Stability and Passivity Analysis, Application of Vector Fitting in RLC Equivalent Circuit Synthesis

A Thesis

Presented in Partial Fulfillment of the Requirements for the

Degree of Master of Science

with a

Major in Electrical Engineering

in the

College of Graduate Studies

University of Idaho

by

Tao Duy Nguyen

Major Professor: Ata Zadehgol, Ph.D.

Committee Members: Herbert Hess, Ph.D.; Feng Li, Ph.D.

Department Administrator: Mohsen Guizani, Ph.D.

May 2016

Authorization to Submit Thesis

This thesis of Tao Duy Nguyen, submitted for the degree of Master of Science with a major in Electrical Engineering and titled "Stability and Passivity Analysis, Application of Vector Fitting in RLC Equivalent Circuit Synthesis," has been reviewed in final form. Permission, as indicated by the signatures and dates given below, is now granted to submit final copies to the College of Graduate Studies for approval.

Major Professor _____Date_____
Ata Zadehgol, Ph.D.

Committee Members _____Date_____
Herbert Hess, Ph.D.

_____Date_____
Feng Li, Ph.D.

Department Administrator _____Date_____
Mohsen Guizani, Ph.D.

Abstract

Vector Fitting (VF) is one of the most well-known techniques in calculating a rational approximation based on given frequency responses, and VF has been used in modeling different electrical systems. This thesis focuses on three main areas. First, the VF algorithm will be studied in depth regarding its advantages, weaknesses and strengths. Second, the implementation of the VF algorithm using python scripting language, and validation of the performance and the functionality of VF in python through numerical examples; this includes the original VF and the relaxed VF. Third, the analysis of the stability and the passivity of the results of each numerical example that is generated by VF. Semi-analytic and cellular method is used to analyze the stability and passivity of each RLC circuit branch, while the Hamiltonian Matrix method is used to verify the passivity of the network. Also, one of applications of VF discussed in this thesis is using VF to approximate poles and residues for Scattering (S-) and Admittance (Y-) parameters of a two-port network to synthesize the RLC circuit based on the approximated poles and residues.

The results show that the VF method demonstrates accurate approximation with very small error between actual frequency response and fitted frequency response. The relaxed VF (RVF) method produces a more accurate approximation compared to non-relaxed VF when the values of responses are small. The VF feature to enforce stability is confirmed and its result is in agreement with the result from stability analysis (i.e., the real part of all poles is less than zero). Also, ability of the VF algorithm to detect unstable systems is investigated and verified. Passivity analysis shows that even though there might be no passivity at one or more RLRC circuit branches in the network, the network could still be passive. Finally, the frequency responses of S-parameters of synthesized equivalent RLC circuits, which are approximated by VF, are well matched against the given tabulated frequency responses.

Acknowledgements

I deeply would like to thank Dr. Zadehgol for being my major professor and for his excellent advice throughout my graduate school. As an engineering outreach student, a full time employee, and a father of two kids, it has been very difficult from time to time, but Dr. Zadehgol is always been patient with me and has guided me in the right direction.

I also would like to thank Dr. Hess and Dr. Li for being my MS committee members.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Introduction

With the rapid growth in microelectronic systems, multi-port network systems, and many other high-speed systems, the design becomes more complicated and challenged. To have an accurate approximation of the system, a good simulation tool or approximation method is always a benefit in the design process as well as in understanding the characteristics of the system. And in the designs, frequency dependent effects play a major role because they impact the performance of the system. Therefore, the accuracy of a model of a system is very important. These frequency dependent responses can often be obtained through physical measurement or calculation and present them as discrete functions of frequency, and this often might not provide a precise system model.

$$f(s) = \frac{r_0 + r_1 * s + r_2 * s^2 + ... + r_N * s^N}{p_0 + p_1 * s + p_2 * s^2 + ... + p_N * s^N} \tag{1.1}$$

A better method is to replace the frequency domain responses with rational function approximation, which is shown in equation - Fitting a ratio of two polynomials to the obtained data (tabulated data). The rational function approximation is a nonlinear [1], but it can be rewritten as a linear system Ax=b by multiplying both sides by the denominator. The convolution can be achieved through a recursively method. This method is known as the Vector Fitting (VF) method (macromodeling technique) in state space to approximate the original frequency response f(s). Vector Fitting (VF) [1] is one of the well-known and excellent techniques in calculating a rational approximation based on given frequency responses. VF has also been used in modeling different electrical systems [1, 4, 8, 26, 45].

In the high-speed interconnects, packages, vias, on-chip passive components, trans-

mission lines, and other high frequency devices, stability and passivity of a system are very important properties. They go hand in hand when describing the characteristics of a system. For example, if the subsystem is stable but nonpassive, this might lead to an unstable system when it connects to other passive components within the system. Therefore, stability and passivity of the results from VF should be discussed. [2, 51]. And Matlab is been used to implement VF algorithms and it has been widely used. However, each user has pay for license fees for using Matlab. Python scripting language is another excellent language script for VF algorithms. Python is simple, easy to learn and use. The edit test debug cycle is very fast, and low cost. Users can use python script without cost. According to math.harvard.edu, program development using python is 5-10 times faster than with C/C++ and 3-5 times faster than using Java. So python is another good programming language, which can be used for VF algorithm.

This thesis endeavors to focus on three main points: 1) an in depth study of the VF algorithm and its advantages, weaknesses and strengths; 2) an implementation of the VF algorithm using python scripting language and validation of the performance and the functionality of VF in python codes through numerical examples, (and this includes comparing the results of the original VF and the relaxed VF [1, 4]); and 3) analysis of the stability and the passivity of the results of each numerical example that is generated by VF. Semi-Analytic and Cellular method is used to analyze the stability and the passivity for each RLC branch, and the Hamiltonian Matrix method is used to verify the passivity of the network. A simple flow diagram is presented and shows the application of VF combined with RLRC equations in [3] when converting frequency domain of s (scattering) parameters of a two-port network to a synthesized RLRC equivalent circuit. (i.e. use VF to approximate poles and residues from frequency responses of s and y parameters, and use RLRC equations in [3] to determine the value of each R, L, and C based on the approximated poles and residues from VF, and then use LTSpice software to synthesize the pi-equivalent circuit for the two-port network).

Chapter 2

Vector Fitting Algorithm

Another way to describe about Vector Fitting is that the Vector Fitting (VF) method is a robust numerical technique to approximate a model in state space of a system, which is using poles and residues (rational partial-fraction transfer function), based on the calculated or measured frequency responses. The form of rational partial-faction transfer function in state space representation can be seen in equation (2.1). Poles and residues can be either real, complex conjugate numbers, or both.

The VF method requires using the system identification procedure to approximate poles and residues, which is the iterative linear Least Square technique [10]. In other words, the procedure is to replace the approximated poles with an improved set of poles through the pole relocation technique, which would improve the approximation iteratively. Once the approximated poles of the system are identified, the residues can be identified from the approximated poles through linear Least Square technique.

In this chapter, the VF will be explored thoroughly as following: identify initial poles, stage 1 − pole relocation method, stage 2 − residue identification, pseudo-codes of VF. Both pole identification and residue identification methods are solved linearly. Therefore, linear transfer function will be reviewed before pole identification and residue identification can be described in detail.

2.1 Formulation of Vector Fitting in State Space

Consider a model of a system in equation (1.2) is written in rational transfer function form below [1].

$$f(s) = \sum_{n=1}^{N} \frac{r_n}{s - a_n} + d + sh \tag{2.1}$$

Where $s = j\omega$, which is a complex frequency. ω is angular frequency. r_n, p_n, d, h are

residues, poles, constant term, and linear term, respectively. N is the order of the approximation of the system. Obviously, the (2.1) is nonlinear due to unknown variables, and plus, a_n appears in the denominator. So in order to resolve the rational approximation method, it is required to introduce another unknown rational transfer function $\sigma(s)$. Without σ, Least Square solution might not able to provide a precise approximation [1]. And at very high frequency, the $\sigma(s)$ becomes very small or approaches zero unity. More in depth discussion will be discussed held later on in this section.

2.2 Stage 1 - Poles Relocation

In this stage, VF method will resolve the poles, p_n, but first, $\sigma(s)$ and the product of $\sigma(s)f(s)$ can be assumed that they can be approximated as rational approximation functions below in (2.2) and also both f(s) and σ(s)f(s) have a same set of poles [1], [4], [22].

$$\begin{vmatrix} \sigma(s)f(s) \\ \sigma(s) \end{vmatrix} = \begin{vmatrix} \sum_{n=1}^{N} \frac{r_n}{s-\bar{p}_n} + d + sh \\ \sum_{n=1}^{N} \frac{\widetilde{r}_n}{s-\bar{p}_n} + 1 \end{vmatrix} \tag{2.2}$$

Now, we take the 2nd row of the equation (2.2) and multiply it by f(s), and the equation can be re-arranged [1] such as:

$$\left(\sum_{n=1}^{N} \frac{r_n}{s-\bar{p}_n} + d + sh \right) \approx \left(\sum_{n=1}^{N} \frac{\widetilde{r}_n}{s-\bar{p}_n} + 1 \right) f(s) \tag{2.3}$$

$$(\sigma f)_{fit}(s) = \sigma_{fit}(s) f(s) \tag{2.4}$$

Solve for f(s) from equation (2.3)

$$f(s) \approx \frac{\left(\sum_{n=1}^{N} \frac{r_n}{s-\bar{p}_n} + d + sh \right)}{\left(\sum_{n=1}^{N} \frac{\widetilde{r}_n}{s-\bar{p}_n} + 1 \right)} \tag{2.5}$$

The rational function in (2.3) can be re-written as a fraction as follow,

$$(\sigma f)_{fit}(s) = h\frac{\prod_{n=1}^{N+1} s-z_n}{\prod_{n=1}^{N} s-\bar{p}_n} \tag{2.6}$$

$$(\sigma)_{fit}(s) = \frac{\prod_{n=1}^{N} s-\bar{z}_n}{\prod_{n=1}^{N} s-\bar{p}_n} \tag{2.7}$$

$$f(s) = \frac{(\sigma f)_{fit}(s)}{(\sigma)_{fit}(s)} = h\frac{\prod_{n=1}^{N+1} s-z_n}{\prod_{n=1}^{N} s-\bar{z}_n} \tag{2.8}$$

Based on equation (2.8), the initial guessed poles or previous set of poles $(s-\bar{p}_n)$ of f(s) has been cancelled out after the division. The new set of poles of f(s) is now the $\bar{z}_n$, which are the zeros of $(\sigma)_{fit}(s)$.

Now transform equation (2.5) into a format that would be easier to add into the matrices by multiplying both sides of (2.5) by denumerator of (2.5). f(s) is now transformed as following.

$$\left(\sum_{n=1}^{N}\frac{r_n}{s-\bar{p}_n} + d + sh\right) \approx \left(\sum_{n=1}^{N}\frac{\widetilde{r}_n}{s-\bar{p}_n} + 1\right) f(s) \tag{2.9}$$

Expand equation (2.9) into

$$\left(\sum_{n=1}^{N}\frac{r_n}{s-\bar{p}_n} + d + sh\right) \approx f(s)\left(\sum_{n=1}^{N}\frac{\widetilde{r}_n}{s-\bar{p}_n}\right) + f(s) \tag{2.10}$$

Solve for f(s)

$$f(s) \approx \left(\sum_{n=1}^{N}\frac{r_n}{s-\bar{p}_n} + d + sh\right) - \left(\textstyle\sum_{n=1}^{N}\frac{\widetilde{r}_n}{s-\bar{p}_n}\right) f(s) \tag{2.11}$$

Equation 2.11 can be represented in linear matrix algebra as follow:

$$\begin{bmatrix} \frac{1}{s_1-p_1} & \cdots & \frac{1}{s_1-p_N} & 1 & s_1 & \frac{-f(s_1)}{s_1-p_1} & \cdots & \frac{-f(s_1)}{s_1-p_N} \\ \frac{1}{s_2-p_1} & \cdots & \frac{1}{s_2-p_N} & 1 & s_2 & \frac{-f(s_2)}{s_2-p_1} & \cdots & \frac{-f(s_2)}{s_2-p_N} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{1}{s_k-p_1} & \cdots & \frac{1}{s_k-p_N} & 1 & s_k & \frac{-f(s_k)}{s_k-p_1} & \cdots & \frac{-f(s_k)}{s_k-p_N} \end{bmatrix} \begin{bmatrix} r_1 \\ \vdots \\ r_N \\ d \\ h \\ \widetilde{r}_1 \\ \vdots \\ \widetilde{r}_N \end{bmatrix} = \begin{bmatrix} f(s_1) \\ f(s_2) \\ \vdots \\ f(s_{k-1}) \\ f(s_k) \end{bmatrix} \tag{2.12}$$

In equation (2.11), there are too many unknown variables $r_n, p_n, d, h, \widetilde{r}_n$, and it is considered as a nonlinear system. VF method wants the equation in linear system form, which is,

$$Ax = b \tag{2.13}$$

The first step in pole identification is providing the guessed starting poles. When the guessed starting poles, $\bar{p}_n$, are in place, the equation (2.11) now becomes a linear system with four unknowns, r_n, d, h, and $\widetilde{r}_n$. As mentioned before, f(s) is a set of data points (tabulated data), which can be obtained either via measurement or calculation. This set of data points must exceed number of frequency samples (N), which results in an overdetermined set of equations for Ax=b. The Least Square method can solve this system without any problem.

The poles of f(s) are the poles of $1/\sigma(s)$ or zeros of $\sigma(s)$ based on previous derivation in (2.8); so the $1/\sigma$ also can be defined as 1/y(t)/u(t), which is u(t)/y(t), where u(t) is input and y(t) is output. We can replace y(t) with u(t) and u(t) with y(t) in the following conventional linear system in state space equation (2.14). The linear system equation in (2.15) is after interchanging output y(t) with input u(t), and vice versa.

Conventional linear system equations can be seen in (2.14), where x'(t)=dx/dt, u(t)

= input, y(t) = output. The ($\sigma(s)$) can be represented as following: A is the diagonal of poles matrix. C contains residues. B is the column of ones, and D is the unity constant.

$$\begin{aligned} x'(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) + Du(t) \end{aligned} \tag{2.14}$$

After linear system equation has interchanged input and output, we are going to solve for y(t) in second equation and plug y(t) into first equation and re-arrange the equation to match with first equation in (2.14). [18, 19, 20] also show derivation in similar way [1, 8].

$$\begin{aligned} x'(t) &= Ax(t) + By(t) \\ u(t) &= Cx(t) + Dy(t) \end{aligned} \tag{2.15}$$

Solve the 2nd equation in term of y(t)

$$\begin{aligned} u(t) &= Cx(t) + Dy(t) \\ u(t) - Cx(t) &= Dy(t) \\ (u(t) - Cx(t))D^{-1} &= y(t) \end{aligned} \tag{2.16}$$

Now substitute y(t) into first equation of (2.15), and solve for x'(t) as following

$$\begin{aligned} x'(t) &= Ax(t) + B((u(t) - Cx(t))D^{-1}) \\ x'(t) &= Ax(t) + Bu(t)D^{-1} - BD^{-1}Cx(t) \\ x'(t) &= (A - BD^{-1}C)x(t) + BD^{-1}u(t) \end{aligned} \tag{2.17}$$

According appendix A of [1] (how to solve for poles of a linear system transfer function), the poles of $1/\sigma$ are the roots of determinant of equation (2.18), i.e. determinant can be solved through equation (2.18). A is row vector, NxN matrix, which contains $diag(p_n)$ and C is the row vectors, 1xN matrix, which contains residues. B is the column vectors of ones, and D is a constant term (assume D is unity in most cases).

Now the poles can be solved by determining eigenvalues of equation (2.18). If the given frequency responses are greater than N, then equation is the overdetermined linear system [1],[4].

$$det[sI - (A - BD^{-1}C)] \tag{2.18}$$

Since D is set to unity in this case

$$p_n = eigen(A - BD^{-1}C) = egen(A - BC) \tag{2.19}$$

Define variables A, C, and B in equations

$$A = \begin{bmatrix} \bar{p}_1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & \bar{p}_2 & 0 & 0 & \cdots & 0 \\ \cdots & & \ddots & & \cdots & 0 \\ \cdots & & & \ddots & \cdots & 0 \\ \cdots & & & & & 0 \\ 0 & \cdots & & & & \bar{p}_N \end{bmatrix}$$

$$C = \begin{bmatrix} \bar{r}_1 & \bar{r}_2 & \bar{r}_3 & \cdots & \bar{r}_N \end{bmatrix}$$

$$B = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{bmatrix}$$

The poles are identified through results of (2.19). If the error is not within acceptable range when compare with tabulated set of frequency samples, use the new set of poles as

the starting poles and repeat the pole identification procedure until the convergence is achieved. This is also called iteration method. Normally, it takes about 2 to 4 iterations before it reaches convergence or within acceptable error. In some cases(unstable systems), the VF might not able to achieve the convergent point. There is more discussion on this later on.

Keys points in the poles identification procedure:

1. Assume sampled frequency domain responses, f(s) and s, are given, and if number of data points is greater than N order approximation, equation is overdetermined linear system. Else, system is undertermined linear system.

2. Initial poles must be guessed (for complex poles, will discuss in later section).

3. $(\sigma f)_{fit}(s)/(\sigma)_{fit}(s)$ would cause the set of poles cancelled out, and that makes the system becoming a linear system [1].

4. The poles of f(s) are poles of $1/\sigma(s)$ or zero of $\sigma(s)$.

5. Solve for poles (eigenvalues) through least square method.

2.3 Stage 2 - Residue Identification

In stage 2, the residues will be identified. Technically, the residues can be identified from stage one, but in order to get the most accurate approximation for the residues, the newly starting poles from stage 1 should be used in calculating residues in this stage. Again, looking back to equation (2.1), the poles, p_n, has been identified. So the remaining unknowns are residues (c_n), constant term (d), and linear constant (h). All these parameters are in the matrix [C], which is the column vector. The number of sampled frequency data points is more than number of unknowns; therefore, this is the overdetermined linear equation. Least square method is used again to identify these parameters through equation (2.20), which is linear Ax=b [1], [4].

$$\begin{bmatrix} \frac{1}{s_1-p_1} & \frac{1}{s_1-p_2} & \cdots & \frac{1}{s-p_N} & 1 & s_1 \\ \frac{1}{s_2-p_1} & \frac{1}{s_2-p_2} & \cdots & \frac{1}{s-p_N} & 1 & s_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \frac{1}{s_k-p_1} & \frac{1}{s_k-p_2} & \cdots & \frac{1}{s_k-p_N} & 1 & s_k \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_N \\ d \\ h \end{bmatrix} = \begin{bmatrix} f(s_1) \\ f(s_2) \\ \vdots \\ f(s_{k-1}) \\ f(s_k) \end{bmatrix} \tag{2.20}$$

Note that data points (tabulated data set) of f(s) are given through either calulcation or measurement. So now the residues are identified, and the VF method is completed. Of course, the results might not exactly matched with actual data because the VF method only provides approximated results, which are close to actual data. If the error is not within acceptable error or not within error range when compared with actual data (tabulated data set), the pole identification and residue identification procedure can be repeated using iteration method. Normally, it might take 2-5 iterations to achieve an accurate result within error range.

2.4 VF Modification for Complex Poles

When fitting the very smooth function, the initial poles or starting poles can be guessed real starting poles over equal spaces of frequency. However, for functions with the resonance peaks, we need to guess starting poles carefully; else, it might take longer time to get to a convergent point or lead to inaccurate result. Guessing a starting pole is a critical step in the pole relocation process because if poor starting poles are selected and the starting poles are real, the linear system might become ill-conditioned and lead to an inaccurate solution. Second problem is that if poor starting poles are selected, there could be large variations in $\sigma(s)$ and $\sigma(s)f(s)$, and the poor fitting may result. However, the paper in [1] has addressed the technique for overcoming these problems. The ill-conditioned problem can be overcome by using the complex starting poles, and

the poor fitting also can be minimized by using the new poles as starting poles in an iterative procedure.

One recommendation from the author of VF method [1] in selecting complex starting poles is the imaginary part β linearly distributed over the interested frequency interval. The real part, α, should be small about 1/100 of imaginary part. This small real is sufficient enough to prevent the ill-conditioning problem. Guessing starting poles are defined as following,

$$p_n = -\alpha + j\beta, \;\; p_n = -\alpha - j\beta \tag{2.21}$$

where

$$\alpha = \beta/100 \tag{2.22}$$

The poles are either real or occurred in form of complex-conjugate pairs. For the complex poles, the matrix A needs to be reformulated as following when we want to solve for residues.

$$A = \begin{bmatrix} A^r \\ A^i \end{bmatrix} \; x = \begin{bmatrix} f^r \\ f^i \end{bmatrix} \tag{2.23}$$

The results of residues are now in terms of real terms.

$$C = \begin{bmatrix} r_1^r & r_1^i & \cdots & r_N^r & r_N^i, d, h, \widetilde{r_1}^r & \widetilde{r_1}^i & \cdots & \widetilde{r}_N^r & \widetilde{r}_N^i \end{bmatrix} \tag{2.24}$$

In order to solve for the new set of poles, we need to modify the matrices A, B, C as sub-matrices as follows.

$$A = \begin{bmatrix} p^r & p^i \\ -p^i & p^r \end{bmatrix} \tag{2.25}$$

$$B = \begin{bmatrix} 2 \\ 0 \end{bmatrix} \tag{2.26}$$

$$C = \begin{bmatrix} \widetilde{r}^r & \widetilde{r}^i \end{bmatrix} \tag{2.27}$$

2.5 Relaxed VF Method

Relaxed VF method will provide an accurate approximation for the fitting elements. On paper [4], the function used in relaxed VF method is called array weight function, which includes independent elements of f(s) and frequency samples, s. Weight=(1,Ns) or weight(n,Ns), where n is the number elements to be fitted, and Ns is the number of frequency samples used in VF. There are three schemes that can be used in the weight function: 1) No Weight, 2) Strong inverse weight, 3) Weaker inverse weight [1], [4], [22].

Now the weight scheme does not need the inversed weight. That means it assumes that the original VF method would able to provide the accurate result. Strong inverse weight is to provide strong weight on elements in f(s), where they are small. The main reason is to minimize the deviation during fitting computation. Lastly, weaker inverse weight is often used when response f(s) contains noise. In many cases, when the elements of f(s) are small and inverse weight is not used, the fitting variables would not be convergent even high number of iteration is used. These are illustrated in numerical example sections.

The weighting scheme approaches are presented in the table below [21].

Table 2.1: RVF Weighting Schemes

Scheme	Independent Weighting	Common Weighting
1) No Weight	weight=ones(Nc,Ns)	weight=ones(1,Ns)
2) Strong inverse weight	weight=1./(abs(f))	weight=zeros(1,Nx) for k=1:Ns weight(1,k)=1/norm(f(:,k)) end
3) Weaker inverse weight	weight=1./sqrt(abs(f))	weight=zeros(1,Nx) for k=1:Ns weight(1,k)=1/sqrt(norm(f(:,k))) end

As we will be going through numerous numerical examples in later section, the RVF method is used, and majority weight function is used with strong inverse weight scheme.

Chapter 3

Python Codes

As mentioned briefly earlier, Python is a programming language, which is very well known in the computer programing today. Python is simple, easy to learn syntax, and low cost of program maintenance. It highly supports modules and packages. The edit-test-debug cycle is very fast, and debugging Python programs is quite easy than most other programming languages in terms of how the error messages displace for user to understand. Because Python is easy to learn and has fast debugging, most users feel productive and efficient when they use Python. Python also consists of an extensive standard library. This library helps programing in general. When it comes to scientific computation, Python allows the use of additional packages, such as, Numpy, Scrip, and Matplotlib. These packages are very useful, powerful, and fast.

3.1 VF Algorithm in Python Language

3.1.1 Introduction

In this section, python codes [23] will be explained briefly in term of how the program is designed and worked for VF algorithm. The program consists of main function and multiple other module functions. Module functions include main function, `vectfit_step function, calculate_residues function, vectfit_auto function`.

At the end of this section, a simple test is demonstrated on 18th poles system to ensure the codes are worked correctly, and the result is matched against the result from the paper [1].

3.1.2 Main Function

This module represents the scope of the entire program. When the program is executed, it will run all the codes within the main function. In other words, this main function allows users to block certain codes from being run when the modules are imported.

Once the input data is calculated, this data is sent to `vectfit_auto` module, and this modue will return residues, poles, h, and d values. The following remaining codes [23] in the main function are to plot fitted data and their errors compared to actual f(s).

3.1.3 Vectfit Auto Function

In `vectfit_auto` function, it receives responses, frequencies, number of iteration from the main function. This data is then passed into `vectfit_step` function and `calculate_residues` function, which will be discussed in detailed in later sections. And `vectfit_step` function [23] returns the new set poles of fitted f(s). The `calculate_residues` function [23] returns the residues, h value, and d value of the fitted f(s).

3.1.4 Vectfit Step Function

With the variables (responses, frequency, initial poles) are passed to this module function from the `vectfit_auto`, the codes in this `vectfit_step` approximate the poles of the fitted f(s). The first part is to determine the initial poles. The second part is the algorithm for appendix A in paper [1], and the third part is the algorithm of the Appendix B of paper [1]. The algorithm is identical and equivalent to the Matlab algorithm codes in original paper, and the python algorithm used in paper [14], [21, 22, 23].

3.1.5 Calculate Residues Function

Computing residues is part of computing the poles. The codes will compute the residues based on the fitted poles from calculated previously. The main difference of codes in this module compared to `vectfit_step` module is that it does not request going through iteration or forcing stability. However, it is required to assign right array values for x, h, and d. This function returns the residues, h, and d after it completed [21, 22, 23].

3.1.6 Codes Verification for VF with Stable System

The vectfit.py has been explained in previous sections. This section will verify the codes to make sure it can generate the fitted data and match with the result of the Matlab codes generated. To do this, we use data from 18th poles system that was provided in paper [1], which can be seen below, and calculated the f(s) for 100 frequency samples. These data are used as frequency responses to be fitted by vectfit.py. The result of vectfit.py should be matched with the given poles-residues and input f(s).

