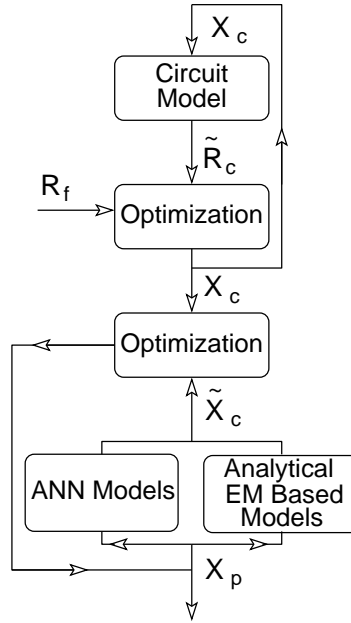


Figure 2.1: Current methodologies for parameter extraction

represents the fine model response of the filter. Through the optimization, we have a new set of coarse model parameters. The entire process can be repeated until required error has been achieved. The error in the system is always reduced. The output X_c parameters can also be optimized against some stored parameters X_c . These parameters are the result of previous model generation and are either ANN or EM based.

The goal of such methods is to reduce the development time of a filter. For the average filter the physical dimensions can be derived directly from the diagnosed model. However, in the odd cases where there is a requirement for an uncommon configuration filter, rigorous models must still be used.

Chapter 3

Proposed Implementation

3.1 Introduction

In this chapter we discuss the proposed implementation. We start by discussing the requirements for the parameter extraction software. We then discuss each of the modular components that create our program. Finally, we quickly discuss the software details and show why our implementation provides an advantage over other existing software.

3.2 Proposed subcomponents

For our parameter extraction program we have the following components:

- circuit model of device to analyze
- objective function construction

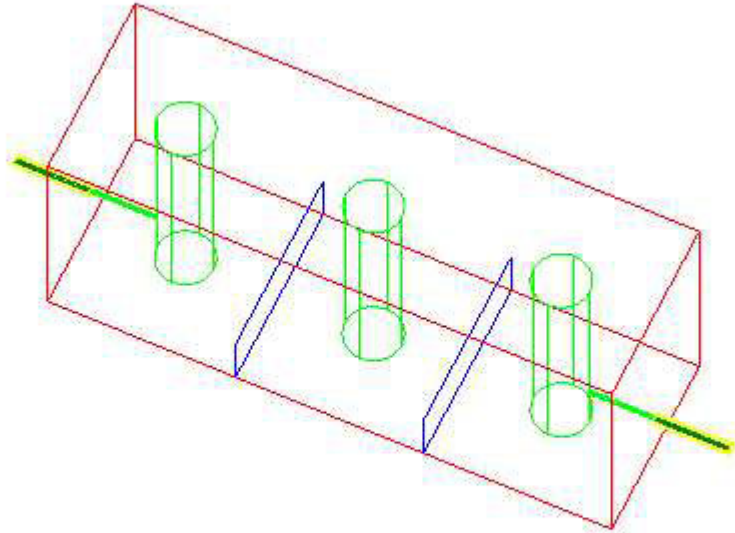


Figure 3.1: 3D view of combline filter structure

- input file parser
- optimization routine

3.3 Circuit function

We proceed in deriving a circuit equivalent model for a filter based on loop equations. We do this for a filter structure with input and output coupling loops.

In current design software created at University of Waterloo, ideal transformers are used for input and output. This greatly affects the system response.

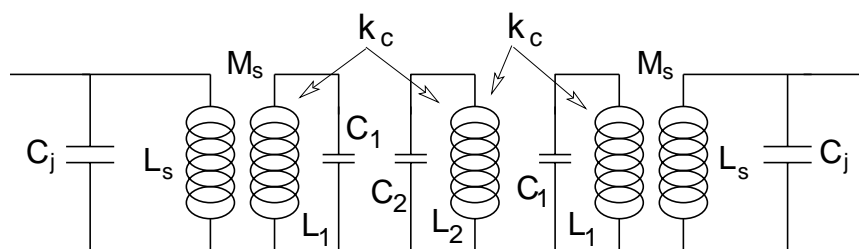


Figure 3.2: 3 pole circuit model with in/out loops

Here we include equivalent circuit models for physical input/output loop couplings and discontinuities at coaxial cavity junctions.

A typical filter structure with 3 poles is given in figure 3.1. The input and output loops are present in such a structure.

Figure 3.2 represents a circuit equivalent model for the physical filter. We write the loop equations in the clockwise direction:

$$\begin{aligned}
I_1[R_s + \frac{1}{j\omega C_j}] + I_2[\frac{-1}{j\omega C_j}] &= E \\
I_1[\frac{-1}{j\omega C_j}] + I_2[\frac{1}{j\omega C_j} + j\omega L_s] + i_1[-j\omega M_s] &= 0 \\
I_2[-j\omega M_s] + i_1[j\omega L_1 + \frac{1}{j\omega C_1} + R_1] + i_2[j\omega M_{12}] + \dots + i_n[j\omega M_{1n}] &= 0 \\
i_1[-j\omega M_{12}] + i_2[j\omega L_2 + \frac{1}{j\omega C_2} + R_2] + i_3[j\omega M_{23}] + \dots + i_n[j\omega M_{2n}] &= 0 \\
\vdots & \\
i_1[j\omega M_{1n}] + i_2[j\omega M_{2n}] + \dots + i_n[j\omega L_n + \frac{1}{j\omega C_n + R_n}] + I_3[j\omega M_s] &= 0 \\
i_n[j\omega M_s] + I_3[j\omega L_s + \frac{1}{j\omega C_j}] + I_4[\frac{-1}{j\omega C_j}] &= 0 \\
I_3[\frac{-1}{j\omega C_j}] + I_4[R_L + \frac{1}{j\omega C_j}] &= 0
\end{aligned} \tag{3.1}$$

In the above equations, I_1 and I_4 are the loop currents at the source and at the load respectively, while i_i represent the loop currents inside resonating cavity i . As well, R_s and R_l are the resistances of the source and load, respectively.

Note that in 3.1 we already consider that

$$M_{ij} = M_{ji}, i \text{ and } j = 1, 2, \dots, n \tag{3.2}$$

This consideration is valid because of reciprocity. The coupling between two cavities is the same regardless whether we measure from cavity i to cavity j or vice-versa.

Note that for this development we allow any topology for the filter. The topology is set by specifying some initial value and a range for the M_{ij} values.

In current literature [13, 5], as well as the current software developed by University of Waterloo (RF/Microwave group) as part of a related project, L and C are normalized. This is done by having $w_0 = 1$ and $Z_0 = 1$ for each resonator. This leads to $L = C = 1$.

Current literature [13] also assumes equation 3.3 for narrow-band approximation.

$$wM_{ij} = w_0M_{ij}, i \text{ and } j = 1, 2, \dots, n \quad (3.3)$$

However, in our development we allow non-normalized values, that is L, C are different for each resonating cavity. As well since we are creating the objective function over a range of frequencies (see the section on objective function), we consider each frequency and we do not consider equation 3.3.

The effect of the normalization assumption in literature is is that when designing for the coupling matrix of the filter there are values for M_{ii} which represent the shift in center frequency of the resonator from an ideal case (where L and C would both be normalized to 1). This happens because each cavity does not resonate at the exact same frequency.

However, in our method, L and C are not normalized to 1, and they are allowed to be optimized. Hence $M_{ii} = 0$ for all our cases.

If we let

$$wL_i - \frac{1}{wC_i} = Z_i \left[\frac{w}{w_i} - \frac{w_i}{w} \right] = \lambda \quad (3.4)$$

where

$$Z_i = \sqrt{\frac{L_i}{C_i}}, w_i = \frac{1}{\sqrt{L_i C_i}}, \quad (3.5)$$

then we can further write the loop equations in matrix form

$$[Z][J] = [E] \quad (3.6)$$

where

$$[J] = \begin{pmatrix} I_1 \\ I_2 \\ i_1 \\ \vdots \\ i_i \\ \vdots \\ i_n \\ I_3 \\ I_4 \end{pmatrix} \quad [E] = \begin{pmatrix} E \\ 0 \\ \vdots \\ 0 \\ \vdots \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix}$$

and

$$[Z] = [R] + j(\lambda[I] - w[M]) \quad (3.7)$$

where

$$[R] = \begin{pmatrix} R_s & 0 & 0 & 0 & \dots & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & R_1 & 0 & \dots & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & R_2 & \dots & 0 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 0 & \dots & R_n & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & \dots & 0 & 0 & R_L \end{pmatrix}$$

and

$$[M] = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & \dots & \dots & 0 & 0 & 0 \\ 0 & 0 & -M_s & 0 & 0 & \dots & \dots & 0 & 0 & 0 \\ 0 & -M_s & 0 & M_{12} & M_{13} & \dots & \dots & M_{1n} & 0 & 0 \\ 0 & 0 & M_{12} & 0 & M_{23} & \dots & \dots & M_{2n} & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & M_{1n} & M_{2n} & M_{3n} & \dots & \dots & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \dots & \dots & 0 & M_s & 0 \\ 0 & 0 & 0 & 0 & 0 & \dots & \dots & M_s & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \dots & \dots & 0 & 0 & 0 \end{pmatrix}$$

The R_i in cavity i is related to the Q of the resonator by the simple equation

$$R_i = \frac{w_0 L}{Q_i} = \frac{Z_0}{Q_i} \quad (3.8)$$

where w_0 and Z_0 are given by equation 3.5 [14].

