POLITECNICO DI TORINO

Master of Science in Mechatronic Engineering

Master Thesis

Fast Data-Driven Algorithms for Parameterized Macromodeling of Multiport Systems

Supervisor
Stefano Grivet-Talocia

Candidate
Tommaso BRADDE
ID Number 230415

Academic Year 2017-2018

To my family and to all of them who inspired my curiosity...

Acknowledgements

The results reported in this Thesis would have never been achieved without the efforts of my colleagues Alessandro Zanco and Marco De Stefano and the enthusiastic supervision of Professor Stefano Grivet-Talocia, whose dedication and patience is out of the ordinary.

Summary

The thesis project focuses on the study and the implementation of a high-performance rational fitting algorithm for the construction of parameterized macromodels. By macromodel we understand a behavioral simplified model of some complex structure, intended to provide a compact, accurate and reliable approximation of the system dynamics, in a form that can be easily handled by circuit simulation software such as SPICE or EMTP-type tools.

During the last few years, the possibility to include a closed-form, suitably approximated, dependency of a system response on an arbitrary number of physical parameters in a compact model has been actively pursued by many researchers. Such a description would enable new possibilities in the fields of design, optimization and worst-case scenario simulations of large electrical interconnects.

An effective approach to obtain a parameterized macromodel is to collect a set of input-output frequency responses, which are processed by a rational fitting algorithm of choice; we extended the formulation of the Generalized Sanathanan-Koerner algorithm (GSK) to the case of parameterized macromodels, implementing the so-called Parametric Sanathanan-Koerner (PSK) iteration, which casts a hard nonlinear least squares problem as a sequence of reweighted linear least squares systems. Since our objective was to guarantee the possibility to model large electrical interconnect structures coming from real design scenarios, our main effort has been to improve the scalability properties of the algorithm with respect to the number of electrical ports. Indeed, the dimensions of the least square problem to be solved at each iteration of standard PSK can grow very large when increasing the number of ports; as a consequence, the computational and memory requirements of the algorithms can become unaffordable.

We first observed how the main reason for such bad scalability is an algebraic coupling between some numerator and denominator unknowns embedded in the model structure, which leads however to a particular sparsity pattern in the regression matrix structure. Using an ad hoc decoupling and compression procedure achieved through a set of QR factorizations, we were able to split the fully-coupled least square problem into two separate, decoupled, and significantly simpler problems. The complexity of resulting algorithm, that we named Fast PSK, scales only linearly with the number of port responses of the system under modeling, opposed to the standard PSK, for which this scaling is cubic.

Further, the structure of the new algorithm is suitable for future improvements, achievable through a straightforward parallelization of the code; also the memory requirements have been improved as well.

We completed the model identification algorithm by allowing some flexibility in the definition of the cost function that is minimized at each iteration, and which represents the model vs data error. Based on a user-defined response- and frequency-dependent weighting coefficient, our implementation can optimize the model accuracy based on absolute or relative error, and possibly even any arbitrary frequency-dependent system norm. An immediate application was found by extracting models for loosely coupled electrical interconnect systems, which are characterized by responses with a very large dynamic range.

In conclusion: the new algorithm that we formulated and demonstrated in this work can be considered as key enabling factors for a reliable construction of surrogate parameterized compact models of those complex structures that are commonly found in production-level electronic designs. These results will be presented at the 2018 flagship conference of the IEEE Electromagnetic Compatibility Society, Long Beach (CA), USA.

Contents

List of Figures

Preface

The present thesis project is self-consistent and it has been developed independently by the Author, under the supervisor guidance. However, the results that have been achieved are of practical interest only if cast in a more general framework, in which other two thesis projects are involved: the shared effort of a team composed of Tommaso Bradde, Marco De Stefano and Alessandro Zanco enabled each member of the group to finalize his work. Being part of a joint effort, each of the three thesis projects shares a common background, which has been summarized in Chapters 1 and 4. These two chapters were written jointly and are common to all three thesis projects. The remaining chapters of each dissertation are the core of each project and are original for each individual team member.

Chapter 1

General Framework and Motivations

This Chapter is co-authored by T.Bradde, M. De Stefano and A. Zanco.

1.1 Data Driven Modeling

This thesis project concerns mathematical modeling of linear dynamic systems, namely, systems that are governed by linear differential equations. By "mathematical modeling", we refer to the procedure by means of which a representation of a physical phenomenon or structure is given in a numerically (i.e quantitatively) exploitable form. This kind of representation grants us the opportunity to describe and predict what would happen in a given scenario in which the described object is involved; we can say that such a procedure is at the same time the foundation and the objective of science and a necessary step of the design process in every engineering field.
Although the first-principle laws of science are theoretically able to properly describe a broad range of dynamic phenomena, usually making use of partial differential equations, it is often inappropriate or impossible to derive from them a model able to satisfy the requirements of a current design process: the (exponentially) increasing complexity of the structures to be modeled would lead to an excessive computational cost with respect to the need of an easily manageable description of the item under design. Further, a model derived from first-principle laws must take care of all the physical quantities involved in the system dynamic, while often, only a subset of them is of practical interest.
The Data Driven Modeling techniques are intended to overcome these issues and to provide simpler yet accurate descriptions, able to catch the case-relevant aspects of the structures under investigation by exploiting, as common ground, a set of data collected to extract information about the system behaviour. Making use of proper algorithms, a suitable reconstruction of the relations underlying the data is achieved.
To gather the data, one can either carry out physical measurements of the desired quantities to be tracked or perform (once) a set of first-principle simulations from which the simplified model can be derived.

The most appropriate algorithm to process the data is always a matter of purposes,

since the structure of the algorithm influences, in some measure, the structure of the final model.
Beside the possible implementations, a broad spectrum classification of those algorithms can be based upon the a priori assumptions about the structure of the system: in the so called white and gray box approaches, a total or partial knowledge of the structure is assumed and the algorithm is expected to give back some quantities that characterize the imposed structure from the physical point of view; on the other hand, black box approaches make no assumptions on this structure and make no claims towards a physical description of the system, focusing only into the construction of models that fit numerically the data of the input-output relationship.
The first class of methods can give a deeper insight into the system behaviour, but they rely on the goodness of the a priori assumptions, that can result to be inaccurate or not possible to be made at all. Conversely, the lack of physical meaning of a black box model is counterbalanced by the opportunity to derive an input-output description without any assumption beyond linearity.
From now on, we will treat the black box methods and we will refer to the obtained model as "Macromodel".

1.1.1 Macromodels: Construction Flow and Advantages

In the following, we will focus on macromodels devoted to the behavioural simulation of complex electrical interconnects, or, more generally, electromagnetic structures. The main objective of the macromodeling procedure is to obtain a macromodel that replaces the high complexity dynamics of the structure with a lower complexity model, which catches only the main features of the relationship between the electrical inputs and outputs of interest.
If we are modeling the system in the frequency domain, our starting point is a set of input-output data:

$$\breve{H}_k = \breve{H}(s_k) \quad for \quad k = 1,2,...,K \tag{1.1}$$

where s_k denotes a complex frequency and $\breve{H}(s_k)$ is the transfer function of the system sampled at s_k. The total number of measurements is K.
In most cases, the measurements are performed at the real frequencies $\mathrm{j}\omega_k$, with $s_k = \mathrm{j}\omega_k$. In this case we have:

$$\mathrm{j}\omega_1 = \mathrm{j}\omega_{min}, \quad \mathrm{j}\omega_K = \mathrm{j}\omega_{max} \tag{1.2}$$

The objective is then to reconstruct the response by means of an interpolation or approximation procedure that returns a model:

$$H(\mathrm{j}\omega) \approx \breve{H}(\mathrm{j}\omega) \quad for \quad \omega \in [\omega_{min}, \omega_{max}] \tag{1.3}$$

Throughout this text, we will denote with the symbol $\breve{H}(\cdot)$ the true system response, while with the symbol $H(\cdot)$ the model response. The obtained model is intended to be exploited in a circuit simulation software such as SPICE or EMTP in a fast and reliable way.
We now present a brief overview of how a macromodel is usually obtained and of the strong points that makes it useful.

- **Macromodeling from field solver data**: a full-wave solver is used to obtain the input-output data; detailed knowledge of the structures and of the characteristics of the actual system is required to perform the primary simulation. The data can be collected both in the time domain or in the frequency domain.
  This method is not properly a black-box one, since the structure of the model must be known to perform the full-wave simulation; anyway, we can say that it is a black box method for what concerns the macromodeling algorithm, that receives only data as inputs, without additional informations about structure. This scenario is common in industrial design environments where commercial field solvers are used.

- **Macromodeling from measurements**: a physical realization of the system under modeling is provided; the data are collected and reconstructed by performing measurements over the electrical ports that we wish to characterize. Also in this case, both frequency and time domain data can be gathered. This approach is truly black-box, in every step of the identification procedure.

Once the data are processed by the chosen algorithm, one can dispose of the obtained macromodel with the following advantages:

1. A closed form expression for the behaviour of the system is obtained from the discrete set of data points collected.

2. The macromodel describes the system behaviour without disclosing any insight about the physical structure: sharing a macromodel doesn't represent a risk for the diffusion of proprietary information.

3. Whatever is the nature of the data set used for the fitting, the resulting macromodel is intended to permit fast time domain simulations.

4. The obtained macromodel can be interfaced with other macromodels for simulation of large interconnects system, allowing the possibility to simulate and optimize various design scenarios.

1.1.2 Macromodel Requirements for Simulations

Some features are required on the macromodel, in order to guarantee its exploitability and reliability. In particular, since we are dealing with the modeling of linear systems, a suitable model structure should be chosen among all the possible ones; indeed, we know that when a system is governed by ordinary differential equations, all the transfer functions that can be derived for its input-output description result to be rational functions of the Laplace variable s.

The choice of a model structure of this type not only catches the underlying governing laws of the system, but results also particularly appropriate to be exploited to perform simulations driven by linear circuit simulation software.
The numerical precision of the model must always be consistent with some physical characteristics of the modeled structure to reproduce its behavior correctly; here, we list the most relevant in an intuitive fashion, leaving a more precise description to later sections.

- **Realness.** Although the rational macromodels make use of complex variables to describe the input-output behaviours, all the simulated quantities must be real numbers when observed in the time domain.

- **Causality.** Any physical system at rest can change its state only as a result of an external stimulus; for an input-output description, this fact implies the necessity of the output to be temporally preceded by its cause, the input.

- **Stability.** The concept of stability can be provided with various definitions; in the following we define stable a model whose poles show negative real part, that is, if $\{p_i\}$ is the set of poles of the model, then:

$$\mathrm{Re}\,\{p_i\} < 0 \quad for \quad i = 1,2,...,n \tag{1.4}$$

  where n is the order of the associated transfer function. The lack of the stability property can imply numerically unbounded simulations that clearly do not reflect the behavior of a real system.

- **Passivity.** A system is passive if it is not able to generate energy on its own; it can realease energy to the outer environment only if that energy was previously provided and stored inside it. The property of passivity can be regarded as the most general, since it automatically implies stability, causality and realness.

1.1.3 Rational Fitting Algorithms

The choice of a particular fitting strategy is the first step in any modeling procedure: we must first fix the structure of our model in order to restrict the set of all the possible candidate representations. Since our aim is to model electrical interconnects and their frequency-dependent behavior, the system will intrinsically exhibit a linear relationship between input and output, due to the nature of the electromagnetic phenomena.
It is well known that any linear system is fully characterized by a rational function of the complex variable s through its input-output transfer function:

$$H(s) = \frac{N(s)}{D(s)}, \tag{1.5}$$

Where $N(s)$ and $D(s)$ are polynomials. Therefore, a natural choice is to try to reconstruct the system through a rational fitting procedure, that returns a model potentially able to catch all the information of interest.

Rational fitting algorithms make use of rational functions as basis for the model. Rational functions are universal approximators: any set of data can be fitted by a series of rational functions if a suitable order (i.e. number of basis functions) is considered. Even if this is for sure an encouraging starting point, several issues affect a modeling process relying on rational fitting:

- The behavior of the returned model is very accurate at the fitting points, but might show an unwanted and improper oscillating nature between the data points and beyond the limits of the data interval; this is particularly common when a very high

order for the interpolating function is chosen.
This phenomenon is known as over-fitting and must be taken into account during the identification procedure: one should use a subset of the available data to test the model quality at points of the domain that are not exploited for the fitting procedure.

- The imposition of constraints that ensure the model physical consistency can prevent the convergence of rational fitting algorithms or, most often, be the cause of a poor quality of the fitting.

From now on, we will assume that the model to be identified is a proper rational function of the variable s, although an extension to the improper case is straightforward. The unknowns that the rational fitting algorithm is intended to return depend on the formulation of the rational function that we want to use. This formulation is fundamental because, as we will see, it can cast the model in forms that are more suitable with respect to others to achieve a good approximation. We now present the most common formulations of rational functions together with the unknowns that an algorithm is expected to return when such formulations are used as starting point.