Table 3.1: Given 18 Poles System

Given Poles	Given Residues
-4500	-3000
-41000	-83000
-100 + j5000	-5 + j7000
-100 - j5000	-5 + j7000
-120 + j15000	-20 + j18000
-120 - j15000	-20 - j18000
-3000 + j35000	6000 + j45000
-3000 - j35000	6000 - j45000
-200 + j45000	40 + j60000
-200 - j45000	40 - j60000
-1500 + j45000	90 + j10000
-1500 - j45000	90 - j10000
-500 + j70000	50000 + j80000
-500 - j 70000	50000 - j80000
-1000 + j73000	1000 + j45000
-1000 - j73000	1000 - j45000
-2000 + j90000	-5000 + j92000
-2000 - j90000	-5000 - j92000
d=0.2, h=2E-5	

We use the above poles-residues, h, and d to compute 100 frequency responses. These 100 responses along with its respective frequencies are fed to vectfit.py and the results are showed in figure 3.1. The error is well within the range, which also can be seen in

figure 3.2. Total time to execute the entire program is 3.49 seconds.

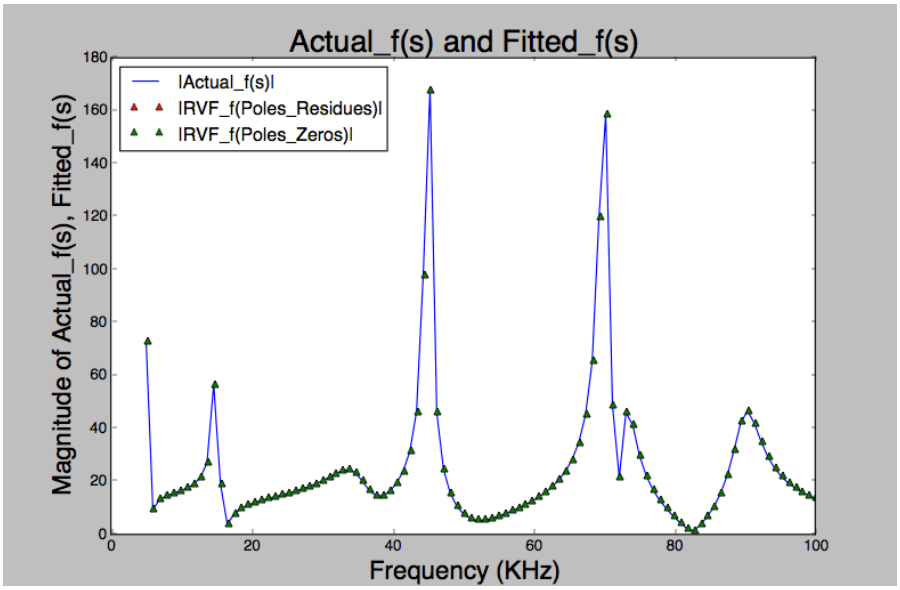


Figure 3.1: Fitted f(s) from Vecfit.py compared with Actual f(s) of 18 Poles System.

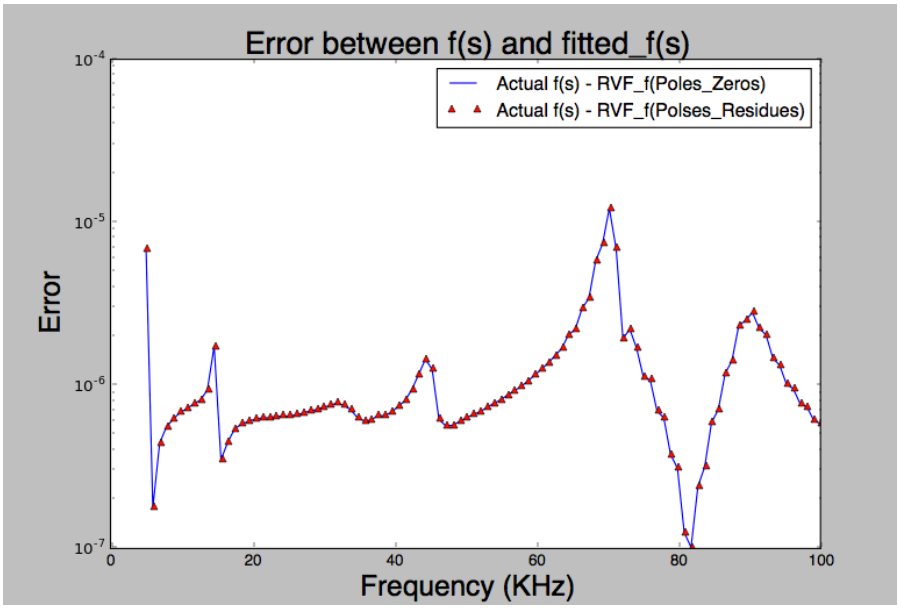


Figure 3.2: Error between Fitted f(s) from vectfit.py and Actual f(s) of 18 Poles System.

The fitted poles, residues, and zeros are output and stored in the Output.csv file, which locates in the same directory where the vectfit.py saved, and it is in the format

similar to the tables below. The program also prints out the Poles, Residues, h, and d. `relaxed vector fitting codes` also has been tried to run this data, and it yields the results similarly as vectfit.py.

A. VF Poles Real	B.VF Poles Imag	C. Blank	D. VF Residue Real	E. VF Residue Imag
-41001.28836	0		-83002.70313	0
-4500.72068	0		-3000.340332	0
-3000.08855	-35001.03228		6000.175293	-45001.32031
-3000.08855	35001.03228		6000.175293	45001.32031
-2000.058975	-90002.6544		-5000.150391	-92002.71875
-2000.058975	90002.6544		-5000.150391	92002.71875
-1500.044363	-45001.32676		90.00389099	-10000.29004
-1500.044363	45001.32676		90.00389099	10000.29004
-1000.029483	-73002.15302		1000.027039	-45001.32813
-1000.029483	73002.15302		1000.027039	45001.32813
-500.0147481	-70002.06456		50001.47266	-80002.35938
-500.0147481	70002.06456		50001.47266	80002.35938
-200.0058978	-45001.32719		39.99911499	-60001.76953
-200.0058978	45001.32719		39.99911499	60001.76953
-120.0035313	-15000.44241		-19.99964523	-18000.53125
-120.0035313	15000.44241		-19.99964523	18000.53125
-100.0049309	-5000.147542		-4.990383625	-7000.345215
-100.0049309	5000.147542		-4.990383625	7000.345215
-	-		-	-

F. Blank	G. VF Zeros Real	H. VF Zeros Imag	J. Blank	K. D Value	L. H value
	-59856.23884	-125146.794		0.500002	2.00E-05
	-59856.23884	125146.794		-	-
	104974.3274	5.96E-11		-	-
	-456.8676756	-82488.22928		-	-
	-456.8676756	82488.22928		-	-
	-461.3565688	72063.40862		-	-
	-461.3565688	-72063.40862		-	-
	-51935.59789	-2.17E-11		-	-
	-2685.957785	51631.99401		-	-
	-2685.957785	-51631.99401		-	-
	-1304.825343	45018.25845		-	-
	-1304.825343	-45018.25845		-	-
	-2660.334426	37548.65124		-	-
	-2660.334426	-37548.65124		-	-
	-301.7688971	16244.68885		-	-
	-301.7688971	-16244.68885		-	-
	-124.5914409	-5422.327983		-	-
	-124.5914409	5422.327983		-	-
	-4678.304482	5.28E-13		-	-

3.1.7 VF Verification for Unstable System

One of the VF features is able to detect unstable system, not converge accurately if the given model (tabulated data) is unstable. Following is a given tabulated data set (Insertion loss data of a differential microstrip transmission line), which is known as an unstable system. This data is simulated by VF to see if VF can converge or not. And

as you can see in figure 3.3 that VF is able to detect unstable system.

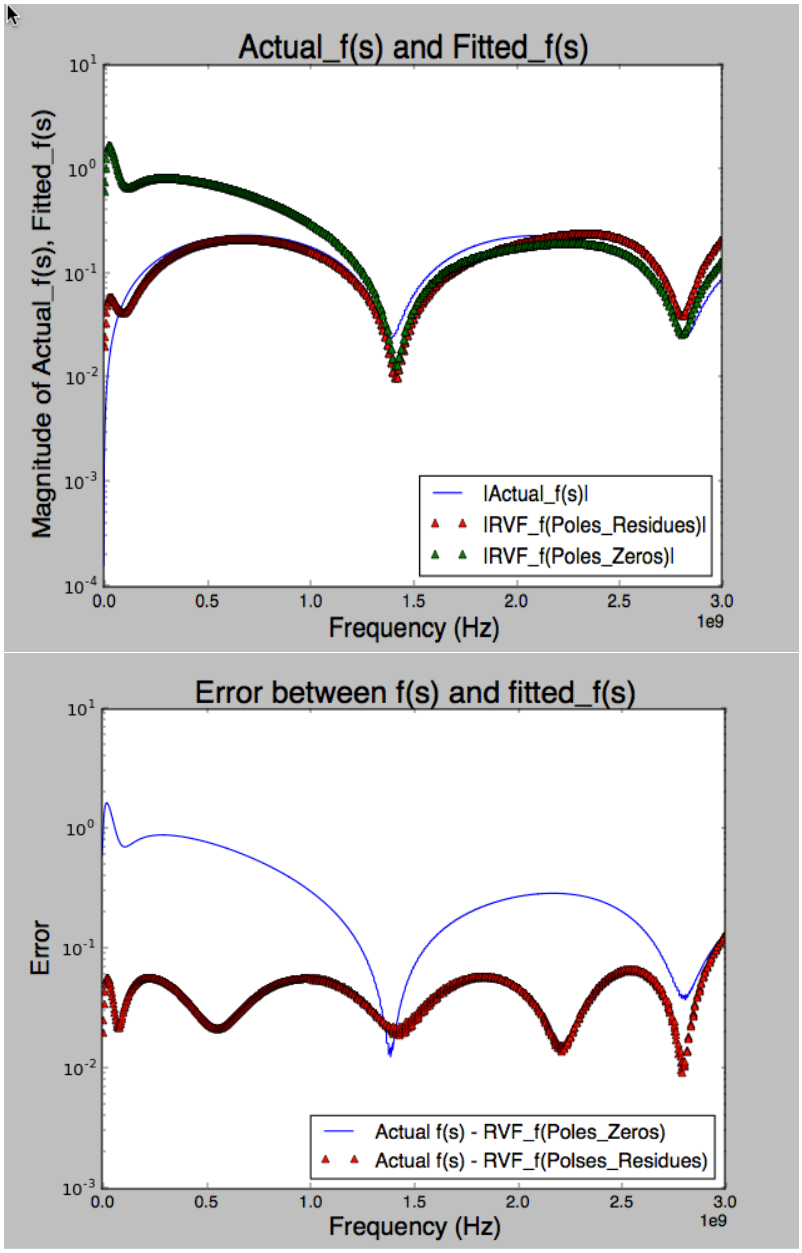


Figure 3.3: Fitted f(s) vs. Actual f(s) of an Unstable System.

3.1.8 Conclusion

Based on the result from Output.csv and plots of Actual f(s) versus fitted f(s), the error is very small and the mrse is very low. The VF program is working as expected. There will be more numerical examples used to justify functionality of two programs.

Chapter 4

Stability and Passivity

4.1 Introduction

The VF can generate a good approximated rational transfer function representation based on measured or calculated data, and the transfer function is presented in terms of poles and residues or poles and zeros. From the approximated rational transfer function, an equivalent circuit or macromodel/netlist can be synthesized through Spice-compatible. The VF algorithm is written to force the results to be stabilized. However, how do we know that the results from VF are stable? What is the passivity of the approximated model? Note that the stability is achieved from VF approximated macromodel, if even one component of a circuit is nonpassive, it might lead to unstable system when it is connected to other network within the system. Therefore, passivity is a critical property for the macromodel circuit.

As mentioned earlier, other contributions of this thesis are to verify that the approximated models, which are generated by VF algorithm written in python, are stable as desired. There are two ways to look at to make sure the system is stable or not. One way is to look at the poles of the transfer function. The second way is to look at the stability of each branch if the transfer function is written in such a way that each partial fraction term is in term of admittance (Y), and the approach in paper [3] will be used to determine the stability of the system per branch. In this thesis, we are going to look into how to present poles and residues rational TF as a model for an equivalent RLCG circuit in terms admittance (Y), and the RLCG parameters will be determined. In order to understand if the approximated model is passive or not, this paper will check the passivity of the approximated TF model in two ways. One is to look at the passivity of the TF model by using the passivity verification technique in paper [2], which is to confirm the value of the Re(Y), and the second way is to verify the passivity of the

each branch of the synthesized macromodel via approaches provided in paper [3]. This process might help the user to determine stability and passivity of the results of VF. Later in this thesis, there are numerical examples presented and discussions on stability and passivity for each example.

4.2 Formulation

4.2.1 Stability

The stability of a system or a branch within a circuit is defined by bounded input and bounded output (BIBO). Basically, if the variables are increased without a boundary ended, they are consider unstable, whereas, if the variables are increased within a boundary over time, then those are considered stable. One simple way to describe the stability of a system is to look at the poles of the transfer function. The transfer function is a rational function in the complex variables, s=σ+jω, a ratio of two polynomial functions [18, 19].

$$H(s) = \frac{b_m s^m + b_{m-1} s^{m-1} + \cdots + b_1 s + b_0}{a_n s^n + a_{n-1} s^{n-1} + \cdots + a_1 s + a_0} \tag{4.1}$$

The transfer function can be rewriten in the poles-zeros TF 4.2 or partial fraction TF 4.3.

$$H(s) = \frac{N(s)}{D(s)} = K \frac{s - z_1)(s - z_2) \cdots (s - z_{m-1}(s - z_m)}{(s - p_1)(s - p_2) \cdots (s - p_{n-1})(s - p_n)} \tag{4.2}$$

$$H(s) = \sum_{n=1}^{N} \frac{c_n}{s - p_n} \tag{4.3}$$

Now looking at each term of the partial fraction TF in time domains by taking inverse Fourier transfer function, and it yields to equation (4.4). Now if $p_i > 0$, the (4.4) continues increasing as t approaches ∞, and this means the system is unstable. While if $p_i < 0$, as t approaches ∞, the y(t) is decayed to 0. In other words, in order for response y(t) to be stable, the p_i must be negative ($p_i < 0$). If $p_i > 0$, the y(t) is

unstable. When plotting the poles on the s-plane (Laplace-plane) plot, the poles must be on the left half of the s-plane. Also if the poles are lying on the imaginary axis of the s-plane, it's considered an imaginary stable [1], [4], [26].

$$y(t) = \lim_{t\to\infty} \sum_{i=1}^{N} c_i e^{p_i t} \tag{4.4}$$

4.2.2 Passivity

The passivity is a critical property of a network or a system. If a macromodel is nonpassive and it connects to other passive components, it might lead to unstable system. This section will discuss an existing method of how to verify the passivity of a network. Assume that the network studied here contains Y-parameters and the data is given in form of tabulated data. In other words, we use this method to determine the passivity of the macromodel (approximated TF model), which is generated by VF. A numerical example in a later section will present more in depth.

There are several ways to determine the passivity of a macromodel: 1) Considered traditional method, which is based on a frequency sweep of eigenvalues of the real part of the admittance matrix, $Re(Y(j\omega))$; 2) determine based on tabulated data set [9], 3) based on the eigenvalues of Hamiltonian matrix (M). However, in this paper, method 3 will be used to verify the passivity of the macromodel because methods 1 and 2 depend on frequency or involved with frequency whereas method 3 does not depend on frequency, and is easy to compute.

Where A is row vector, NxN matrix, which contains diagonal poles, C is the row vectors, 1xN matrix, which contains residues. D is an unity constant term, and B is the column vectors of ones [2], [52].

$$M = \begin{bmatrix} A - B(D + D^t)^{-1}C & B(D + D^t)^{-1}B^t \\ (-C^t(D + D^{-}t) - 1 & -A^t + C + (D + D^t)^{-1}B^t \end{bmatrix} \tag{4.5}$$

There are two theorems for passivity verification in using the Hamiltonian matrix technique. Theorem 1: If the Hamiltonian matrix (M) has no imaginary eigenvalues, then the state space system (A, B, C, D) is passive. Theorem 2: If $j\omega$ is an imaginary eigenvalue corresponding to Hamiltonian Matrix, M [2, 3], then the state space system is nonpassive.

4.2.3 Stability and Passivity Per RLCG Branch

As mentioned before, the stability of a system or a branch is defined by bounded input and bounded output (BIBO). Basically, if the variables are increased without a boundary ended, they are considering unstable. The passivity of a passive system is where its average power is greater than zero watt ($S_{avg} > 0$), whereas, a nonpassive system is where the average power is less than zero Watt ($S_{avg} < 0$). The equation of S_{avg} will be presented later in this section. Most formulas in this section have been derived in [3].

The transfer function can be written in partial fraction form in terms of admittances and where unit of impedance is ohms and unit of admittance is Siemen [3].

$$H(s) = \sum_{n=1}^{N} \frac{c_n}{s - p_n} = \sum_{n=1}^{N} Y_n = Y_1 + \cdots + Y_N = Y_{Network} = \frac{1}{Z_1} + \cdots + \frac{1}{Z_N} \tag{4.6}$$

Expanding each term Y_n in terms of poles and residues transfer function, it shows below, where c and p can be real number or complex number:

$$Y = \frac{c}{s - p} \tag{4.7}$$

Therefore,

$$Z = Y^{-1} = -\frac{p}{c} + \frac{1}{c}s \tag{4.8}$$

For complex poles and residues, sometimes they are presented in pairs of complex conjugate as showed below [3].

$$Y = \sum_{n=1}^{N} \frac{c_n}{s - p_n} + \frac{c_n^*}{s - p_n^*} = \sum_{n=1}^{N} Y_n = Y_1 + \cdots + Y_N = Y_{Network} = \frac{1}{Z_1} + \cdots + \frac{1}{Z_N} \quad (4.9)$$

For the single term Y with a pair complex conjugate, it can be expressed as [3]

$$\begin{aligned} Y &= \frac{c_r + jc_i}{s - (p_r + p_i)} + \frac{c_r - jc_i}{s - (p_r - jp_i)} \\ Z = Y^{-1} &= \frac{c_i p_i - c_r p_r}{2c_r^2} + \frac{1}{2c_r} s + \left(\left(\frac{(c_i^2 + c_r^2)p_i^2}{2c_r^2(c_i p_i + c_r p_r)} \right)^{-1} + s \frac{2c_r^3}{(c_i^2 2 + c_r^2)p_i^2} \right)^{-1} \end{aligned} \quad (4.10)$$

For real poles and residues, each term Y is a single arbitrary RL branch. The R, L, and S_{ave} are expressed in equations 4.11 - 4.13. And readers want more details how to derive these equations, please refer to the paper [3].

$$R = -\frac{p}{c}, \qquad L = \frac{1}{c} \qquad c = \frac{1}{L} \qquad p = -\frac{R}{L} \quad (4.11)$$

From (4.11), the inverse Fourier transform will yield to this:

$$y(t) = \frac{sgn(R)}{|L|} e^{-\frac{R}{L}t} us\left(tsgn\left(\frac{R}{L} \right) \right) \quad (4.12)$$

$$S_{avg}^{ITF^{RL}} = \frac{i_0^2 R}{2} \quad (4.13)$$

The ITF stands for Impedance Transfer Function, which is mainly driven by Ideal Current Source (ICS). Now for Admittance Transfer Function (ATF), which is driven by Ideal Voltage Source (IVS), and its cumulative average power can be seen in equation (4.14) below and where m is integer number of the cumulative periods [3].

$$S_{cavg}^{ATF^{RL}} = \frac{LR^2 v_0^2 \omega_0 (e^{-\frac{2\pi m R}{L\omega_0}} - 1)}{2\pi m (L^2\omega_0^2 + R^2)^2} + \frac{R v_0^2}{2(L^2\omega_0^2 + R^2)} \quad (4.14)$$

Since R and L of each term Y have been determined, we can synthesize equation 4.9 to a RL equivalent circuit.

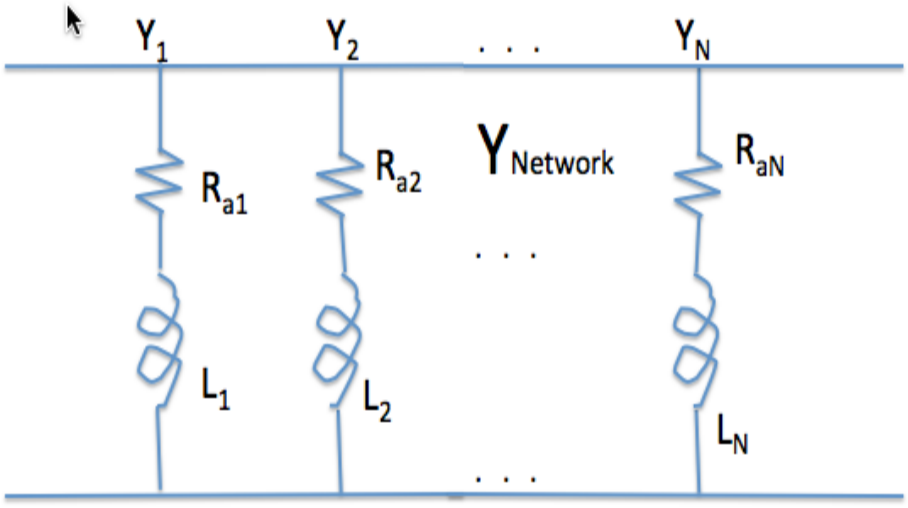


Figure 4.1: RL Equivalent Circuit

Each branch of $Y_{Network}$, there are four paramatric states that can be defined. From definitions previously defined, stability and passive of each parametric state are described in table below. For ATF with real pole and residue, only states 1 and 4 are stable [3].

Table 4.1: Summary of parametric states of a single arbitrary branch of RL equivalent circuit presented with R and L condition

State No.	Equivalent Circuit
1	$R > 0$ and $L > 0$
2	$R > 0$ and $L < 0$
3	$R < 0$ and $L > 0$
4	$R < 0$ and $L < 0$

Table 4.2: Summary of pole/residue states of a single arbitrary branch of RL equivalent circuit with pole and residue condition

State No.	Poles/Residue
1	$p < 0$ and $c > 0$
2	$p > 0$ and $c < 0$
3	$p > 0$ and $c > 0$
4	$p < 0$ and $c < 0$

Table 4.3: Stability and Passivity of each branch for ATF case

State No.	Stability	Passive
1	Yes	Yes
2	No	Omit
3	No	Omit
4	Yes	No

Table 4.4: Stability and Passivity of each branch for ITF case

State No.	Stability	Passive
1	Yes	Yes
2	Yes	Yes
3	Yes	No
4	Yes	No

These are brief discussions for each parametric state for the real poles and raly residues of TF. For the ATF, states 1 and 4 are stable given equation (4.12) will decay to zero. And only state 1 is passive due to equation (4.14) is greater than zero as m (number of cumulative periods) approaches to infinity. States 2 and 3 are omitted and not to discuss due to states 2 and 3 are unstable.

For the ITF, all four states are stable because there are no poles for each state. Passivity occurs on states 1 and 2. Again, this is because average power (defined in equation (4.13)) is greater than zero.