Using knowledge from [13] we can write the relationship between loop currents and filter responses.

We can define the network transfer function as

$$\frac{P_L}{P_s} = \frac{|I_4|^2 R_L}{|E|^2 / (4R_s)} = 4R_s R_L \frac{|I_4|^2}{|E|^2} \quad (3.9)$$

From this we can derive the transmission

$$S_{21} = 2\sqrt{R_s R_L} \frac{I_4}{E} \quad (3.10)$$

and reflection

$$S_{11} = 1 - \frac{2R_s}{E/I_1} \quad (3.11)$$

functions respectively.

Hence we have derived the system response in terms of the circuit elements. We realize only I_1 and I_4 are needed to calculate the system response.

All that is needed to obtain the circuit response for one given frequency is an inversion of the matrix Z . Due to the fact that practical filter size is relatively small ($order \leq 20$), we do not have a matrix larger than 24 elements to invert. Hence in this thesis we will not consider fast matrix inversions such as ones based on the Cayley-Hamilton Theorem [15] but instead use a simple Gauss-Jordan elimination method for complex matrices.

```

//two lines of comments
//specify f in GHz, C and L in nanoF/H
0 //0 for CITI file, 1 for user file
ads_opt.txt //specification file
1 1 //optimize S11 S21
3 //degree of filter
50 //RE
50 //RL
1.58100 0 1.2 2 //Ms flag lower upper
5.05591 0 4 6 //Ls
4.21732 0 2.5 6.5 //Cj
0.04 0 0.03 0.05 //r1
0.04 0 0.04 0.04 //r2
0.04 0 0.03 0.05 //r3
30.1897 0 29 31 //L1
30 0 5 37 //L2
30.1882 0 29 31 //L3
1 0 0.001 1.5 //C1
1 0 0.001 1.5 //C2
1 0 0.001 1.5 //C3
0.598 0 0.598 0.598 //M12
0.001 0 0.001 0.001 //M13 - weak coupling
0.597 0 0.597 0.597 //M23

```

Figure 3.3: Typical user input file

3.4 File Parser

In this section we talk about two parsers. First we have a parser for the user data file. This data file specifies such things as degree of filter, starting values, and ranges for optimization of required parameters.

Second, we have the actual measurement file parser. This parser loads the measurement data from a standard file specified by the user.

3.5 User Data File Parser

In this section we present the data file parser. In Figure 5.2 we show a typical input file for a 3 pole filter.

First, the user has the choice of using a CITI file to specify the frequency

points, or to load the set of frequency points from a custom file. If a custom file is provided, the user must also specify the value of the S-parameters at those frequency points (S_{11} , and S_{21}).

The reason for having this option is that we might choose to reduce the optimization time by using a select set of frequency points. Usually, the CITI files which are output from network analyzer measurements contain more data than is necessary to extract the circuit parameters. Although this is not relevant for a few calculations, if we consider performing a large batch of parameter extractions in order to build an artificial neural network model, this work becomes tedious.

Second, the user must specify the data file. Either CITI file or custom frequency file is accepted.

After the data file, the flags for the optimization function are specified. User has a choice on whether to optimize for S_{11} , S_{21} , or both. This will become more clear in the section on the optimization function, which follows.

The next few sections of the input file contain the actual filter specification. They contain details about the circuit model. In our case (see Figure 5.2), the filter is a 3-pole. The impedance of the source ($R_E = R_s$) and load (R_L) are defined to be standard 50Ω .

Following this, the circuit elements of the actual filter are defined. As we can see in Figure 3.2 we consider a symmetric structure for our filter. Each of the input and output coupling loops are determined by an inductance (L_s) and a capacitance (C_j). As well the coupling between the input and output loops is represented by M_s .

Upon definition of the loop values, the resonator circuit values are defined. This can be seen in Figure 5.2.

Finally the coupling values between each resonator are defined. The actual coupling values also define the topology of the filter, for example if $M_{ij} = 0$ that implies there is no coupling between cavity i and j of the filter.

3.6 Measurement File Parser

In order to have the measurement data input into our parameter extraction program we need a method to parse the input data. In this section we quickly describe the parser design. Due to the fact that a parser is a software engineering tool, we consider the actual implementation details to be beyond the scope of this thesis. However, the full implementation is given in the appendix.

In order to standardize our input data, we choose the CITI file format as input for our program. This file is supported as both input and output by a general network analyzer, such as the HP 85XX vector network analyzers. It is as well supported by all major CAD and EM simulation software such as Agilent ADS and Ansoft HFSS.

3.7 Objective Function

Recall that our objective is to obtain a set of circuit parameters which provide a system response as close as possible to our desired (measured) response. In

the previous section we showed how to calculate the S-parameters from the circuit model. In this section we show how we get the error function from the two sets of S-parameters - one from the circuit model, and the other from measurements or EM simulations.

For our purpose, we consider the following optimization/error function

$$\varepsilon = optS_{11}\Sigma|S_{11measured} - S_{11circuit}|^2 + optS_{21}\Sigma|S_{21measured} - S_{21circuit}|^2 \quad (3.12)$$

In equation (3.12) $optS_{11}$ and $optS_{21}$ are specified by the user in the input file. This can be seen in the previous section.

The reason for having the option of optimizing on S_{11} or S_{21} (or both) gives more flexibility to the user. The effect of this can be seen in the results chapter.

As an alternative to using equation (3.12), we can use

$$\varepsilon = optS_{11}\Sigma(|S_{11measured}| - |S_{11circuit}|)^2 + optS_{21}\Sigma(|S_{21measured}| - |S_{21circuit}|)^2 \quad (3.13)$$

As we explain later in the Chapter 4, using equation (3.13) instead of (3.12) above can improve the convergence speed. This is due to the fact that a small variation in circuit elements from their ideal values introduce a relatively small magnitude error, but a large phase error. In equation (3.13) we only consider the magnitude of S_{11} and S_{21} .

3.8 Optimization Routine

For our optimization routine we use the NAG library E04UCF function. This function is designed to minimize an arbitrary smooth function subject to constraints - in our case simple lower and upper bounds on the variables - using a sequential quadratic programming (SQP) method [16].

The problem is required to be stated in the form:

$$\text{Minimize } F(x) \text{ subject to } l \leq \begin{pmatrix} x \\ A_L x \\ c(x) \end{pmatrix} \leq u \quad (3.14)$$

In this case $F(x)$ is the nonlinear objective function. A_L and $c(x)$ represent the linear and non-linear constraints respectively. In our case they are null, and we only consider the lower/upper bounds on the optimizing variables. The function approximates unspecified derivatives by finite differences.

3.9 Software Engineering Considerations

In the goal of every software engineer is to create a modular, expendable program. In this section we show how we meet these requirements with our design.

Our program is designed with modular, interconnecting components.

As we have shown at the beginning of this chapter, the composed of:

- circuit model of device to analyze
- optimization function
- input file parser
 - user data file
 - measurement file
- optimization routine

Each of the above components are modules (in the form of C/C++ functions) which integrate together to form the optimization program.

For one class of parameter extraction problems, we only need to replace the circuit model and the parser for the user data file, and maintain everything else the same. As long as the measurement is in the CITI format, the program will act as a 'black box' and extract the parameters for the problem.

Similarly, if we desire to use a new optimization routine (such as routines based on stochastic/genetic programming algorithms [11], all that is needed is to replace the optimization module and the program will extract the circuit parameters.