- **Ratio of polynomials**: in this case, we assume that the model is representative of an underlying dynamics expressed as:

$$H(s;\boldsymbol{x}) = \frac{N(s;\boldsymbol{x})}{D(s;\boldsymbol{x})} = \frac{a_0 + a_1 s + a_2 s^2 + \cdots + a_m s^m}{b_0 + b_1 s + b_2 s^2 + \cdots + b_{n-1} s^{n-1} + s^n} \tag{1.6}$$

  in this case, the unknown vector $\boldsymbol{x}$ collects the $2n$ parameters:

$$\boldsymbol{x} = (a_0, a_1, a_2, \ldots, a_m, b_0, b_1, b_2, \ldots, b_{n-1})^T \tag{1.7}$$

  and the quality of the fitting can be evaluated by means of the residual quantity

$$r_k(\boldsymbol{x}) = \breve{H}_k - \frac{a_0 + a_1 s_k + a_2 s_k^2 + \cdots + a_m s_k^m}{b_0 + b_1 s_k + b_2 s_k^2 + \cdots + b_{n-1} s_k^{n-1} + s_k^n} \tag{1.8}$$

  evaluated for each of the data samples.

- **Pole-zero form**: with this formulation the rational function reads

$$\breve{H}(s,\boldsymbol{x}) = \alpha \frac{\prod_{j=1}^{n-1}(s - z_j)}{\prod_{j=1}^{n}(s - p_j)}; \tag{1.9}$$

  the $2n$ unknown vector is now:

$$\boldsymbol{x} = (\alpha, z_1, z_2, \ldots, z_{n-1}, p_1, p_2, \ldots, p_n)^T \tag{1.10}$$

  and each residual quantity is evaluated as:

$$r_k(\boldsymbol{x}) = \breve{H}_k - \alpha \frac{\prod_{j=1}^{n-1}(s - z_j)}{\prod_{j=1}^{n}(s - p_j)}. \tag{1.11}$$

- **Partial fractions form**: in this case, the rational function is expressed as a series of partial functions of the form:

$$H(s, \boldsymbol{x}) = \sum_{j=1}^{n} \frac{c_j}{s - p_j}, \tag{1.12}$$

with the assumption that the multiplicity of each pole equals one, that is:

$$p_i \neq p_j \quad \forall i \neq j. \tag{1.13}$$

The $2n$ unknown vector is now defined as:

$$\boldsymbol{x} = (c_1, c_2, \ldots, c_n, p_1, p_2, \ldots, p_n)^T, \tag{1.14}$$

and the residual quantities are:

$$r_k(\boldsymbol{x}) = \breve{H}_k - \sum_{j=1}^{n} \frac{c_j}{s - p_j}. \tag{1.15}$$

- **Ratio of rational functions**: to formulate the model in this form, we observe first that any rational function of the variable s can be expressed as a ratio of other two rational functions in s; for this reason, we can cast the model in a more general form that reads:

$$H(s; \boldsymbol{x}) = \frac{N(s; \boldsymbol{x})}{D(s; \boldsymbol{x})} = \frac{\sum_{i=1}^{n} c_i \varphi_i(s)}{\sum_{i=1}^{n} d_i \varphi_i(s)} \tag{1.16}$$

where both numerator and denominator are expressed as a sum of *rational basis functions* $\varphi_i(s)$. In this case, the unknowns vector embeds the $2n$ coefficients of the series expansions:

$$\boldsymbol{x} = (c_1, c_2, \ldots, c_n, d_1, d_2, \ldots, d_n)^T, \tag{1.17}$$

while the residual vector is defined as:

$$r_k(\boldsymbol{x}) = \breve{H}_k - \frac{\sum_{i=1}^{n} c_i \varphi_i(s)}{\sum_{i=1}^{n} d_i \varphi_i(s)} \tag{1.18}$$

1.2 Rational Fitting with Fixed Poles

Our main attempt is to formulate the rational fitting problem in such a way that a linear dependence holds between the unknowns and the basis functions that we want to use to fit the data. If this linear relation holds, then the rational fitting problem can be solved by means of a standard least squares problem: the basis functions are sampled in the points of the domain for which data points are available and the resulting numerical values are used to build the *regressor* matrix of the least square problem.
We can see how, among all the formulations of a rational function, the only one that can guarantee linearity between the unknowns and the basis functions is the partial fractions expansion (1.12) under the assumption that the poles p_j are fixed apriori. This formulation will be deeply exploited in the following since it allows the formulation of the rational fitting problem as a standard least squares problem.

1.2.1 Partial Fractions

We usually define the frequency-dependent basis functions, due to their very convenient numerical properties, as a set of partial fractions with a fixed set of poles. In particular, we realize a prescribed set of distinct $\bar{n}_r$ real poles $q_i \in \mathbb{R}^-$ and $\bar{n}_c$ complex pole pairs $q_{i,i+1} = q_i' \pm \mathrm{j}q_i'' \in \mathbb{C}^-$, where $\varphi_0(s) = 1$. The total number of basis functions is assumed to be $n = 1 + \bar{n}_r + 2\bar{n}_c$, including the constant term. We can define,

$$\begin{aligned} &\text{if } \bar{q}_i \in \mathbb{R} \rightarrow \varphi_i(s) = (\bar{s} - \bar{q}_i)^{-1}; \\ &\text{if } \bar{q}_i \in \mathbb{C} \rightarrow \begin{cases} \varphi_i(s) = (\bar{s} - \bar{q}_i)^{-1} \\ \varphi_{i+1}(s) = \varphi_i^*(s) = (\bar{s} - \bar{q}_i^*)^{-1} \end{cases} \end{aligned} \tag{1.19}$$

To improve numerical conditioning, this basis definition is based on normalized independent variables and poles throughout

$$\bar{s} = \frac{s}{\omega_0}, \quad \bar{q}_i = \frac{q_i}{\omega_0}, \tag{1.20}$$

where ω_0 is a scaling frequency, which is in general obtained considering the largest model pole.

1.2.2 Least Squares Formulation of the Fitting Problem

Denoting with $\varphi_i(s)$ the generic element of our basis of partial fraction defined over a set of poles $\{q_i\}$, with $i = 0,1,2,...,n$, then the residual quantities related to each data sample can be written as:

$$r_k(\boldsymbol{x}) = \breve{H}_k - \boldsymbol{\varphi}_k^T \boldsymbol{x} \tag{1.21}$$

with

$$\boldsymbol{\varphi}_k^T = (\varphi_1(s_k), \varphi_2(s_k), ..., \varphi_n(s_k)), \boldsymbol{x} = (c_1, c_2, ..., c_n)^T \tag{1.22}$$

If we drop the dependency of the residuals on $\boldsymbol{x}$ we can write the above relationship in matrix form by writing:

$$\begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_K \end{pmatrix} = \begin{pmatrix} \breve{H}_1 \\ \breve{H}_2 \\ \vdots \\ \breve{H}_K \end{pmatrix} - \begin{pmatrix} \boldsymbol{\varphi}_1^T \\ \boldsymbol{\varphi}_2^T \\ \vdots \\ \boldsymbol{\varphi}_K^T \end{pmatrix} \boldsymbol{x}. \tag{1.23}$$

We can use the more compact and general notation:

$$\boldsymbol{b} = \begin{pmatrix} \breve{H}_1 \\ \breve{H}_2 \\ \vdots \\ \breve{H}_K \end{pmatrix}, \quad \boldsymbol{r} = \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_K \end{pmatrix}, \quad \boldsymbol{\Phi} = \begin{pmatrix} \boldsymbol{\varphi}_1^T \\ \boldsymbol{\varphi}_2^T \\ \vdots \\ \boldsymbol{\varphi}_K^T \end{pmatrix} \tag{1.24}$$

and write:

$$\boldsymbol{r} = \boldsymbol{b} - \boldsymbol{\Phi}\boldsymbol{x} \tag{1.25}$$

Since our goal is to minimize the value of the residuals, we can solve the least squares problem [27, 28]:

$$\boldsymbol{\Phi x} \approx \boldsymbol{b} \tag{1.26}$$

that returns an unknown vector $\boldsymbol{x}^*$ such that the euclidean norm of the vector $\boldsymbol{r}$ is minimized.
By writing the matrix $\boldsymbol{\Phi}$ in extended form we obtain the Cauchy matrix:

$$\boldsymbol{\Phi} = \begin{pmatrix} 1 & \frac{1}{s_1 - q_1} & \frac{1}{s_1 - q_2} & \dots & \frac{1}{s_1 - q_n} \\ 1 & \frac{1}{s_2 - q_1} & \frac{1}{s_2 - q_2} & \dots & \frac{1}{s_2 - q_n} \\ 1 & \vdots & \vdots & & \vdots \\ 1 & \frac{1}{s_K - q_1} & \frac{1}{s_K - q_2} & \dots & \frac{1}{s_K - q_n} \end{pmatrix}, \tag{1.27}$$

It is well known that the condition number [12, 21] of the normal equations associated to the regressor matrix,

$$\kappa(\boldsymbol{\Phi}) = \sqrt{\frac{\sigma_{max}(\boldsymbol{\Phi}^H \boldsymbol{\Phi})}{\sigma_{min}(\boldsymbol{\Phi}^H \boldsymbol{\Phi})}}, \tag{1.28}$$

strongly influences the quality of the solution of the least squares problem. Fortunately, being the partial fraction basis linearly independent (although not orthogonal) the Cauchy matrix is usually well conditioned.

1.3 General Rational Fitting

The situation explained in the previous section is desirable to solve the fitting problem, but it is very uncommon to known a priori the set of poles of the underlying system. For this reason, black box rational fitting algorithms must be able to return a model without any initial assumption beyond linearity. Two example of such algorithms are the Generalized Sanathanan-Koerner iteration (GSK), introduced in the following, and the Vector Fitting Iteration, for which a discussion can be found in [16].

1.3.1 Generalized Sanathanan Koerner Iteration

The GSK [33]iteration makes use of the model formulation (1.16) to iteratively solve a linearized version of the rational fitting problem. At each iteration ν of the algorithm a modified residual quantity, defined as:

$$r_k^{\nu}(\boldsymbol{x_\nu}) = \frac{D(s_k; \boldsymbol{x}_\nu)\breve{H}_k - N(s_k; \boldsymbol{x}_\nu)}{D(s_k; \boldsymbol{x}_{\nu-1})}; \quad for \quad k = 1, 2, .., K \tag{1.29}$$

is minimized in LS sense. In this formulation $D(s_k; \boldsymbol{x}_\nu)$ is the denominator of the model at the current iteration (that is the one that will be found after the solution of the LS problem), while $D(s_k; \boldsymbol{x}_{\nu-1})$ is the denominator of the model computed at the previous iteration, evaluated at the fitting points. We denote with $\boldsymbol{x}_\nu$ an iteration-dependent

unknowns vector.
The idea behind the GSK algorithm is that as the number of iteration increases, the estimate of the denominator stabilizes, implying that the residual quantity becomes for $\nu \to \infty$

$$r_k^{\nu}(\boldsymbol{x}_{\infty}) = \breve{H}_k - \frac{N(s_k;\boldsymbol{x}_{\infty})}{D(s_k;\boldsymbol{x}_{\infty})} \quad for \quad k = 1, 2, .., K, \tag{1.30}$$

which coincides with the residual that we actually want to minimize. When the model is cast in the form (1.16), then the components of the residual vector $\boldsymbol{r}^{\nu}(\boldsymbol{x}_{\nu})$ at iteration ν will read:

$$r_k^{\nu}(\boldsymbol{x}_{\nu}) = \frac{[\varphi_0(s_k) + \sum_{j=1}^{n} d_j^{\nu}\varphi_j(s_k)]\breve{H}_k - \sum_{j=0}^{n} c_j^{\nu}\varphi_j(s_k)}{\varphi_0(s_k) + \sum_{j=1}^{n} d_j^{\nu-1}\varphi_j(s_k)}. \tag{1.31}$$

Here we imposed $d_0 = 1$ to guarantee a unique solution of the system since the component φ_0 is usually associated with a constant term. We made all the coefficients iteration-dependent.
The iterative minimization of $||\boldsymbol{r}^{\nu}(\boldsymbol{x}_{\nu})||$ is achieved through the least square solution of the system:

$$(\mathbf{M}_{\nu-1}\boldsymbol{\Psi})\boldsymbol{x}_{\nu} \approx \mathbf{M}_{\nu-1}\boldsymbol{b} \tag{1.32}$$

where:

$$\mathbf{M}_{\nu-1} = diag\{m_1^{\nu-1}, m_2^{\nu-1}, ..., m_K^{\nu-1}\}, \quad m_k^{\nu-1} = \frac{1}{D(s_k;\boldsymbol{x}_{\nu-1})},$$

$$\boldsymbol{b} = (\breve{H}_1\varphi_0(s_1), \breve{H}_2\varphi_0(s_2), ..., \breve{H}_K\varphi_0(s_K))^T,$$

$$\boldsymbol{\Psi} = (\boldsymbol{\Phi}_0 - \breve{\mathbf{H}}\boldsymbol{\Phi}_1),$$

$$\boldsymbol{\Phi}_0 = \begin{pmatrix} \varphi_0(s_1) & \varphi_1(s_1) & ... & \varphi_n(s_1) \\ \varphi_0(s_2) & \varphi_1(s_2) & ... & \varphi_n(s_2) \\ \vdots & \vdots & & \vdots \\ \varphi_0(s_K) & \varphi_1(s_K) & ... & \varphi_n(s_K) \end{pmatrix} \tag{1.33}$$

$$\boldsymbol{\Phi}_1 = \begin{pmatrix} \varphi_1(s_1) & \varphi_2(s_1) & ... & \varphi_n(s_1) \\ \varphi_1(s_2) & \varphi_2(s_2) & ... & \varphi_n(s_2) \\ \vdots & \vdots & & \vdots \\ \varphi_1(s_K) & \varphi_2(s_K) & ... & \varphi_n(s_K) \end{pmatrix}$$

The rational basis functions used as a basis is often the partial fractions basis.
We end this chapter by pointing out that the formulations of GSK we presented is given for the scalar case; anyway, a straightforward extension is possible to the multiport systems. For details see [16]. From now on, we will denote with the symbol $\breve{\mathbf{H}}(\cdot) \in \mathbb{C}^{P\times P}$ the multiport response of the true system and with $\mathbf{H}(\cdot) \in \mathbb{C}^{P\times P}$ the multiport response of our models, where the symbol P denotes the number of ports of the system. In the following, we will describe the main model formulations used to characterize a multiport system macromodel.