Now we are moving on to the ATF and the ITF with pair complex conjugates. With the same approach, the Ra, L, Rb, C are determined from equation 4.10, which yield to following equations [3]:

$$R_a = -\frac{c_i p_i - c_r p_r}{2c_r^2}, \quad R_b = \frac{p_i^2(c_i^2 + c_r^2)}{2c_r^2(c_i p_i + c_r p_r)} \quad L = \frac{1}{2c_r} \quad C = -\frac{2c_r^3}{p_i^2(c_i^2 + c_r^2} \tag{4.15}$$

With Ra, Rb, L, and C are determined, the RLCG equivalent circuit can be synthesized as following:

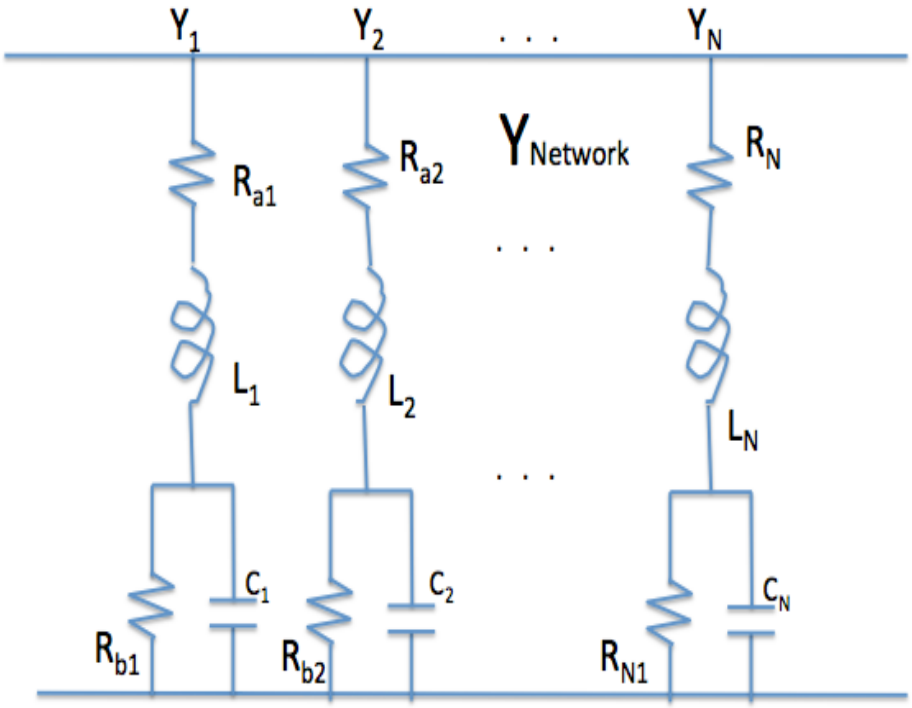


Figure 4.2: RLCG Equivalent Circuit

According to paper [3], there are a total of 16 parametric states that can occur both in terms of R_a, R_b, L, C and p_r, p_i, c_r, c_i. Each could be negative or positive. So since R_a, R_b, L, C have been determined in equation 4.15, p_r, p_i, c_r can be determined in term of R, L, C as show below [3].

$$\begin{aligned} c_r &= \frac{1}{2L}, \\ p_r &= \frac{-R_a}{2L} + \frac{-1}{2CR_b} \\ c_i &= \frac{L - CR_aR_b)}{2L\sqrt{-C^2R_a^2R_b^2 + 2CLR_b(R_a + 2R_b) - L^2}} \\ p_i &= \frac{\sqrt{-C^2R_a^2R_b^2 + 2CLR_b(R_a + 2R_b) - L^2}}{2CLR_b} \end{aligned} \tag{4.16}$$

Next are the cumulative average power of the ITF and the cumulative average power

of the ATF, respectively [3].

$$S_{cavg}^{ITF^{RLCG}} = \frac{i_0^2(\pi m(R_a+R_b) - C\omega_0 R_b^2)}{2\pi m(C^2\omega_0^2 R_b^b + 1)^2} + \frac{Ci_0^2\omega_0 R_b^2 e^{(\frac{-1}{CR_b}\frac{2\pi m}{\omega_0})}}{2\pi m(C^2\omega_0^2 R_b^2+1)^2} + \frac{C^2 i_0^2\omega_0^2 R_b^2(R_a(C^2\omega_0^2 R_b^2+2)+R_b)}{2(C^2\omega_0^2 R_b^2+1)^2} \tag{4.17}$$

$$S_{cavg}^{ATF^{RLCG}} = \frac{f_4 v_0^2}{2\pi f_6^2 m} + \frac{f_5 v_0^2}{2m} + \frac{v_0^2\omega_0}{2\pi m} e^{\frac{\pi m}{\omega_0}(\frac{-R_a}{L}+\frac{-1}{CR_b})} + \left((\frac{f_2}{f_6^2\sqrt{f_1}} sin\left(\frac{\pi\sqrt{f_1}m}{CL\omega_0 R_b}\right) + \frac{f_3}{f_6^2} cos\left(\frac{\pi\sqrt{f_1}m}{CL\omega_0 R_b}\right)\right) \tag{4.18}$$

And where f's are defined as follows [3]:

$$\begin{aligned}
f1 &= -C^2R_a^2R_b^2 + 2CLR_b(R_a+2R_b) - L^2 \\
f2 &= -(R_a+R_b)^2(C^2R_aR_b^3 + CLR_b(R_a+3R_b) - L^2) \\
&\quad +2C^2L\omega_0^2R_b^2(R_a+R_b)^2(L-CR_aR_b) \\
&\quad +C^3L\omega_0^4R_b^4(-C^2R_a^3R_b + CLR_a(R_a+3R_b) + L^2) \\
f_3 &= C^3L\omega_0^4R_b^4(-C^2R_a^3 - L) - 2C^2L\omega 0^2R_b^2(R_a+R_b)^2 + (R_a+R_b)^2(L-CR_b^2) \\
f_4 &= C^3L\omega_0^5R_b^4(L-CR_a^2) - 2C^2L\omega_0^3R_b^2(R_a+R_b)^2 + \omega_0(R_a+R_b)^2(CR_b^2-L) \\
f_5 &= C^2\omega_0^2R_aR_b^2 + R_a + R_b \\
f_6 &= R_z^2(C^2\omega_0^2R_b^2+1) + 2R_aR_b + R_b^2(CL\omega_0^2-1)^2 + L^2\omega_0^2
\end{aligned} \tag{4.19}$$

$$S_{cavg}^{ITF^{RLCG}} = \lim_{m\to\infty} S_{cavg}^{ATF^{RLCG}} = \frac{Rv_0^2}{2L^2\omega_0^2 + 2R^2} \tag{4.20}$$

$$S_{cavg}^{ATF^{RLCG}} = \lim_{m\to\infty} S_{cavg}^{ITF^{RLCG}} = \frac{i_0^2(C^2\omega_0^2R_aR_b^2 + Ra + Rb)}{2C^2\omega_0^2R_b^2+2} \tag{4.21}$$

The equation 4.21 can be simplified to[3]:

$$R_a + \frac{R_b}{1 + C^2\omega_0^2 R_b^2} > 0 \tag{4.22}$$

And if $p_r < 0$, the condition is equivalent to the following:

$$\frac{-R_a}{2L} + \frac{-1}{2CR_b} < 0 \tag{4.23}$$

In term of stability and passivity of each state, they are described in following table for both ATF and ITF. Note that states 2, 3, 6, 7, 10, 10, 11, 14, and 15 are not producing pair complex conjugate for poles and residues. However, they produce real poles and real residues; therefore, they are not included in the tables because they can use tables 4.1, 4.2 , 4.3, and 4.4 to determine the stability and passivity.

Table 4.5: Summary of R/L states of a single arbitrary branch of RLRC equivalent circuit [3]

State No.	Equivalent Circuit
1	$R_a > 0$ and $R_b > 0$ and $L > 0$ and $C > 0$ and $cond_C$
2	$R_a > 0$ and $R_b > 0$ and $L > 0$ and $C < 0$
3	$R_a > 0$ and $R_b > 0$ and $L < 0$ and $C > 0$
4	$R_a > 0$ and $R_b > 0$ and $L < 0$ and $C < 0$ and $cond_C$
5	$R_a > 0$ and $R_b < 0$ and $L > 0$ and $C > 0$ and $cond_C$
6	$R_a > 0$ and $R_b < 0$ and $L > 0$ and $C < 0$
7	$R_a > 0$ and $R_b < 0$ and $L < 0$ and $C > 0$
8	$R_a > 0$ and $R_b < 0$ and $L < 0$ and $C < 0$ and $cond_C$
9	$R_a < 0$ and $R_b > 0$ and $L > 0$ and $C > 0$ and $cond_C$
10	$R_a < 0$ and $R_b > 0$ and $L > 0$ and $C < 0$
11	$R_a < 0$ and $R_b > 0$ and $L < 0$ and $C > 0$
12	$R_a < 0$ and $R_b > 0$ and $L < 0$ and $C < 0$ and $cond_C$
13	$R_a < 0$ and $R_b < 0$ and $L > 0$ and $C > 0$ and $cond_C$
14	$R_a < 0$ and $R_b < 0$ and $L > 0$ and $C < 0$
15	$R_a < 0$ and $R_b < 0$ and $L < 0$ and $C > 0$
16	$R_a < 0$ and $R_b < 0$ and $L < 0$ and $C < 0$ and $cond_C$

Table 4.6: Summary of pole/residue states of a single arbitrary branch of RLRC equivalent circuit [3]

State No.	Pole/Residue
1	$p_r < 0$ and $c_r > 0$ and $p_i \neq 0$ and $c_r p_r < c_i p_i$ and $c_r p_r + c_i p_i < 0$
2	Incompatible with a complex-conjugate pole/residue pair
3	Incompatible with a complex-conjugate pole/residue pair
4	$p_r > 0$ and $c_r < 0$ and $p_i \neq 0$ and $c_r p_r < c_i p_i$ and $c_i p_i + c_r p_r < 0$
5	$c_r > 0$ and $p_i \neq 0$ and ($c_r p_r < c_i p_i$ or $p_r \leq 0$) and ($p_r > 0$ or $c_i p_i + c_r p_r > 0$)
6	Incompatible with a complex-conjugate pole/residue pair
7	Incompatible with a complex-conjugate pole/residue pair
8	$c_r < 0$ and $p_i \neq 0$ and ($c_r p_r < c_i p_i$ or $p_r > 0$) and ($p_r \leq 0$ or $c_i p_i + c_r p_r > 0$)
9	$c_r > 0$ and $p_i \neq 0$ and ($c_i p_i < c_r p_r$ or $p_r > 0$) and ($c_i p_i + c_r p_r < 0$ or $p_r \leq 0$)
10	Incompatible with a complex-conjugate pole/residue pair
11	Incompatible with a complex-conjugate pole/residue pair
12	$c_r < 0$ and $p_i \neq 0$ and ($c_i p_i < c_r p_r$ or $p_r <= 0$ and ($c_i p_i + c_r p_r < 0$ or $p_r > 0$
13	$p_r > 0$ and $c_r > 0$ and $p_i \neq 0$ and $c_i p_i < c_r p_r$ and $c_i p_i + c_r p_r > 0$
14	Incompatible with a complex-conjugate pole/residue pair
15	Incompatible with a complex-conjugate pole/residue pair
16	$p_r < 0$ and $c_r < 0$ and $pi \neq 0$ and $c_i p_i < c_r p_r$ and $c_i p_i + c_r p_r > 0$

Table 4.7: Stability and Passivity of each branch for IFT case for pair complex-conjugate of poles/residues [3].

State No.	Stable	Passive
1	Yes	Yes
4	No	Omit
5	No	Omit
8	Yes	$Cond_{P_{IFT}}$
9	Yes	$Cond_{P_{IFT}}$
12	No	Omit
13	No	Omit
16	Yes	No

Table 4.8: Stability and Passivity of each branch for AFT case for pair complex-conjugate of poles/residues [3].

State No.	Stable	Passive
1	Yes	Yes
4	No	Omit
5	$Cond_{SC}$	$Cond_{P_{AFT}}$
8	$Cond_{SC}$	$Cond_{P_{AFT}}$
9	$Cond_{SC}$	$Cond_{P_{AFT}}$
12	$Cond_{SC}$	$Cond_{P_{AFT}}$
13	No	Omit
16	Yes	No

As you notice that some of the states need to meet condition ($Cond_{P_{ATF}}$) in order to be passive in ATF, and similarly to ITF, it must meet condition ($Cond_{P_{IFT}}$) in order to be passive in ITF. What that means is that if the power of each TF branch must greater than zero according to equation 4.22 or 4.21 for ITF, and 4.20 for ATF, the circuit branch is passive. Readers are asked to accept the definitions listed in the table for parametric states. All the mathematical equation derivations are provided in the paper [3]. In states 5, 8, 9, 12 are conditional stable ($Cond_{SC}$), the equivalent circuit branch is stable or unstable is depending on the condition of parameters so that the exponential term goes to zeros as time increases (i.e. only the real part of pole value, p_r, is less than 0), then system is stable.

Chapter 5

Numerical Examples

5.1 Numerical Example 1 - Buck Power Converter

5.1.1 Introduction

The switching voltage regulator is very popular in circuit system and design, such as, motherboard, communication board, and other devices. It also helps to increase the design flexibilty, where it can provide localized converting power from high voltage to lower voltage depending on the need. One of the most common uses in switching voltage regulator is the "Buck Power Converter." The buck converter sends energy from input to output, and store it in the inductors and capacitors in a fraction of a period, and use it in the remaining time of the full period. The buck converter is a simple step-down voltage converter, where it provides output voltage less than input voltage. The authors in [15] had modeled transfer function for Buck Power Converter to ensure the stability between input and output voltage, the stability between load and output voltage, and a good transient response between different disturbance signals. From the modeled transfer funtion of Z_0, VF will generate the poles and residues of transfer function Z_0 based on actual given discrete Z_0 frequency responses.

5.1.2 Overview

Through this section, we will look at the formulation of load impedance transfer function, and use that to generate actual frequency samples. The data is then fed into VF Python codes to identify poles and residues. The generated poles and residues from Python VF codes will be used to compute the fitting response, and then compare the fitting responses (fitted Z_0) against the actual (Z_0). The stability and the passivity will

be discussed based on result [15].

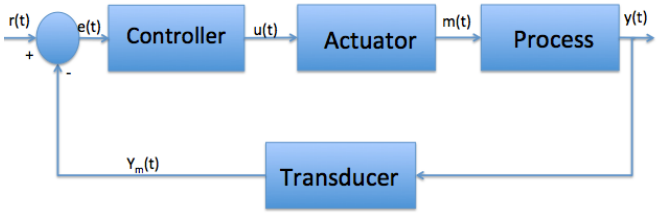


Figure 5.1: Buck Control Loop Block Schematic

.

5.1.3 Formulation

The buck converter, in most cases, the control loop is a closed loop circuit. The figure 5.1 is the control loop block schematic, where Y(t) represents the output, or the process variable (PV), yM(t) are the detected values of the PV, and r(t) is the set point (SP) of the process. Also, the circuit schematic for the buck converter is showed in 5.2 [15].

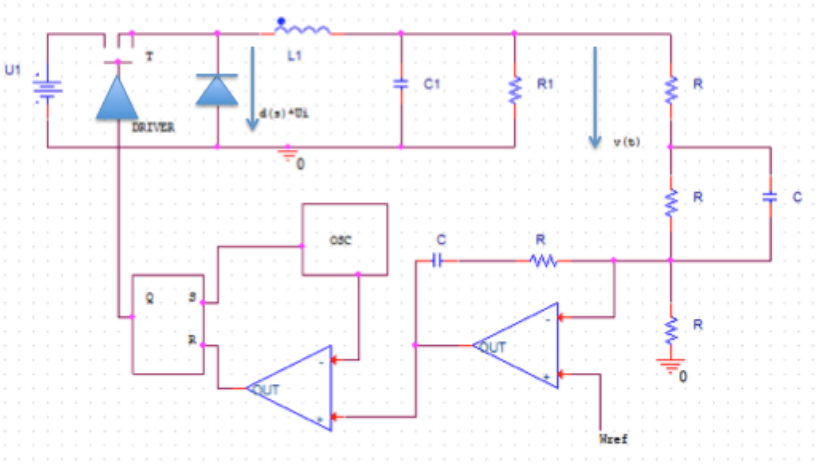


Figure 5.2: Voltage Control Loop for Buck Converter.

For the transfer function between input and output (voltage/current), it is obtained as:

$$H_{th}(s) = \frac{D}{1+s(L/R)+s^2LC} \tag{5.1}$$

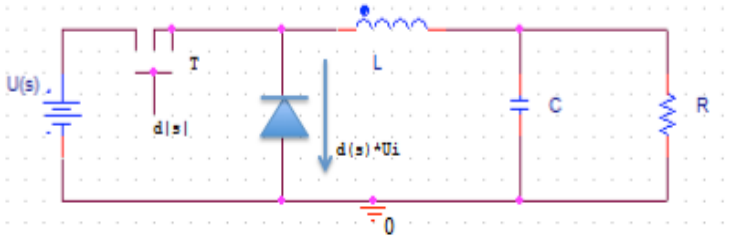


Figure 5.3: Block Converter

To obtain the output impedance, Z_0, we need to obtain it from following circuit, 5.4. From the circuit schematic, the circuit equivalent as following:

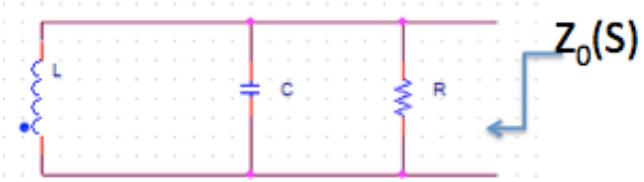


Figure 5.4: Output Impedance for Buck Converter.

Expanding equation (5.2) into

$$\begin{aligned} Z_0(s) &= R||Z_L||Z_C \\ Z_0(s) &= \frac{R}{1-jR(1/X_C+1/X_L)} \\ Z_0(s) &= \frac{R}{1-jR(1/wC-+1/wL)} \\ Z_0(s) &= \frac{sL}{1+s(L/R)+s^2LC} \end{aligned} \tag{5.2}$$

5.1.4 Generate Frequency Response

In the paper, the author used following parameter values: R=30 Ohms, L = 2mH, and C = 100uF. S = jω. With these values, the frequency responses for N samples

(N=100) are calculated. In this case the discrete samples and responses are calculated from original function from equation 5.2.

The magnitude of actual f(s), VectFit f(s) and Relaxed VectFit f(s) are shown in figure 5.5.

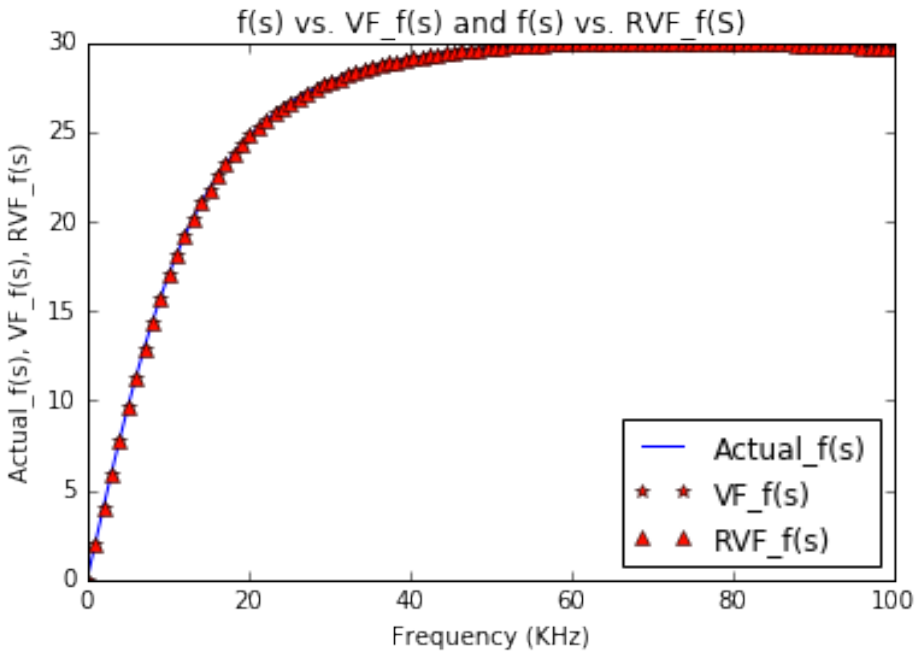


Figure 5.5: Responses of Buck Power Converter

The errors are very small between them, which also can be seen in figure 5.6. The error between f(s) and RVF f(s) is smaller than the error between f(s) and VF f(s). RVF codes generate data as expected, which is more accurate compared to VF codes.

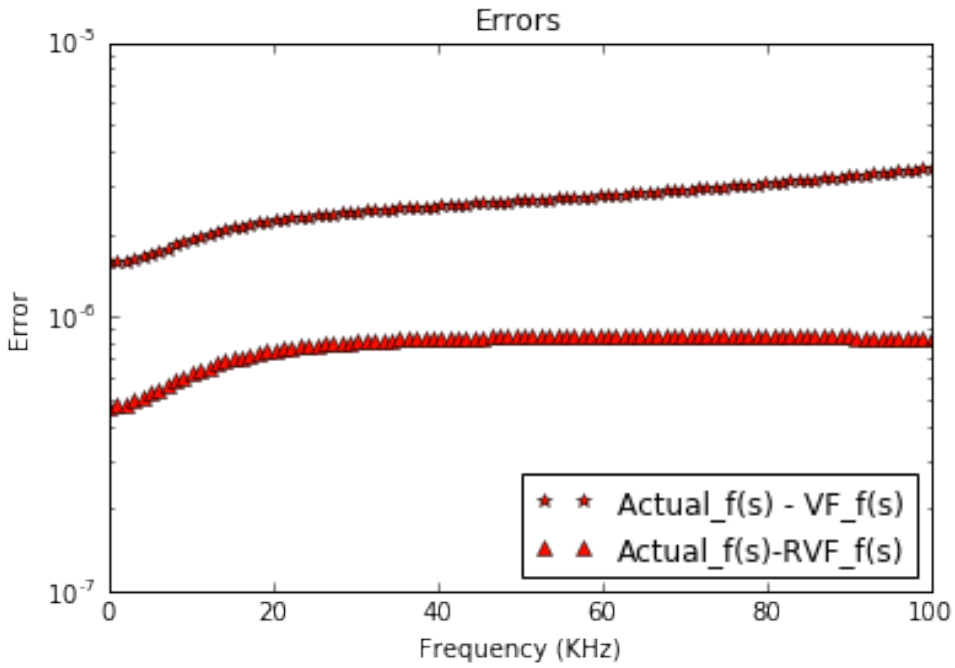


Figure 5.6: Errors of VF and RVF

5.1.5 Poles and Residues Identification by VF and RVF using Python Codes

From the VF codes, fitted poles and fitted residues are identified as showed in the table below. Poles and Residues from table 5.1 and table 5.2 are almost identical. The fitted d and h values are 8.55572e-06 , -5.82077e-11, respectively. However, when compare the error of the two, RVF provides a more accurate approximation compared to VF. One side note, the total time for this program to run this computation is 0.665 second.

Table 5.1: VF Poles and Residues of 2nd Order Buck Power Converter

VF Poles	VF Residues
-317589.7523023	10521573.00
-15743.58103104	-521576.25

Table 5.2: RVF Poles and Residues of 2nd Order Buck Power Converter

RVF Poles	RVF Residues
-317589.7523023	10521576.00
-15743.58103104	-521576.28

5.1.6 Stability and Passivity Discussion for VF Result

Based on fitted poles and fitted residues data generated from VF and RVF methods, we might able to conclude that the macromodel is stable. There are three main reasons to support that the fitted data is stable: 1) the fitted real pole values are negative, which land on left half plane of real/imaginary plane [18] - [19], [33] - [36] and this result is expected because the VF algorithm is written to force stability; 2) since stability is defined as the conventional method, bounded input bounded output (BOBO) [3]. And as t increases, y(t) decays to zero, and from figure 5.5, it can be concluded that the impedance is approaching a constant as frequency or time increases; 3) if the Z is converted to Y (i.e. Y=Z^{-1}), and according to paper [3] or table 4.1, the response falls into for state one ($p < 0$ and $c > 0$) and state 4 ($p < 0$ and $c < 0$), therefore, fitted impedance generated by VF is stable.

In term of passivity, according to paper [3] or table 4.1, the first pole/residue pair is passive. The second pair pole/residue is not passive due to its average power ($Ri_0^2/2$) is less than zero [3]. However, that is passivity for per branch. We are now exploring the passivity of the network instead of per branch level. According to theorem 1 and 2 of paper [2], it is concluded that the results generated from VF algorithm is passive because the eigenvalues of Hamiltonian matrix (4.5) are not $j\omega_0$. The eigenvalues of the

Hamiltonian is showed in table below. Based on these data, the branch level could be nonpassive but the whole network is passive.

Table 5.3: Eigenvalues of Hamiltonian for Buck Converter

Eigenvalues
-5.84404335e+11 + 5.84404335e+11j
-5.84404335e+11 - 5.84404335e+11j
-4.27962435e-03 + 4.20289549e-03j
-4.27962435e-03 - 4.20289549e-03j

5.2 Numerical Example 2 - Efficient Symbolic Circuit Analysis Based Transfer Function and Input Impedance Computation

5.2.1 Introduction

In the high performance systems such as microprocessor, IC packages or others, the methods or tools using in the design play a significant role in the design process. The more efficient technique would provide advantages to design engineers to understand the effects of the electrical parameters of the device. In the high performance logic circuits design, one of tools that has been used is called, the symbolic circuit analysis methodology. This method computes various transfer functions of the circuits, such as, current, voltage, input impedance, etc. One of the commercial available software's that is highly used in generating transfer functions for a complicated circuit system is field solver and SPICE. However, these softwares are not powerful built-in functions that could to provide more efficient computation of transfer function. In other words, the softwares can generate the transfer function in term of the symbolic circuit variables, which requires additional computation to transform the transfer function to a simplified form. In paper [17], the author has developed a new methodology, which will compute

the required transfer functions without any algebraic manipulations of circuit equations, and the final transfer function equation is in form of two polynomials in terms of 's', and the power of 's' in both numerator and denominator are in the descending format. And the coefficients are in term of circuit parameters (i.e. R, L, C). A physical system of high performance system, microprocessor, will be discussed in later section.

With the VF method, it can be used to approximate the transfer function based on input frequencies and responses. And transfer function can be in different forms either a ratio of two polynomials, a poles-zeros, or pole-residues.

5.2.2 Overview

In this section, it is a brief discussion of transfer function in electrical circuit system. A microprocessor system will be discussed and the novel symbolic circuit analysis methodology given in paper [17] is applied to generate a transfer function for an input impedance. Next, the VF method is used to approximate the transfer function of input impedance and compare it with the given numerical transfer function, which is generated by the novel symbolic method. The stability and the passivity are discussed at the end. In a microprocessor system, the input impedance of the Core-PDN (Core-Power Delivery Network) must be resonant free and the value must be less than or equal to a target impedance value, specified over a frequency range; therefore, knowing input impedance is critical for characterization.

The Core-PDN model of a microprocessor system consists of ground planes, Via of the PCB, the IC-package, and the microprocessor die. The parasitic parameters (resistances, capacitances, inductances) associated with each component above play important role in the power integrity performance of the Core-PND. In this specific example, we want to look at the Core-PDN in the IC Package and the PCB, which can be represented by a single-section stage R-L-C lump linear circuit model or multiple stages R-L-C lump circuit model.

5.2.3 Transfer Function of Electrical Circuit

In the electrical system, the linear differential equation can be described in 5.3. The y(t) is the response corresponding to an excitation x(t) in a linear electrical. Assuming that the response $y(t) = Ye^{st}$ and the excitation $x(t) = Xe^{st}$ [17].

$$\begin{aligned} a_n\frac{d^{n-1}y(t)}{dt^n} + a_{n-1}\frac{d^{n-1}y(t)}{dt^{n-1}} + \cdots + a_2\frac{d^2y(t)}{dt^{n-2}} + a_1\frac{d^1y(t)}{dt^{n-1}} + a_0y(t) = b_n\frac{d^{n-1}x(t)}{dt^n} \\ +b_{n-1}\frac{d^{n-1}x(t)}{dt^{n-1}} + \cdots + b_2\frac{d^2x(t)}{dt^{n-2}} + b_1\frac{d^1x(t)}{dt^{n-1}} + b_0x(t) \end{aligned} \tag{5.3}$$

The transfer function can be defined as following:

$$H(s) = \frac{Y}{X} = \frac{\text{Magnitude of the response}}{\text{Magnitude of excitation}} \tag{5.4}$$

So the transfer function can be obtained as a ratio of two polynomials in term of s, which is the response to excitation in electrical circuit system, and $s = j\omega$:

$$H(s) = \frac{Y}{X} = \frac{b_ns^n + b_{n-1}s^{n-1} + \cdot + b_2s^2 + b_1s + b_0}{a_ma^m + b_{m-1}s^{m-1} + \cdot + a_2s^2 + a_1s + a_0} = \frac{B(s)}{A(s)} \tag{5.5}$$

So in order to obtain frequency domain transfer function for the impedance parameters of circuit elements, R, L, C, we need to convert these circuit elements into transfer function impedances accordingly.

5.2.4 Transfer Functions of Microprocessor System

Before getting into actual numerical analysis of the transfer functions of the microprocessor system, the basic concepts of the microprocessor system will be discussed in this section. As mentioned earlier, the microprocessor system is considered as a high-performance digital system. It consists of a printed circuit board (PCB) mounted on a switch voltage regulator (VRMs) and PCV Decaps, and the core/die and the package

Decaps are attached to microprocessor package. The PCB and the microprocessor circuits are connected by microprocessor package using flip-chip ball-grid array (FCBGA) and wirebond-BGA.