Chapter 4

Numerical Results and Validation

4.1 Introduction

In this chapter we use the previously introduced parameter extraction program to solve some specific examples. We will specifically use it for:

- A forward, circuit-model designed 3-pole filter
- An EM simulated 3-pole filter
- A measured industry-standard 6-pole filter

4.2 Circuit-designed 3-pole filter

In this section we design a 3 pole filter with $Degree = 3$, $f_0 = 918.88MHz$, $Bandwidth = 15MHz$, and $ReturnLoss = 25dB$

Using Cavity - a software available to the university as part of a related research project [1, 5], we design the filter and obtain the response. The software provides the coupling matrix values as its output, as well as the S-parameters of the filter.

As well if we choose a filter having a Q of approximately 4500, using a previous chapter formula, we have:

$$R = \frac{Z_0}{Q} = \sqrt{\frac{L}{C}} \frac{1}{Q} = \sqrt{\frac{30}{0.001}} \frac{1}{4500} \approx 0.04 \quad (4.1)$$

This represents the value of the resistor in each of the cavities. Note that for small variations this resistor does not greatly affect the system response.

We assume coupling between cavities 1-2 and 2-3. There is also a small parasitic coupling 1-3.

$$f_0 = \frac{w_0}{2\pi} = \frac{1}{2\pi\sqrt{LC}} = \frac{1}{2\pi\sqrt{3e-20}} = 918.88MHz \quad (4.2)$$

Note that M_{13} is very small like we expected. We use these values along with the R, L, C values for each resonator (from equations 4.2 and 4.1) in our software and we extract the input/output coupling values, as well as adjust the circuit element values. The result of this optimization is given in Figure 4.4

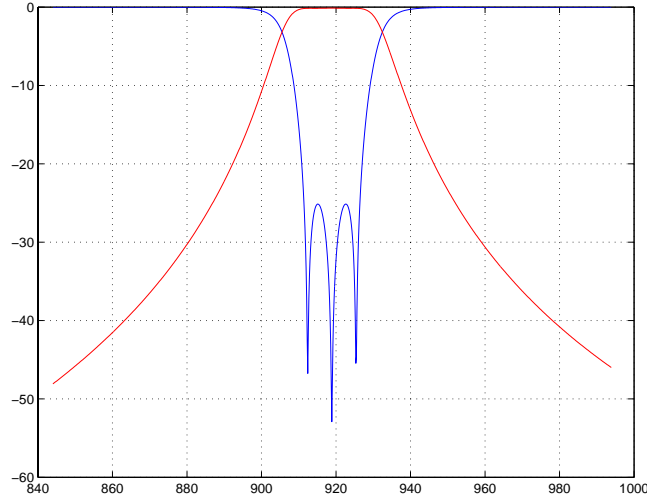


Figure 4.1: 3 pole filter - design response

Coupling	normalized	nH
M_{12}	1.2197163	0.597
M_{23}	1.2199133	0.598
M_{13}	0.0000002	0.001

Figure 4.2: Coupling values between cavity resonators

Also, transforming the Cavity coupling values:

$$M_{ij}(nH) = L * k_{ij}(Cavity) \quad (4.3)$$

Hence the we can summarize the coupling values in Figure 4.2.

Using the approximate models already implemented [1][11] we derive the physical dimensions for our structure. The cavity cross section is $50 \times 50 \text{ mm}$ and it has a height of 50 mm . The center resonator is 49 mm and it has a diameter of 10 mm . There is a 1 mm gap between the top of the resonator

and the top of the cavity.

The calculated aperture windows (cavity 1-2 and 2-3) are $30.9mm$ (width) and $29.7mm$ (height) each. Designing the loop dimensions, we get the input/output loops to be $12.4mm$ (width) and $14.8mm$ (height).

With the S-parameters and our coupling values, we use our method to extract the loop coupling values, and optimize the circuit. The reason for this is that in Cavity the input and output to the filter is represented by transformers which do not exist in practice.

We use the circuit model given by Figure 4.3 in ADS to plot the response of the filter. The values are to be input from our optimization result. The results of our method are summarized in Figure 4.4.

In this experiment we want to extract only the coupling values and optimize the first and last resonator of our structure - where the frequency shift will be most noticeable.

Based on our design we choose the starting values for each resonator to be $L = 30nH$ and the capacitance to be $C = 1pF$. This gives a filter design frequency of 918.88 MHz, as can be seen in equation (4.2).

We choose to optimize in 3 iterations. In this experiment, this number of optimizations represents an optimum value between convergence of results and computation time. For the first iteration we fix all parameters except the input/output loop elements (M_s , L_s and C_j). The program converges to a closer solution as can be seen by the results of the first iteration (It_1) in the table.

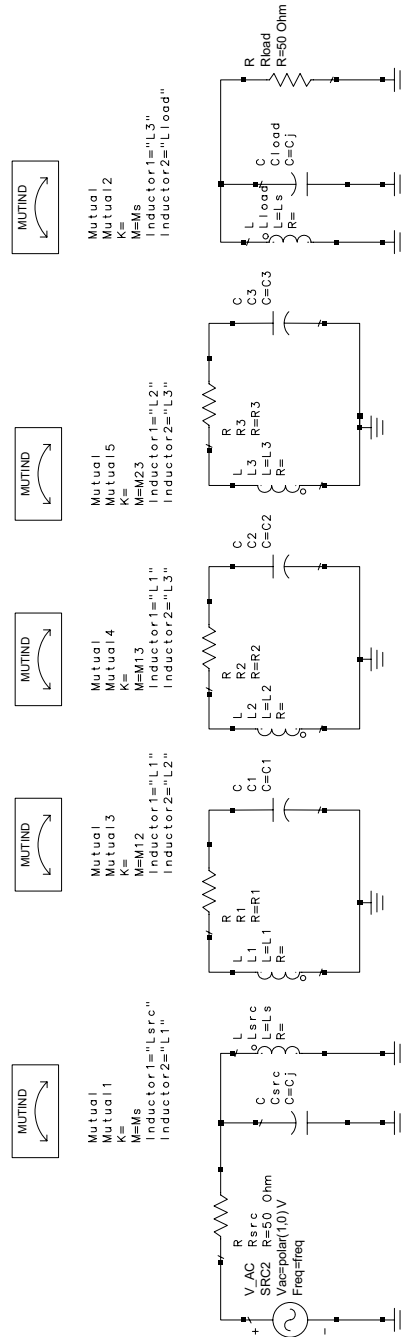


Figure 4.3: 3 Pole filter – circuit model

El. (unit)	Opt.	Start	LB	UB	Final
$M_s(nH)$	yes(1)	1	0.5	2.5	1.58100
$L_s(nH)$	yes(1)	4	2.5	6	5.05591
$C_j(pF)$	yes(1)	3	2.5	6.5	4.21732
$R_1(\Omega)$	no	0.04	N/A	N/A	0.04
$R_2(\Omega)$	no	0.04	N/A	N/A	0.04
$R_3(\Omega)$	no	0.04	N/A	N/A	0.04
$L_1(nH)$	yes(2)	30	29	31	30.1897
$L_2(nH)$	no	30	N/A	N/A	30
$L_3(nH)$	yes(2)	30	29	31	30.1882
$C_1(pF)$	no	1	N/A	N/A	1
$C_2(pF)$	no	1	N/A	N/A	1
$C_3(pF)$	no	1	N/A	N/A	1
$M_{12}(nH)$	no	0.598	N/A	N/A	0.598
$M_{13}(nH)$	no	0.001	N/A	N/A	0.001
$M_{23}(nH)$	no	0.597	N/A	N/A	0.597

Figure 4.4: Optimization results for 3 pole filter

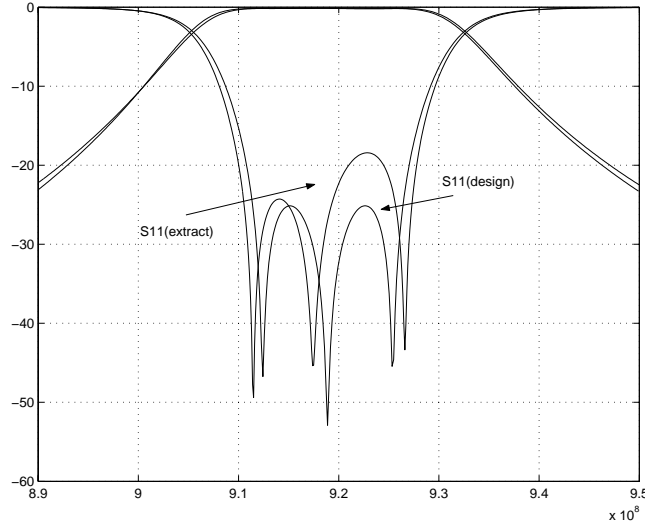


Figure 4.5: S-parameter comparison between ideal and extracted filters

Next, we fix the new coupling values and choose to optimize the first and last resonators of the structure. Optimizing for the inductance of each of these resonators, we have $L_1 = 30.1897nH$ and $L_3 = 30.1882nH$.