1.4 Multiport (MIMO) Model Formulations

Approximating the true system response in a suitable macromodel form is fundamental to include the curve fitting result in system-level simulations using standard circuit solver such as SPICE. Several mathematical structures are available: the identification algorithm efficiency, in frequency and time domain, is affected by this choice.
In this Section we are going to describe the model formulation through a transfer matrix and a state space realization; the latter will be useful for the macromodel characterization.

1.4.1 Transfer Function Formulation

Recalling to the scalar model of (1.5), we extend now the formulation realizing a rational model of a MIMO system. Considering a generic MIMO LTI system with rational transfer function, we can adopt the so-called *Generalized Sanathanan-Koerner (GSK)* form [34] [16] as

$$\mathbf{H}(s) = \frac{\mathbf{N}(s)}{\mathsf{D}(s)} = \frac{\sum_{n=0}^{\bar{n}} \mathbf{R}_n \, \varphi_n(s)}{\sum_{n=0}^{\bar{n}} r_n \, \varphi_n(s)}, \tag{1.34}$$

where we denoted with $\mathbf{R}_{n,\ell} \in \mathbb{R}^{P \times P}$ and $r_{n,\ell} \in \mathbb{R}$ the numerator and denominator model (real-valued) coefficients, respectively.
Frequency variations are induced by chosen basis function $\varphi_n(s)$, which are rational functions of s , with $\bar{n}$ frequency basis order.
Avoiding an explicit parameterization of model poles is a critical and necessary condition: in fact, non-smooth behaviours may occur, e.g. when bifurcations are present, with a pair of coincident real poles that split into two complex conjugate poles, or viceversa (see [15] for details).
Both numerator and denominator of (5.16) share the same basis poles set, which are assumed stable.

1.4.2 State Space and Descriptor Form

We now explore the state space and descriptor realizations of a MIMO LTI system, starting from the pole-residue or GSK form of the model $\mathbf{H}(s)$ in the Laplace domain.

State Space for Pole-Residue Form

Considering a general $P \times P$ model in a pole-residue form, we can write

$$\mathbf{H}(s) = \mathbf{H}_\infty + \sum_{n=1}^{\bar{n}} \frac{\mathbf{R}_n}{s - q_n}, \tag{1.35}$$

where $\mathbf{H}_\infty = \mathbf{R}_0$ and $\bar{n}$ is the overall number of poles. We can denote

$$\mathbf{A} = \text{blkdiag}\{q_n \mathbb{I}_P\}_{n=1}^{\bar{n}_r} \tag{1.36}$$

$$\mathbf{B} = \big[1, \ldots, 1\big]^\mathsf{T} \otimes \mathbb{I}_P \tag{1.37}$$

$$\mathbf{C} = \begin{bmatrix} \mathbf{R}_1 & \cdots & \mathbf{R}_{\bar{n}} \end{bmatrix} \tag{1.38}$$

$$\mathbf{D} = \mathbf{H}_\infty \,. \tag{1.39}$$

Starting from the definitions of the matrices above, we can define a regular state-space realization of the system as

$$\begin{cases} \dot{\boldsymbol{x}}(t) = \mathbf{A}\boldsymbol{x}(t) + \mathbf{B}\boldsymbol{u}(t) \\ \boldsymbol{y}(t) = \mathbf{C}\boldsymbol{x}(t) + \mathbf{D}\boldsymbol{u}(t) \end{cases} \tag{1.40}$$

where $\boldsymbol{u}, \boldsymbol{y} \in \mathbb{R}^P$ are the system input and output, respectively, and $\boldsymbol{x} \in \mathbb{R}^N$ are the system internal states.
The notation that we provide for the state-space realization is the following

$$\mathbf{H}(s) = \mathbf{D} + \mathbf{C}(s\mathbb{I} - \mathbf{A})^{-1}\mathbf{B} \leftrightarrow \left(\begin{array}{c|c} \mathbf{A} & \mathbf{B} \\ \hline \mathbf{C} & \mathbf{D} \end{array}\right) \tag{1.41}$$

Considering now the model of (5.16), with $\varphi(s)$ defined as the partial-fraction basis with a prescribed set of real and complex poles q_n (see Section 1.2.1), we can write

$$\mathbf{N}(s) = \mathbf{R}_0 + \sum_{n=1}^{\bar{n}} \frac{\mathbf{R}_n}{s - q_n} \tag{1.42}$$

$$\mathsf{D}(s) = r_0 + \sum_{n=1}^{\bar{n}} \frac{r_n}{s - q_n} \,. \tag{1.43}$$

We now construct the two separate state-space realizations as

$$\mathbf{N}(s) \leftrightarrow \left(\begin{array}{c|c} \mathbf{A}_0 & \mathbf{B}_0 \\ \hline \mathbf{C}_1(s) & \mathbf{D}_1(s) \end{array}\right) \tag{1.44}$$

$$\mathsf{D}(s)\mathbb{I}_P \leftrightarrow \left(\begin{array}{c|c} \mathbf{A}_0 & \mathbf{B}_0 \\ \hline \mathbf{C}_2(s) & \mathbf{D}_2(s) \end{array}\right) , \tag{1.45}$$

where

$$\mathbf{A}_0 = \text{blkdiag}\{\mathbf{A}_{0r}, \mathbf{A}_{0c}\} \tag{1.46}$$

$$\mathbf{B}_0^\mathsf{T} = \big[\mathbf{B}_{0r}^\mathsf{T}, \mathbf{B}_{0c}^\mathsf{T}\big] \tag{1.47}$$

$$\mathbf{C}_1 = \begin{bmatrix} \mathbf{R}_1 & \cdots & \mathbf{R}_{\bar{n}} \end{bmatrix} \tag{1.48}$$

$$\mathbf{C}_2 = \begin{bmatrix} \mathbb{I}_P\, r_1 & \cdots & \mathbb{I}_P\, r_{\bar{n}} \end{bmatrix} \tag{1.49}$$

$$\mathbf{D}_1 = \mathbf{R}_0 \tag{1.50}$$

$$\mathbf{D}_2 = \mathbb{I}_P\, r_0. \tag{1.51}$$

with

$$\mathbf{A}_{0r} = \text{blkdiag}\{q_n \mathbb{I}_P\}_{n=1}^{\bar{n}_r} \tag{1.52}$$

$$\mathbf{A}_{0c} = \text{blkdiag}\left\{ \begin{bmatrix} q'_n \mathbb{I}_P & q''_n \mathbb{I}_P \\ -q''_n \mathbb{I}_P & q'_n \mathbb{I}_P \end{bmatrix} \right\}_{n=1}^{\bar{n}_c} \tag{1.53}$$

$$\mathbf{B}_{0r} = \big[1,\ldots,1\big]^{\mathsf{T}} \otimes \mathbb{I}_P \tag{1.54}$$

$$\mathbf{B}_{0c} = \big[2,0,\ldots,2,0\big]^{\mathsf{T}} \otimes \mathbb{I}_P \tag{1.55}$$

where real-valued matrices have been used for complex conjugate poles.
We finally obtain the (compact) model state-space realization by the cascade of expression (1.44) as

$$\mathbf{H}(s) = \mathbf{N}(s)(\mathsf{D}(s)^{-1}\mathbb{I}_P) \leftrightarrow \left(\begin{array}{c|c} \mathbf{A}_0 - \mathbf{B}_0\mathbf{D}_2^{-1}\mathbf{C}_2 & \mathbf{B}_0\mathbf{D}_2^{-1} \\ \hline \mathbf{C}_1 - \mathbf{D}_1\mathbf{D}_2^{-1}\mathbf{C}_2 & \mathbf{D}_1\mathbf{D}_2^{-1} \end{array} \right) \tag{1.56}$$

We recall [16] and [23] for more details.

Descriptor Form

We now define an alternative descriptor form (or differential-algebraic system of equations (DAE), see [16]) to (1.56) as

$$\begin{cases} \mathbf{E}\dot{\boldsymbol{x}}(t) = \mathbf{A}\boldsymbol{x}(t) + \mathbf{B}\boldsymbol{u}(t) \\ \boldsymbol{y}(t) = \mathbf{C}\boldsymbol{x}(t) \end{cases} \tag{1.57}$$

where $\boldsymbol{u}$ and $\boldsymbol{y}$ are the system input and output, respectively, and the system internal states are $\boldsymbol{x} \in \mathbb{R}^{N+P}$, with $N = \bar{n}P$: the number of states changes with respect to the state-space realization, increasing the problem dimension. The descriptor matrices of (1.57) are realized as

$$\mathbf{E} = \begin{pmatrix} \mathbb{I}_N & \mathbf{0}_{N,P} \\ \mathbf{0}_{P,N} & \mathbf{0}_{P,P} \end{pmatrix} \quad \mathbf{A} = \begin{pmatrix} \mathbf{A}_0 & \mathbf{B}_0 \\ \mathbf{C}_2 & \mathbf{D}_2 \end{pmatrix}$$
$$\mathbf{C} = \begin{pmatrix} \mathbf{C}_1 & \mathbf{D}_1 \end{pmatrix} \quad \mathbf{B} = \begin{pmatrix} \mathbf{0}_{N,P} \\ -\mathbb{I}_P \end{pmatrix} \tag{1.58}$$

with $\mathbf{0}_{J,K}$ null matrix of size $J \times K$. The other matrices of (1.58) denote the state-space realization of the model numerator $\mathbf{N}(s)$, described by the set $\{\mathbf{A}_0, \mathbf{B}_0, \mathbf{C}_1, \mathbf{D}_1\}$, and the (extended) denominator $\mathsf{D}(s)\mathbb{I}_P$, described by $\{\mathbf{A}_0, \mathbf{B}_0, \mathbf{C}_2, \mathbf{D}_2\}$, which are exactly the same elements of (1.44).
It can be proven that the model expression of (5.16) is equivalent to

$$\mathbf{H}(s) = \mathbf{C}(s\mathbf{E} - \mathbf{A})^{-1}\mathbf{B} \tag{1.59}$$

as detailed in [16].
The descriptor form is particularly useful because it requires no block matrix inversion and moreover all matrix elements depend linearly on the model coefficients, in opposition with the regular state space realization of (5.16).
In the following sections we are going to describe in more details how the model should reflect the physical properties of the true system.

1.5 Stability

Several stability definitions may be formulated for an LTI system, analysing the general properties of all the possible solutions of a system. During our work we only modelled black-box systems, which can be characterized, from a stability standpoint, through the matrix $\mathbf{A}$ of the state-space realization (1.40) of the model $\mathbf{H}(s)$.
For this reason, we can define an LTI system [22] [31] [41] as

- *asymptotically stable* if and only if all the poles have a strictly negative real part, $\mathrm{Re}\{q_n\} < 0 \; \forall n$;
- *stable* if and only if all the poles have a non-positive real part, $\mathrm{Re}\{q_n\} \leq 0 \; \forall n$, and all the purely imaginary poles have a multiplicity that is at most one;
- *unstable* if at least one pole has either a strictly positive real part $\mathrm{Re}\{q_n\} > 0$ or a null real part with a multiplicity higher than one.

Furthermore, since the eigenvalues of $\mathbf{A}$ are the model poles q_n from (1.36), the matrix $\mathbf{A}$ can be denoted as (*asymptotically*) *stable* if its eigenvalues have a (strictly) negative real part.

1.6 Passivity

In electronic systems engineering, it is a common practice to deal with many interconnected sub-systems. Especially during high-speed electronic devices design, it is fundamental to assess the signal and power integrity (SI, PI), when all the sub-systems are connected together, since even individual components like vias and packages may strongly affect SI and PI performances if the design is poor. In general, it is common to perform in-depth analyses of these components and, to speed-up the whole process, surrogate macro-models for each sub-system are used, that will be connected together just in simulation phases. Such analyses of interconnected systems may suffer from instabilities, even if all the models are internally asymptotically stable. In fact, if one or more of the single macro-models is not passive, an un-physical energy generation may occur, leading to a distorted output signal which may have detrimental effects on the whole system. This fact, under suitable load conditions, may be responsible of an uncontrolled amplification of the output signal, resulting in an unstable simulation.
Model passivity turns out to be a fundamental requirement that must be carefully analysed when such macro-models are synthesized to ensure reliable simulations under any working condition.