The core/die power delivery network is combined of the Vias of the PCB, the IC-package, microprocessor die, and the power planes, and ground planes.

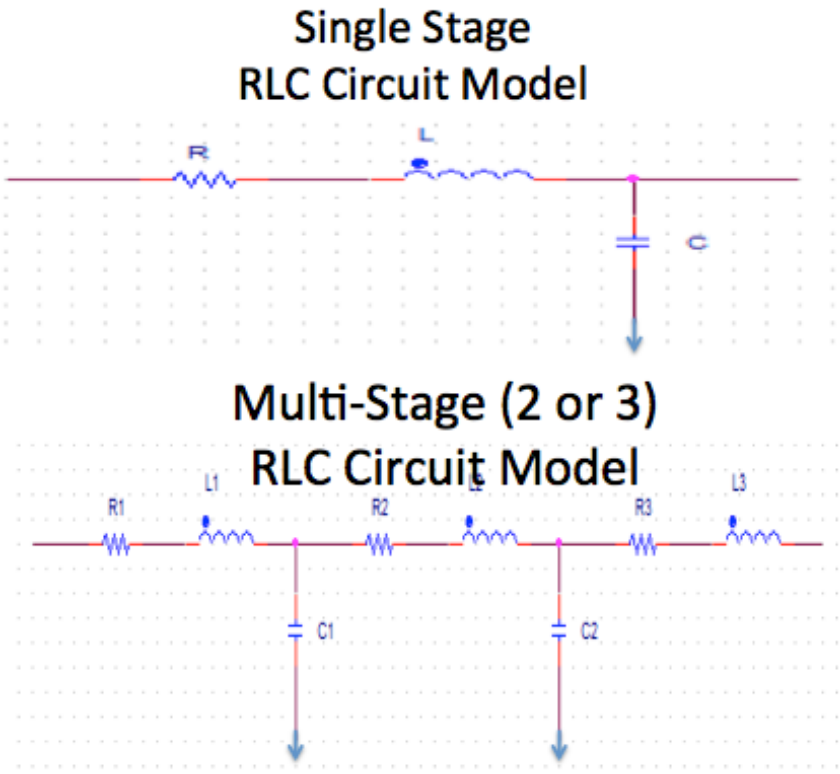


Figure 5.7: Single Stage and Multi Stage R-L-C Circuit Model

Because of the complication of the structure of power and ground planes, PCB, Package of microprocessor system, the parasitic elements play significant roles in the power performance of the Core-DPN and for doing the characterization of the system [17]. And the characterization is using lumped linear circuit model either single section

or multiple sections

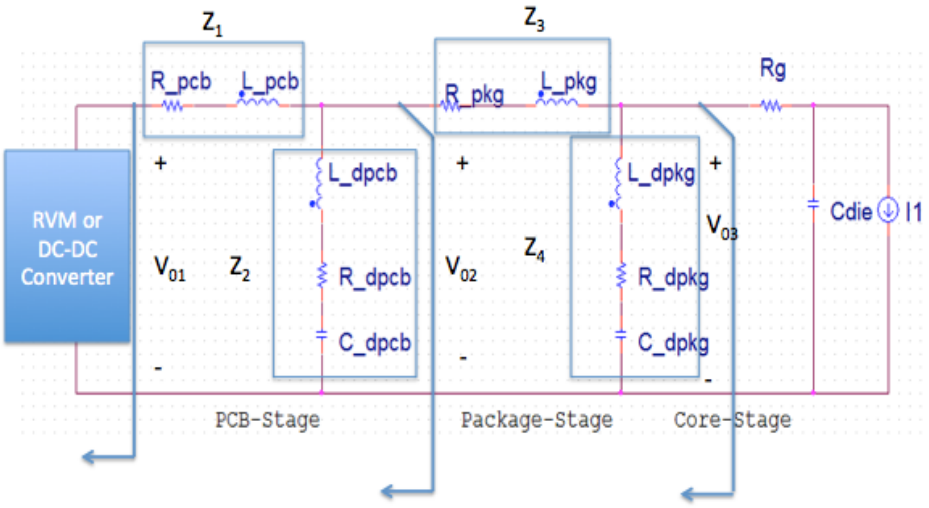


Figure 5.8: Single Stage and Multi Stage R-L-C Circuit Model for low frequency range [17]

In the case of optimizing the input impedance, use small signal closed loop circuit like in figure 5.8. However, when there is the impedance in the low frequency range, it might need to be analyzed with the small-close loop linear circuit model with VRM.

Based on the discussion above, the input impedance parameter is very critical in the microprocessor system. Input impedance can be simulated by Field Solver or PSPICE software without problem. In the complicated system, the computation is a little time consuming. One example is that the conventional commercial symbolic analysis did not put transfer function in the best known form which is power of s descended. In paper [17], the author has been modified the method to provide the transfer function in the form of descending the power of s, and it is easier to read and interpret.

Here is an overview of how available commercial Field Solver software works. Assume that using Field Solver to solve for transfer function of figure 5.9, in order to compute an equivalent input impedance for this circuit, Z1, Z2, Z3, and Z4 are computed and Zd is computed from Z1 and Z4. The symbolic functions below in Field Solver software

outputs the transfer function of Zd. However, the transfer function is not generated in the descending power of s, and the coefficients are described in terms of passive elements, which are little hard to read.

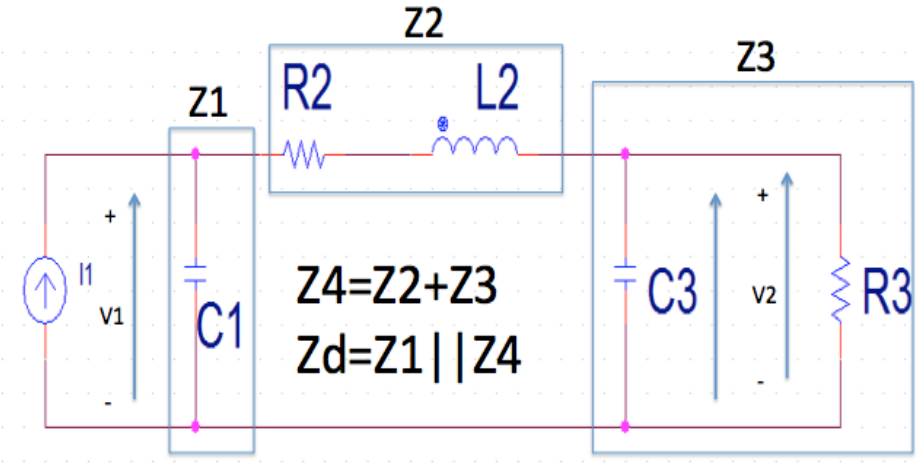


Figure 5.9: Simple R-L-C Linear Lump Circuit Model, [17]

5.2.5 Generate Frequency Response

Here we use a specific example, which computes the input impedance of IC-Package-core-PDN. The transfer function of input impedance is derived using field solver application, which is also shown below [17].

$$
\begin{aligned}
Z_{input} \;=\; & \big(1.78e-071s^7 + 9.76e-062s^6 + 4.06e-050s^5 + 2.15e-40s^4 \\
& +1.98e-029s^3 + 1.02e-19s^2 + 3.54e-011s + 0.000357\big) \\
& \big/\big(9.72e-61s^6 + 5.33e-51s^5 + 1.87e-39s^4 \\
& +9.87e-30s^3 + 5.78e-19s^2 + 2.97e-9s + 1\big)
\end{aligned}
$$

The VF and the RVF methods can be used to approximate the poles- residues, and the poles-zeros for this transfer function. Assume the sample frequency and response

are given or calculated from input impedance above, which are 1000 samples (could be measured from physical device or calculated from the given transfer function), and this is 6 poles transfer function with 5 iterations. In this case, the VF is not able to approximate neither poles nor residues because the response elements are very small. As shown in figure 5.10 that the VF method does not converge for approximated response, even though the stability has been forced, and the error is still very large, which is displaced in figure 5.11.

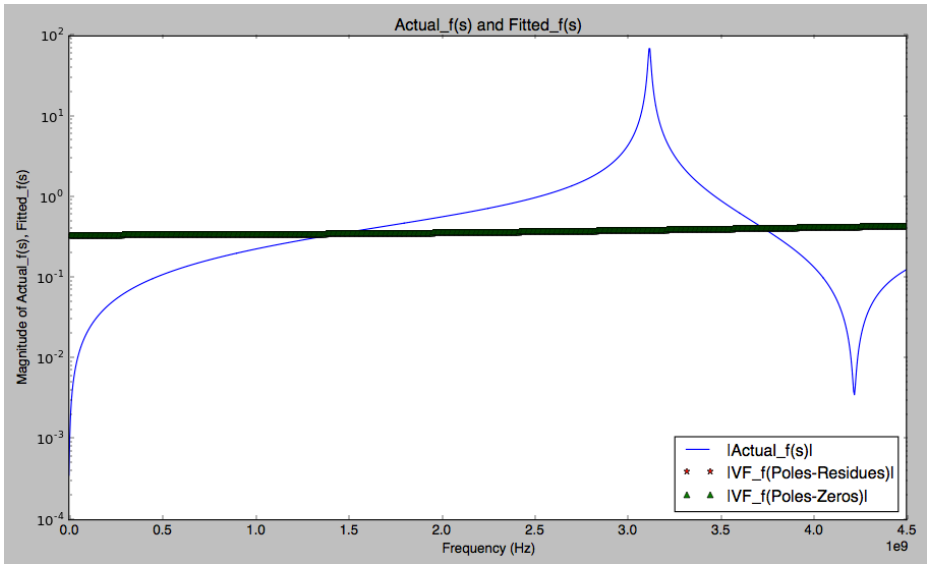


Figure 5.10: Magnitude of Response Computed by VF Method with Stability Forced, 6 poles and 5 iterations

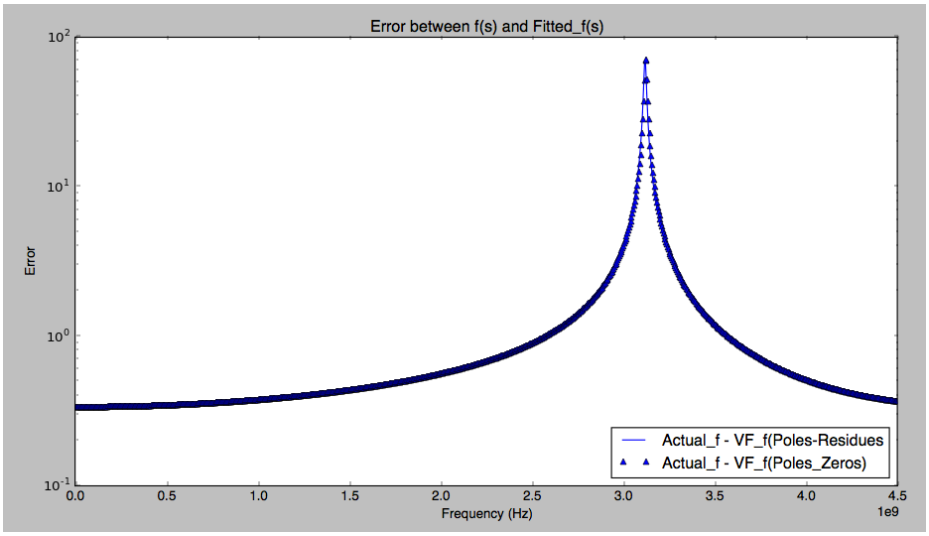


Figure 5.11: Error between Actual f(s) and Fitted f(s)

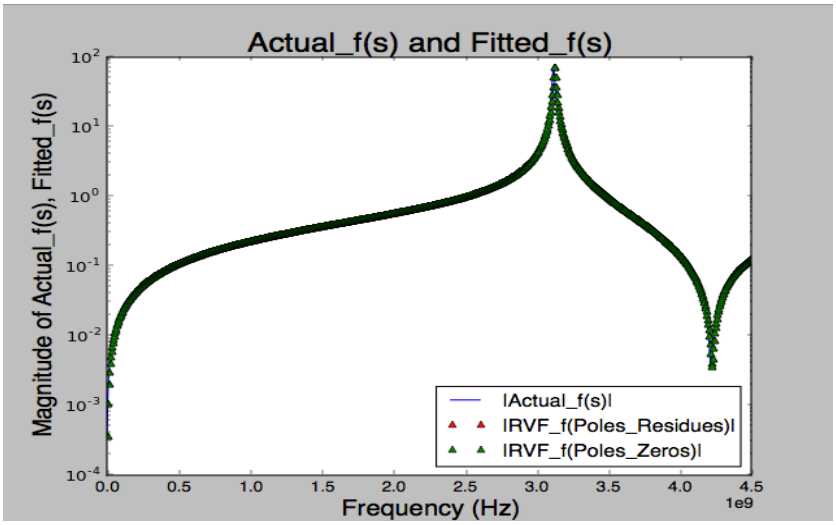


Figure 5.12: Magnitude of Response Computed by RVF Method with Stability Forced, 6 poles and 5 iterations

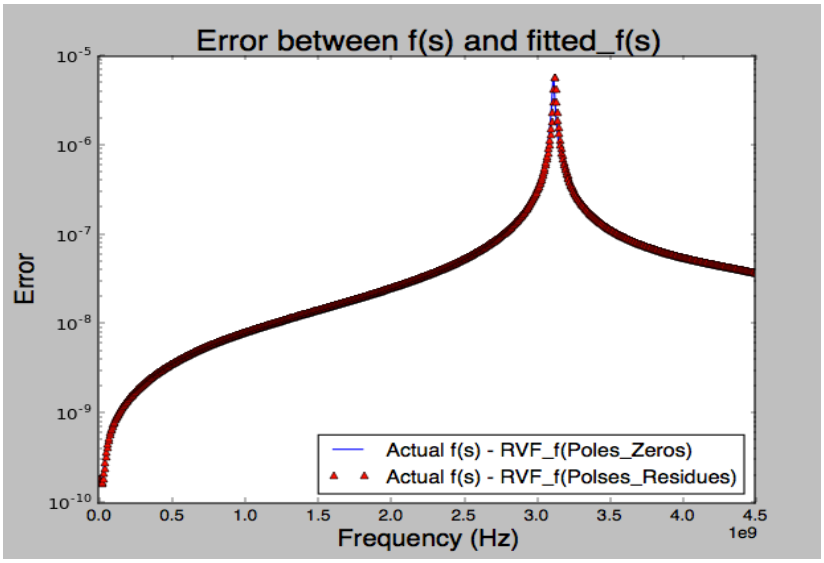


Figure 5.13: Error between Actual f(s) and Fitted f(s)

In this numerical example, `R_VF` method would be used to provide a more accurate results, and the outcome is very precise and well matched with the actual response. As figure 5.12 shows very close to actual response, and the error in figure 5.13. Total time to execute the program is 5.5 seconds.

5.2.6 Poles and Residues Identification by VF and RVF

Since the VF method is not providing convergence for the response, so poles-residues and poles-zeros would not be mentioned here. However, for the `R_VF` method, the poles-residues and the poles-zeros can be seen in the table below. And d, and h values are -7.19776927223e-06 , and 1.83127576937e-11, respectively.

Table 5.4: RVF Poles and Residues of Input Impedance of Core-PDN

RVF Poles	RVF Residues
-4814418747	1722058
-362069418.1	-9836.78
-112647209.5 - 39196245466 j	253312864 - 671121.68 j
-112647209.5 + 39196245466 j	253312864 + 671121.68 j
-40877395.58 - 19599895649 j	3014957312 - 5986366.00 j
-40877395.58 + 19599895649 j	3014957312 + 5986366.00 j

Table 5.5: RVF Poles and Zeros of Input Impedance of Core-PDN

RVF Poles	RVF Zeros
-4814418747	-111659588.2- 3.968E+10 j
-362069418.1	-111659588.2 + 3.9681E+10 j
-112647209.5 - 39196245466 j	-41420407.46 - 2.651E+10 j
-112647209.5 +39196245466 j	-41420407.46 + 2.6512E+10 j
-40877395.58 - 19599895649 j	-4803702837.34 - 3.94E-08 j
-40877395.58 + 19599895649 j	-362885999.58 + 7.78E-10 j
-	-10395500.18 +1.08E-12 j

5.2.7 Stability and Passivity Discussion for VF Results

Once again, the reals part of the poles are negative values; therefore, it is a guarantee that this system is stable, which is expected because the VF algorithm is desired to force stability; also, since stability is defined as the conventional method bounded input bounded output (BOBO) [3] and equation 4.4, as t increases, y(t) decays to zero, so it can be concluded that the output impedance is approaching a constant as frequency increases. According to paper [3], branches a and b (first pole and second pole) are stable because $p < 0$ and $c > 0$, $p < 0$ and $c > 0$, and branches c, d, e, f (3rd-6th poles/residues), they are also stable and matched conditions stated in [3].

Regarding to passivity, according to paper [3], table 4.1 and table 4.5 to 4.8, the second pole/residue is not passive and all remaining pole/residue pairs are passive. For the entire system, according to theorem 1 and 2 of paper [2], it is concluded that the

result generated from VF algorithm is passive because the eigenvalues of Hamiltonian matrix (4.5) are not $j\omega_0$. The eigenvalues of the Hamiltonian is showed in table below. Again, even though one out of five branches in this system is nonpassive, and the network is still passive.

Table 5.6: Eigenvalues of Hamiltonian M for Core PDN

Eigenvalues
4.54186034e+14 +4.54186036e+14j
4.54186034e+14 -4.54186036e+14j
-1.07272308e+08 +3.80402482e+10j
1.07344511e+08 +3.80402478e+10j
-1.07272308e+08 -3.80402482e+10j
1.07344511e+08 -3.80402478e+10j
-4.79068474e+09 +4.38433209e-06j
4.79068001e+09 -1.38753286e-04j
-3.63885401e+08 -9.76290750e-04j
-2.22641203e+07 +8.52829265e-04j
2.31617811e+07 +3.97974731e-04j
3.63890214e+08 -5.56722350e-06j])

5.3 Numerical Example 3 - Two-Port Network S and Y Parameters

5.3.1 Introduction

The two-port network is an electrical network (electrical circuit) and been widely used in circuit analysis for electrical engineering, microelectronic, wireless communication, filter, transformer, and small-signal models. In the circuit analysis, the two-port network model is used to isolate portions of larger circuits. The properties of two-port network can be specified by a matrix of numbers. In other words, it describes the how the network responds at each port when signals are applied to the ports, and the responses can be calculated through the matrix form as well. Scattering parameters (S-parameters) matrix is commonly used in the electrical network characterization for two-port network,

three-port network, four-port network, or n-port network. The S-parameter allows a device to be treated as a "black box" with inputs and resulting outputs, and makes it possible to model a system without having to deal with complexity of the actual structure inside of the device. For a two-port network, the network is described by 2x2 square matrix. There are many other parameter models that can be used to describe the n-port networks, such as, Z-parameters, Y-parameters, H-parameters, G-parameters, and ABCD-parameters.

In the electrical engineering field, engineers are always looking for an efficient approach or an improved circuit analysis technique for their works either in design, model simulation, or data analysis. There are many ways to synthesize the network into equivalent circuit and ensure the system is stable such as in papers [1] [31]. However, it appears that the combined VF approximation method and approach converting numerical TF to synthesized RLRG circuit in paper [3] is a simple procedure, straight forward, and guarantee that the system will be stable. Therefore, in this numerical example, the S-parameters and Y-parameters of a two-port network will be discussed. And the vector fitting method demonstrates that it can be used in analyzing TF for the S and Y parameters. With the combination of the technique in paper [3] and the VF method, they can determine the equivalent circuit elements of the two-port network based on the given calculated or the measured S-parameter data. Stability and passivity of this two-port system will be discussed.

5.3.2 Overview

Assuming that the S-parameters of a two-port network are measured or calculated, and in this case, the data set of S-parameters of a power plane square 500mil x 500mil is used for this study purpose. The relaxed vector fitting (RVF) is used to determine the transfer function (TF) for each S-parameter at each port (i.e. S11, S12, S21, S22). And again, the RVF is used to determine the transfer function for Y-parameters at

each port (i.e. Y11, Y12, Y21, Y22). Once the approximated transfer function of each Y-parameter is found, the Python script is used to convert the network system from S-parameters to Y-parameters (admittance parameters). With the Y-parameters available, they are arranged in a pi-circuit configuration, which represents a Y-equivalent circuit for a reciprocal two-port network. Y11+Y11, -Y12, Y22+Y12 are three main components of pi-equivalent circuit. We use equations in paper [3] and along with Python codes to determine the RLRC elements. At the end, the stability and the passivity will be discussed for each component..

5.3.3 Formulation of Two-Port Network

S-Parameters:

Considering a two-port network typical example in figure 5.14 below, where I_1 is current into port 1; V_1 is voltage across port 1; V_2 is voltage across port 2; and I_2 is current into port 2. The current enters at one terminal and leaves at other terminal of the port. Sometime, it is called a black box, and its properties are described by parameter matrix.

Figure 5.14: Two-Port Network Configuration

Figure 5.14 is simple and mainly for a low frequency network where parameters are defined in term of currents and voltages at ports. However, for the high frequency network, such as in ultra high frequency range or microwave frequency (GHz) range, there is difficulty in measuring the voltages, the currents, the admittance and the impedance

directly from the network, the s-parameters are used and defined as described in figure 5.15, where a_1 and a_2 are the incident power waves, b_1 and b_2 are the reflected power waves [27]-[29].

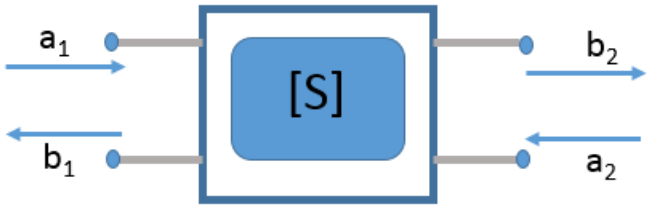


Figure 5.15: S-Parameters Two-Port Network Configuration

In this case, the relationship of these parameters is determined in equation 5.6 and equation

$$\begin{pmatrix} b_1 \\ b_2 \end{pmatrix} = \begin{pmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \tag{5.6}$$

Expanding the s-matrices, and we have:

$$\begin{aligned} b_1 &= S_{11}a_1 + S_{12}a_2 \\ b_2 &= S_{21}a_1 + S_{22}a_2 \end{aligned} \tag{5.7}$$

S_{11} is the input port voltage reflection coefficient. $S_{11} = b_1/a_1 = V_1^-/V_1^+$.

S_{12} is the reverse voltage gain. $S_{12} = b_1/a_2 = V_1^-/V_2^+$.

S_{21} is the forward voltage gain. $S_{21} = b_2/a_1 = V_2^-/V_1^+$.

S_{22} is the output port voltage reflection coefficient. $S_{22} = b_2/a_2 = V_2^-/V_2^+$.

Where V^+ and V^- are forward wave and reverse wave, respectively.

Y-Parameters:

Y-parameters are also one of the common models used for the two-port network. The equivalent circuit for an arbitrary two-port admittance matrix and relationship between i-v characteristics are described below.

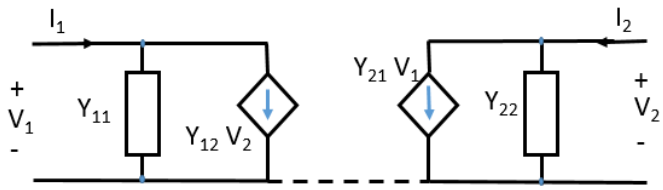


Figure 5.16: Equivalent circuit of an arbitrary two-port network

$$\begin{pmatrix} I_1 \\ I_2 \end{pmatrix} = \begin{pmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \end{pmatrix} \tag{5.8}$$

Expand the y-matrices, and we have:

$$\begin{aligned} I_1 &= Y_{11}V_1 + Y_{12}V_2 \\ I_2 &= Y_{21}a_1 + Y_{22}V_2 \end{aligned} \tag{5.9}$$

From the following equations and the circuit in figure 5.16, we can derive and obtain the Y-equivalent circuit for a reciprocal two-port network with pi-topology in terms of Y-parameters. So in order to measure the admittance parameters, we apply excitation at one port and short circuit at other port [33]-[38]. That means when measurement or computation occurs at port 1, the port 2 is shorted ($V_2 = 0$):

$$\begin{aligned} y_{11} &= (I_1/V_1)|_{V_2=0} \\ y_{21} &= (I_2/V_1)|_{V_2=0} \end{aligned} \tag{5.10}$$

When measurement or computation occurs at port 2, the port 1 is shorted ($V_1 = 0$):

$$\begin{aligned} y_{12} &= (I_1/V_2)|_{V_1=0} \\ y_{22} &= (I_2/V_2)|_{V_1=0} \end{aligned} \tag{5.11}$$

To summarize the equations of Y-parameters above, the y_{11} is the input admittance with the output port shorted. y_{21} is the forward transfer admittance with the output

port shorted. y_{12} is the reverse transfer admittance with the input port shorted. y_{22} is output admittance with the input port shorted. To simplify the circuit configuration, Y-equivalent circuit for a reciprocal two-port network can be achieved, which can be seen in figure 5.17.

To derive Y_A, Y_B, and Y_C, output port 2 shorted, and now short circuit at port 2 connects admittances Y_A and Y_B in parallels. So the admittance looking in at port 1, $y_{11} = Y_A + Y_B$. Also, $I_2 = -Y_B V_1$; therefore, $y_{21} = -Y_B$.

With the same approach, output port 1 shorted, and now short circuit at port 2 connects admittances Y_B and Y_C in parallels. So the admittances looking in at port 2, $y_{22} = Y_B + Y_C$. Also, $I_1 = -Y_C V_2$. Therefore, $y_{12} = -Y_C$.

$$\begin{aligned} y_{11} &= Y_A + Y_B \\ y_{21} = y_{11} &= -Y_B \\ y_{22} &= Y_B + Y_C \end{aligned} \tag{5.12}$$

Solve for Y_A, Y_B, and Y_C:

$$\begin{aligned} Y_A &= y_{11} + y_{12} \\ Y_B &= -y_{12} \\ Y_C &= y_{22} + y_{12} \end{aligned} \tag{5.13}$$

Now Y_A, Y_B and Y_C of the pi circuit can be replaced by Y-parameters, which are shown in figure 5.17.