Finally we fix all parameters except for the input/output loop elements and re-run our method. This provides a very slight change in the loop element values. The final results are given in Figure 4.4

In Figure 4.5 we compare the S-parameter results between the ideal filter (designed) and the extracted (optimized) case.

4.3 EM Simulated 3-pole Filter

In this section we show how our method extracts the circuit parameters from an actual EM simulator.

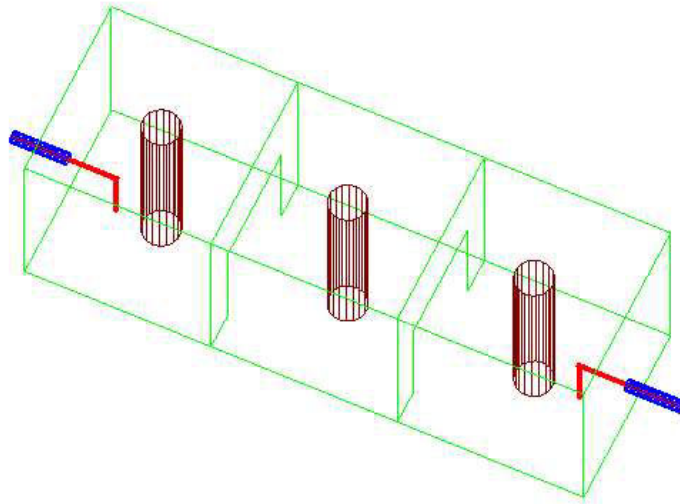


Figure 4.6: 3D view of 3-pole simulated filter

We use the design from the previous section and draw the filter in Ansoft HFSS. A picture of this is shown in Figure 4.6. The response of the filter after simulation is shown in Figure 4.7

We proceed as previously. This time we optimize over more variables and with more iterations. The reason for this is that in the EM model there are additional effects that can change the resonant frequency of each resonator or the Q .

The results and iteration table are given in Figure 4.8. The S parameter response is given in Figure 4.9. In this case we have chosen to start by iterating on the loop coupling values.

After we reduce the error by obtaining newer values, we optimize each resonator equivalent resistor value.

Next we optimize the resonant frequency of each resonator, and our results

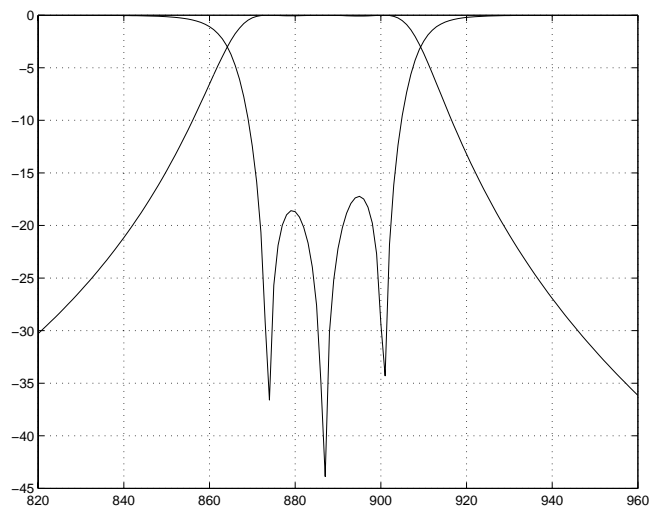


Figure 4.7: 3 pole filter - EM simulation response

El. (unit)	Opt.	Start	LB	UB	Final
$M_s(nH)$	yes(1)	2	1.5	2.6	2.22770
$L_s(nH)$	yes(1)	7.5	5	9	5.80688
$C_j(pF)$	yes(1)	5	2.5	8	5.63115
$R_1(\Omega)$	yes(2)	0.04	0.03	0.06	0.05
$R_2(\Omega)$	yes(2)	0.04	0.03	0.06	0.04
$R_3(\Omega)$	yes(2)	0.04	0.03	0.06	0.05
$L_1(nH)$	yes(3)	30	25	35	32.3879
$L_2(nH)$	yes(3)	30	25	35	32.1954
$L_3(nH)$	yes(3)	30	25	35	32.3889
$C_1(pF)$	yes(3)	1	0.9	1.1	1.02402
$C_2(pF)$	yes(3)	1	0.9	1.1	1
$C_3(pF)$	yes(3)	1	0.9	1.1	1.02404
$M_{12}(nH)$	yes(4)	1	0.5	1.5	1.1
$M_{13}(nH)$	yes(4)	0.01	0.001	0.1	0.08
$M_{23}(nH)$	yes(4)	1	0.5	1.5	1.1

Figure 4.8: Optimization results for 3 pole EM simulated filter

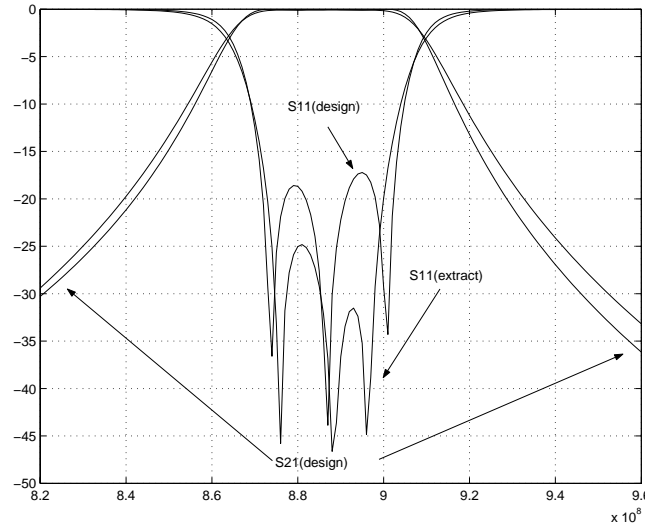


Figure 4.9: S-parameter comparison between EM and extracted filters

are consistent with the ones in the previous section. We then optimize the coupling values between each of the cavity resonators, and we finalize by one more iteration on the loop circuit model in order to obtain a better response.

4.4 Industry Standard 6-pole filter

Finally in this section we show an application of our method to industry.

Six pole filters are commonly used in a variety of base stations for mobile communications. We present a six pole filter from one of today leading manufacturers. The measured response using a common network analyzer is given in Figure 4.10. Please note the noisy response in this case.

Current diagnosis software performs a model based parameter estimation from the measured system response. This provides a function estimation to

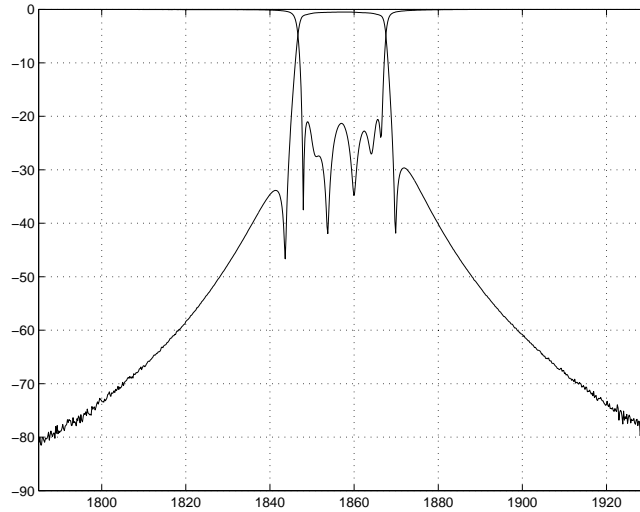


Figure 4.10: Industry 6-pole filter - measured response

the noisy data, which then is used to reverse the design process and extract the coupling values.

In our case we do not need a functional estimation of our data, but rather proceed with the parameter extraction from the measured response directly. It is also important to note that our method is more general and it directly links a user predefined circuit function to the system response - as we have shown in the previous chapter.

The design specifications for this filter are $f_0 = 1857.5\text{MHz}$ and $\text{bandwidth} = 15\text{MHz}$. There are also two transmission zeros at $f_1 = 1845\text{MHz}$ and $f_2 = 1870\text{MHz}$. These are representative of a coupling between cavity 2 and 5. Hence for our analysis the topology of the filter is given in Figure 4.11.