The passivity of a system is strongly related to the net power it absorbs at any time instant t. Considering a P-ports system, the absorbed instantaneous power is

$$p(t) = \sum_{k=1}^{P} p_k(t) = \sum_{k=1}^{P} v_k(t) i_k(t) \tag{1.60}$$

that can be written in compact form as

$$p(t) = \mathbf{v}(t)^{\mathsf{T}} \mathbf{i}(t) = \mathbf{i}(t)^{\mathsf{T}} \mathbf{v}(t) \tag{1.61}$$

where $\mathbf{v} = [v_1, \ldots, v_k]^\mathsf{T}$ and $\mathbf{i} = [i_1, \ldots, i_k]^\mathsf{T}$.

In case the system is in immittance representation, input and output variables, denoted respectively as $u_k(t)$ and $y_k(t)$ may be either voltage or currents. The instantaneous power is thus

$$p(t) = \mathbf{y}^\mathsf{T}(t)\mathbf{u}(t) = \mathbf{u}^\mathsf{T}(t)\mathbf{y}(t) \tag{1.62}$$

Considering input and output as complex valued signals, the instantaneous power definition can be generalized, as

$$p(t) = \mathrm{Re}\left\{\mathbf{v}^\mathsf{H}(t)\mathbf{i}(t)\right\} = \mathrm{Re}\left\{\mathbf{i}^\mathsf{H}(t)\mathbf{v}(t)\right\} \tag{1.63}$$

For scattering representations, voltages v_k and currents i_k are transformed in incident and reflected scattering waves, respectively a_k and b_k. To this end we recall that

$$a_k = \frac{1}{2\sqrt{R_{ref,k}}}(v_k + R_{ref,k} i_k) \tag{1.64}$$

$$b_k = \frac{1}{2\sqrt{R_{ref,k}}}(v_k - R_{ref,k} i_k) \tag{1.65}$$

where $R_{ref} > 0$ is the normalization resistance of each port.
The power $p(t)$ for scattering representation is thus

$$p(t) = \sum_{k=1}^{P} \sqrt{R_{ref,k}}[a_k(t) + b_k(t)] \frac{1}{\sqrt{R_{ref,k}}}[a_k(t) - b_k(t)] = \mathbf{a}(t)^\mathsf{T}\mathbf{a}(t) - \mathbf{b}(t)^\mathsf{T}\mathbf{b}(t) \tag{1.66}$$

with $\mathbf{a}(t) = [a_1(t), \ldots, a_k(t)]$ and $\mathbf{b}(t) = [b_1(t), \ldots, b_k(t)]$.
Defining generic input and output signals as $\mathbf{u}(t) = \mathbf{a}(t)$ and $\mathbf{y}(t) = \mathbf{b}(t)$, it follows that

$$p(t) = \mathbf{u}(t)^\mathsf{T}\mathbf{u}(t) - \mathbf{y}(t)^\mathsf{T}\mathbf{y}(t) \tag{1.67}$$

Generalizing this definition to the case in which $\mathbf{u}$ and $\mathbf{y}$ are complex-valued signals, the instantaneous power is

$$p(t) = \mathbf{u}(t)^\mathsf{H}\mathbf{u}(t) - \mathbf{y}(t)^\mathsf{H}\mathbf{y}(t) \tag{1.68}$$

The net energy absorbed by a P-ports system in a time interval $[t_1,\ t_2]$ is defined as

$$\mathcal{E}(t) = \int_{t_1}^{t_2} p(\tau) d\tau \tag{1.69}$$

If the energy for $t_1 \to -\infty$ is vanishing, the cumulative net energy at an arbitrary time instant t is

$$\mathcal{E}(t) = \int_{-\infty}^{t} p(\tau) d\tau \tag{1.70}$$

The definition for passivity now can be stated.

Definition 1.1 *[16, 41, 42] A P-ports system is passive if the cumulative net energy in* (1.70) *is non-negative for any time t*

$$\mathcal{E}(t) \geq 0, \ \forall t \tag{1.71}$$

and for any input signal $\mathbf{u}(t)$.

The term "passivity" is often replaced by its synonym "dissipativity", so that a passive system is also denoted as "dissipative".

1.6.1 The Dissipation Inequality

The passivity definition given in the previous section regards only the net input/output energy flow, without making any reference to the system internal energy. An alternative way to describe the passivity of a system is to relate the amount of energy it stores and exchanges with the environment, for any time t. Considering a generic system (described in its state space representation) the following dissipativity definition holds:

Definition 1.2 *[16] A system (expressed in its state space representation) is dissipative with respect to the supply rate* $p(t)$ *if there exist a scalar-valued function* $V(\mathbf{x})$*, with* $\mathbf{x}$ *the system states, such that*

$$V(\mathbf{x}(t_1)) \leq V(\mathbf{x}(t_0)) + \int_{t_0}^{t_1} p(t)dt, \ \forall \ t_0 \leq t_1 \, and \, \forall \ \mathbf{u}, \mathbf{y}, \mathbf{x}. \tag{1.72}$$

.

The integral term in (1.72) is exactly the net cumulative energy entering the system in the time interval $[t_0, \ t_1]$, as defined in (1.69). The function $V(\mathbf{x}(t))$ is recognized to be the energy that is stored by the system at any time instant t. Equation (1.72) states that in a system, to be dissipative, the variation on internal energy $V(\mathbf{x}(t_1)) - V(\mathbf{x}(t_0))$ can not exceed the energy that is supplied from the environment to the system during the time interval $[t_0, \ t_1]$.
If the storage function is differentiable, Equation (1.72) can be rewritten in differential form as

$$\frac{d}{dt}V(\mathbf{x}(t)) \leq p(t), \ \forall t \tag{1.73}$$

Under the assumption that the energy stored for $t \to -\infty$ is vanishing, inequality (1.72) reduces to the passivity condition in Equation (1.71). This way to characterize the passivity of a system will turn out to be useful later on, when advanced algebraic passivity assessment methods will be derived

1.6.2 Passivity Characterization

Considering now the class of MIMO (Multi Input-Multi Output) lumped LTI systems with input $\mathbf{u}(t)$ and output $\mathbf{y}(t)$, for which there exist a transfer matrix representation,

the previous dissipativity definition can be written in terms of the transfer function, denoted as $\mathbf{H}(s)$, for both immittance and scattering representations.
For an immittance system, in order to derive passivity conditions in terms of its transfer matrix $\mathbf{H}(s)$, we can explicitly write the instantaneous absorbed power under cisoidal excitation $\mathbf{u}(s) = \mathbf{u}\, e^{st}$ using (1.63) as

$$p(t) = \mathrm{Re}\left\{\mathbf{u}^{\mathsf{H}}\mathbf{H}\mathbf{u}\right\} e^{2\sigma t},\ \sigma = \mathrm{Re}\{s\} \tag{1.74}$$

The cumulative net energy can be computed as

$$\mathcal{E}(t) = \int_{-\infty}^{t} p(\tau)d\tau = \mathrm{Re}\left\{\mathbf{u}^{\mathsf{H}}\mathbf{H}(s)\mathbf{u}\right\} \frac{e^{2\sigma t}}{2\sigma} \tag{1.75}$$

where $\sigma > 0$ to ensure the integral convergence.
Recalling the passivity condition in (1.71), it must hold $\mathcal{E}(t) \geq 0,\ \forall t$. Thus, being $\frac{e^{2\sigma t}}{2\sigma} > 0$ by assumption, it follows that

$$\mathrm{Re}\left\{\mathbf{u}^{\mathsf{H}}\mathbf{H}(s)\mathbf{u}\right\} = \mathbf{u}^{\mathsf{H}}\left[\frac{1}{2}(\mathbf{H}(s) + \mathbf{H}^{\mathsf{H}}(s))\right]\mathbf{u} \geq 0,\ \forall \mathbf{u} \in \mathbb{C}^{P} \tag{1.76}$$

We can conclude that an immittance system is dissipative if

$$\mathbf{H}(s) + \mathbf{H}^{\mathsf{H}}(s) \geq 0,\ \mathrm{Re}\{s\} > 0. \tag{1.77}$$

For further details on these derivations see [16].

To derive passivity conditions for scattering systems, as for the immittance case, the instantaneous power is written in terms of $\mathbf{H}(s)$. Under cisoidal excitation $\mathbf{u}(t)$, recalling Equation (1.68), it reads

$$p(t) = \mathbf{u}(t)^{\mathsf{H}}\mathbf{u}(t) - \mathbf{y}(t)^{\mathsf{H}}\mathbf{y}(t) = \mathbf{u}^{\mathsf{H}}[\mathbb{I} - \mathbf{H}(s)^{\mathsf{H}}\mathbf{H}(s)]\mathbf{u}\, e^{2\sigma t}. \tag{1.78}$$

As for the immittance case, we compute the cumulative net energy absorbed by the system at time instant t as

$$\mathcal{E}(t) = \int_{-\infty}^{t} p(\tau)d\tau = \mathbf{u}^{\mathsf{H}}\left[\mathbb{I} - \mathbf{H}^{\mathsf{H}}(s)\mathbf{H}(s)\right]\mathbf{u}\frac{e^{2\sigma t}}{2\sigma} \tag{1.79}$$

with $\sigma > 0$. The passivity condition in (1.71) implies that

$$\mathbb{I} - \mathbf{H}(s)^{\mathsf{H}}\mathbf{H}(s) \geq 0,\ \mathrm{Re}\{s\} > 0. \tag{1.80}$$

The two passivity conditions for immittance and scattering representation given above are now generalized with reference to Positive Real and Bounded Real matrices [2,16,40].

Definition 1.3 *A transfer matrix* $\mathbf{H}(s)$ *is Positive Real if:*

1. *each element of* $\mathbf{H}(s)$ *is defined and analytic in* $\mathrm{Re}\{s\} > 0$

2. $\mathbf{H}^*(s) = \mathbf{H}(s^*)$
3. $\mathbf{\Theta}(s) = \mathbf{H}(s) + \mathbf{H}^{\mathsf{H}}(s) \geq 0$ *for* $\mathrm{Re}\{s\} > 0$

Definition 1.4 *A transfer matrix* $\mathbf{H}(s)$ *is Bounded Real if:*

1. *each element of* $\mathbf{H}(s)$ *is defined and analytic in* $\mathrm{Re}\{s\} > 0$
2. $\mathbf{H}^*(s) = \mathbf{H}(s^*)$
3. $\mathbf{\Theta}(s) = \mathbb{I} - \mathbf{H}^{\mathsf{H}}(s)\mathbf{H}(s) \geq 0$ *for* $\mathrm{Re}\{s\} > 0$

Condition 1 is related to stability and causality. In fact both causality and stability requires the transfer function to be analytic (must not have poles) in the closed right half complex plane.

Condition 2 may be interpreted as a "consistency" one, since it implies that the transfer matrix is real for any $s \in \mathbb{R}$. This condition strongly affects the residues of $\mathbf{H}(s)$: in fact, to be satisfied, they must be real, for real poles, or must appear in complex conjugate pairs, when corresponding to complex conjugate poles.

Finally, Condition 3 is exactly the one we derived above in Equations 1.77 and 1.80, related to the energy of the system described by $\mathbf{H}(s)$.

We are now ready to re-formulate LTI system passivity conditions in terms of Positive Real and Bounded Real matrices, as stated in Theorem 1.1 ([2, 16, 40])

Theorem 1.1 *A LTI system with transfer matrix* $\mathbf{H}(s)$ *is defined to be passive if and only if* $\mathbf{H}(s)$ *is Positive Real (for immittance representations) or Bounded Real (for scattering representations).*

Theorem 1.1 provides a powerful theoretical tool to check the passivity of an LTI system through its transfer matrix. However, verifying that the three conditions are concurrently fulfilled in the open complex plane, implies considerable computational efforts.
In the following, we derive some simpler conditions, based on the rational nature of the model underlying the transfer matrix $\mathbf{H}(s)$ to assess whether the model is passive, for both immittance and scattering representations.

Considering immittance systems, the following Theorem holds [2, 16, 40]

Theorem 1.2 *A rational matrix* $\mathbf{H}(s)$ *is Positive Real if and only if*

1. $\mathbf{H}(s)$ *has no poles in* $\mathbb{C}_+$
2. $\mathbf{H}^*(\mathrm{j}\omega) = \mathbf{H}(-\mathrm{j}\omega)$
3. $\mathbf{H}(\mathrm{j}\omega) + \mathbf{H}^{\mathsf{H}}(\mathrm{j}\omega) \geq 0,\ \forall \omega \in \mathbb{C}$*, except for simple poles* $\mathrm{j}\omega_i$ *of* $\mathbf{H}(s)$ *where the transfer matrix must be Hermitian and nonnegative definite.*

4. *for* $\omega \to \infty$, $\mathbf{H}(s) \sim \mathbf{R}_\infty s$ *in* $\mathrm{Re}\{s\} > 0$, *with* $\mathbf{R}_\infty$ *real, symmetric and non-negative definite*

The main advantage of this theorem with respect to the more general one, as shown in [16], is evident from the third condition. In fact, comparing it with the one defined in 1.3, it turns out that the non-negative definiteness of $\mathbf{H}(s) + \mathbf{H}^{\mathsf{H}}(s)$ can be checked just along the imaginary axis rather than in the right half open complex plane.
If Conditions 1,2,4 are satisfied (as usually are), the only thing we need to check is Condition 3, whose statement can be cast as follows

$$\lambda_{min}(\mathrm{j}\omega) \geq 0, \ \forall \omega \in \mathbb{R} \tag{1.81}$$

with

$$\lambda_{min}(\mathrm{j}\omega) = \min\{\lambda(\mathbf{H}(\mathrm{j}\omega) + \mathbf{H}^{\mathsf{H}}(\mathrm{j}\omega)\}, \ \forall \omega \in \mathbb{R} \tag{1.82}$$

Assuming the transfer matrix to be asymptotically stable, the above eigenvalues are continuous functions of frequency, therefore $\lambda_{min}(\mathrm{j}\omega)$ is a continuous function of frequency. This fact enables the use frequency sampling techniques in advanced passivity assessment algorithms.
In the next sections we will go through a set of fundamental results to perform advanced and reliable passivity verifications.