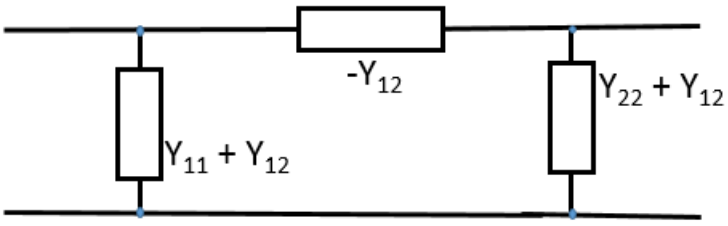


Figure 5.17: Y-equivalent circuit with Y-parameters for a reciprocal two-port network

Finally, the relationship between S-parameters and Y-parameters can be described by following equations, where 1_N is the identity matrix and $\sqrt{y}$ is a diagonal matrix:

$$\begin{aligned} Y &= \sqrt{y}(1_N - S)(1_N + S)^{-1}\sqrt{y} \\ \sqrt{y} &= (\sqrt{z})^{-1} \end{aligned} \tag{5.14}$$

5.3.4 Poles and Residues and Zeros Identification for S-Parameter

The goal of this numerical example is to use RVF algorithm method combined with other approach to determine the circuit elements, which are converted from the TF of the Y-parameters, and arrange them in form of equivalent pi-circuit in figure 5.17. The set of S-parameters of square power plane 500milx500mil, `(500mil_planes_Log500_10M5G.s2p)`, is used. It contains of 1351 frequency response samples, and the frequency range are from 10MHz to 5GHz. In order to pull S11, S12, S21, S21 from this s2p file, the relaxed vector fitting has been modified by adding the following codes (note that `Test1.s2p is the 500mil_planes_Log500_10M5G.s2p`:

```
Test1 = rf.Network(rf.data.pwd+'/Test1.s2p') ####read s2p file.
S11=Test1.s[:,0,0] ####assign S11 in Test1 to S11 variable.
freq=Test1.f ####read the frequency from Test1.s2p
Actual_f = S11 ####assign S11 as the actual f(s)
```

Also, note that if we want to pull S21, the codes we need to use is `S21=Test1.s[:,1,0]`. After modification, the relaxed vector fitting is able to read the S-Parameters file, pull S-parameter at each terminal and generate the fitted poles-residues, poles-zeros, d, and h values. The Python codes write an output.csv file. The outpout.csv file contains variables, which are shown in tables 5.7 and 5.8. The values of d, and h are 0.998605269104,

and -2.03719477524e-15, respectively.

Table 5.7: Fitted Poles and Residues of S11 Parameter from RVF

Pole Real	Pole Imag	Residue Real	Residue Imag
-2851596767	0	-46426280	0
-1698304537	-50538923338	-180649840	42968088
-1698304537	50538923338	-180649840	-42968088
-1040642002	0	-964501120	0
-791828239.4	-22878305474	-675556608	25730972
-791828239.4	22878305474	-675556608	-25730972
-273616394.7	0	-13356906	0
-11467071.63	0	-143.1921387	0

Table 5.8: Fitted Poles and Zeros of S11 Parameter from RVF

Pole Real	Pole Imag	Zeros Real	Zero Imag
-2851596767	0	4.90184E+14	-4.64E-11
-1698304537	-50538923338	-1515710193	50571773456
-1698304537	50538923338	-1515710193	-50571773456
-1040642002	0	-116045216.6	-22866650812
-791828239.4	-22878305474	-116045216.6	22866650812
-791828239.4	22878305474	-2821734193	-1.09E-10
-273616394.7	0	-312663902	5.28E-10
-11467071.63	0	-11501203.65	-1.51E-09
-	-	-7659768.42	1.29E-09

To validate the result of the RVF python codes, the magnitude of the Fitted f(s) of Poles-Residues and the magnitude of the Fitted f(s) of the Poles-Zeros are plotted against the actual f(s) of S11, S12, S21, and S22. Also, the error between the actual f(s) and the fitted f(s) have been plotted. The plots showed that the fitted and the actual data are very matched.

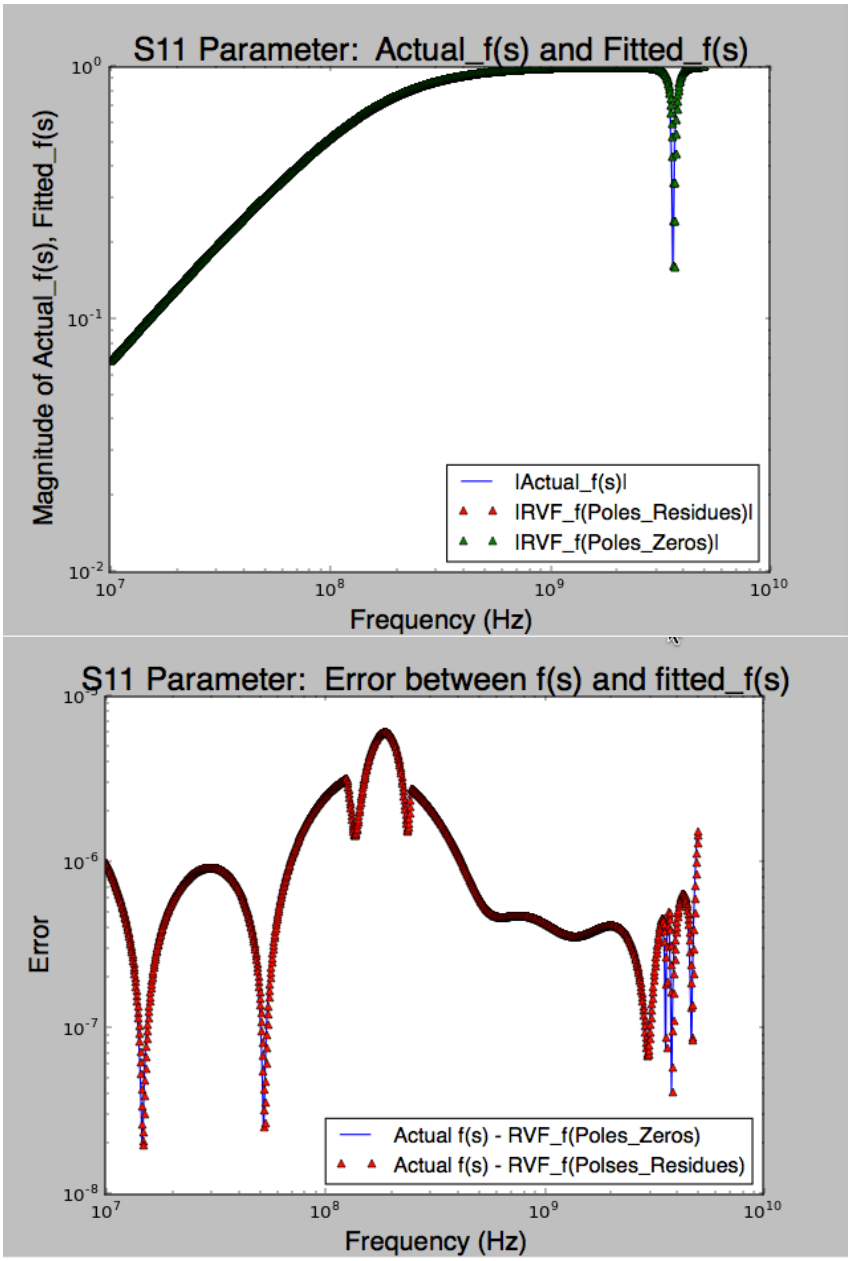


Figure 5.18: S11 Parameter - Top: Fitted f(s) and Actual f(s); Bottom: Error between Fitted f and Actual f

Next are the fitted poles-residues and fitted poles-zeros, which are generated from actual S12 and shown in table below, and once again, the plots have shown the fitted data and the actual data are matched. The values of d and h are 0.998605269104, and

-2.03719477524e-15, respectively.

Table 5.9: Fitted Poles and Residues of S12 Parameter from RVF

Pole Real	Pole Imag	Residue Real	Residue Imag
-2846964496	0	49799248	0
-1594938745	-50411707643	-122971424	-7664338
-1594938745	50411707643	-122971424	7664338
-1040536841	0	963040512	0
-791827676.7	-22878304727	-414793824	2670030
-791827676.7	22878304727	-414793824	-2670030
-273429909.7	0	13376316	0
-13042530.8	0	-148.7841797	0

Table 5.10: Fitted Poles and Zeros of S12 Parameter from RVF

Pole Real	Pole Imag	Zeros Real	Zero Imag
-2846964496	0	4.65094E+11	1.52588E-05
-1594938745	-50411707643	56826375803	91174822188
-1594938745	50411707643	56826375803	-91174822188
-1040536841	0	-76466213959	6.47459E-06
-791827676.7	-22878304727	-770071042.5	37373914015
-791827676.7	22878304727	-770071042.5	-37373914015
-273429909.7	0	-2758301643	0
-13042530.8	0	-283781274.9	0
-	-	-13042382.89	0

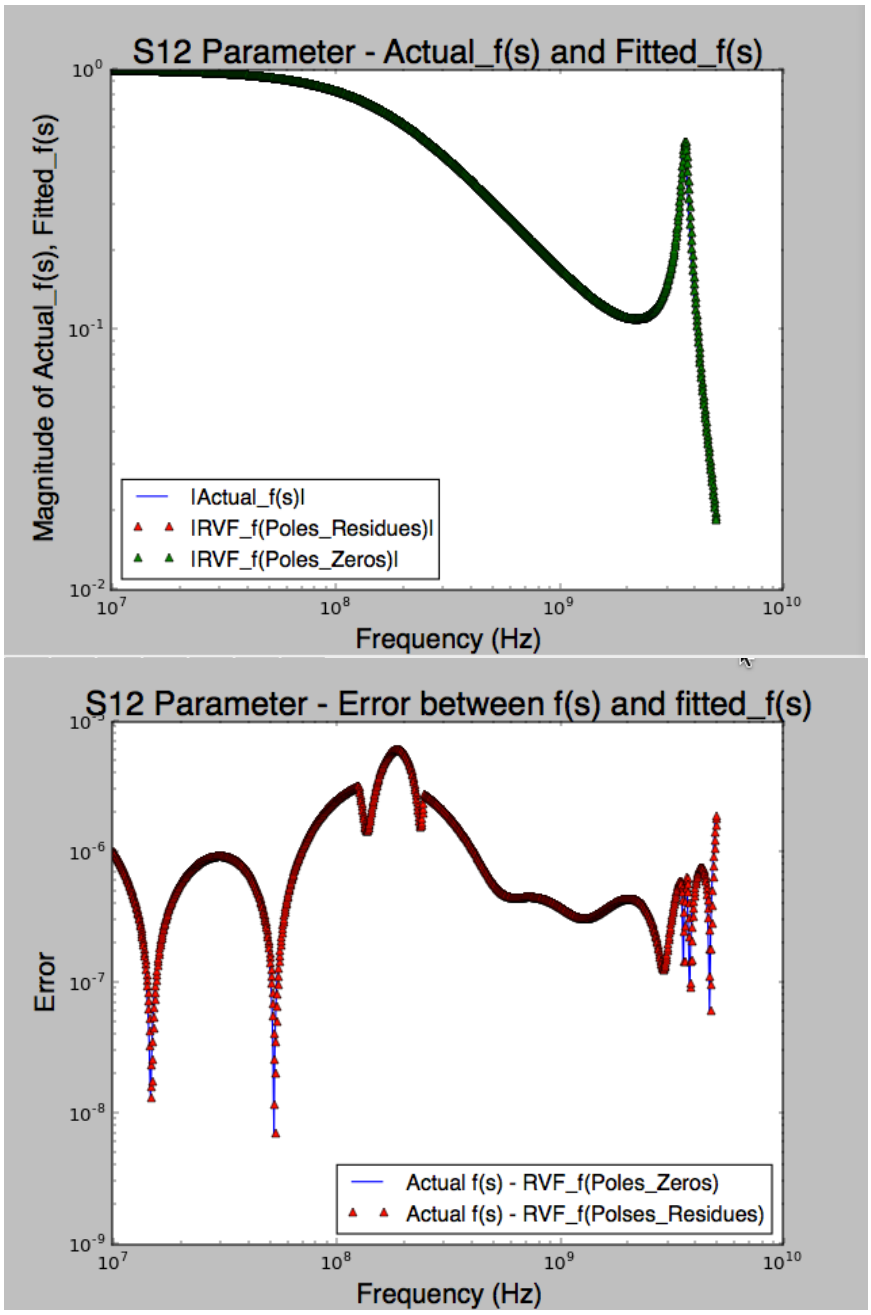


Figure 5.19: S12 Parameter - Top: Fitted f(s) and Actual f(s); Bottom: Error between Fitted f and Actual f

Below is the S21 parameter result. The results have demonstrated the data is matched among the actual f(s) (i.e. S21) and the fitted S21. The values of d and h values are 0.00120827885705, and -2.38492259281e-15, respectively. S21 is matched

with S12 result.

Table 5.11: Fitted Poles and Residues of S21 Parameter from RVF

Pole Real	Pole Imag	Residue Real	Residue Imag
-2846964496	0	49799248	0
-1594938745	-50411707643	-122971424	-7664338
-1594938745	50411707643	-122971424	7664338
-1040536841	0	963040512	0
-791827676.7	-22878304727	-414793824	2670030
-791827676.7	22878304727	-414793824	-2670030
-273429909.7	0	13376316	0
-13042530.8	0	-148.7841797	0

Table 5.12: Fitted Poles and Zeros of S21 Parameter from RVF

Pole Real	Pole Imag	Zeros Real	Zero Imag
-2846964496	0	4.65094E+11	1.52588E-05
-1594938745	-50411707643	56826375803	91174822188
-1594938745	50411707643	56826375803	-91174822188
-1040536841	0	-76466213959	6.47459E-06
-791827676.7	-22878304727	-770071042.5	37373914015
-791827676.7	22878304727	-770071042.5	-37373914015
-273429909.7	0	-2758301643	0
-13042530.8	0	-283781274.9	0
-	-	-13042382.89	0

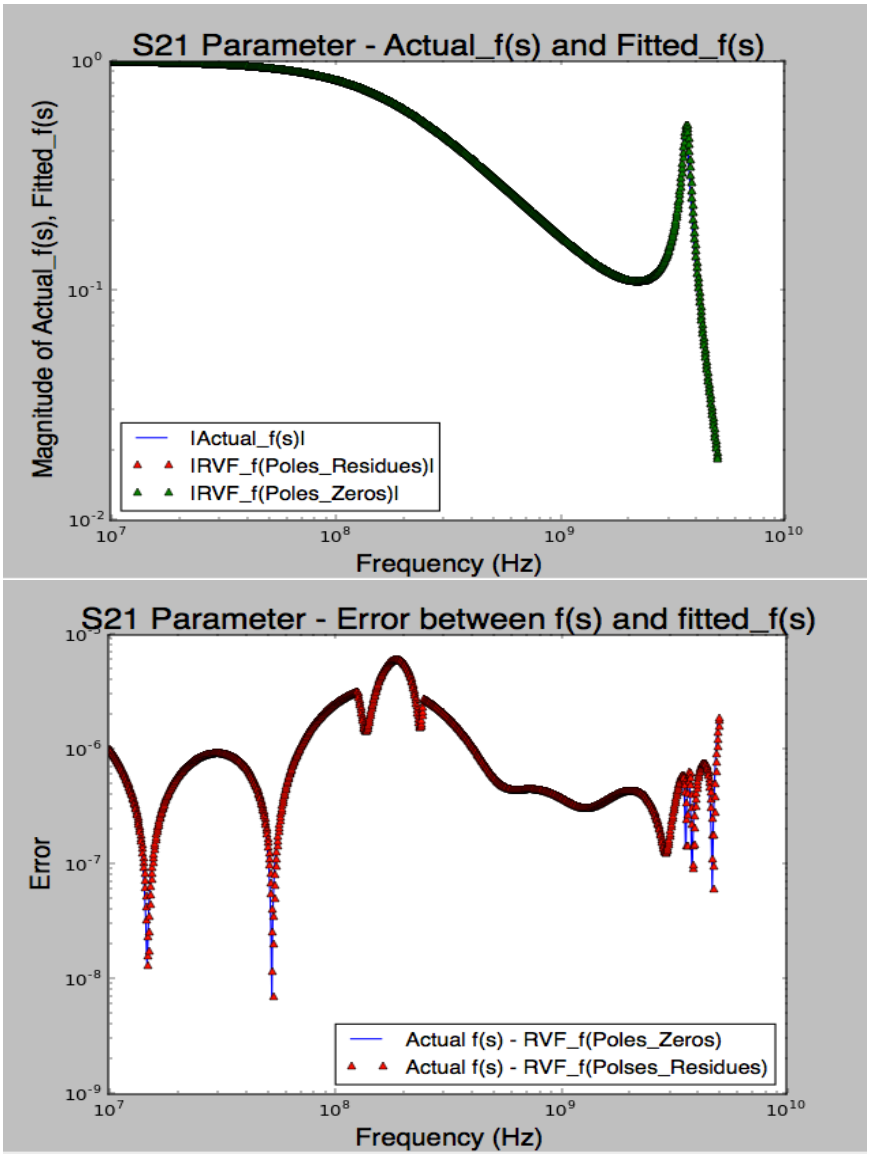


Figure 5.20: S21 Parameter - Top: Fitted f(s) and Actual f(s); Bottom: Error between Fitted and Actual

Finally, the fitted f(s) of S22 parameter can be seen below. And the values of d and

h are 0.998295676941 and -1.3293793129e-15, respectively.

Table 5.13: Fitted Poles and Residues of S22 Parameter from RVF

Pole Real	Pole Imag	Residue Real	Residue Imag
-2851314294	0	-47917824	0
-1702969011	-50700454303	-741793216	82072056
-1702969011	50700454303	-741793216	-82072056
-1040640565	0	-962818560	0
-791828110.8	-22878305092	-254282720	-6251064.5
-791828110.8	22878305092	-254282720	6251064.5
-273614312.5	0	-13372677	0
-11145627.82	0	-139.6633301	0

Table 5.14: Fitted Poles and Zeros of S22 Parameter from RVF

Pole Real	Pole Imag	Zeros Real	Zero Imag
-2851314294	0	7.50946E+14	-4.68390E-11
-1702969011	-50700454303	-958480539	50752788294
-1702969011	50700454303	-958480539	-50752788294
-1040640565	0	-537989016.8	-22863508461
-791828110.8	-22878305092	-537989016.8	22863508461
-791828110.8	22878305092	-2820396136	1.59553E-10
-273614312.5	0	-312730200.5	5.17530E-10
-11145627.82	0	-11181956.84	1.32842E-10
-	-	-7655107.036	1.73907E-10

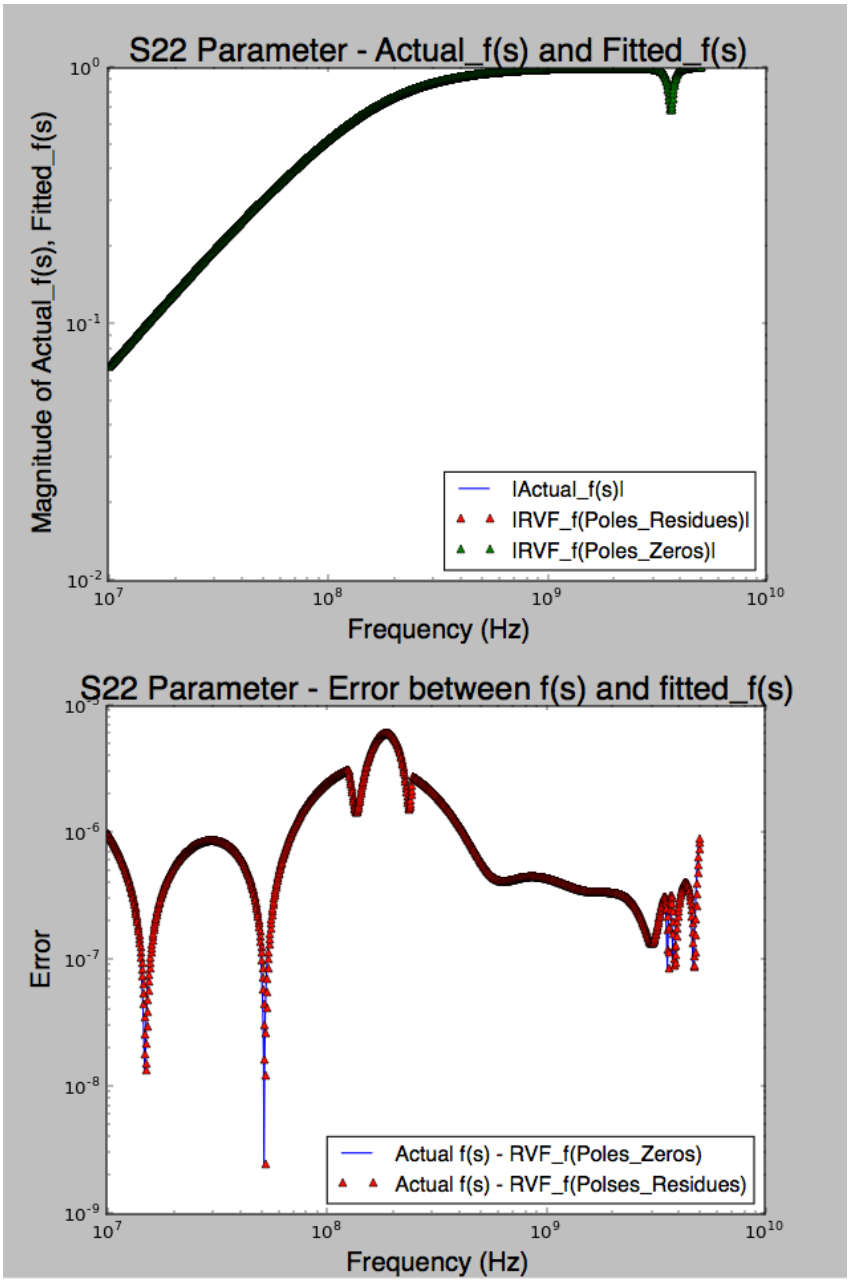


Figure 5.21: S22 Parameter - Top: Fitted f(s) and Actual f(s); Bottom: Error between Fitted and Actual

As the results of S-parameters, the python program has forced the real part of the poles to be negative; therefore, the poles-residues of the S-parameters are stable. We will discuss further both stability and passivity of this system when it gets to Y-parameter

analysis.

5.3.5 Poles and Residues Identification for Y-Parameter

In order to determine and find the pi equivalent circuit in term of the admittances, the Y-parameters need to be determined. To do this, the Python codes convert S-parameters of the system to Y-parameters. The code `[y_param=rf.network.s2y(Test1.s, z0=[0.1+0.j, 0.1+0.j])]` can do the job easily, where file Test1 contains S-parameters, and z0 is the characteristic impedance of the network. Once S-parameters are converted to Y-parameters, poles-residues and poles-zeros transfer function of Y11, Y12, -Y21, and Y22 can be generated through RVF Python codes, similarly to how transfer functions of S11, S12, S21, and S22 are generated. The fitted poles, fitted residues, and fitted zeros can be seen in table 5.15 and table 5.16, and the magnitude of fitted f(s) and actuals f(s) are shown in 5.22. The values of d and h are 0.00756353757021 and -1.0305001589e-14, respectively.

Table 5.15: Fitted Poles and Residues of Y11 Parameter from RVF

Pole Real	Pole Imag	Residue Real	Residue Imag
-2811484324	0	91698000	0
-1760229740	-50915726227	949572672	-182932560
-1760229740	50915726227	949572672	182932560
-1065467427	0	1196216	0
-326964909.8	-22884492884	3378653952	-21642676
-326964909.8	22884492884	3378653952	21642676
-312594039.9	0	472312768	0
-7676854.43	0	4554145280	0

Table 5.16: Fitted Poles and Zeros of Y11 Parameter from RVF

Pole Real	Pole Imag	Zeros Real	Zero Imag
-2811484324	0	1.579249E+12	1.81980E-06
-1760229740	-50915726227	-8.468016E+11	3.90498E-06
-1760229740	50915726227	-1.141373E+09	47760702506
-1065467427	0	-1.141373E+09	-47760702506
-326964909.8	-22884492884	-228392976	14871406390
-326964909.8	22884492884	-228392976	-14871406390
-312594039.9	0	-2762703086	2.79397E-08
-7676854.43	0	-1065222896	2.80845E-08
-	-	-283882537.4	1.08143E-08

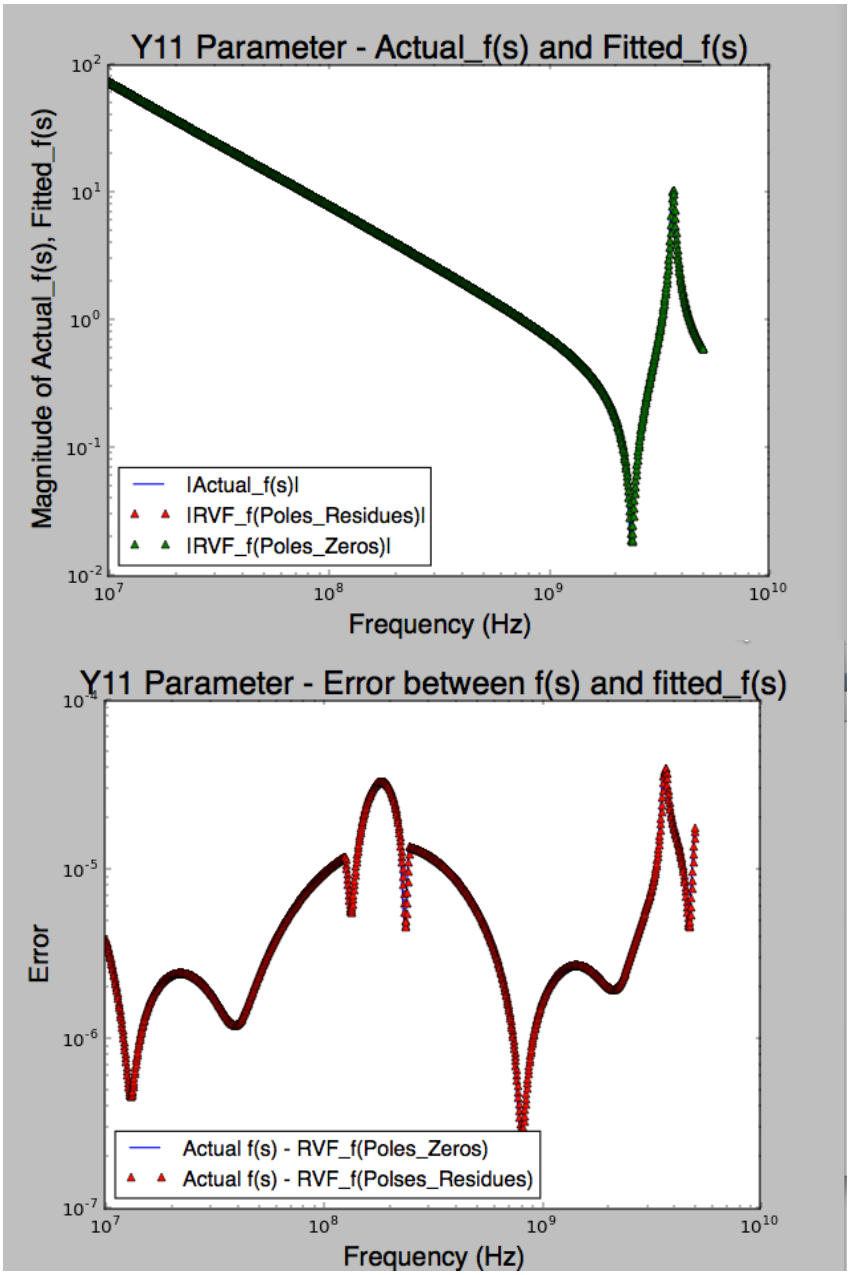


Figure 5.22: Y11 Parameter - Top: Fitted f(s) and Actual f(s); Bottom: Error between Fitted and Actual

Next are the results of Y12, Y21, and Y22 of Y-parameters.