The results of the final 5 iterations of our method are given in Figure 4.12

	C_1	C_2	C_3	C_4	C_5	C_6
C_1	0	1	0	0	0	0
C_2	1	0	1	0	1	0
C_3	0	1	0	1	0	0
C_4	0	0	1	0	1	0
C_5	0	1	0	1	0	1
C_6	0	0	0	0	1	0

Figure 4.11: 6-pole filter physical layout

El. (unit)	Opt.	Start	LB	UB	It_1	It_2	It_3	It_4	It_5	Final
$M_s(nH)$	yes(1)	3.5	1	7	4.3231	N/A	N/A	N/A	4.4295	4.4295
$L_s(nH)$	yes(1)	6	1	10	7.3421	N/A	N/A	N/A	6.9688	6.9688
$C_j(pF)$	yes(1)	5	3	7	5.1326	N/A	N/A	N/A	4.9523	4.9523
$R_1(\Omega)$	yes(2)	0.05	0.03	0.1	N/A	0.06521	N/A	N/A	N/A	0.06521
$R_2(\Omega)$	yes(2)	0.05	0.03	0.1	N/A	0.05545	N/A	N/A	N/A	0.05545
$R_3(\Omega)$	yes(2)	0.05	0.03	0.1	N/A	0.05309	N/A	N/A	N/A	0.05309
$R_4(\Omega)$	yes(2)	0.05	0.03	0.1	N/A	0.05237	N/A	N/A	N/A	0.05237
$R_5(\Omega)$	yes(2)	0.05	0.03	0.1	N/A	0.05510	N/A	N/A	N/A	0.05510
$R_6(\Omega)$	yes(2)	0.05	0.03	0.1	N/A	0.06489	N/A	N/A	N/A	0.06489
$L_1(nH)$	yes(3)	14.6829	13	16	N/A	N/A	14.9837	N/A	N/A	14.9837
$L_2(nH)$	yes(3)	14.6829	13	16	N/A	N/A	14.6903	N/A	N/A	14.6903
$L_3(nH)$	yes(3)	14.6829	13	16	N/A	N/A	14.6687	N/A	N/A	14.6687
$L_4(nH)$	yes(3)	14.6829	13	16	N/A	N/A	14.6879	N/A	N/A	14.6879
$L_5(nH)$	yes(3)	14.6829	13	16	N/A	N/A	13.9439	N/A	N/A	13.9439
$L_6(nH)$	yes(3)	14.6829	13	16	N/A	N/A	14.9543	N/A	N/A	14.9543
$C_1(pF)$	yes(3)	0.5	0.45	0.6	N/A	N/A	0.5011	N/A	N/A	0.5011
$C_2(pF)$	yes(3)	0.5	0.45	0.6	N/A	N/A	0.4984	N/A	N/A	0.4984
$C_3(pF)$	yes(3)	0.5	0.45	0.6	N/A	N/A	0.4910	N/A	N/A	0.4910
$C_4(pF)$	yes(3)	0.5	0.45	0.6	N/A	N/A	0.4939	N/A	N/A	0.4939
$C_5(pF)$	yes(3)	0.5	0.45	0.6	N/A	N/A	0.4995	N/A	N/A	0.4995
$C_6(pF)$	yes(3)	0.5	0.45	0.06	N/A	N/A	0.5019	N/A	N/A	0.5019
$M_{12}(nH)$	yes(4)	0.15	0.1	0.25	N/A	N/A	N/A	0.13066	N/A	0.13066
$M_{23}(nH)$	yes(4)	0.1	0.05	0.15	N/A	N/A	N/A	0.08762	N/A	0.08762
$M_{34}(nH)$	yes(4)	0.1	0.05	0.15	N/A	N/A	N/A	0.11205	N/A	0.11205
$M_{45}(nH)$	yes(4)	0.1	0.05	0.15	N/A	N/A	N/A	0.08230	N/A	0.08230
$M_{56}(nH)$	yes(4)	0.15	0.1	0.25	N/A	N/A	N/A	0.13206	N/A	0.13206
$M_{25}(nH)$	yes(4)	-0.01	-0.05	-0.005	N/A	N/A	N/A	-0.02838	N/A	-0.02838

Figure 4.12: 6-pole filter extraction results

Chapter 5

Neural Network Parameter Estimation

5.1 Introduction

This chapter is an extension of our work on parameter extraction.

In this chapter we talk about the application of neural networks to modelling fast coupling between circuit model and physical dimensions. We start by defining what neural networks are, and we continue by showing how we can apply their advantages to modelling a specific problem.

In our case we choose to characterize the excitation input and output loop models (L_s , M_s and C_j) which we extracted with our proposed method.

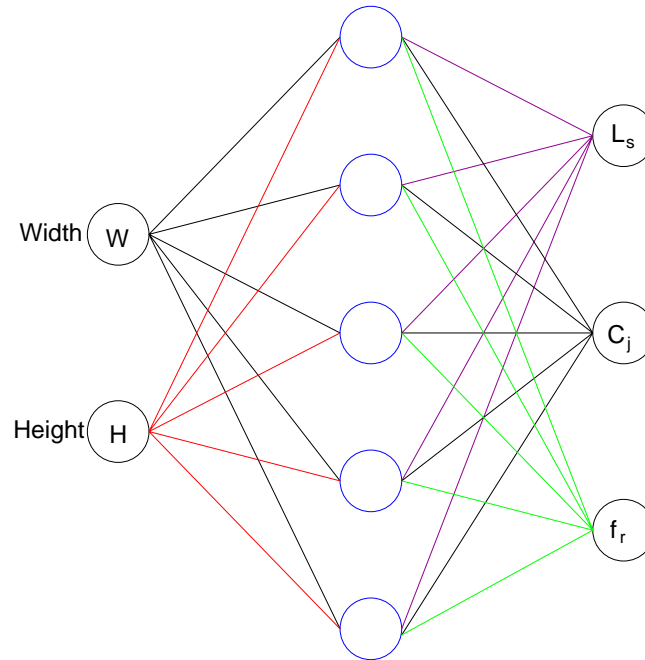


Figure 5.1: Multilayer perceptron for input/output loop modelling

5.2 Neural Networks

Neural networks are composed of simple elements operating in parallel. These elements are inspired by biological nervous systems. As in nature, the network function is determined largely by the connections between elements. We can train a neural network to perform a particular function by adjusting the values of the connections (weights) between elements.

Commonly neural networks are adjusted, or trained, so that a particular input leads to a specific target output. Such a situation is shown below. There, the network is adjusted, based on a comparison of the output and the target, until the network output matches the target. Typically many such

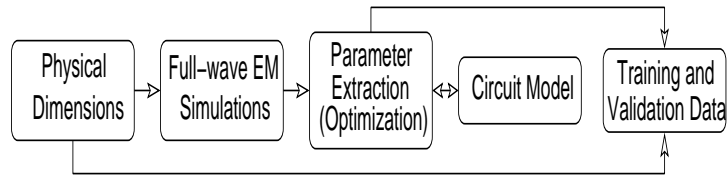


Figure 5.2: Generating the training and validation data for ANN

input/target pairs are used, in this supervised learning, to train a network.

5.3 Model discussion

In previous chapters we have shown how we can model the physical input/output loops by a parallel LC block.

For our analysis we have performed a total of 30 parameter extractions to train our neural network model. We choose a neural network composed of a multi-layer perceptron (MLP) with one hidden layer. This network, as shown in Figure 5.1 is used to model the self-inductance (L_s), parasitic capacitance (C_j) and mutual-inductance (M_s) of our input/output loops.

In Figure 5.2 we show a general neural network building procedure. The main work presented in this document is the parameter extraction/optimization (center block). In this chapter we discuss the last block of Figure 5.2 which represents the use of the training/validation data for creating the neural network model for this problem.

W	H	Ls (nH)		Cj (pF)		Fr (MHz)	
		ANN	Extracted	ANN	Extracted	ANN	Extracted
15.0	7.5	6.981	7.165	2.361	2.291	894.207	894.333
10.0	12.5	7.686	7.723	2.039	2.039	893.023	892.702
8.5	15.0	8.139	8.163	1.871	1.852	892.186	892.134
12.5	17.5	11.080	11.095	1.452	1.445	902.019	901.458
15	22.5	14.115	14.052	1.064	1.047	918.234	919.180

Figure 5.3: Generating the training and validation data for ANN

5.4 Numerical results

We use a rectangular coaxial cavity with two symmetric ports. The cross section of the cavity was square with dimensions of $a = b = 50mm$. The diameter of the center rod was $D = 15mm$ with the height of $L = 62mm$. There is also a $3mm$ air gap between the top of the center rod and the top of the cavity. The resonant frequency of the resonator (prior to introducing the loop excitations) was calculated with Ansoft HFSS to be $881MHz$.