Immittance Systems

Particularizing now the dissipation inequality 1.73 for immittance LTI systems, we will derive a condition to assess system passivity in terms of the state-space representation matrices.
To this end, we need to have an analytic expression of the supplied power that is given by Equation 1.74 and reads

$$p(t) = \frac{1}{2}[\mathbf{u}^{\mathsf{T}}(\mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}) + (\mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u})^{\mathsf{T}}\mathbf{u}] \tag{1.83}$$

If the storage function is defined as $V(\mathbf{x}) = \frac{1}{2}(\mathbf{x}^{\mathsf{T}}\mathbf{P}\mathbf{x})$, with $\mathbf{P}$ a symmetric positive definite matrix, its derivative (rate of change of the internal energy) will be

$$\frac{d}{dt}V(\mathbf{x}(t)) = \frac{1}{2}[(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u})^{\mathsf{T}}\mathbf{P}\mathbf{x} + \mathbf{x}^{\mathsf{T}}\mathbf{P}(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u})] \tag{1.84}$$

Let us now impose the dissipativity condition defined in Equation 1.71. Splitting input and state signals, with trivial algebraic manipulations we get to the following LMI form, known as *Positive Real Lemma* [2, 35].

Lemma 1.1 *A LTI system in immittance form is passive if and only if, for any signal* $\mathbf{x}, \mathbf{u}$ *satisfying the state equations, it holds that:*

$$\exists \mathbf{P} = \mathbf{P}^{\mathsf{T}} > 0 : \quad \begin{pmatrix}\mathbf{x}\\ \mathbf{u}\end{pmatrix}^{\mathsf{T}} \begin{pmatrix}\mathbf{A}^{\mathsf{T}}\mathbf{P} + \mathbf{P}\mathbf{A} & \mathbf{P}^{\mathsf{T}}\mathbf{B} - \mathbf{C}^{\mathsf{T}} \\ \mathbf{B}^{\mathsf{T}}\mathbf{P} - \mathbf{C} & -(\mathbf{D} + \mathbf{D}^{\mathsf{T}})\end{pmatrix} \begin{pmatrix}\mathbf{x}\\ \mathbf{u}\end{pmatrix} \leq 0 \tag{1.85}$$

We now derive a fundamental result, originally proposed in [16], used extensively in LTI passivity assessment algorithms, that enables the use of algebraic methods to spot passivity violations. In details, it will be shown that the imaginary eigenvalues of a particular Hamiltonian-structured matrix are strongly related to the location of passivity violations along the frequency axis.
First, let us define a support matrix function, called *Popov function*, $\mathbf{\Psi}(s)$ as

$$\mathbf{\Psi}(s) = \mathbf{H}(s) + \mathbf{H}^{\mathsf{T}}(-s) \tag{1.86}$$

Recalling that, to be passive, the transfer matrix of an immittance system must satisfy

$$\mathbf{\Theta}(s) = \mathbf{H}(s) + \mathbf{H}^{\mathsf{H}}(s) \geq 0 \tag{1.87}$$

in turns out that $\mathbf{\Theta}(s)$ and $\mathbf{\Psi}(s)$ are equal when evaluated on the imaginary axis. This enables us to check the non-negative definiteness of $\mathbf{\Psi}(\mathrm{j}\omega)$ instead of $\mathbf{\Theta}(\mathrm{j}\omega)$.
The condition that must be verified to guarantee passivity is thus

$$\mathbf{\Psi}(\mathrm{j}\omega) \geq 0, \forall\omega \tag{1.88}$$

Focusing our attention to this last equation, we see that the frequencies at which $\mathbf{\Psi}(\mathrm{j}\omega_i)$ becomes singular, algebraically pinpoint passivity violations, being $\mathbf{\Psi}(\mathrm{j}\omega_i)$ singular exactly when $\mathbf{H}(\mathrm{j}\omega) + \mathbf{H}^{\mathsf{T}}(-\mathrm{j}\omega) = 0$.
These frequencies $\mathrm{j}\omega_i$ are defined to be the solutions of

$$\mathbf{\Psi}(\mathrm{j}\omega_i)\mathbf{u} = 0 \tag{1.89}$$

for some vector $\mathbf{u}$.
In order to find these frequencies, we derive a state-space realization of $\mathbf{\Psi}(s)$, the analytic extension to the open complex plane of $\mathbf{\Psi}(\mathrm{j}\omega)$. This turns out to be useful since the solutions of Equation (1.89) are the poles of $\mathbf{\Psi}^{-1}(s)$, for which a simple state space realization is readily computed. The poles of $\mathbf{\Psi}^{-1}(s)$ are the eigenvalues of its dynamic matrix, that reads

$$\boldsymbol{\mathcal{N}}_0 = \mathbf{A}_{\mathbf{\Psi}^{-1}} = \mathbf{A}_\Psi - \mathbf{B}_\Psi \mathbf{D}_\Psi^{-1} \mathbf{C}_\Psi \tag{1.90}$$

where $\mathbf{A}_\Psi,\ \mathbf{B}_\Psi,\ \mathbf{C}_\Psi,\ \mathbf{D}_\Psi$ are the state-space realization matrices of $\mathbf{\Psi}(s)$.
Expanding $\boldsymbol{\mathcal{N}}_0$ in terms of the system realization matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}$ we get the following matrix

$$\boldsymbol{\mathcal{N}}_0 = \begin{pmatrix} \mathbf{A} - \mathbf{B}(\mathbf{D} + \mathbf{D}^{\mathsf{T}})^{-1}\mathbf{C} & -\mathbf{B}(\mathbf{D} + \mathbf{D}^{\mathsf{T}})^{-1}\mathbf{B}^{\mathsf{T}} \\ \mathbf{C}^{\mathsf{T}}(\mathbf{D} + \mathbf{D}^{\mathsf{T}})^{-1}\mathbf{C} & -\mathbf{A}^{\mathsf{T}} + \mathbf{C}^{\mathsf{T}}(\mathbf{D} + \mathbf{D}^{\mathsf{T}})^{-1}\mathbf{B}^{\mathsf{T}} \end{pmatrix} \tag{1.91}$$

Defining as $\mathbf{J}$ the following matrix

$$\mathbf{J} = \begin{pmatrix} 0 & \mathbb{I}_n \\ -\mathbb{I}_n & 0 \end{pmatrix} \tag{1.92}$$

it holds that

$$(\mathbf{J}\boldsymbol{\mathcal{N}}_0)^{\mathsf{T}} = \mathbf{J}\boldsymbol{\mathcal{N}}_0 \tag{1.93}$$

which shows that $\boldsymbol{\mathcal{N}}_0$ has a Hamiltonian structure.
Because of that, $\boldsymbol{\mathcal{N}}_0$ has some peculiar characteristics. In particular, its eigen-spectrum is symmetric with respect to both imaginary and real axes. In fact the set of poles of $\boldsymbol{\Psi}(s)$ includes the ones of $\mathbf{H}(s)$ which are symmetric with respect to the real axis, and their mirror images, symmetric with respect to the imaginary axis.
The following theorem, proposed in [5,13,16], provides a fundamental results that relates the eigenvalues of $\boldsymbol{\mathcal{N}}_0$ with the ones of $\boldsymbol{\Psi}(\mathrm{j}\omega)$.

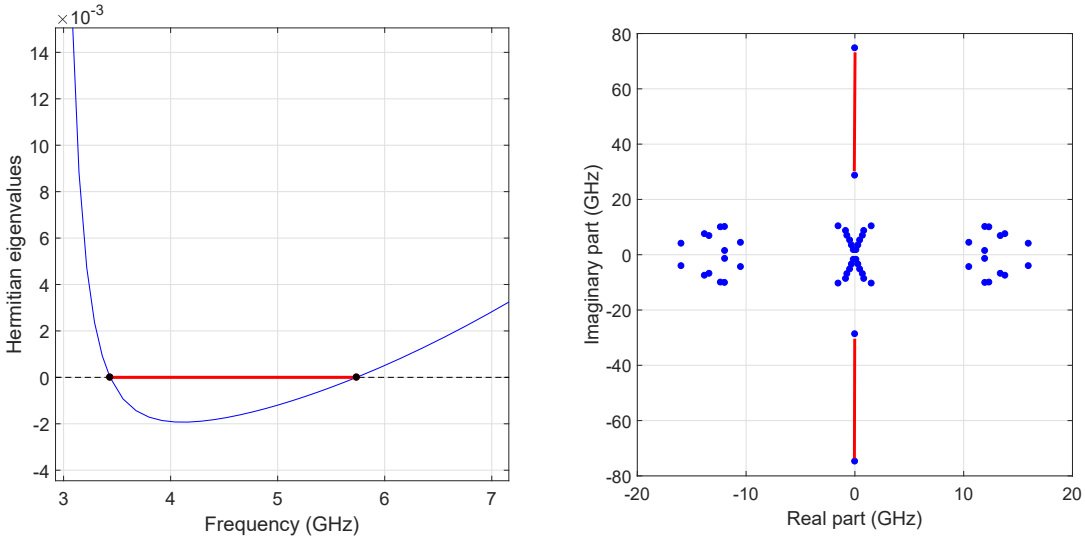
Theorem 1.3 *Let $\mathbf{H}(s)$ be the transfer matrix of an immittance system, whose state space matrices are $\{\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D}\}$, where $\mathbf{A}$ has no purely imaginary eigenvalues and $\mathbf{D}+\mathbf{D}^{\mathsf{T}}$ is non-singular. Then, $\mathrm{j}\omega_0$ is an eigenvalue of $\boldsymbol{\mathcal{N}}_0$ if and only if 0 is an eigenvalue of $\boldsymbol{\Psi}(\mathrm{j}\omega_0)$.*

It follows that, if $\boldsymbol{\mathcal{N}}_0$ has imaginary eigenvalues, the related LTI system is not passive for some frequency bands.
This result is formally stated in Theorem 1.4.

Theorem 1.4 *Let $\mathbf{H}(s)$ be the transfer matrix of an immittance system, whose state space matrices are $\{\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D}\}$, where $\mathbf{A}$ has no purely imaginary eigenvalues and $\mathbf{D}+\mathbf{D}^{\mathsf{T}}$ is positive definite. Then the system is passive if the Hamiltonian matrix $\boldsymbol{\mathcal{N}}_0$ has no purely imaginary eigenvalues.*

Theorems 1.3 and 1.4 provide an algebraic tool that is able to precisely verify system passivity and enables us to easily localize violation areas along the frequency axis.
To this end we must notice that Hamiltonian imaginary eigenvalues correspond to the complex frequencies at which at least one eigenvalue of $\boldsymbol{\Psi}(\mathrm{j}\omega)$ crosses the zero-threshold. These frequencies induce a partition of the frequency axis in disjoint sub-bands, where $\boldsymbol{\Psi}(\mathrm{j}\omega)$ is either positive definite or not. This means that, being the Hamiltonian eigenvalues the edges of these sub-bands, the frequency axis is now partitioned in passive and not-passive areas, so that a detailed passivity characterization is available.
In Figure 1.1 we show the described partitioning of the frequency axis in passive and non-passive bands induced by imaginary Hamiltonian eigenvalues. In the left panel we show an eigenvalue of $\mathbf{H}(\mathrm{j}\omega)+\mathbf{H}^{\mathsf{H}}(\mathrm{j}\omega)$ that, becoming negative, denote a non-passive frequency band, shown in red. Imaginary Hamiltonian eigenvalues are represented as black dots and bound this violation area. In the right panel we show the Hamiltonian eigen-spectrum, where it is possible to see that the magnitude of purely imaginary eigenvalues coincides with the edges of the violation interval discussed before. The violation bands in the complex plane are represented with red lines.

The main result presented here relies on the strong assumption that $\mathbf{D}+\mathbf{D}^{\mathsf{T}}$ is not singular. However, the same approach can be extended to the case in which $\mathbf{D}+\mathbf{D}^{\mathsf{T}}$ is singular with minor modifications. For details see [16].
In order to relax the non-singularity condition on $\mathbf{D}+\mathbf{D}^{\mathsf{T}}$, it is necessary to slightly modify Theorem 1.3 resulting in an extended eigenvalue problem where, now, no inversions on $\mathbf{D}+\mathbf{D}^{\mathsf{T}}$ are required. The new problem, shown in Equation (1.94) is cast in what is usually called a "generalized eigenvalue problem", where the unknowns are no more the eigenvalues of a matrix, but the ones of a matrix pencil $(\boldsymbol{\mathcal{N}}_0^{ext},\boldsymbol{\mathcal{K}})$.