Table 5.17: Fitted Poles and Residues of Y12 Parameter from RVF

Pole Real	Pole Imag	Residue Real	Residue Imag
-2.81932E+09	0	-106823688	0
-1.11898E+09	0	1330723.25	0
-698174696.9	-4.98129E+10	563616320	87218288
-698174696.9	4.98129E+10	563616320	-87218288
-326983812.8	-2.28845E+10	2070813568	-10916837
-326983812.8	2.28845E+10	2070813568	10916837
-312573133.7	0	-472273280	0
-7676873.839	0	-4554151936	0

Table 5.18: Fitted Poles and Zeros of Y12 Parameter from RVF

Pole Real	Pole Imag	Zeros Real	Zero Imag
-2.81932E+09	0	1.74336E+11	1.52588E-05
-1.11898E+09	0	3.73045E+10	94536643107
-698174696.9	-4.98129E+10	3.73045E+10	-94536643107
-698174696.9	4.98129E+10	-7.26920E+10	3.30444E-05
-326983812.8	-2.28845E+10	-737797632.7	3.73453E+10
-326983812.8	2.28845E+10	-737797632.7	-3.73453E+10
-312573133.7	0	-2760703379	1.75473E-07
-7676873.839	0	-1119264726	-1.78448E-07
-	-	-283856339.3	-3.35363E-09

The values of d and h are -0.00646493582417 and 3.65411910549e-14, respectively.

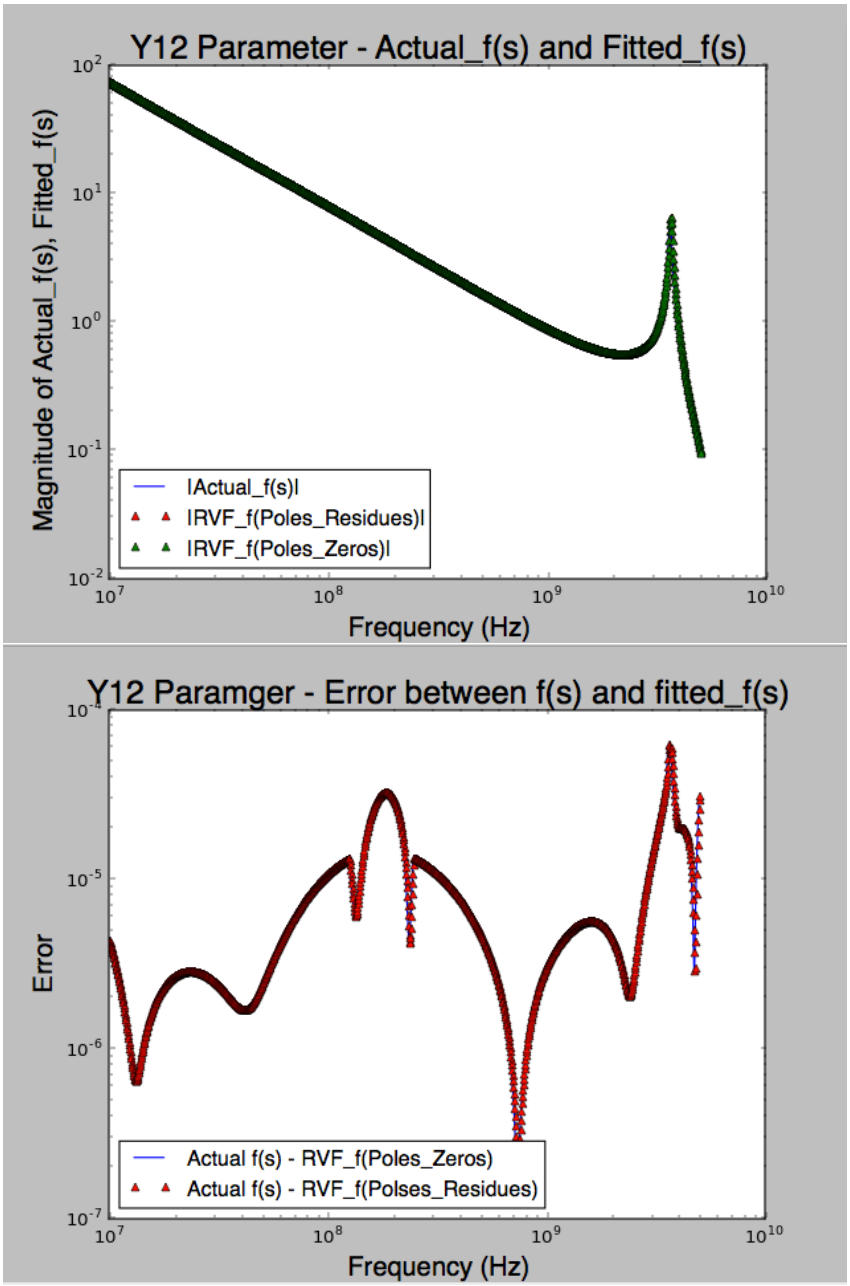


Figure 5.23: Y12 Parameter - Top: Fitted f(s) and Actual f(s); Bottom: Error between Fitted and Actual

The Y12 parameter and the Y21 parameter are identical, which indicates that the S-parameter to Y-parameter conversion is working correctly. The values of d and h are

-0.00646493582417, and 3.65411910549e-14, respectively.

Table 5.19: Fitted Poles and Residues of Y21 Parameter from RVF

Pole Real	Pole Imag	Zeros Real	Zero Imag
-2.81932E+09	0	-106823688	0
-1.11898E+09	0	1330723.25	0
-698174696.9	-4.98129E+10	563616320	87218288
-698174696.9	4.98129E+10	563616320	-87218288
-326983812.8	-2.28845E+10	2070813568	-10916837
-326983812.8	2.28845E+10	2070813568	10916837
-312573133.7	0	-472273280	0
-7676873.839	0	-4554151936	0

Table 5.20: Fitted Poles and Zeros of Y21 Parameter from RVF

Pole Real	Pole Imag	Zeros Real	Zero Imag
-2.81932E+09	0	1.74336E+11	3.05176E-05
-1.11898E+09	0	37304454248	94536643107
-698174696.9	-4.98129E+10	37304454248	-94536643107
-698174696.9	4.98129E+10	-72692007427	7.09240E-06
-326983812.8	-2.28845E+10	-737797632.7	37345323544
-326983812.8	2.28845E+10	-737797632.7	-3.73453E+10
-312573133.7	0	-2760703379	1.81106E-08
-7676873.839	0	-1119264726	-6.18774E-08

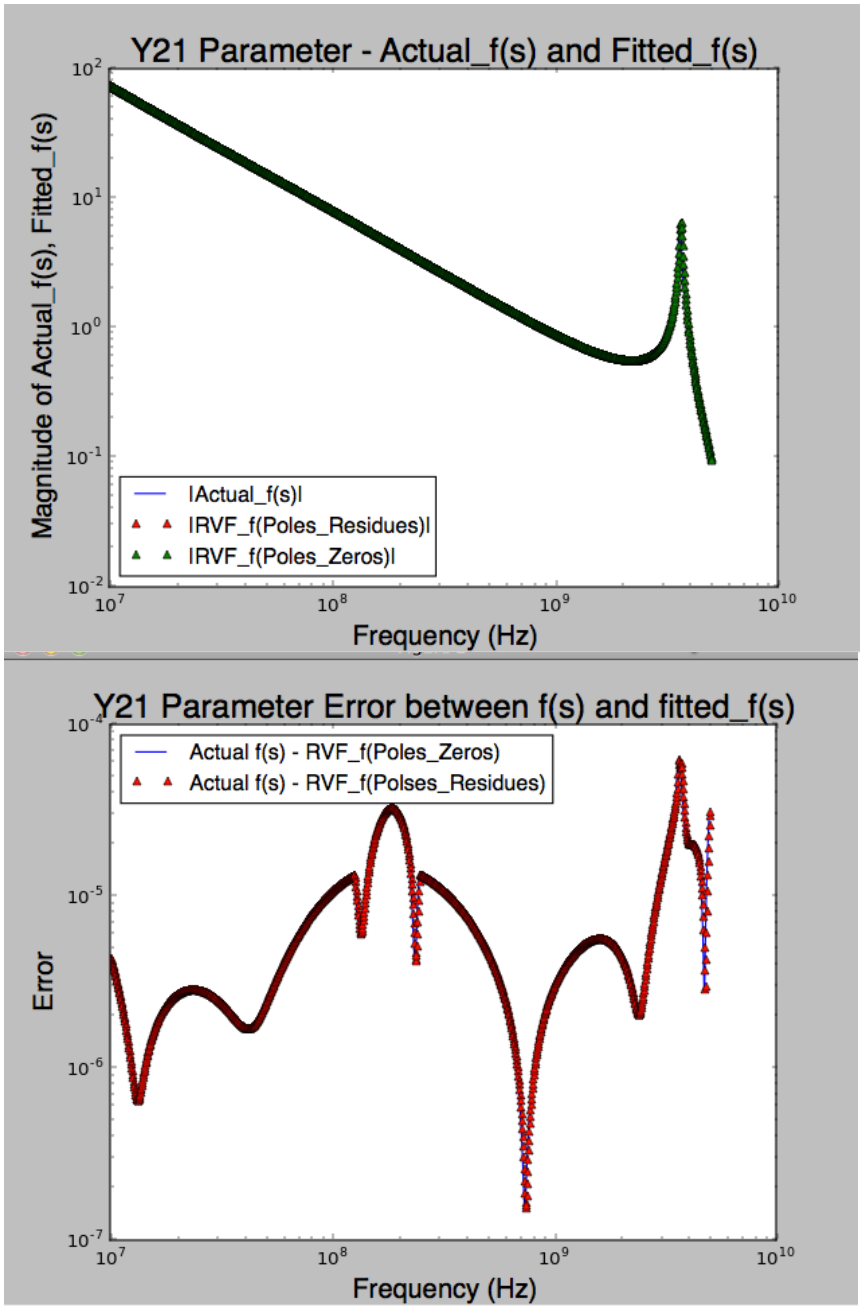


Figure 5.24: Y21 Parameter - Top: Fitted f(s) and Actual f(s); Bottom: Error between Fitted and Actual

Here is the result of Y22-parameter. And the values of d and h are 0.0097563591181

and -3.99638749067e-14, respectively.

Table 5.21: Fitted Poles and Residues of Y22 Parameter from RVF

Pole Real	Pole Imag	Residue Real	Residue Imag
-2.80925E+09	0	96972304	0
-1.44866E+09	-5.09386E+10	3808147968	-303361248
-1.44866E+09	5.09386E+10	3808147968	303361248
-1.03592E+09	0	1130995	0
-326931309.2	-22884450854	1268715264	-5905514.5
-326931309.2	22884450854	1268715264	5905514.5
-312592839	0	471964416	0
-7676854.767	0	4554157568	0

Table 5.22: Fitted Poles and Zeros of Y22 Parameter from RVF

Pole Real	Pole Imag	Zeros Real	Zero Imag
-2.80925E+09	0	7.50416E+11	-1.34706E-05
-1.44866E+09	-5.09386E+10	-5.08245E+11	-1.77806E-06
-1.44866E+09	5.09386E+10	-570352505.5	-3.76696E+10
-1.03592E+09	0	-570352505.5	3.76696E+10
-326931309.2	-22884450854	-270218451.2	-1.79462E+10
-326931309.2	22884450854	-270218451.2	1.79462E+10
-312592839	0	-2757340774	-2.12006E-08
-7676854.767	0	-1035691407	9.02205E-08
-	-	-283896062	-2.31489E-08

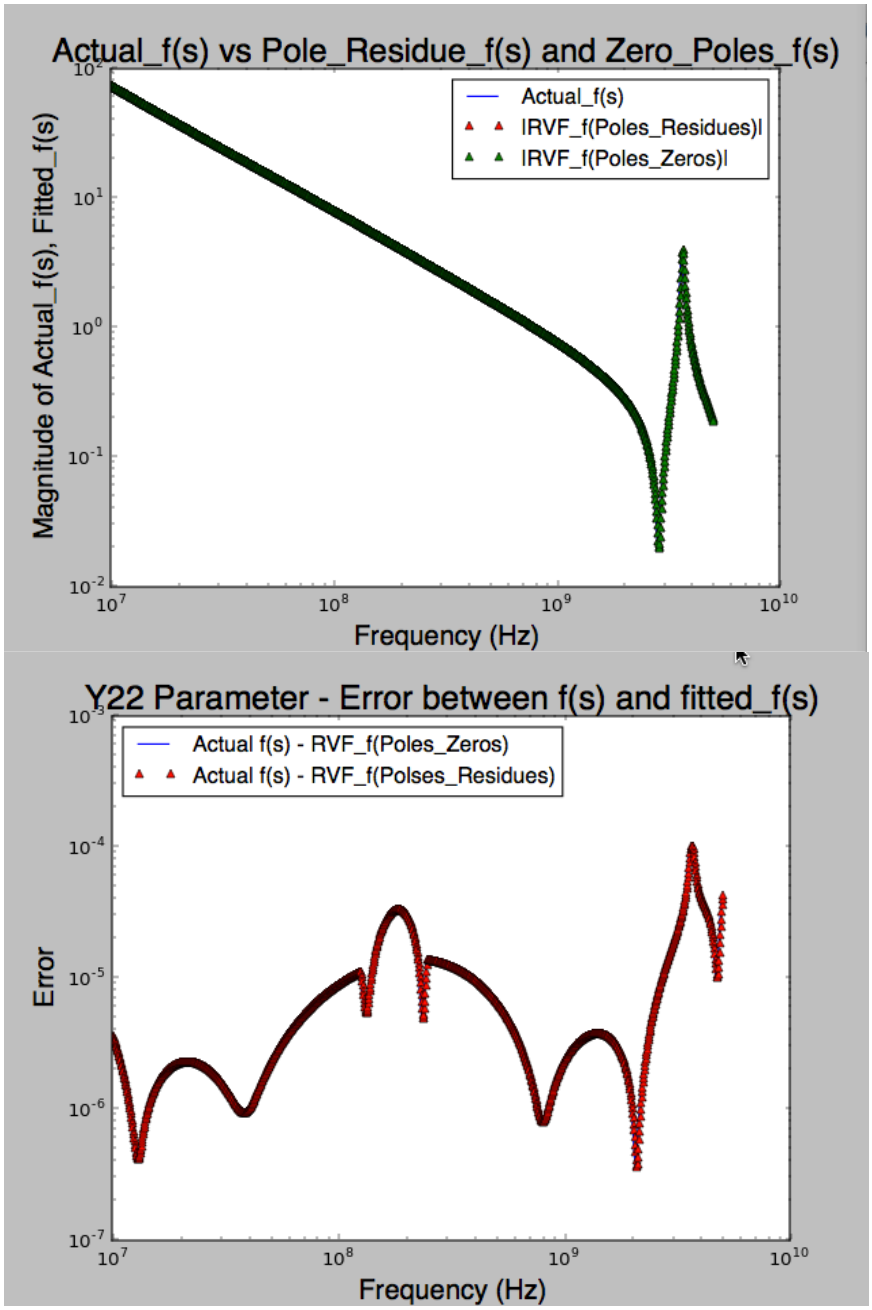


Figure 5.25: Y22 Parameter- Top: Fitted f(s) and Actual f(s); Bottom: Error between Fitted and Actual

5.3.6 Y-Equivalent Pi-Network Circuit

Recall Y-equivalent circuit network:

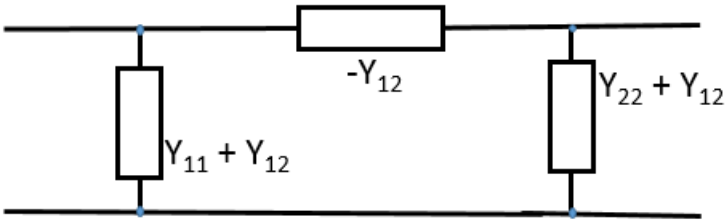


Figure 5.26: Y-equivalent circuit with Y-parameters for a reciprocal two-port network

One of the approaches to generate an equivalent circuit from S-parameters is to convert S-parameters to Y-parameters and generate an equivalent circuit in terms of Y-parameters. We have already converted S-parameters to Y-parameters in previous section. Let's recall figure 5.26, which represents an equivalent circuit with pi network circuit configuration. Again, Python RVF codes can be used to generate transfer functions for $Y_{11} + Y_{12}$ and $Y_{22} + Y_{12}$. Once the poles and residues are determined from the $Y_{11} + Y_{12}$ and $Y_{22} + Y_{12}$, another set of Python codes program is used to determine the circuit elements (i.e. R, L, and C) of each component ($Y_{11} + Y_{12}$, Y_{12}, $Y_{22} + Y_{12}$).

From the Python codes, the RVF algorithm generates fitted poles, residues, and zeros as shown in table 5.23 and the magnitude of fitted f(s) and actual f(s) and error can be seen in figure 5.27. The values of d and h are 0.000896109079092, and 2.65256325898e-14,

respectively.

Table 5.23: Fitted Poles and Residues of Y11+Y12 Parameter from RVF

Pole Real	Pole Imag	Residue Real	Residue Imag
-2847315292	0	-15116824	0
-1369399005	-50427689311	1513837952	-88409112
-1369399005	50427689311	1513837952	88409112
-1078295408	0	2620513.75	0
-326972025.89	-22884502843	5449466880	-32557866
-326972025.89	22884502843	5449466880	32557866
-71455124.96	0	-10195.80	0
-790696.34	0	-2717.10	0

Table 5.24: Fitted Poles and Zeros of Y11+Y12 Parameter from RVF

Pole Real	Pole Imag	Zeros Real	Zero Imag
-2847315292	0	-17511681669	7.24743E+11
-1369399005	-50427689311	-17511681669	-7.24743E+11
-1369399005	50427689311	-1007048174	-45855678069
-1078295408	0	-1007048174	45855678069
-326972025.89	-22884502843	-3078864114	-3.19787E-07
-326972025.89	22884502843	-980068627.3	2.46886E-07
-71455124.96	0	-78063521.48	2.41686E-07
-790696.34	0	11351876.19	1.69054E-08
-	-	-10254490.07	-1.79889E-08

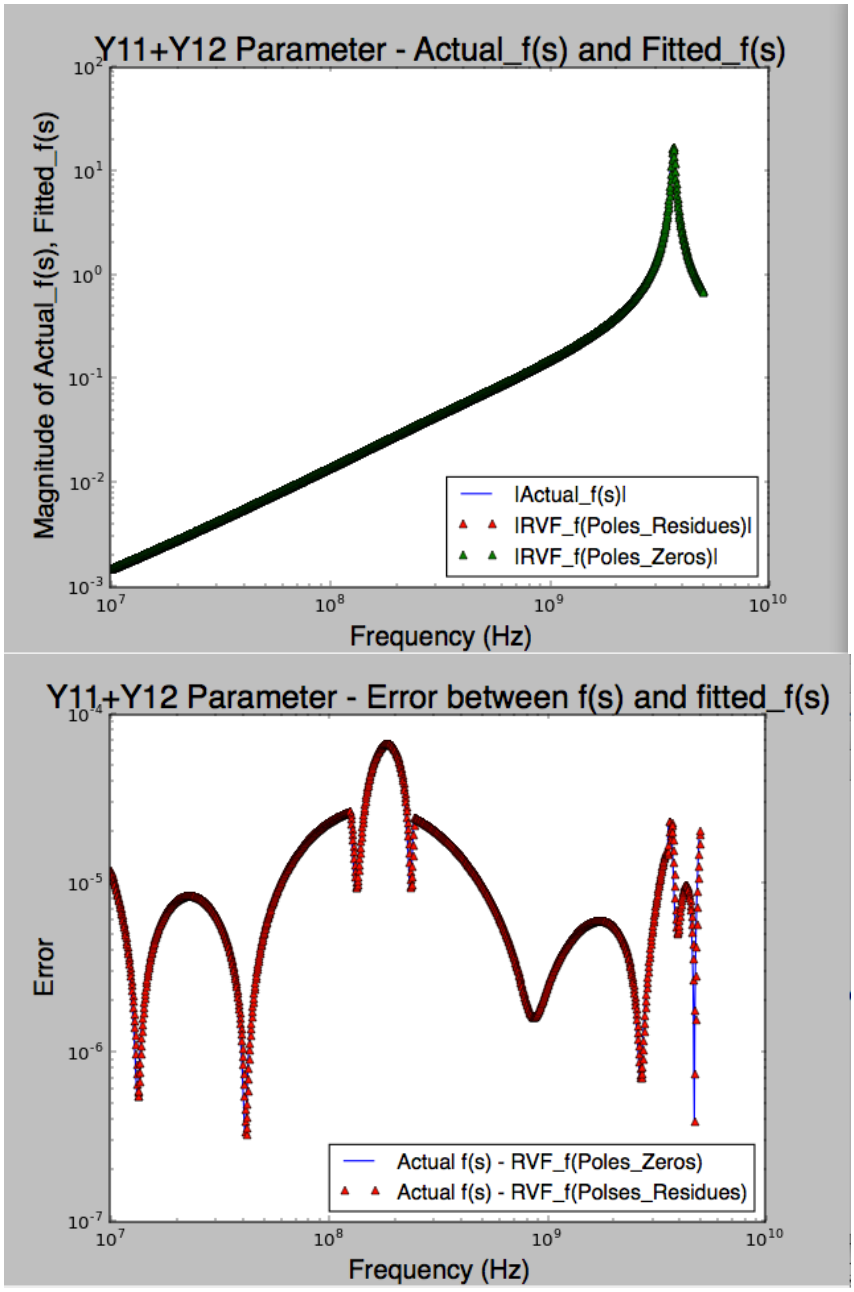


Figure 5.27: Y11+Y12 Parameter - Top: Fitted f(s) and Actual f(s); Bottom: Error between Fitted and Actual

Now let's find the poles and residues of $-Y_{12}$ parameter using RVF Python codes, which are showed below. The values of d and h are 0.00646493582417, and -3.65411910549e-

14, respectively.