In order to generate the training and validation data the same EM solver was used for different values of width W and H of the loop. We vary the height between 5 and 25 mm and the width between 7.5 and 17.5 mm. Note that when $W = 17.5mm$ the loop is a tap into the center post.

After generating the S-parameters from the EM solver, we extract the circuit parameters. We use 25 of our parameter extractions for training the network and 5 for validation. Figure 5.3 shows the extracted circuit values for a few EM simulations, as well as the circuit values which are obtained from the neural network after training.

In Figures 5.4 - 5.6 we show the response of our neural network for the specific training data that we provided.

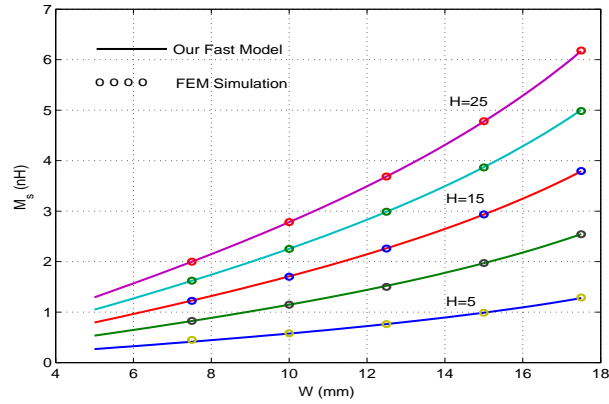


Figure 5.4: Mutual inductance obtained from full-wave solution and fast EM based model

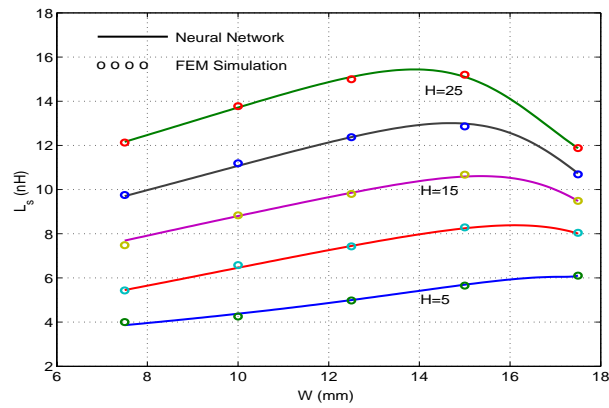


Figure 5.5: Self inductance of the loop obtained from two methods

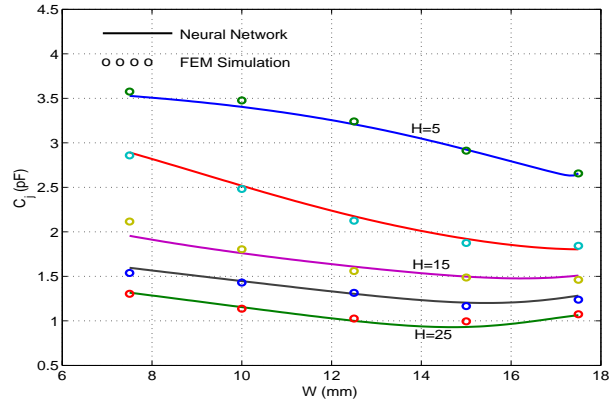


Figure 5.6: Parasitic capacitance obtained from two methods

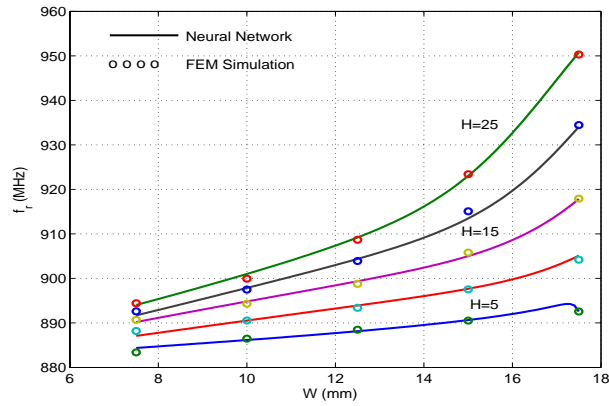


Figure 5.7: Resonant frequency of the basic resonator as the excitation loops are introduced

Figure 5.7 shows an extension of the neural network to modelling the resonant frequency shift of our resonators with the different loop sizes introduced.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this work we have shown a general method for parameter extraction. We started the work by considering current methodologies for modelling of microwave filters. We discussed the issues involved with current methods, as well as requirements for a more efficient method.

We followed the introduction by proposing a general parameter extraction method which is robust, fast, modular, and expandable. Following the development of the method, we applied to three current problems (circuit-designed filter, EM simulated filter, and industry-standard measured filter).

Finally we discussed an extension of our work through the use of neural networks. This has an advantage that we can readily implement the neural network model and obtain a tool for filter development and diagnosis.

6.2 Future Work

Future work can be summarized in three different sections:

- Engine improvement through the use of novel optimization techniques
- Development of new circuit models for filters and/or other microwave devices of interest
- Inclusion of generated models into current development software

6.2.1 Engine Improvement

The current engine is based on the SQP method and it fails for a high number of variables. Additional engines can be investigated using the requirements given in this work. They can be implemented easily due to the modularity of the program. This can ensure we have better and faster convergence of results, for higher complexity problems.

6.2.2 New Model Development

We can extend the use of our program by developing additional circuit models for required problems. Due to the modularity of the program these new models can be easily implemented and new circuit parameters extracted from simulations and/or measurements. Such an example would be the development of an antenna/probe coupling model for couplings in cavity filters.

6.2.3 Model Inclusion

The loop coupling model for cavity filters discussed in this work, as well as new models developed need to be included into current software to be useable. It remains to further research to obtain the best methods for including such models.

Chapter 7

Appendix A

7.1 Parser

This module inputs a standard CITI file and loads the S11 and S21 data values for all frequencies.