(a) Passive/non-passive frequency axis partitioning

(b) Violation bands in the Hamiltonian eigenspectrum

Figure 1.1: Frequency axis partitioning induced by Hamiltonian imaginary eigenvalues

$$\boldsymbol{\mathcal{N}}_0^{ext}\mathbf{v} = \mathrm{j}\omega_0\boldsymbol{\mathcal{K}}\mathbf{v} \tag{1.94}$$

where

$$\boldsymbol{\mathcal{N}}_0^{ext} = \begin{pmatrix} \mathbf{A} & \mathbf{0} & \mathbf{B} \\ \mathbf{0} & -\mathbf{A}^\mathsf{T} & -\mathbf{C}^\mathsf{T} \\ \mathbf{C} & \mathbf{B}^\mathsf{T} & \mathbf{D}+\mathbf{D}^\mathsf{T} \end{pmatrix}, \qquad \boldsymbol{\mathcal{K}} = \begin{pmatrix} \mathbb{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbb{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix} \tag{1.95}$$

This matrix pencil is denoted as "*Skew-Hamiltonian/Hamiltonian*", because $\boldsymbol{\mathcal{N}}_0^{ext}$ has Hamiltonian structure while $\boldsymbol{\mathcal{K}}$ is skew-Hamiltonian.

Up to now, just a state-space realization for $\mathbf{H}(s)$ has been considered. However there are several situations for which a descriptor realization is preferable, e.g., when using MNA (Modified Nodal Analysis) method to automatically solve electrical circuits. For this reason, a generalization of the Hamiltonian approach to descriptor realization is needed. Minor modifications to Theorem 1.3 allow to state that, for immittance systems in descriptor form, the complex frequencies at which passivity violations occur are the purely imaginary generalized eigenvalues of this generalized eigen-problem:

$$\boldsymbol{\mathcal{N}}_0^{ext}\mathbf{v} = \mathrm{j}\omega_0\boldsymbol{\mathcal{K}}\mathbf{v} \tag{1.96}$$

where

$$\boldsymbol{\mathcal{N}}_0^{ext} = \begin{pmatrix} \mathbf{A} & \mathbf{0} & \mathbf{B} \\ \mathbf{0} & -\mathbf{A}^\mathsf{T} & -\mathbf{C}^\mathsf{T} \\ \mathbf{C} & \mathbf{B}^\mathsf{T} & \mathbf{D}+\mathbf{D}^\mathsf{T} \end{pmatrix}, \qquad \boldsymbol{\mathcal{K}} = \begin{pmatrix} \mathbf{E} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{E}^\mathsf{T} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix} \tag{1.97}$$

Scattering systems

We now focus on scattering systems.
Recalling Theorem 1.1, a scattering system, to be passive, must have a Bounded Real transfer matrix. Again, verifying system passivity throughout the complex plane is too expensive in terms of computational effort.
As for the Positive Real Lemma, a formulation of the Bounded Real Lemma exists for rational matrices [2, 16, 40], that are the main focus of this work.

Theorem 1.5 *A rational matrix* $\mathbf{H}(s)$ *is Bounded Real if and only if*

1. $\mathbf{H}(s)$ *has no poles in* $\mathbb{C}_+$
2. $\mathbf{H}^*(\mathrm{j}\omega) = \mathbf{H}(-\mathrm{j}\omega)$
3. $\mathbb{I} - \mathbf{H}(\mathrm{j}\omega)^{\mathsf{H}}\mathbf{H}(\mathrm{j}\omega) \geq 0, \ \forall\omega \in \mathbb{R}$

No further conditions are required, as in the immittance case, for purely imaginary poles, because passive scattering systems can not have poles on the imaginary axis. As in Theorem 1.2, the main advantage that the rational nature of the system brings with it, is that Conditions 2 and 3 can be checked just along the imaginary axis. Assuming the system to be asymptotically stable (all the poles of $\mathbf{H}(s)$ has strictly negative real part) and that the state-space realization matrices real, the first two conditions are verified and only the third remains to be checked.
Here, in contrast with the immittance case, a product of transfer matrices appears, so a direct eigenvalues calculation, to guarantee that the smaller one is above the zero threshold, should be avoided. An alternative formulation for Condition 3 is based on the SVD (Singular Values Decomposition) of $\mathbf{H}(\mathrm{j}\omega)$, that reads

$$\mathbf{H}(\mathrm{j}\omega) = \mathbf{U}(\mathrm{j}\omega)\mathbf{\Sigma}(\mathrm{j}\omega)\mathbf{V}(\mathrm{j}\omega)^{\mathsf{H}} \tag{1.98}$$

The third condition is then re-formulated in terms of the singular values of $\mathbf{H}(\mathrm{j}\omega)$:

$$\mathbb{I} - \mathbf{H}(\mathrm{j}\omega)^{\mathsf{H}}\mathbf{H}(\mathrm{j}\omega) \geq 0 \Leftrightarrow \sigma_{max}(\mathbf{H}(\mathrm{j}\omega)) = \|\mathbf{H}(\mathrm{j}\omega)\|_2 \leq 1, \ \forall\omega \in \mathbb{R}. \tag{1.99}$$

Being additionally, by assumption, the transfer matrix $\mathbf{H}(\mathrm{j}\omega)$ regular in an open subset of the complex plane containing the imaginary axis, singular values are continuous functions of $\mathrm{j}\omega$, enabling the use of frequency sampling techniques.
Since any passive system must satisfy the dissipation inequality in (1.73), to derive a precise passivity characterization, it must be particularized for scattering systems.
The supplied power $p(t)$ is

$$p(t) = \mathbf{u}^{\mathsf{T}}\mathbf{u} - \mathbf{y}^{\mathsf{T}}\mathbf{y} = \mathbf{u}^{\mathsf{T}}\mathbf{u} - (\mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u})^{\mathsf{T}}(\mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{u}), \tag{1.100}$$

where $\mathbf{u}, \mathbf{y}$ are respectively the input and output signals and the time dependency has been omitted for readability.
The storage function $V(\mathbf{x})$, defined as $V(\mathbf{x}) = \mathbf{x}^{\mathsf{T}}\mathbf{P}\mathbf{x}$, with $\mathbf{P} = \mathbf{P}^{\mathsf{T}} \geq 0$, leads to the

following equation

$$\frac{d}{dt}V(\mathbf{x}(t)) = (\mathbf{A}\mathbf{x}+\mathbf{B}\mathbf{u})^{\mathsf{T}}\mathbf{P}\mathbf{x} + \mathbf{x}^{\mathsf{T}}\mathbf{P}(\mathbf{A}\mathbf{x}+\mathbf{B}\mathbf{u}) \leq p(t),\ \forall t. \tag{1.101}$$

Combining the previous relation with the dissipation inequality, and splitting the input and state signals, the so-called *Bounded Real Lemma* [2,35] can be stated.

Lemma 1.2 *A LTI system in scattering form is passive if and only if, for any signal* $\mathbf{x},\mathbf{u}$ *satisfying the state equations, it holds that:*

$$\exists \mathbf{P}=\mathbf{P}^{\mathsf{T}}>0: \quad \begin{pmatrix}\mathbf{x}\\ \mathbf{u}\end{pmatrix}^{\mathsf{T}} \begin{pmatrix}\mathbf{A}^{\mathsf{T}}\mathbf{P}+\mathbf{P}\mathbf{A}+\mathbf{C}^{\mathsf{T}}\mathbf{C} & \mathbf{P}\mathbf{B}+\mathbf{C}^{\mathsf{T}}\mathbf{D}\\ \mathbf{B}^{\mathsf{T}}\mathbf{P}+\mathbf{D}^{\mathsf{T}}\mathbf{C} & -(\mathbb{I}-\mathbf{D}^{\mathsf{T}}\mathbf{D})\end{pmatrix}\begin{pmatrix}\mathbf{x}\\ \mathbf{u}\end{pmatrix} \leq 0 \tag{1.102}$$

In the following we derive, as for immittace representations, a set of results that enables to cast the passivity verification problem in a closed algebraic form. See [16] for details.
Defining $\boldsymbol{\Theta}(s)$ as

$$\boldsymbol{\Theta}(s) = \mathbb{I} - \mathbf{H}^{\mathsf{H}}(s)\mathbf{H}(s), \tag{1.103}$$

and denoting the Popov function as

$$\boldsymbol{\Psi}(s) = \mathbb{I} - \mathbf{H}^{\mathsf{T}}(-s)\mathbf{H}(s), \tag{1.104}$$

it is easy to see that, when evaluating these functions for $s=\mathrm{j}\omega$, they are equal:

$$\boldsymbol{\Psi}(\mathrm{j}\omega) = \boldsymbol{\Theta}(\mathrm{j}\omega). \tag{1.105}$$

Passivity condition can be cast in terms of the Popov function as

$$\boldsymbol{\Psi}(\mathrm{j}\omega) \geq 0,\ \forall\omega \tag{1.106}$$

Equation 1.106 exactly matches the one for immittance representations, where passivity violations are solutions of

$$\boldsymbol{\Psi}(\mathrm{j}\omega)\mathbf{u} = 0 \tag{1.107}$$

for some vector $\mathbf{u}$.
To find the zeros of $\boldsymbol{\Psi}(\mathrm{j}\omega)$, a state space realization for $\boldsymbol{\Psi}(s)$ (whose matrices are $\mathbf{A}_\Psi,\mathbf{B}_\Psi,\mathbf{C}_\Psi,\mathbf{D}_\Psi$) is derived, from which it is possible to get a realization for $\boldsymbol{\Psi}^{-1}(s)$, whose purely imaginary poles are the solutions of Equation (1.107). The poles of $\boldsymbol{\Psi}^{-1}(s)$ are the eigenvalues of its state-space dynamic matrix, that reads:

$$\boldsymbol{\mathcal{M}}_1 = \mathbf{A}_\Psi - \mathbf{B}_\Psi\mathbf{D}_\Psi^{-1}\mathbf{C}_\Psi. \tag{1.108}$$

Writing now this matrix in terms the state-space realization matrices $\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D}$ of $\mathbf{H}(s)$, we get the following matrix:

$$\boldsymbol{\mathcal{M}}_1 = \begin{pmatrix}\mathbf{A}+\mathbf{B}(\mathbb{I}-\mathbf{D}^{\mathsf{T}}\mathbf{D})^{-1}\mathbf{D}^{\mathsf{T}}\mathbf{C} & \mathbf{B}(\mathbb{I}-\mathbf{D}^{\mathsf{T}}\mathbf{D})^{-1}\mathbf{B}^{\mathsf{T}}\\ -\mathbf{C}^{\mathsf{T}}(\mathbb{I}-\mathbf{D}\mathbf{D}^{\mathsf{T}})^{-1}\mathbf{C} & -\mathbf{A}^{\mathsf{T}}-\mathbf{C}^{\mathsf{T}}\mathbf{D}(\mathbb{I}-\mathbf{D}^{\mathsf{T}}\mathbf{D})^{-1}\mathbf{B}^{\mathsf{T}}\end{pmatrix} \tag{1.109}$$

Matrix $\boldsymbol{\mathcal{M}}_1$ has Hamiltonian structure, since it satisfies the condition in (1.93).

What relates matrix $\boldsymbol{\mathcal{M}}_1$ with system passivity is given by the following theorem [5,13,16]:

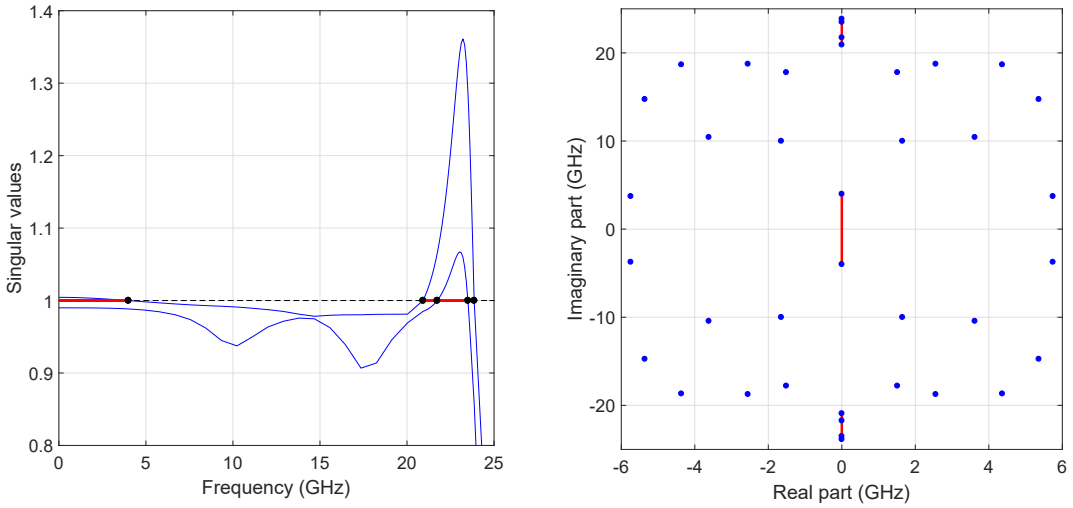
Theorem 1.6 *Let* $\mathbf{H}(s)$ *be the transfer matrix of a scattering system, whose state space matrices are* $\{\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}\}$*, where* $\mathbf{A}$ *has no purely imaginary eigenvalues and* $\mathbb{I} - \mathbf{D}^\mathsf{T}\mathbf{D}$ *is non-singular. Then,* $\mathrm{j}\omega_0$ *is an eigenvalue of* $\boldsymbol{\mathcal{M}}_1$ *if and only if 0 is an eigenvalue of* $\boldsymbol{\Psi}(\mathrm{j}\omega_0)$ *and 1 a singular value of* $\mathbf{H}(\mathrm{j}\omega_0)$*.*

This result allows us to derive the following theorem, that provides a sufficient passivity condition for scattering systems:

Theorem 1.7 *Let* $\mathbf{H}(s)$ *be the transfer function of an asymptotically passive (*$\|\mathbf{D}\|_2 < 1$*) and stable scattering system, whose state-space matrices are* $(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D})$*. The system is uniformly passive if* $\boldsymbol{\mathcal{M}}_1$ *has no purely imaginary eigenvalues*

Furthermore, the frequencies ω_i solving $\boldsymbol{\Psi}(\mathrm{j}\omega_i)\mathbf{u} = 0$, i.e., the Hamiltonian imaginary eigenvalues, induce a partition of the frequency axis in passive and not-passive sub-bands. These considerations allow to characterize in details the passivity of a system for any frequency value.