Table 5.25: Fitted Poles and Residues, of -Y12 Parameter from RVF

Pole Real	Pole Imag	Residue Real	Residue Imag
-2819321842	0	106823688	0
-1118976014	0	-1330723.25	0
-698174696.9	-49812907680	-563616320	-87218288
-698174696.9	49812907680	-563616320	87218288
-326983812.8	-22884518839	-2070813568	10916837
-326983812.8	22884518839	-2070813568	-10916837
-312573133.7	0	472273280	0
-7676873.839	0	4554151936	0

Table 5.26: Fitted Poles and Zeros of -Y12 Parameter from RVF

Pole Real	Pole Imag	Zeros Real	Zero Imag
-2819321842	0	1.74336E+11	1.52588E-05
-1118976014	0	37304454248	94536643107
-698174696.9	-49812907680	37304454248	-94536643107
-698174696.9	49812907680	-72692007426	3.13721E-06
-326983812.8	-22884518839	-737797632.7	37345323544
-326983812.8	22884518839	-737797632.7	-37345323544
-312573133.7	0	-2760703379	1.69425E-08
-7676873.839	0	-1119264726	-3.15608E-08
-	-	-283856339.3	-2.84612E-08

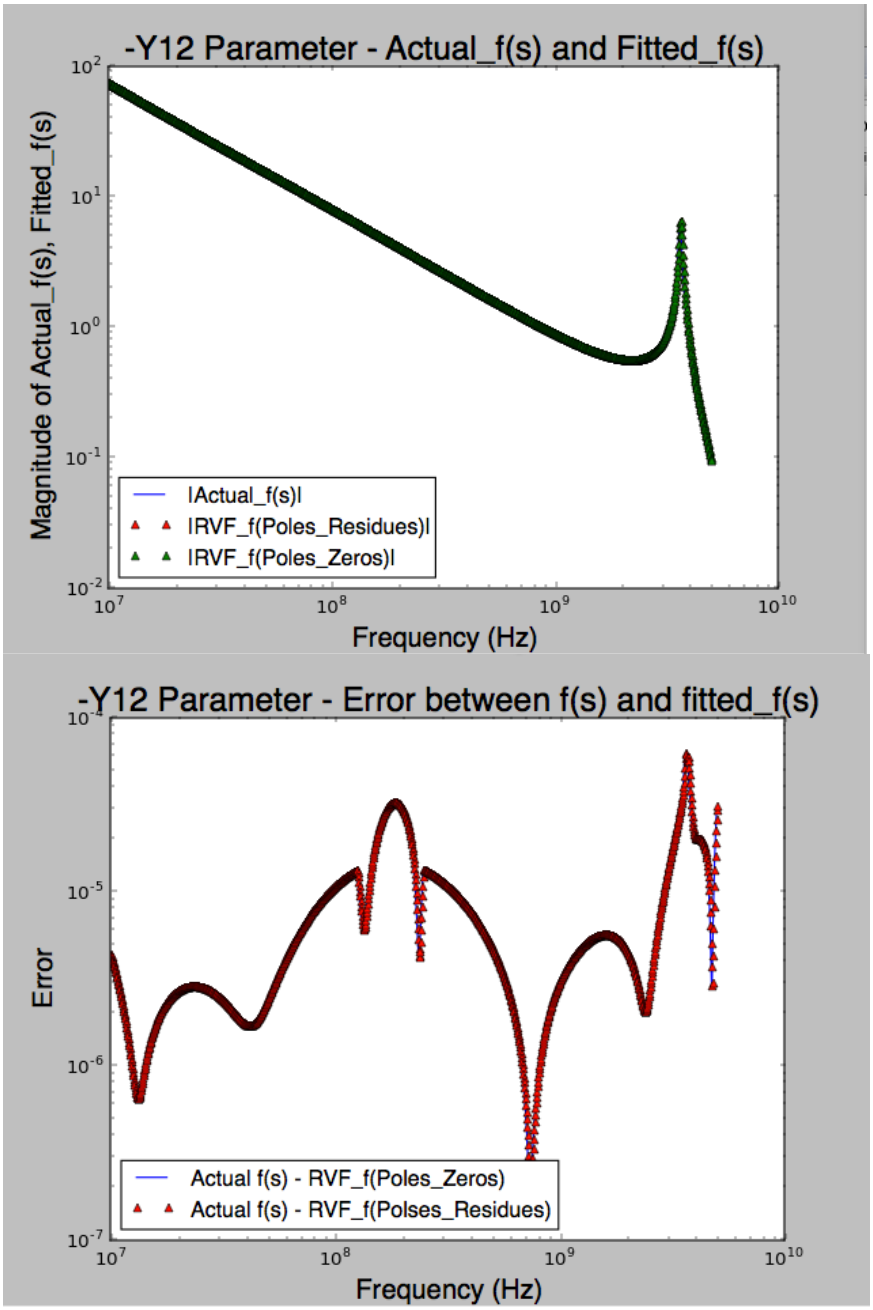


Figure 5.28: -Y12 Parameter - Top: Fitted f(s) and Actual f(s); Bottom: Error between Fitted and Actual

Finally, the pi-network circuit component $Y_{22} + Y_{12}$ is generated. Table of the data

and its figure are showed following.

Table 5.27: Fitted Poles and Residues of (Y22+Y12) Parameter from RVF

Pole Real	Pole Imag	Residue Real	Residue Imag
-2725880182	0	-10641444	0
-1334360533	-50754631329	4363649536	-201806288
-1334360533	50754631329	4363649536	201806288
-1250467832	0	3389945.75	0
-326963589.2	-22884493357	3339526144	-16818378
-326963589.2	22884493357	3339526144	16818378
-260985683.8	0	-256793.9844	0
-5050197.20	0	7016.69	0

Table 5.28: Fitted Poles and Zeros of (Y22+Y12) Parameter from RVF

Pole Real	Pole Imag	Zeros Real	Zero Imag
-2725880182	0	-6.95203E+11	2.64154E+12
-1334360533	-50754631329	4363649536	-201806288
-1334360533	50754631329	-612711131.4	37591660961
-1250467832	0	-612711131.4	-37591660961
-326963589.2	-22884493357	-2952492846	-2.43714E-06
-326963589.2	22884493357	-1094557689	3.62225E-06
-260985683.8	0	-326478655	-2.07617E-06
-5050197.20	0	-1628477.185	18809537.21
-	-	-1628477.185	-18809537.21

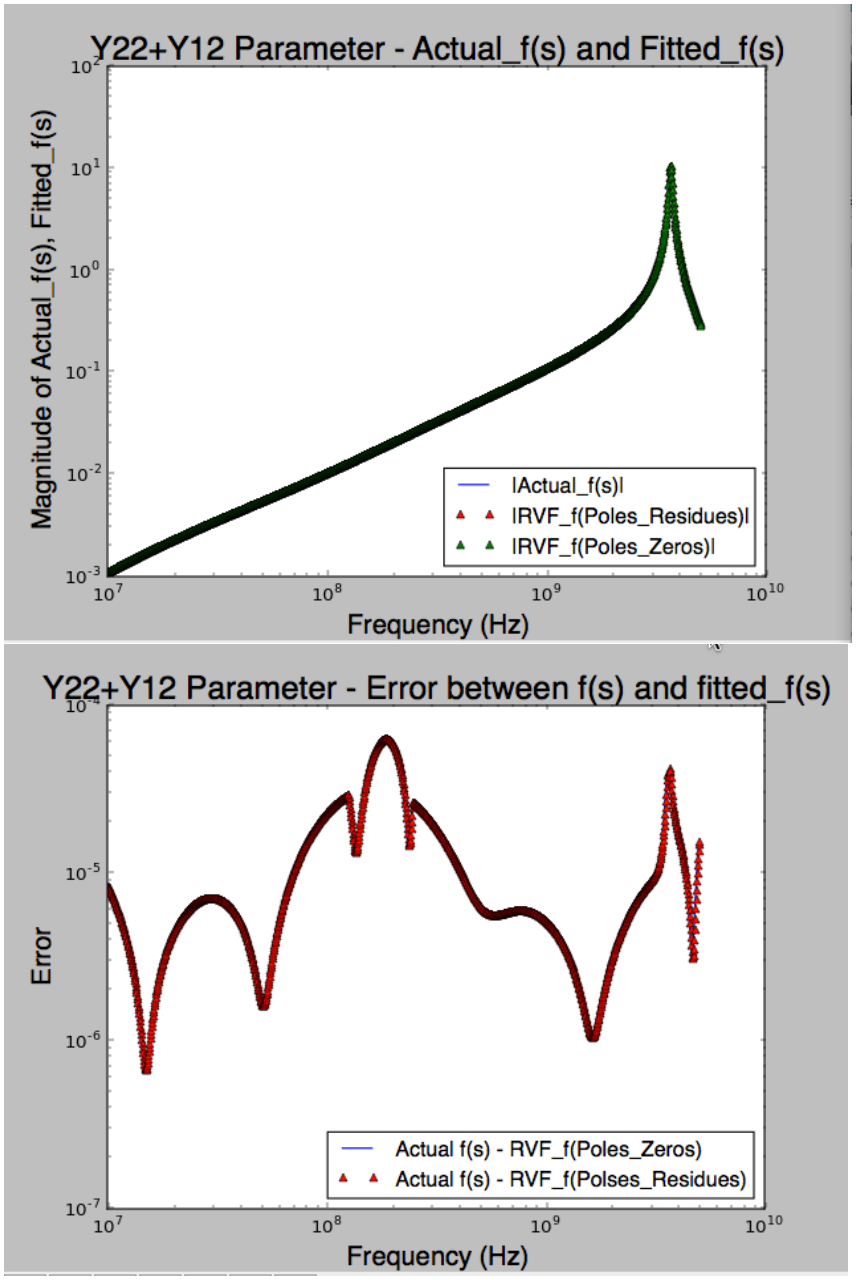


Figure 5.29: Y22+Y12 - Left: Fitted f(s) and Actual f(s); Right: Error between Fitted and Actual

5.3.7 Network Synthesis-RLRG Derivation

Paper [3] presents how to synthesize the RLRC network circuit based on poles and residues of the transfer functions. Since the poles and residues of $Y_{11} + Y_{12}, -Y_{12}, Y_{22} + Y_{12}$ have been determined in a previous section, the RLRC netlist will be determined using equation in paper [3]. Also, the Python algorithm codes have been generated and provided by the author of [3] to compute RLRC elements from numerical TF. Also, [32] - [39] have explained a lot of basic RLC circuits and fundamentals of two port networks, and synthesis of electrical networks.

For $Y_{11} + Y_{12}$, the RLRC results show in table below.

Table 5.29: RLRC Netlist for Y11+Y12

Branch	Ra	L	Rb	C
1	-188.35	-6.62E-08	0	0
2	1.425	3.30E-10	-534.88	-1.1903E-12
3	411.48	3.82E-07	0	0
4	0.04255	9.18E-11	252.58	2.08E-11
5	-7008.29	-9.81E-05	0	0
6	-291.01	-0.000368	0	0

Here is the RLRC for -Y_{12}, which is showed in table below

Table 5.30: RLRC Netlist for -Y12

Branch	Ra	L	Rb	C
1	26.39	9.36E-09	0	0
2	-840.88	-7.51E-07	0	0
3	6.22	-8.87E-10	-268.12	-4.44E-13
4	-0.11	-2.41E-10	-612.83	-7.91E-12
5	0.662	2.12E-09	0	0
6	0.001686	2.2E-10	0	0

Here is the RLRC for $Y_{22} + Y_{12}$, which is showed in table below.

Table 5.31: RLRC Netlist for Y22+Y12

Branch	Ra	L	Rb	C
1	-256.16	-9.40E-08	0	0
2	0.42	1.15E-10	-292.03	1.38E-12
3	368.875	2.95E-07	0	0
4	0.0662	1.50E-10	370.36	1.28E-11
5	-1016.32	-3.89E-06	0	0
6	719.74	0.000143	0	0

5.3.8 Stability and Passivity Discussion

First, look at the stability of the equivalent pi circuit components Y11+Y12, -Y12, and Y22+Y12. As we can see from the fitted poles and the fitted residues, all real parts of the poles are negative (see tables: 5.25, 5.23, 5.27), which indicates that the system is stable.

However, what about the passivity of each component of pi-circuit? To understand if each component or the whole network is passive or not, we need to find the eigenvalues of the Hamiltonian matrix (M). Eigenvalues of Hamiltonian matrix (M) below show that none of the eigenvalues is the $j\omega_0$, so according to paper [2], these components are passive. We are confident to say that whole pi equivalent circuit is passive.

To conclude about stability and passivity in this study, the system is passive based on result from Hamiltonian matrix method. The real parts of poles carry negative values; therefore, the system is stable.

Table 5.32: Eigenvalues of Hamiltonian for Y11+Y12

Eigenvalues
7.76355782e+12 -7.76368174e+12j
7.76355782e+12 +7.76368174e+12j
-9.87452101e+08 +4.58721159e+10j
-9.87452101e+08 -4.58721159e+10j
1.00864946e+09 +4.58700574e+10j
1.00864946e+09 -4.58700574e+10j
-3.07329561e+09 +1.25050818e-05j
3.07909388e+09 +3.25418843e-05j
-9.86860017e+08 -1.00798197e-06j
9.80126408e+08 +1.07002448e-05j
6.77990842e+07 +9.88511854e+06j
-7.41750424e+07 -1.44523847e-06j
6.77990842e+07 -9.88511854e+06j
2.73891623e+07 -6.47077136e-05j
-3.12010769e+06 -1.66621961e+06j
-3.12010769e+06 +1.66621961e+06j

Below are the eigenvalues of M matrix

Table 5.33: Eigenvalues of Hamiltonian for -Y12

Eigenvalues
-2.26182090e+10 +8.87221298e+10j
-5.54646116e+10 +5.74262205e+10j
-2.26182090e+10 -8.87221298e+10j
-5.54646116e+10 -5.74262205e+10j
6.87170470e+10 +1.88085180e+10j
6.87170470e+10 -1.88085180e+10j
-1.82832431e+09 +3.73360691e+10j
6.03207398e+08 +3.74838486e+10j
-1.82832431e+09 -3.73360691e+10j
6.03207398e+08 -3.74838486e+10j
-2.76111140e+09 +2.43431824e-07j
2.76070729e+09 -9.40589053e-07j
-1.11926391e+09 +8.59622259e-06j
-2.83876792e+08 -1.45739142e-06j
1.11926472e+09 -5.34145912e-07j
2.83856360e+08 -6.87523973e-07j

Below are the eigenvalues of M matrix of Y22+Y12

Table 5.34: Eigenvalues of Hamiltonian for Y22+Y12

Eigenvalues
1.19024324e+12 -1.19165813e+12j
1.19024324e+12 +1.19165813e+12j
-3.68665039e+08 -3.76059855e+10j
6.75852729e+08 -3.75881930e+10j
-3.68665039e+08 +3.76059855e+10j
6.75852729e+08 +3.75881930e+10j
-2.91405135e+09 -3.21731932e-07j
2.96442701e+09 -4.98772979e-07j
-1.15097574e+09 -7.37879016e-06j
1.10273072e+09 +9.27526180e-07j
4.21039452e+08 -3.85067182e+08j
4.21039452e+08 +3.85067182e+08j
-2.75652141e+08 +1.35063450e-06j
2.32387266e+08 +3.69009059e-07j
5.89611161e+06 -1.12235337e-10j
-4.68144136e+06 -3.64021417e-07j

Based on the results of RLRC synthesis for Y11+Y12, -Y12, and Y11+Y12, the equivalent synthesized RLRC circuit can be generated, and the synthsized netlist is shown in appendix B and C, respective.

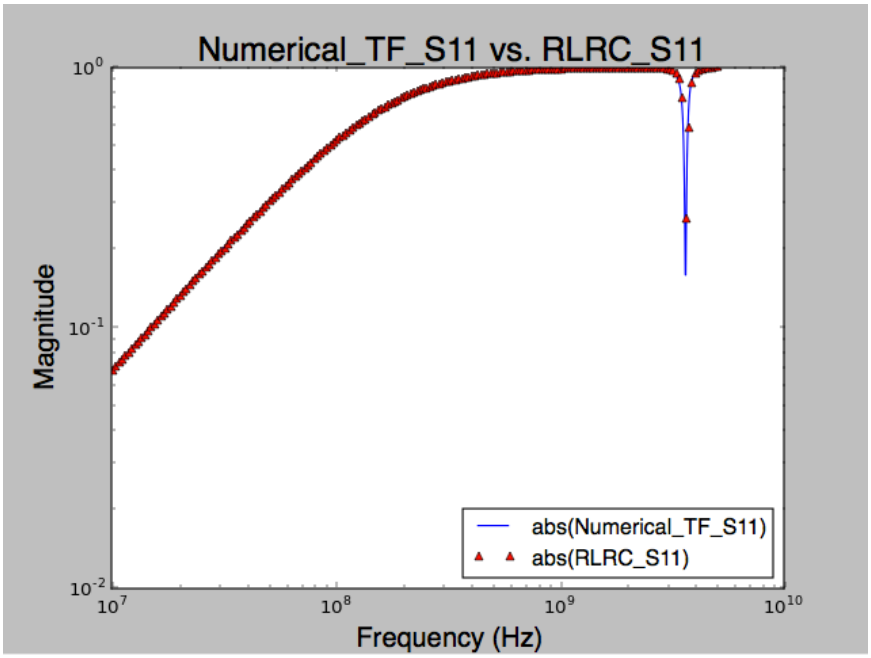


Figure 5.30: S11 Parameter comparison between RLRC circuit and given Numerical TF.

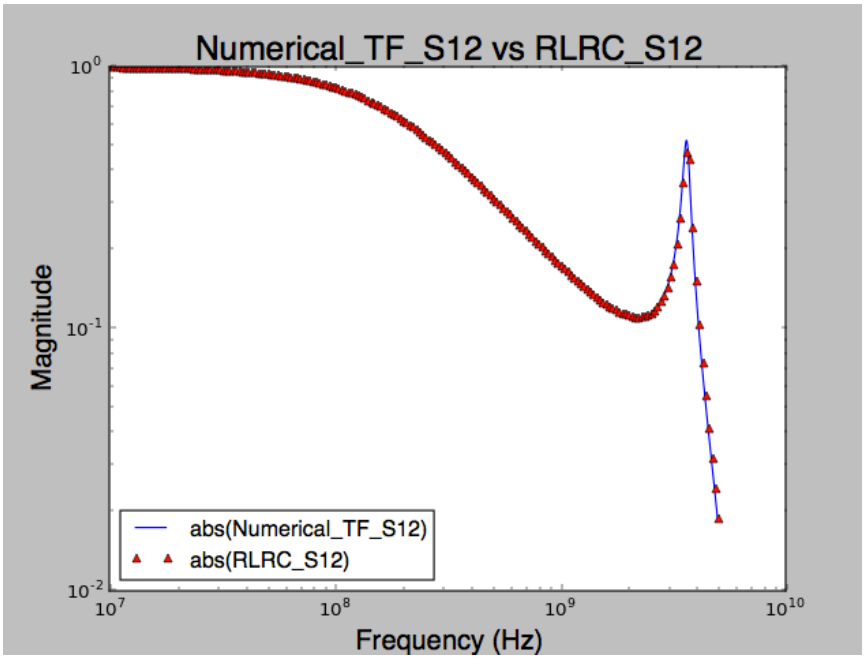


Figure 5.31: S12 Parameter comparison between RLRC circuit and given Numerical TF.

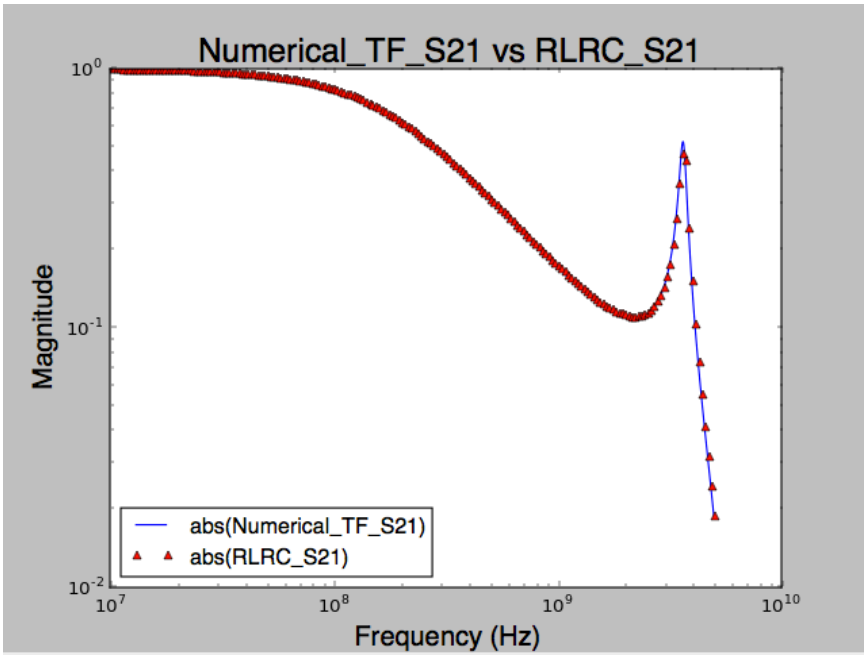


Figure 5.32: S21 Parameter comparison between RLRC circuit and given Numerical TF.

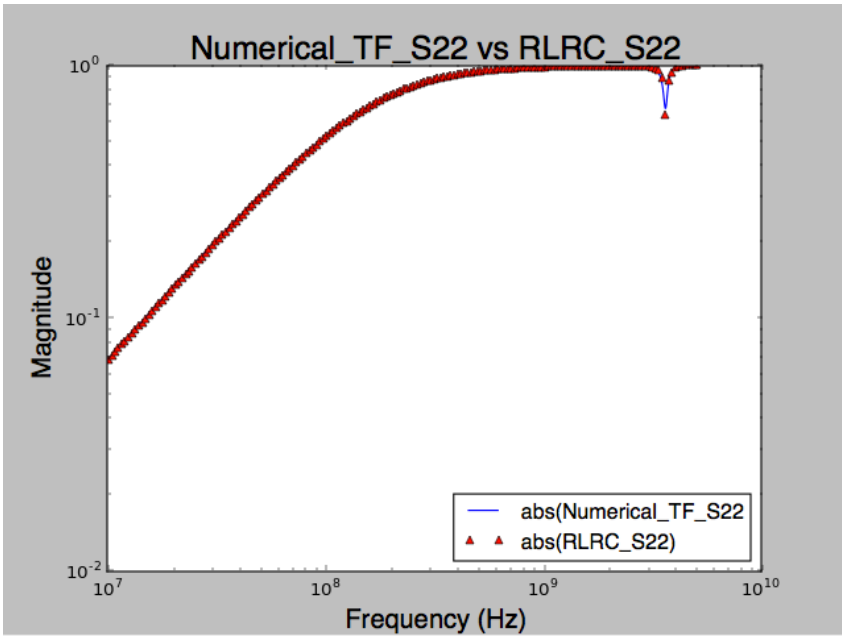


Figure 5.33: S22 Parameter comparison between RLRC circuit and given Numerical TF.

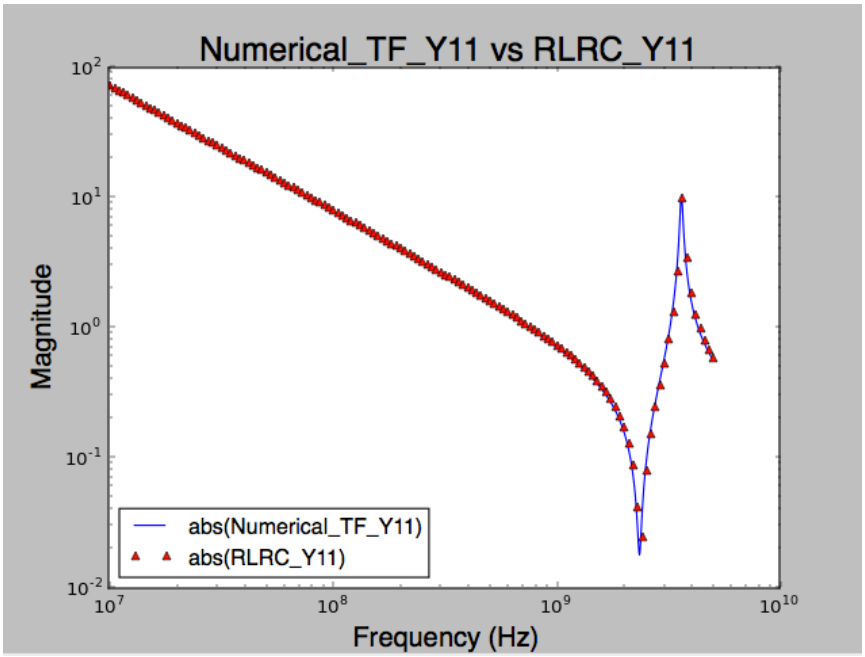


Figure 5.34: Y11 Parameter comparison between RLRC circuit and given Numerical TF.

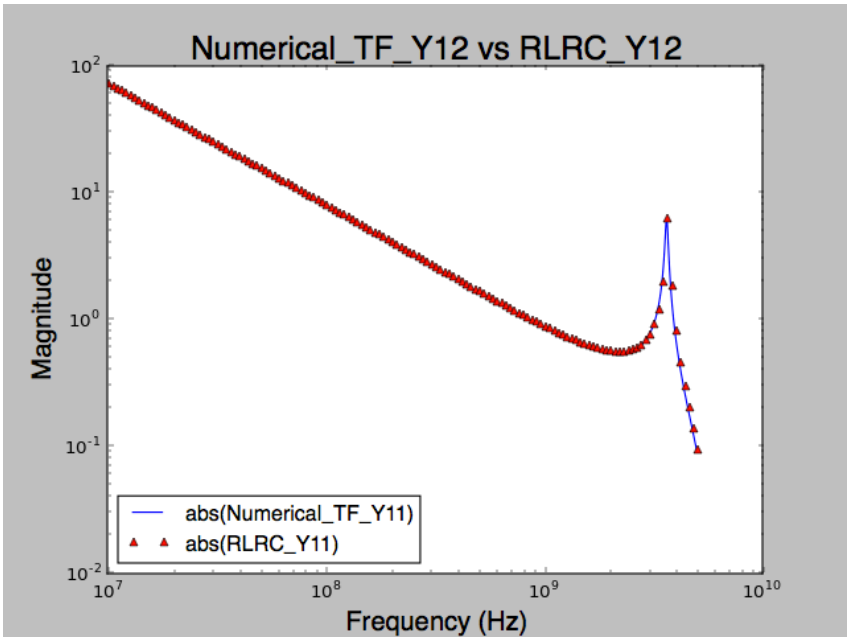


Figure 5.35: Y12 Parameter comparison between RLRC circuit and given Numerical TF.

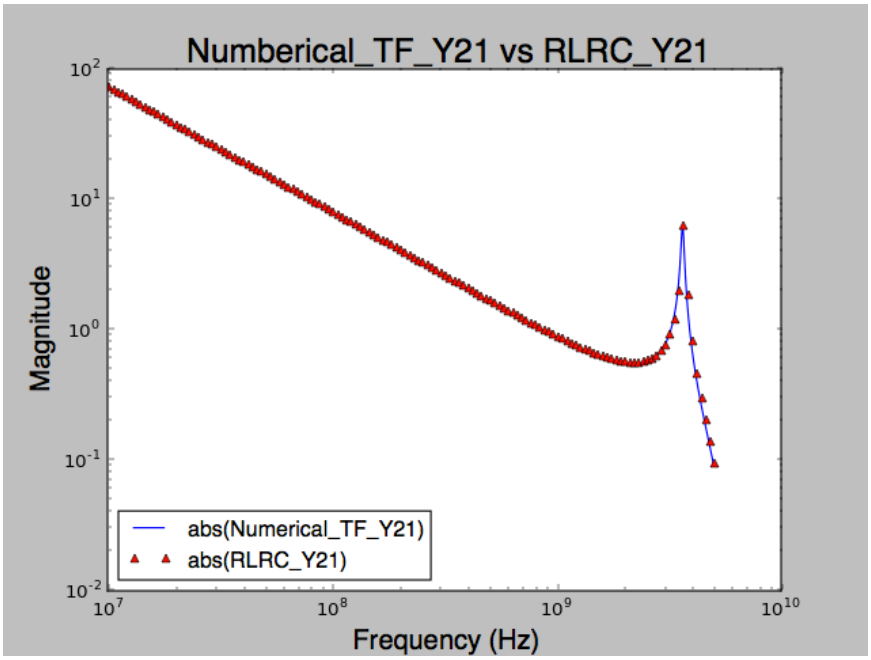


Figure 5.36: Y21 Parameter comparison between RLRC circuit and given Numerical TF.