7.2 Code

```
#include <complex> #include <math.h> #include <stdio.h> #include  
<fstream.h>
```

```
using std::complex; using std::arg; using std::abs; using  
std::polar;
```

```
void analyzeData(istream inputFile, int *npoints,
                double *nFreq, complex<double> *S11,
                complex<double> *S21);

void analyzeUserData(istream inputFile, int *npoints,
                    double *nFreq, complex<double> *S11,
                    complex<double> *S21);

void analyzeData(istream inputFile, int *npoints,
                double *nFreq, complex<double> *S11,
                complex<double> *S21)
{
    char readStr[80];
    char seps[] = " ,\t\n";
    char *strRealCoeff, *strRealExp;
    char *strImaginaryCoeff, *strImaginaryExp;
    char *strReal, *strImag;
    double lRealCoeff, lImaginaryCoeff;
    int lRealExp, lImaginaryExp;
    double lReal, lImaginary;
    char *strPoints, *strtemp;

    //skip the initial lines in the file
    for(int i=0;i<16;i++) {
        inputFile.getline(readStr,80);
        strtemp = strstr(readStr, "VAR freq MAG");
```

```
        if (strtemp != NULL) {
            strtemp = strtok(strtemp, " ");
            strtemp = strtok(NULL, " ");
            strtemp = strtok(NULL, " ");
            strPoints = strtok(NULL, " ");
            *npoints = atoi(strPoints);
        }
    }

    //nFreq = new double[*npoints];
    //S11 = new complex<double>[*npoints];
    //S21 = new complex<double>[*npoints];

    inputFile.getline(readStr, 80);
    for (i = 0; i<*npoints; i++) {
        inputFile.getline(readStr, 80);
        nFreq[i] = atof(strtok( readStr, seps ));
    }

    inputFile.getline(readStr, 80); //this is VAR_LIST_END
    inputFile.getline(readStr, 80); //this is BEGIN

    for (i=0; i<*npoints; i++) {
        //if (i%10 == 0) printf(".");
        inputFile.getline(readStr, 80);
```

```
strReal = strtok(readStr, ",");
strImag = strtok(NULL, "\\t\\n");

strRealCoeff = strtok(strReal, "E");
strRealExp = strtok(NULL, ",");
strImaginaryCoeff = strtok(strImag, "E");
strImaginaryExp = strtok(NULL, "\\t\\n");

lRealCoeff = atof(strRealCoeff);
if (strRealExp != NULL) {
    lRealExp = atoi(strRealExp);
}
else {
    lRealExp = 0;
}

lImaginaryCoeff = atof(strImaginaryCoeff);
if (strImaginaryExp != NULL) {
    lImaginaryExp = atoi(strImaginaryExp);
}
else {
    lImaginaryExp = 0;
}
```

```
    lReal = lRealCoeff * pow(10,lRealExp);
    lImaginary = lImaginaryCoeff * pow(10,lImaginaryExp);
    S11[i] = complex<double>(lReal, lImaginary);
}
```

```
inputFile.getline(readStr, 80); //this is END
inputFile.getline(readStr, 80); //this is BEGIN
```

```
for (i= 0; i<*npoints; i++) {
    //if (i%10 == 0) printf(".");
    inputFile.getline(readStr,80);

    strReal = strtok(readStr, ",");
    strImag = strtok(NULL, "\t\n");

    strRealCoeff = strtok(strReal, "E");
    strRealExp = strtok(NULL, ",");
    strImaginaryCoeff = strtok(strImag, "E");
    strImaginaryExp = strtok(NULL, "\t\n");

    lRealCoeff = atof(strRealCoeff);
    if (strRealExp != NULL) {
        lRealExp = atoi(strRealExp);
    }
    else {
```

```
        lRealExp = 0;
    }

    lImaginaryCoeff = atof(strImaginaryCoeff);
    if (strImaginaryExp != NULL) {
        lImaginaryExp = atoi(strImaginaryExp);
    }
    else {
        lImaginaryExp = 0;
    }

    lReal = lRealCoeff * pow(10,lRealExp);
    lImaginary = lImaginaryCoeff * pow(10,lImaginaryExp);
    S21[i] = complex<double>(lReal, lImaginary);
}
}

void analyzeUserData(ifstream inputFile, int *npoints,
                    double *nFreq, complex<double> *S11,
                    complex<double> *S21)
{ }

//this is particular to this circuit model
void analyzeInput(ifstream inputFile, int *npoints,
                 int *obj_S11, int *obj_S21,
```

```
        double *USER, double *BL, double *BU,
        double *nFreq, complex<double> *S11,
        complex<double> *S21)
{
    char readStr[80];
    char seps[] = " ,\t\n";
    char *strTemp, *strOpt;
    char *strUpper, *strLower;
    int userData;
    int n;

    //skip the initial lines in the file
    for(int i=0;i<2;i++) {
        inputFile.getline(readStr,80);
    }

    inputFile.getline(readStr,80);
    strTemp = strtok(readStr, "\t");
    userData = atoi(strTemp);

    inputFile.getline(readStr,80);
    strTemp = strtok(readStr, "\t");

    //get the data file
    ifstream inputDataFile(strTemp);
```

```
if (userData == 0) {
    analyzeData (inputDataFile, npoints, nFreq, S11, S21);
}
else {
    analyzeUserData (inputDataFile, npoints, nFreq, S11, S21);
}
inputDataFile.close();
//end of get data

inputFile.getline(readStr,80);
strTemp = strtok(readStr," ");
*obj_S11 = atoi(strTemp);
strTemp = strtok(NULL, "\t");
*obj_S21 = atoi(strTemp);

inputFile.getline(readStr,80);
strTemp = strtok(readStr, " ");
n = atoi(strTemp);

/*USER = new double[3*n + n*(n-1)/2 + 8];
BL = new double[3*n + n*(n-1)/2 + 3];
BU = new double[3*n + n*(n-1)/2 + 3]; */

USER[0] = n;
USER[1] = NULL; //to be used as f later
```



```
USER[2] = NULL;

inputFile.getline(readStr,80);
strTemp = strtok(readStr, " ");
USER[3] = atof(strTemp); //RE

inputFile.getline(readStr,80);
strTemp = strtok(readStr, " ");
USER[4] = atof(strTemp); //RL

for (i=5; i< 3*n + n*(n-1)/2 + 8; i++){
    inputFile.getline(readStr,80);
    strTemp = strtok(readStr," ");
    USER[i] = atof(strTemp);
    strOpt = strtok(NULL, " ");
    if (atoi(strOpt) == 0){
        BL[i-5] = USER[i];
        BU[i-5] = USER[i];
    }
    else {
        strLower = strtok(NULL, " ");
        strUpper = strtok(NULL, " ");
        BL[i-5] = atof(strLower);
        BU[i-5] = atof(strUpper);
    }
}
```

```
    }  
}
```

7.3 Circuit function

This module creates the loop equations for an n-pole filter. It assumes a model for the input/output loop which is discussed earlier.

This function makes use of a matrix inversion routine which is not provided here due to copyright purposes.

7.4 Code

```
#include <complex> #include <math.h>  
  
using std::complex; using std::arg; using std::abs; using  
std::polar;  
  
double PI = 3.141592654;  
  
extern int cmatinv(complex<double> **a,int n);  
  
void circfun(double *X, double *USER, complex<double> *S11,  
complex<double> *S21) {  
    int i;
```

```
double w;

int n = USER[0];
double f = USER[1] / 1E9;
double f0 = USER[2]; //not used
double RE = USER[3];
double RL = USER[4];
double Ms = X[0];
double Ls = X[1];
double Cj = X[2]/1E3;

double *R_vector = new double[n];
for (i=0; i<n; i++) {
    R_vector[i] = X[i+3];
}

double *L_vector = new double[n];
for (i=0; i<n; i++) {
    L_vector[i] = X[i+n+3];
}

double *C_vector = new double[n];
for (i=0; i<n; i++) {
    C_vector[i] = X[i+2*n+3]/1E3;
}
```

```
double *M_vector = new double[n*(n-1)/2];
for (i=0; i<n*(n-1)/2; i++) {
    M_vector[i] = X[i+3*n+3];
}

int indexM = 0;
complex<double> temp;
int success;

complex<double> **pA;

w = 2*PI*f;

pA = new complex<double>* [n+4];

for(i = 0; i < n+4; ++i) {
    pA[i] = new complex<double>[n+4];
}

for(i = 2; i < (n+2); ++i) {
    for(int j = i+1; j<(n+2); ++j) {
        temp.real(0);
        temp.imag(w*M_vector[indexM]);
        pA[i][j] = temp;
    }
}
```

```
        pA[j][i] = temp;
        indexM++;
    }
    //we need to do the diagonals here...
    temp.real(R_vector[i-2]);
    temp.imag(w*L_vector[i-2] - 1/(w*C_vector[i-2]));
    pA[i][i] = temp;
}

temp.real(RE);
temp.imag(-1/(w*(Cj)));
pA[0][0] = temp;

temp.real(RL);
temp.imag(-1/(w*(Cj)));
pA[n+3][n+3] = temp;

temp.real(0);
temp.imag(1/(w*(Cj)));
pA[0][1] = temp;
pA[1][0] = temp;
pA[n+2][n+3] = temp;
pA[n+3][n+2] = temp;

temp.real(0);
```

```

temp.imag(w*(Ls) - 1/(w*(Cj)));
pA[1][1] = temp;
pA[n+2][n+2] = temp;

//and finally
temp.real(0);
temp.imag(w*(Ms));
pA[1][2] = -temp;
pA[2][1] = -temp;
pA[n+1][n+2] = temp;
pA[n+2][n+1] = temp;

success = cmatinv(pA, n+4);

*S11 = 1.0 - 2*(RE)* pA[0][0];
*S21 = 2.0 * pA[n+3][0] * sqrt((RE)*(RL));
}

```

7.5 Objective function

The objective function is the error function. We are trying to minimize the difference between the S parameters given by the measurement (industry measurement/HFSS simulation/etc.) and the S parameters given by the circuit function.

Minimization of the objective function will result in a good match for our final parameters.

7.6 Code

```
#include <complex> #include <math.h>

using std::complex; using std::arg; using std::abs; using
std::polar;

extern void circfun(double *X, double *USER, complex<double> *S11,
complex<double> *S21);

void __stdcall objfun(int *MODE, int *N, double *X_Array, double
*OBJ_value, double *OBJ_grad,
                    int *NSTATE, int *IUSER, double *USER)
{
    complex<double> S11_circ, S21_circ;
    int npoints, optS11, optS21;
    double *FIXED;

    FIXED = new double[5];

    for (int i = 0; i<5; i++){
        FIXED[i] = USER[i];
```

```
}

*OBJ_value = 0;

if ((*MODE == 0) || (*MODE == 2)) {
    npoints = IUSER[0];
    optS11 = IUSER[1];
    optS21 = IUSER[2];

    for (i = 0; i < npoints; i++)
    {
        FIXED[1] = USER[5+5*i];

        circfun(X_Array, FIXED, &S11_circ, &S21_circ);

        *OBJ_value = *OBJ_value +
            optS11 * pow(abs(complex<double>
                (USER[5 + 5*i + 1], USER[5 + 5*i + 2])
                - S11_circ),2) +
            + optS21 * pow(abs(complex<double>
                (USER[5 + 5*i + 3], USER[5 + 5*i + 4])
                - S21_circ),2);
    }
}
}
```


7.7 Optimization routine

The optimization routine is called in the main body of the program.