Figure 1.1 shows the partitioning of the frequency axis in passive and non-passive bands induced by imaginary Hamiltonian eigenvalues. In the left panel we show singular values of $\mathbf{H}(\mathrm{j}\omega)$ that, denote non-passive areas when exceed the unit threshold, represented in red. Imaginary Hamiltonian eigenvalues are represented as black dots and bound these violation areas. In the right panel we show the Hamiltonian eigen-spectrum, where we can see that the magnitude of purely imaginary eigenvalues coincide with the edges of the violations interval discussed before. The violations band in the complex plane are represented with red lines.



(a) Passive/non-passive frequency axis partitioning

(b) Violation bands in the Hamiltonian eigen-spectrum

Figure 1.2: Frequency axis partitioning induced by Hamiltonian imaginary eigenvalues

As we did for immittance systems, it is possible to relax the non-singularity condition on $\mathbb{I} - \mathbf{D}^\mathsf{T}\mathbf{D}$.

As proposed in [16], Theorem 1.6 can be generalized to the case in which $\mathbf{D}$ is arbitrary. Slightly modifying its proof, it is possible to define an extended eigen-problem shown in (1.110), that does not require any inversion of $\mathbb{I} - \mathbf{D}\mathbf{D}^\mathsf{T}$ and $\mathbb{I} - \mathbf{D}^\mathsf{T}\mathbf{D}$, as:

$$\boldsymbol{\mathcal{M}}_1^{ext}\mathbf{v} = \mathrm{j}\omega_0\boldsymbol{\mathcal{K}}\mathbf{v}, \tag{1.110}$$

where

$$\boldsymbol{\mathcal{N}}_0^{ext} = \begin{pmatrix} \mathbf{A} & \mathbf{0} & \mathbf{B} & \mathbf{0} \\ \mathbf{0} & -\mathbf{A}^\mathsf{T} & \mathbf{0} & -\mathbf{C}^\mathsf{T} \\ \mathbf{0} & \mathbf{B}^\mathsf{T} & -\mathbb{I} & \mathbf{D}^\mathsf{T} \\ \mathbf{C} & \mathbf{0} & \mathbf{D} & \mathbb{I} \end{pmatrix}, \qquad \boldsymbol{\mathcal{K}} = \begin{pmatrix} \mathbb{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbb{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix} \tag{1.111}$$

It can be proven that purely imaginary eigenvalues of 1.110 correspond exactly to the location on the frequency axis of passivity violations.

Previous results are based on the assumption that a state-space realization for $\mathbf{H}(s)$ is used. Here, we provide a generalization of the Hamiltonian-driven passivity characterization to descriptor realizations, that will be used extensively later on in this work, and are of paramount importance in many other applications.
Passivity violations are again defined by complex frequencies $\mathrm{j}\omega_i$ for which $\boldsymbol{\Psi}(\mathrm{j}\omega_i)\mathbf{v} = 0$. Suitably modifying Theorem 1.6, we find that this condition is reached if and only if $\mathrm{j}\omega_i$ is an eigenvalue of the generalized eigenproblem in (1.112)

$$\boldsymbol{\mathcal{M}}_1^{ext}\mathbf{v} = \mathrm{j}\omega_0\boldsymbol{\mathcal{K}}\mathbf{v} \tag{1.112}$$

where

$$\boldsymbol{\mathcal{N}}_0^{ext} = \begin{pmatrix} \mathbf{A} & \mathbf{0} & \mathbf{B} & \mathbf{0} \\ \mathbf{0} & -\mathbf{A}^\mathsf{T} & \mathbf{0} & -\mathbf{C}^\mathsf{T} \\ \mathbf{0} & \mathbf{B}^\mathsf{T} & -\mathbb{I} & \mathbf{D}^\mathsf{T} \\ \mathbf{C} & \mathbf{0} & \mathbf{D} & \mathbb{I} \end{pmatrix}, \qquad \boldsymbol{\mathcal{K}} = \begin{pmatrix} \mathbf{E} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{E}^\mathsf{T} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{pmatrix} \tag{1.113}$$

Chapter 2

Classical Formulation of Generalized Sanathanan Koerner and Vector Fitting Algorithms

In this chapter, we present the main results concerning the rational fitting algorithms that are exploited and improved in this Thesis project.

2.0.1 Sanathanan Koerner Iteration

The Sanathanan Koerner (SK) [33] is a rational fitting algorithm intended to obtain a rational model in the form:

$$H(s;\boldsymbol{x}) = \frac{N(s;\boldsymbol{x})}{D(s;\boldsymbol{x})} = \frac{a_0 + a_1 s + a_2 s^2 + \cdots + a_m s^m}{b_0 + b_1 s + b_2 s^2 + \cdots + b_{n-1} s^{n-1} + s^n} \tag{2.1}$$

that is, to return a vector $\boldsymbol{x}$ of unknown model coefficients:

$$\boldsymbol{x} = (a_0, a_1, a_2, \ldots, a_m, b_0, b_1, b_2, \ldots, b_{n-1})^T \tag{2.2}$$

given a set of measurements $(s_k, \breve{H}_k)$ for $k = 1,2,\ldots,K$, a degree m for the numerator and n for the denominator.
Since our aim is to minimize the difference between the data and the model at the prescribed data points, we want to minimize the residual quantities:

$$r_k(\boldsymbol{x}) = \breve{H}_k - \frac{a_0 + a_1 s_k + a_2 s_k^2 + \cdots + a_m s_k^m}{b_0 + b_1 s_k + b_2 s_k^2 + \cdots + b_{n-1} s_k^{n-1} + s_k^n} \tag{2.3}$$

for every k. The most common choice for the cost function is the sum of the squared errors between the data and the model values:

$$F(\boldsymbol{x}) = \sum_{k=1}^{K} \left| \breve{H}_k - H(s_k, \boldsymbol{x}) \right|^2 = \|\boldsymbol{r}(\boldsymbol{x})\|^2 , \tag{2.4}$$

where we collected the residuals $r_k(\boldsymbol{x})$ in a single vector $\boldsymbol{r}(\boldsymbol{x})$.
The minimization of a quadratic cost function is usually achieved solving a standard least squares problem that returns the optimal value of $\boldsymbol{x}$; however we see that in the case of rational fitting the residual vector does not depend linearly on the model coefficients: we are looking for the solution of a nonlinear least square problem.
Even if several techniques have been investigated for the solution of this kind of problem, it is well known that nonlinear optimization algorithms can not guarantee the achievement of the optimal solution due to the possible prensence of multiple local minima [8] [30]. For this reason, various attempts have been done in order to obtain a linearized expression for the cost function.
In 1959, Levy [26] proposed to modify the cost function with the definition of a novel residual quantity, defined as:

$$\begin{aligned} e_k(\boldsymbol{x}) =& D(s_k;\boldsymbol{x})r_k(\boldsymbol{x}) = (b_0 + b_1 s_k + \cdots + b_{n-1}s_k^{n-1} + s_k^n)\breve{H}_k + \\ & - (a_0 + a_1 s_k + \cdots + a_m s_k^m) \end{aligned} \tag{2.5}$$

where $D(s_k;\boldsymbol{x})$ is the model denominator, that is unknown apriori.
This manipulation ensures the linear dependancy between the model coefficients and the cost function, introducing a frequency-dependent factor that cannot be estimated until the model coefficients are obtained. This procedure is likely to return an optimal value of the coefficients that is far different from the one we expect: the polynomial at the denominator spans a very large range of values as the complex frequency changes, inducing a weighting that magnifies or reduces the fitting error with respect to a quantity that depends on the model to be fitted.
An analytic description of this fact is obtained defining:

$$\begin{aligned} \mathbf{D}(\boldsymbol{x}) &= diag\{D(s_1;\boldsymbol{x}), \ldots, D(s_k;\boldsymbol{x})\} \\ \mathbf{W}(\boldsymbol{x}) &= diag\{|D(s_1;\boldsymbol{x})|, \ldots, |D(s_k;\boldsymbol{x})|\}, \end{aligned} \tag{2.6}$$

and writing the cost function associated with the new residuals:

$$G(\boldsymbol{x}) = \|\boldsymbol{e}(\boldsymbol{x})\|^2 = \|\mathbf{D}(\boldsymbol{x})\boldsymbol{r}(\boldsymbol{x})\|^2 = \boldsymbol{r}(\boldsymbol{x})^H \mathbf{W}^2(\boldsymbol{x})\boldsymbol{r}(\boldsymbol{x}) = \|\mathbf{W}(\boldsymbol{x})\boldsymbol{r}(\boldsymbol{x})\|^2 . \tag{2.7}$$

Intuitively, the least square algorithm would struggle to minimize residuals at frequencies associated with larger denominator values, giving less importance to the others.
Even if the Levy's approach is unsuitable to obtain a good fitting of the data, it is the starting point for the formulation of the SK algorithm, which is thought to iteratively solve the fitting problem trying to compensate the weight introduced by the denominator, better and better at each iteration, until the modified residues vector $\boldsymbol{e}(\boldsymbol{x})$ converges to the nominal one, $\boldsymbol{r}(\boldsymbol{x})$. To reach the goal, SK redefines at each iteration (denoted here as ν) the residuals, dividing them by the value assumed by the denominator estimated at the previous iteration; at the first iteration, no information about the model denominator is avalaible and the residuals vector coincides with the Levy's one. In formulas we have:

$$e_k^{\nu}(\boldsymbol{x_\nu}) = \frac{D(s_k;\boldsymbol{x}_\nu)\breve{H}_k - N(s_k;\boldsymbol{x}_\nu)}{D(s_k;\boldsymbol{x}_{\nu-1})}; \quad for \quad k = 1, 2, .., K \tag{2.8}$$

With $D(s_k; \boldsymbol{x}_1) = 1$. Note that, at each iteration, the value of the denominator estimated at the previous iteration is numerically avalaible and the minimization of the residuals can be achieved through linear least squares.
As the number of iterations increases, if the method converges, the denominator estimate stabilizes and the frequency-dependent bias is compensated so that:

$$e_k^{\nu}(\boldsymbol{x_\nu}) \to r_k(\boldsymbol{x_\nu}) \quad \forall k, \quad for \quad \nu \to \infty \tag{2.9}$$

To build the least squares system that must be solved at each iteration, we collect the residuals in matrix form, defining the following matrices:

- Vandermonde Matrix $\boldsymbol{\Phi}_{n+1}$: it is the matrix that collects in each row the samples of the monomial basis until order n [24], evaluated at the frequency points for which data are avalaible; it takes the form:

$$\boldsymbol{\Phi}_{n+1} = \begin{pmatrix} 1 & s_1 & ... & s_1^n \\ 1 & s_2 & ... & s_2^n \\ \vdots & \vdots & & \vdots \\ 1 & s_K & ... & s_K^n \end{pmatrix} \tag{2.10}$$

- The data diagonal matrix $\breve{\mathbf{H}}$: collects data samples in a diagonal matrix:

$$\breve{\mathbf{H}} = diag\{\breve{H}_1, \breve{H}_2, ..., \breve{H}_K\} \tag{2.11}$$

- The denominator inverse matrix $\mathbf{M}_{\nu-1}$: the diagonal matrix which collects the inverse of the values of the denominator at the previous iteration:

$$\mathbf{M}_{\nu-1} = diag\{m_1^{\nu-1}, m_2^{\nu-1}, ..., m_K^{\nu-1}\}, \quad m_k^{\nu-1} = \frac{1}{D(s_k; \boldsymbol{x}_{\nu-1})} \tag{2.12}$$

With these definitions, we refer to the first iteration of the SK algorithm and we collect the relative residuals in matrix form by writing:

$$\boldsymbol{e}^1(\boldsymbol{x}_1) = \boldsymbol{b} - \boldsymbol{\Psi}\boldsymbol{x}_1 \tag{2.13}$$

Here, $\boldsymbol{b}$ stacks the known term components $\breve{H}_k s_k^n$ in a column vector, while the matrix $\boldsymbol{\Psi}$, defined as:

$$\boldsymbol{\Psi} = (\boldsymbol{\Phi}_{m+1} - \breve{\mathbf{H}}\boldsymbol{\Phi}_n), \tag{2.14}$$

collects the monomial basis for the denominator (properly scaled, row by row, with respect to the data) and the numerator. The resulting least square system reads:

$$\boldsymbol{\Psi}\boldsymbol{x}_1 \approx \boldsymbol{b} \tag{2.15}$$

The solution of this system coincides with the solution found with the Levy's method.
From the second iteration on, an expression for the previous denominator estimate is available and the least squares problem is modified to compensate the bias, multiplying each row of regressor and known term components by the inverse of the denominator values:

$$(\mathbf{M}_{\nu-1}\mathbf{\Psi})\boldsymbol{x}_\nu \approx \mathbf{M}_{\nu-1}\boldsymbol{b}, \quad \nu > 1 \tag{2.16}$$

Even if the SK iteration is theoretically able to compensate the bias induced by the linearization procedure, it still suffers of numerical illness: the Vandermonde matrices that form the regressor can assume a very broad range of values as the order of the polynomials increase; as a consequence, the condition number of the matrix is likely to affect the accuracy of the estimate. The row scaling induced by $\mathbf{M}_{\nu-1}$ is not sufficient to guarantee the required precision. Further, the above formulation would lead to additional numerical precision loss when one tries to convert the ratio of two polynomials into a state space model and to a total lack of control on the model poles, that could exhibit positive real parts, driving the modeled system to instability in time domain simulations.