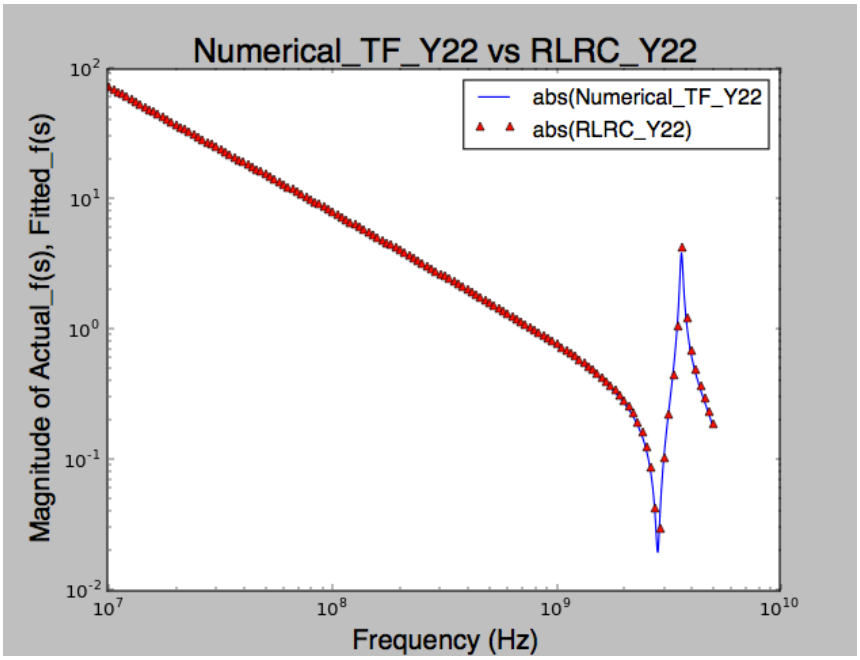


Figure 5.37: Y22 Parameter comparison between RLRC circuit and given Numerical TF.

The flow diagram below summarizes the procedure of how to apply VF to synthesize admittance RLRC pi-equivalent circuit based on S-parameters:

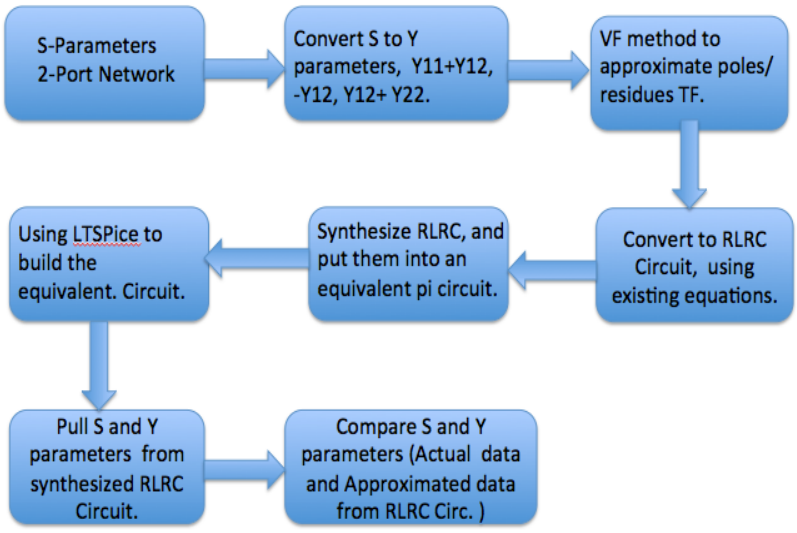


Figure 5.38: Flow Diagram of Conversion from S-parameters to Synthesized RLRC Equivalent Circuit [3], [39].

To ensure the RLRC circuit has the output as desired, the magnitude of the S-parameters and the Y-parameters of the RLRC circuit are compared with the original S-parameters and Y-parameters of numerical transfer function (TF in form of poles and residues). The results are well matched between the two models (TF and Synthesized RLRC circuit). The results are provided in figure from 5.30 to figure 5.38, Therefore, the VF approximation works properly as desired and very accurate.

5.4 Numerical Results and Discussion

In numerical example one, both VF and RVF methods are able to approximate buck converter output impedance, and the error is very small error ($error < 10^{-5}$) between actual frequency response and fitted frequency response. RVF method has a smaller error compared to VF method. This indicates that VF written in Python is working properly. This example contains 100 frequency responses, 2 poles and 5 iterations. Total time of executing Python script is 0.66 seconds. The fitted impedance (i.e. poles and residues TF) of VF is stable because the real part of each pole is negative, which lies on

left-hand side of the Laplace-plane [51, 18, 19]. The stability analysis from approach in [3] also shows that each branch is stable, therefore the stability of the model is confirmed. The passivity analysis using the approach in [3] shows one of the branches nonpassive; however, Hamiltonian Matrix analyzes the results of VF, and it shows the system is passive.

In numerical example two, because the actual responses, f(s), are small, the VF is not able to converge, and it contains a very large error between the fitted f(s) and actual f(s); whereas, the RVF produces a very small error (less than 10^{-5}). The total time of executing this python program of VF with 1000 tabulated frequency responses and 5 iterations and 6 poles is 3.8 seconds. The results show that the model is stable because the real part of poles is negative. And the approach in [3] is also used analyze the stability of each RLC branch, and the results show that the model is stable. This is in agreement with the stability theory. There are a couple branches of the equivalent circuit showing nonpassive according to the approach in paper [3]. However, the Hamiltonian Matrix is used to verify the whole system, and result shows the system is passive.

For numerical example three, RVF successfully approximates each component model for Y parameters (Y11+Y12, -Y12, Y22+Y12), and the error is small less than 10^{-5}. The stability analysis shows that the system is stable. The Hamiltonian Matrix verifies the model and the results show that the model is passive. The total time of executing this program is about 1.02 seconds for each component (this is Synthesized S22 parameters, has 177 frequency samples, 5 iterations, and 8 poles).

Each of the numerical examples has demonstrated that the VF algorithm written in Python working properly and yield accurate approximation. The computational time is proportional to the number of iteration in the program, amount of fitting data (number of samples), and number of poles. The computational time for either RVF method or the VF method is very similar. Based on the data that are generated from numerical examples, there are a few key points to highlight about VF:

- **Advantages** - VF is been widely used in different electrical systems. VF algorithm is not too difficult to understand for a new user. Computational time is fast based on data generated from Matlab and Python. VF Software is available for everyone.

- **Strength** - VF has capability to force stability for the approximated model. That means an unstable pole can be stabilized through a non-linear pole flipping in [1], and the flipping does not affect the algorithm convergence.

- **Weakness** - in these VF methods (original VF method and relaxed VF method), it may generate nonpassive macromodels, which could be due to numerical errors. This has been illustrated in numerical examples. The VF might need integrated with the passivity enforcement to ensure the macromodel is passive. The passive enforcement has been studied in [47], [2].

Chapter 6

Conclusion

In this thesis, we have performed and discussed three main points about vector fitting. 1) We perform an in-depth study of the vector fitting method, and its advantages, weaknesses and strengths; this includes VF method and RVF method. There are some advantages of VF found, such as, being widely used in different electrical systems, being easy to follow and understand, and ease of implementation in Matlab and Python. One of the useful features of the VF method is the stability enforcement feature. One of the weaknesses of the VF method is that it might generate slightly nonpassive macromodel. 2) The VF algorithm has been successfully implemented in Python scripting language and verified through numerical examples. Numerical results show that the VF method is very accurate. The error between fitted responses and actual responses is well within acceptable range ($< 10^{-5}$). A smaller error is seen from the RVF method compared to the normal VF method. Computational time of VF is fast, and it is proportional to the amount of fitting data, number of iterations, and number of poles. 3) The stability and the passivity analyses have been done on the numerical results, obtained through VF. The data confirms that enabling the stability enforcement feature of VF indeed produces a stable approximant, verified by poles being located in the left-hand side of the Laplace plane. We use semi-analytic and cellular methods to analyze passivity for each RLC circuit branch, and we also use the Hamiltonian Matrix method to analyze the passivity of the network. The results from passivity analysis show that even though there might be no passivity at one or more RLC circuit branches in the network, the network could still be passive. Finally, the application of VF method in RLC circuit synthesis for a two-port network is presented in this thesis; from the result analysis, the fitted frequency responses of S-parameters is well matched against the given tabulated frequency responses.

References

[1] B. Gustavsen and A. Semlyen, Rational Approximation of Frequency Domain Responses by Vector Fitting, IEEE Trans. Power Delivery, Vol 14, no. 3, pp. 1052-1061, 1999.

[2] D. Saraswat, R. Achar, and M. Nakhla, Global Passivity Enforcement Algorithm for Macromodels of Interconnect Subnetworks Characterized by Tabulated Data, IEEE Trans. Very Large Scale Integration (VLSI) Systems, **Vol 13, no. 7,** pp. 819-831, 2005.

[3] A. Zadehgol, A semi-analytic and cellular approach to rational system characterization through equivalent circuit, IJNM: Elecronic Network, Devices and Fields, 2015.

[4] B. Gustavsen, Relaxed Vector Fitting Algorithm for Rational Approximation of Frequency Domain Responses, IEEE Workshop, Signal Propagation on Interconnects, pp. 97-100, 2006.

[5] B. Gustavsen and A. Semlyen, Enforcing Passivity for Admittance Matrices Approximatedby Rational Function, IEEE Trans. on Power Delivery, **Vol 16, no. 1,** pp. 97-103, 2001.

[6] B. Gustavsen, Improving The Pole Relocating Properties of Vector Fitting, IEEE Trans. Power Delivery, Vol. 21, no.3, pp. 1587-1592, 2006.

[7] . Gustavsen, T. Dhaene, and D. Deschrijver, Advancements in Iteratie Methods for Rational Approximation in Frequency Domain, IEEE Trans. Power Delivery, Vol. 22, no. 3, pp. 1633-1642, 2007.

[8] B. Gustavsen and A. Semlyen, Simulation of Transmission line Transients Using Vector Fitting and Modal Demoposition, IEEE Tran. on Power Delivery, **Vol 13, no. 2** pp. 605-613, 1998.

[9] A. Charest, M. Nakhla, and R. Achar, Passivity Verification and Enforcement of Delayed Rational Approximations from Scattering Parameter Based Tabulated Data, IEEE EPEPS. 18th Conference, pp. 65-68, 2009.

[10] C. Lei, Y. Wang, Q. Chen, and N. Wong, A Decade of Vector Fitting Development: Application on Signal/Power Ietgrity, AIP Confer. Proceedings (2010).

[11] D. Pozar, Microwave Engineering (3rd edition). Hoboken, NJ, USA: John Wiley and Sons, Inc., 2005.

[12] Python Website: https://github.com/ scikit-rf/scikit-rf/blob/master/skrf/network2.py

[13] Python Website: https://docs.python.org/3/library/csv.html

[14] Python Website: https://github.com/PhilReinhold/vectfit_python

[15] I. Baciu, I Ciocan, and S. Lungu, Modeling Transfer Function for Buck Power Converter, IEEE Conference, Electronic Technology 30th, pp. 541-544, 2007.

[16] E. Fowler, and R. Yarlagadda, A State-Space Approach to RLCT Two-Port Transfer-Function Synthesis, IEEE Tran. on Circuit Theory, Vol 13, no. 2, pp. 15-19, 1972.

[17] O. Mandhana, Efficient Symbolic Circuit Anlysis Based Transfer Functions and Input Impedance Computations for Core-Power Delevery Network with VRM, 2013.

[18] Z. Gajic, Linear Dynamic Systems And Signals, Upper Saddle River, NJ, USA: Pearson Education, Inc. 2003.

[19] W. Brogan, Modern Control Theory (3rd edition), Upper Saddle River, NJ, USA: Prentice Hall, 1991.

[20] D. Lay, Linear Algebra and Its Application (2nd edition). Englewood Cliffs, NJ, USA: Addison-Wesley Longman, Inc. 2000.

[21] B. Gustanvsen, CalTech Website: https://labcit.ligo.caltech.edu/ ~rana/mat/vectfit/ vectfit3_userguide.pdf, 2008.

[22] B, Gustavsen, Vector Fitting Website: http://www.energy.sintef.no/produkt/VECTFIT/index.asp

[23] P. Reinhold, Python Website: https://github.com/PhilReinhold/ \\ vectfit_python/ blob/master/vectfit.py

[24] A. Soysal and A. Semyen, Practical Transfer Function Estimation and Its Application to Wide Frequency Range Representation of Transformer, IEEE Tran. on Power Delivery, Vol. 8, no. 3, pp. 1627-1637, 1993.

[25] S. Lin, H. Yang, and R. Luo, A Novel γd/n RLCG Transmission Line Model Considering Complex RC(L) Loads, IEEE Tran. on Comput.-Aided Design Integr. Circuits Syst., Vol. 26, no. 5, pp 970-977.

[26] B. Gustavesen and A. Semlyen, Application of Vector Fitting to the State Equation Represention of Transformers for Simulation of Electromagetic Transients, IEEE Tran. on Power Delivery, Vol. 13, no. 3, pp 834-842, 1998.

[27] Wikipedia Website: https://en.wikipedia.org/wiki/Two-port_network

[28] Wikipedia Website: https://en.wikipedia.org/wiki/ \\ Scattering_parameters

[29] Wikipedia Website: https://en.wikipedia.org/wiki/ \\ Admittance_parameters

[30] M. Malti, and H. Sun, Synthesis of Transfer Functions with Poles Restricted to the Negative Real Axis into Two Parallel R-C Ladders and an Ideal Transformer, IEEE

Trans. of the American Insitu of Elecrical Engineers, Vol. 75, no. 2, pp. 165-171, 2012.

[31] S. Hakim, Synthesis of RC Active Filters with Prescribed Pole Sensitivity, IEEE in Electrical Engineers, Proceedings of Insititution of Vol. 12, no. 12, pp. 2235-2242, 2010.

[32] I. Herstein and D. Winter, Matrix Theory and Linear Algebra, New York, NY, USA: Macmillan Publishing Company, 1988.

[33] S. Madhu, Linear Circuit Analysis, Eanglewood Cliffs, NJ, USA: Prentice Hall, 1998.

[34] R. Decarlo and P. Lin, Linear Circuit Analysis (2nd edition), New Yor, NY, USA: Oxford University Press, 2001.

[35] C. Meyer, Matrix Analysis and Applied Linear Algebra, Philadelphia, PA, USA: Siam, 2000.

[36] W. Hayt, and J. Kemmerly, Engineering Circuit Analysis (3rd edition)New York, NY, USA: McGraw-Hill Book Company, 2978.

[37] A. Ahmed and P. Spreadbury, Transmission and propagation of electromagnetic waves, New York, NY, USA: Cambridge University Press, 1978.

[38] C. Tse, Linear Circuit Analysis, New York, NY, USA: Addison Wesley Longman Limited, 1998.

[39] H. Baher, Synthesis of Electrical Networks, New York, NY, USA: John Wiley and Sons, 1984.

[40] D. Elgamel, R. Greeff, and D. Ovard, Passivity Verification and Macromodel Interpolation Using Singular Value Decomposition(SVD), IEEE Conf. on Microelectronics and Electron Devices, pp. 1-4, 2005.

[41] C. Walkey, D. Paul, M. Nakhla, R. Achar, and A. Weisshaar, A Noval Passivity Verification and Enforcement Algorithm for Macromodels of Microwave Devices, IEEE Conf. on Microwave Symposium Digest, pp. 611-614, 2008.

[42] M. Nakhla, R. Achar, and D. Saraswat, Fast Passivity Verification and Enforcement via Reciprocal Systems for Interconnects With Large Order macromodels, IEEE Tran. on VLSI Systems, Vol. 16, no. 1, pp. 48-59, 2007.

[43] S. Talocia, and A. Ubolli, A Comparative Study of Passivity Enforcement Schemes for Linear Lumped Macromodels, IEEE Trans. Advanced Packaging, Vol. 33, no. 1, pp.246-256, 2010.

[44] D. Deschrijver, and T. Dhaene, A Note On Multiplicity of Poles In The Vector Fitting Macromodeling Method, IEEE trans. Microwave Theory and Techniques, Vol. 55, no. 10, pp. 736-741, 2007.

[45] D. Deschrijver, M. Mrozowski, T. Dhaene, and D. Zutter, Macromodeling of Multiport System Using a Fast implementation of the Vector Fitting Method, IEEE trans. Microwave Wireless Component Letters, Vol. 18, no. 6, 283-385, 2008.

[46] D. Deschrijver, and T. Dhaene, Generalised Vector Fitting Algorithm for Macromodelling of Passive Electronic Components, IEEE Electronics Letters, Vol. 41, no. 6, pp 299-300, 2005.

[47] S. Mabrok, A. Kallapur, I. Petersen, and A. Lanzon, Passivity Enforcement via Perturbation of Hamiltonian Matrices, IEEE Trans. Conf. Decision and Control and European Control, pp. 3748-3752, 2011.

[48] D. Saraswat, R. Achar, and M. Nakhla, A Fast Algorithm and Practical Considerations for Passive Macromodeling of Measured/Simulated Data, IEEE Trans. on Advanced Packaging, Vol. 27, no. 1, pp. 57-80, 2004.

[49] C. Sanatanan, On The Synthesis of Space Dependent Transfer Fuctions, IEEE Trans. Automatic Control, 1963.

[50] W. R, and B. EJ, Distribution theory as the basis of generlized passive-network analysis, IEEE trans. Circuit Theory, pp. 164-170, 1965.

[51] P. Triverio, S. Talocia, M. Nakhla, F. Canavero, and R. Achar, Stability, Causlity, and Passivity in Electrical Interconnect Models, IEEE Trans. Advanced Packageing, Vol. 30, No. 4, pp. 795-806, 2007.

[52] B. Gustavsen and A. Semlyen, Fast Passivity Assessment for S-Parameter Rational Models Via a Half-Size Test Matrix, IEEE Trans. Microwave Theory and Techniques, Vol. 56, no. 12, pp. 2701-2709.

Appendix A: 100 frequency samples used for VF and RVF codes verification

Solve how to find poles of linear system transfer function ...

Table 6.1: 18 Poles Responses f(s)

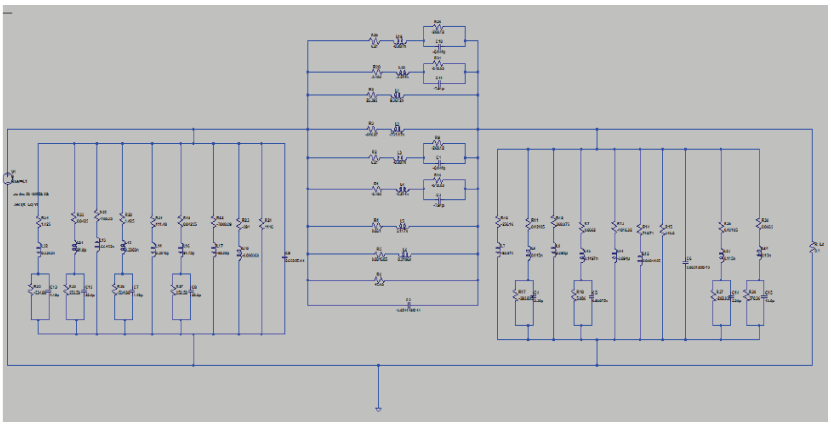
Frequency	f Real	f Imaginary
795.7981856	-16.54213408	70.9116496
948.5271303	-9.34430753	1.77421324
1101.256075	-13.0691022	1.31131556
1253.98502	-14.51599799	1.30875843
1406.713964	-15.47806592	1.38007892
1559.442909	-16.36472356	1.48139626
1712.171854	-17.39306157	1.6097904
1864.900799	-18.82612053	1.78421239
2017.629743	-21.24940308	2.08134727
2170.358688	-26.78252254	2.93435267
2323.087633	-54.79427389	14.0199006
2475.816577	16.94252953	8.74309455
2628.545522	-2.37403994	3.09228969
2781.274467	-7.10946231	2.63323864
2934.003411	-9.33447824	2.6000125
3086.732356	-10.69997763	2.67639163
3239.461301	-11.68171046	2.80128763
3392.190246	-12.46972214	2.9586455
3544.91919	-13.15672127	3.14552261
3697.648135	-13.79516899	3.3647989
3850.37708	-14.41874942	3.62348179
4003.106024	-15.05166922	3.93294955
4155.834969	-15.7129344	4.31038747
4308.563914	-16.41790735	4.78152258
4461.292859	-17.17737145	5.38521256
4614.021803	-17.99249303	6.18075191
4766.750748	-18.84083523	7.25839201
4919.479693	-19.64178029	8.74964187
5072.208637	-20.17879282	10.8150907
5224.937582	-19.96576607	13.5223533

Frequency	f Real	f Imaginary
5377.666527	-18.22557589	16.4207306
5530.395472	-14.67648227	18.0046244
5683.124416	-10.95944955	16.8339056
5835.853361	-9.26429176	13.831542
5988.582306	-9.77493608	10.8999437
6141.31125	-11.63692315	8.82284052
6294.040195	-14.32329386	7.58120464
6446.76914	-17.82397229	7.02096912
6599.498084	-22.62461211	7.12778261
6752.227029	-30.09247529	8.25679985
6904.955974	-44.5747616	12.1473736
7057.684919	-91.32257897	35.4823143
7210.413863	133.4663952	101.589764
7363.142808	43.29869291	15.4199856
7515.871753	23.02220549	8.70283788
7668.600697	14.01927628	6.65687221
7821.329642	8.78309099	5.81832602
7974.058587	5.27466382	5.444665
8126.787532	2.69558826	5.29413996
8279.516476	0.66588779	5.27028616
8432.245421	-1.02027814	5.32979213
8584.974366	-2.48605202	5.45228934
8737.70331	-3.81171618	5.62887099
8890.432255	-5.05417671	5.85734604
9043.1612	-6.25752813	6.14031225
9195.890144	-7.45946902	6.48464799
9348.619089	-8.69582575	6.90195516
9501.348034	-10.00445229	7.40993739
9654.076979	-11.42945567	8.03506286
9806.805923	-13.02679812	8.817382
9959.534868	-14.87289918	9.81940704
10112.26381	-17.0792939	11.1433705
10264.99276	-19.81981613	12.9674774
10417.7217	-23.38527029	15.6304454
10570.45065	-28.30272894	19.8587945
10723.17959	-35.60899953	27.5149684
10875.90854	-47.26321333	44.942373
11028.63748	-55.96972369	105.846808
11181.36643	126.7525404	94.9335986

Frequency	f Real	f Imaginary
11334.09537	48.70422047	1.3178631
11486.82432	14.8379374	15.5796875
11639.55326	30.14631552	34.7877855
11792.2822	39.1601162	12.7079894
11945.01115	29.68803534	2.43498068
12097.74009	22.09976048	-0.450561758
12250.46904	16.7493843	-1.20474037
12403.19798	12.75693749	-1.24202068
12555.92693	9.57145927	-0.989168963
12708.65587	6.86679287	-0.587433261
12861.38482	4.43354579	-0.075792079
13014.11376	2.12038434	0.556028962
13166.84271	-0.19848673	1.35637122
13319.57165	-2.64305302	2.42214171
13472.3006	-5.33790864	3.93338502
13625.02954	-8.40213171	6.23211081
13777.75849	-11.8594193	9.99161398
13930.48743	-15.22269301	16.5197946
14083.21638	-15.94414612	27.7200358
14235.94532	-6.81727567	41.8773721
14388.67426	13.51438774	44.6109977
14541.40321	26.34857474	32.6002477
14694.13215	28.01235285	20.9493745
14846.8611	25.8080161	13.7889644
14999.59004	23.07135807	9.64267685
15152.31899	20.62718431	7.15827006
15305.04793	18.58687808	5.59442973
15457.77688	16.9012785	4.56441844
15610.50582	15.49973731	3.85995397
15763.23477	14.32106448	3.36333753
15915.96371	13.31759474	3.00479948

Appendix B: 2 Port-Network Schematic

The synthesized RLRC circuit from VF and PRtoRLRC algorithm (I don't expect readers to read values on the circuit schematic. This circuit schematic just shows the configuration of Y11+Y12, -Y12, Y22+Y12 in terms of RLRC, which generated by VF and [3]:



Appendix C: 2 port-Network Netlist

Netlist for Synthesized RLRC Circuit

```
R1 N008 N001 -0.108
R2 N006 N001 6.21
R3 N004 N001 -840.87
R4 N009 N001 0.661
R5 N022 N001 0.001685
R6 N003 N001 154.8
R8 N002 N001 26.392
L1 N003 N002 9.3612n
L2 N003 N004 -751.47n
L3 N005 N006 -0.887n
L4 N007 N008 -0.241n
L5 N003 N009 2.117n
L6 N003 N022 0.2196n
R9 N003 N005 -268.12
R10 N003 N007 -612.83
C1 N003 N005 -0.444p
C2 N003 N007 -7.91p
C3 N003 N001 -3.654119E-14
R7 N019 N003 0.0662
R11 N018 N003 0.42185
R12 N016 N003 368.875
R13 N020 N003 -1016.32
R14 N021 N003 719.74
R15 0 N003 348.8
R16 N015 N003 -256.16
L7 0 N015 -93.97n
L8 0 N016 0.295_
L9 N025 N018 0.115n
L10 N026 N019 0.1497n
L11 0 N020 -3.894_
L12 0 N021 0.0001425
R17 0 N025 -292.035
R18 0 N026 370.36
C4 0 N025 3.38p
C5 0 N026 12.8p
C6 0 N003 2.065125E-15
R19 N013 N001 0.04255
R20 N011 N001 1.425
R22 N014 N001 -7008.29
R23 N017 N001 -291
R24 0 N001 1116
R25 N010 N001 -188.35
L13 0 N010 -66.152n
L15 N023 N011 0.3303n
L16 N024 N013 91.75p
L17 0 N014 -98.08_
L18 0 N017 -0.000368
R26 0 N023 -534.88
R27 0 N024 252.58
C7 0 N023 1.19p
C8 0 N024 20.8p
C9 0 N001 2.6525E-14
V1 N001 0 1 Rser=0.1
R_Ld N003 0 0.1
R21 N012 N001 411.48
L14 0 N012 0.3816_
.ac dec 65 10MEG 5G
.net I(R_Ld) V1
.backanno
.end
```

Appendix D: Command to execute and Read CSV File Codes.

Execute vectfit.py under MAC Terminal Prompt: python vectfit.py