Currently it consists of the E04UCF function which minimizes an error function. The E04UCF

7.8 Code

```
#include "c:\FLDLL184z\headers\nagmk18.h" #include "stdafx.h"
#include <complex> #include <math.h> #include <stdio.h> #include
<fstream.h>

using std::complex; using std::arg; using std::abs; using
std::polar;

extern void analyzeInput(ifstream inputFile, int *npoints,
    int *obj_S11, int *obj_S21,
    double *USER, double *BL, double *BU,
    double *nFreq, complex<double> *S11,
    complex<double> *S21);

extern void objfunction(double *error, int *npoints,
    int *optS11, int *optS21, double *nFreq,
    complex<double> *S11_data,
    complex<double> *S21_data, double *USER);

extern void __stdcall objfun(int *MODE, int *N, double *X_Array,
```

```
        double *OBJ_value, double *OBJ_grad,
        int *NSTATE, int *IUSER, double *USER);
void __stdcall E04UDAMIR(int *mode, int *ncnln, int *n, int
        *nrowj, int needc[],
        double x[], double c[], double cjac[],
        int *nstate, int iuser[],
        double user[]);

int main(int argc, char* argv[]) {

    int i, n, npoints, variables;
    int opt_S11, opt_S21;
    double MAXUSER[100];
    double MAXBL[100], MAXBU[100];
    double MAXFREQ[2001];
    complex<double> MAXS11[2001];
    complex<double> MAXS21[2001];

    //e04 crap
    int ITER, IFAIL;
    int IUSER[4];
    int NIWORK, NWORK;
    int LDA = 1;
    int LDCJ = 1;
    int LDR;
```

```
int NCLIN, NCNLN;
double C[1];
double CJAC[1];
double OBJF;
double A[1][1];

NCLIN = 0;
NCNLN = 0;
IFAIL = -1;

if(argc < 1) {
    printf("Usage: e04 <data file>\n");
    return 0;
}

//read in user data
ifstream inputFile(argv[1]);
analyzeInput (inputFile, &npoints, &opt_S11, &opt_S21,
    MAXUSER, MAXBL, MAXBU, MAXFREQ, MAXS11, MAXS21);
inputFile.close();

n = MAXUSER[0];

variables = 3*n + n*(n-1)/2 + 3;
NIWORK = 4*variables;
```

```
NWORK = 25*variables;

double *USER = new double[5 + 5*npoints];
double *BL = new double[variables];
double *BU = new double[variables];
int *ISTATE = new int[variables];
double *OBJGRD = new double[variables];
double *X = new double[variables];
double *CLAMDA = new double[variables];
double *R = new double[variables];
LDR = variables;
int *IWORK = new int[NWORK];
double *WORK = new double[NWORK];

for (i=0; i<5; i++) {
    USER[i] = MAXUSER[i];
}
for (i=5; i<variables+5; i++) {
    BL[i-5] = MAXBL[i-5];
    BU[i-5] = MAXBU[i-5];
    X[i-5] = MAXUSER[i];
}
for (i=0; i<npoints; i++){
    USER[5 + 5*i] = MAXFREQ[i];
    USER[5 + 5*i + 1] = MAXS11[i].real();
```

```

        USER[5 + 5*i + 2] = MAXS11[i].imag();
        USER[5 + 5*i + 3] = MAXS21[i].real();
        USER[5 + 5*i + 4] = MAXS21[i].imag();
    }

    IUSER[0] = npoints;
    IUSER[1] = opt_S11;
    IUSER[2] = opt_S21;
    IUSER[3] = variables;

    char options[] = "Derivative level = 0";

    E04UEF(options, strlen(options));

    E04UCF(&variables, &NCLIN, &NCNLN, &LDA, &LDCJ, &LDR,
          (double *)A, BL, BU, E04UDAMIR, objfun, &ITER,
          ISTATE, C, CJAC, CLAMDA, &OBJF, OBJGRD, R, X,
          IWORK, &NIWORK, WORK, &NWORK, IUSER, USER, &IFAIL);

    return 0;
}

void __stdcall E04UDAMIR(int *mode, int *ncnln, int *n, int
*nrowj, int needc[],
                        double x[], double c[], double cjac[],

```

```
        int *nstate, int iuser[],
        double user[])
{
    printf("Mode is %d\n", *mode);
}
```

Bibliography

- [1] Amir Borji, “Approximate models for coaxial cavity filters,” Tech. Rep., Dept. of ECE, University of Waterloo, Feb. 2000, part of the first annual report for the project sponsored by Ericsson Communications, Inc.
- [2] Q.J.Zhang and K.C.Gupta, *Neural Networks for RF and Microwave Design*, Artech-House, Inc., 2000.
- [3] J.W.Bandler, M.A.Ismail, J.E.Rayas-Sanchez, and Q.J.Zhang, “Neuro-modeling of Microwave Circuits Exploiting Space-Mapping Technology,” *IEEE Trans. Microwave Theory Tech.*, vol. 47, no. 12, pp. 2417–2427, Dec. 1999.
- [4] J.W.Bandler, R.M.Biernacki, S.H.Chen, P.A.Grobelny, and R.H.Hemmers, “Space Mapping Technique for Electromagnetic Optimization,” *IEEE Trans. Microwave Theory Tech.*, vol. 42, no. 12, pp. 2536–2544, Dec. 1994.
- [5] M.Kahrizi, “Accurate and robust methods for synthesis, design optimization, and computer parameter estimation of microwave coupled resonator filters,” Tech. Rep., Dept. of ECE, University of Waterloo,

- Feb. 2000, part of the first report for the project sponsored by Ericsson Communications, Inc.
- [6] Amir Borji, Dan Busuioc, S.Safavi-Naeini, and S.K.Chaudhuri, “ANN and EM Based Models for Fast and Accurate Characterization of Excitation Loops in Compline-type Filters,” *IEEE Trans. Microwave Theory Tech.*, 2002.
- [7] P.T. Boggs and W.J. Tolle, “SEquential Quadratic Programming,” *Acta Numerica*, pp. 1–100, 1996.
- [8] G.L.Matthaei, L.Young, and E.M.T.Jones, *Microwave Filters, Impedance-Matching Networks, and Coupling Structures*, Artech-House, Inc., reprint edition, 1980, chapters 8,9.
- [9] D.Baillargeat, S.Verdeyme, M.Aubourg, and P.Guillon, “CAD Applying the Finite Element Method for Dielectric Resonator Filters,” *IEEE Trans. Microwave Theory Tech.*, vol. 46, no. 1, pp. 10–17, Jan. 1998.
- [10] J.P.Cousty, S.Verdeyme, M.Aubourg, and P.Guillon, “Finite Elements for Microwave Device Simulation: Application to Microwave Dielectric Resonator Filters,” *IEEE Trans. Microwave Theory Tech.*, vol. 40, no. 5, pp. 925–932, May 1992.
- [11] Amir Borji, *Fast Full-wave Electromagnetic Analysis of Multiple Coupled Cavity Filters*, Ph.D. thesis, University of Waterloo, Waterloo, ON, May 2000, PhD Research Proposal.

- [12] J.W. Bandler and A.E. Salama, “Functional Approach to Microwave Postproduction Tuning,” *IEEE Trans. Microwave Theory Tech.*, vol. 33, no. 4, pp. 302–310, April 1985.
- [13] L.Young G.L.Matthaei and E.M.T.Jones, *Microwave Filters, Impedance Matching Networks and Coupling Structures*, McGraw-Hill Book Company, 1964.
- [14] K.C.Gupta, R.Garg, and R.Chadha, *Computer-Aided Design of Microwave Circuits*, Artech-House, Inc., 1981.
- [15] N. Balabanian and T.A. Bickart, *Electrical Network Theory*, John Wiley and Sons, 1969.
- [16] NAG Optimization Group, “E04UCF SQP method user manual,” pp. 1–33, 2000.