2.0.2 Change of Basis Functions: Partial Fractions

The issues that affect the Sanathanan-Koerner iteration can be overcome with a choice of a basis function different from the monomials; an immediate solution could be the use of a family of orthogonal polynomials which would result in a better conditioning of the least square problem.
Anyway, this choice would solve only one of the above-mentioned problems . An optimal set of basis functions should:

1. lead to a well conditioned regressor matrix,
2. limit the dynamic range of the denominator of the model to avoid an induced ill-conditioning,
3. allow some degree of control over the model poles,
4. be able to return a model easily convertible in state space form.

Any polynomials could satisfy only the first condition.
We are looking for a model formulation in the form 1.16: Consider now the equation (1.32), with the definitions in (1.33); except for the definition of more general basis matrices $\mathbf{\Phi}_0$ and $\mathbf{\Phi}_1$, the structure of the least square problem to be solved is the same as in the standard Sanathanan-Koerner iteration.
We now consider the particular choice of partial fractions as rational basis functions for the model construction; these functions are defined as in 1.2.1, even if here we do not consider the frequency scaling in order to simplify the notation. In particular, our partial fractions must satisfy the following conditions:

$$\begin{aligned} &q_i \neq q_j \quad \forall i,j \quad q_{j+1} = q_j^* \quad if \quad \mathrm{Im}\{q_j\} \neq 0 \\ &\mathrm{Re}\{q_j\} < 0 \end{aligned} \tag{2.17}$$

When we make use of these functions, the basis-samples matrices involved in the least square problem become Cauchy matrices, with $\mathbf{\Phi}_0$ defined as in (1.27) and

$$\mathbf{\Phi}_1 = \begin{pmatrix} \frac{1}{s_1 - q_1} & \frac{1}{s_1 - q_2} & \dots & \frac{1}{s_1 - q_n} \\ \frac{1}{s_2 - q_1} & \frac{1}{s_2 - q_2} & \dots & \frac{1}{s_2 - q_n} \\ \vdots & \vdots & & \vdots \\ \frac{1}{s_K - q_1} & \frac{1}{s_K - q_2} & \dots & \frac{1}{s_K - q_n} \end{pmatrix}, \tag{2.18}$$

This choice fits our goal particularly well, since it leads to a well-posed least squares system, is suitable for a state space representation conversion of the model and, moreover, builds it as the sum of nuclear terms that are easily handled by circuit simulation software. This is the most common basis choice for the basis used in the SK framework.
When a generalized basis function is used, the SK algorithm takes the name of Generalized Sanathanan-Koerner (GSK); the structure of the algorithm is very simple, since it reduces to the iterative solution of a standard least squares problems. The accuracy of the resulting model is often satisfactory and the convergence of the residuals is reached in a small number of iterations.

Algorithm 2.1 Generalized Sanathanan-Koerner iteration

Require: frequency data $\{(s_k, \breve{H})\}_{k=1}^{K}$, set of initial poles $\{q_j\}_{j=1}^{n}$, threshold ϵ
  set $\boldsymbol{x}_0 = \mathbf{0}$;
  build the matrices $\mathbf{\Phi}_0$,$\mathbf{\Phi}_1$,$\mathbf{\Psi}$;
  for $\nu = 1,2,...,\nu_{max}$ **do**
    build the weight matrix $\mathbf{M}_{\nu-1}$ and vector $\boldsymbol{b}$
    solve the LS problem $(\mathbf{M}_{\nu-1}\mathbf{\Psi})\boldsymbol{x}_\nu \approx \mathbf{M}_{\nu-1}\boldsymbol{b}$
    if $||\boldsymbol{x}_\nu - \boldsymbol{x}_{\nu-1}|| < \epsilon||\boldsymbol{x}_\nu||$ **then**
      break
    end if
  end for
  return estimated unknowns $\boldsymbol{x}_\nu$.

2.1 The Vector Fitting Algorithm

We now present the Vector Fitting (VF) algorithm [19].
Since its first appearence, VF has become the standard algorithm of choice for rational fitting problems. It has been possible due to its robust numerical properties and its simplicity, that grants accurate macromodels and direct control over the model poles.
We will derive the algorithm starting from its theoretical similarities with the already exposed GSK.

2.1.1 Change of Model Representation

Let us consider again the linearization of the rational fitting residuals making, this time, the hypothesis for which partial fractions are used as basis functions to formulate the model:

$$H(s;\boldsymbol{x}) = \frac{N(s;\boldsymbol{x})}{D(s;\boldsymbol{x})} = \frac{c_0 + \sum_{j=1}^{n} \frac{c_j}{s-q_j}}{1 + \sum_{j=1}^{n} \frac{d_j}{s-q_j}}. \tag{2.19}$$

With this definition we can rewrite the Levy's linearized residuals as follows:

$$e_k(\boldsymbol{x}) = D(s_k;\boldsymbol{x}) r_k(\boldsymbol{x}) = \left(1 + \sum_{j=1}^{n} \frac{d_j}{s-q_j}\right) \breve{H} - \left(c_0 + \sum_{j=1}^{n} \frac{c_j}{s-q_j}\right). \tag{2.20}$$

Since the model is the ratio of two rational functions sharing the same set of poles q_j, by rewriting numerator and denominator in pole-zero form, denoting with z_j the zeros of the numerator and with p_j the zeros of the denominator, we easily observe that the latter coincides with the poles of the final model

$$H(s;\boldsymbol{x}) = \frac{N(s;\boldsymbol{x})}{D(s;\boldsymbol{x})} = c_0 \frac{\prod_{j=1}^{n} \frac{s-z_j}{s-q_j}}{\prod_{j=1}^{n} \frac{s-p_j}{s-q_j}} = c_0 \frac{\prod_{j=1}^{n} s - z_j}{\prod_{j=1}^{n} s - p_j}, \tag{2.21}$$

and that the initial set of poles $\{q_j\}$ cancels out.
The idea behind the vector fitting algorithm is to solve the rational fitting problem, again, in an iterative way, updating at each iteration ν the model poles, by setting, at each iteration:

$$q_j^{\nu} = p_j^{\nu-1}. \tag{2.22}$$

The intuition consists in observing that if the relocation of the poles converges, then almost no difference will occur between the zeros and the poles of the denominator; this would lead to two different outstanding advantages: first, the denominator $D(s;\boldsymbol{x})$ would approach unity for all frequency, compensating implicitly the frequency weighting induced by the linearization [20]; second, we would obtain a set of model poles that coincides with the system's dominant poles, and at the end, also the related residues.

2.1.2 Model Poles Relocation

As already stated, Vector Fitting relies on the update of the model poles at each iteration; since those poles coincide with the zeros of the denominator, their calculation can be performed by finding the eigenvalues of the state space matrix associated to the inverse of the denominator.
Given the transfer function:

$$\xi(s) = d_0 + \sum_{j=1}^{n} \frac{d_j}{s-q_j} = d_0 \frac{\prod_{j=1}^{n}(s-p_j)}{\prod_{j=1}^{n}(s-q_j)} \tag{2.23}$$

suppose we are given with all the coefficients except p_j that we want to find.
We first derive the state-space form of the system (more on this in section 1.4.2):

$$\begin{cases} \boldsymbol{x} = \mathbf{A}\boldsymbol{x} + \mathbf{1}u \\ y = \boldsymbol{c}^T\boldsymbol{x} + d_0 u \end{cases} \tag{2.24}$$

where the notation means:

$$\mathbf{A} = diag\{q_1, ..., q_n\}, \quad \mathbf{1} = (1, ...,1)^T, \quad \boldsymbol{c} = (c_1, ..., c_n). \tag{2.25}$$

From the state space we can write the transfer function as:

$$\xi(s) = d_0 + \boldsymbol{c}^T(s\mathbb{I} - \mathbf{A})^{-1}\mathbf{1}; \tag{2.26}$$

in the same way we can exchange the roles of input and output, to obtain the transfer function $\xi(s)^{-1}$. We write the value of the input u in function of the output y and we obtain:

$$\begin{cases} \boldsymbol{x} = (\mathbf{A} - \mathbf{1}d_0^{-1}\boldsymbol{c}^T)\boldsymbol{x} + \mathbf{1}d_0^{-1}y \\ y = -d_0^{-1}\boldsymbol{c}^T\boldsymbol{x} + d_0^{-1}y \end{cases}, \tag{2.27}$$

Now we can easily obtain the zeros of $\xi(s)$ as the eigenvalues of the above state matrix:

$$\{p_j\} = \lambda(\mathbf{A} - \mathbf{1}d_0^{-1}\boldsymbol{c}^T). \tag{2.28}$$

This procedure represents the main difference between the GSK and the VF iteration.

2.1.3 The Vector Fitting Iteration

We now present the vector fitting algorithm in its most basic form.
The algorithm requires a set of frequency-domain samples $(s_k, \breve{H})$ and a set of *starting poles* $\{q_j^1 \in \mathbb{C}, j = 1, ..., n\}$ where the index 1 means that they are referred to the first iteration; at each iteration ν this set of poles will be refined.
The poles identify the partial fraction basis:

$$\varphi_0^\nu(s) = 1, \quad \varphi_j^\nu(s) = \frac{1}{s - q_j^\nu}, \quad j = 1, ..., n \tag{2.29}$$

by means of which we build the *vector fitting weighting function* $\xi(s)$:

$$\xi^\nu(s) = 1 + \sum_{j=1}^{n} \frac{d_j^\nu}{s - q_j^\nu}, \tag{2.30}$$

with coefficients d_j^ν still unknown.
By means of the VF weighting function we write the following approximation:

$$\xi^\nu(s_k)\breve{H}_k = \left(1 + \sum_{j=1}^{n} \frac{d_j^\nu}{s - q_j^\nu}\right) \breve{H}_k \approx c_0^\nu + \sum_{j=1}^{n} \frac{c_j^\nu}{s_k - q_j^\nu}. \tag{2.31}$$

As already stated, this expression is equivalent to the Levy's approximation when partial fractions are used instead of simple monomials.
Anyway, when partial fractions are used, there is no need to express the model as a ratio of two different functions: the right-hand side o f the above equation already ensures the possibility to obtain a rational expression for the model; for this reason, we refere to $\xi(s)$ as VF weighting function rather than as model denominator.
When for every sample $k = 1, ..., K$ we impose the above approximation, we are actually looking for two different transfer functions, belonging to the set defined by the current set of poles: one of them, the VF weighting function, filters the data to maximize their correlation with the other one, the right-hand side term. The optimal couple of transfer functions is found by means of the solution of a linear least square problem, which returns the coefficients of their partial fraction expansions; this LS system reads:

$$(\boldsymbol{\Phi}_0^\nu - \breve{\mathbf{H}}\boldsymbol{\Phi}_1^\nu)\boldsymbol{x}_\nu \approx \boldsymbol{b} \tag{2.32}$$

Where the terms are defined as in the formulation of the GSK iteration, with the difference that the basis matrices $\boldsymbol{\Phi}_1^\nu$ and $\boldsymbol{\Phi}_0^\nu$ are now iteration-dependent.
The solution provides us the set of coefficients:

$$\boldsymbol{x}_\nu = (c_0^\nu, ..., c_n^\nu, d_1^\nu, ..., d_n^\nu)^T \tag{2.33}$$

that identify the two transfer functions that we are looking for.
Once they are available, we can apply the inverse of the transformation induced by the VF weighting function:

$$\breve{H}_k \approx \frac{c_0^\nu + \sum_{j=1}^{n} \frac{c_j^\nu}{s_k - q_j^\nu}}{1 + \sum_{j=1}^{n} \frac{d_j^\nu}{s - q_j^\nu}} = c_0 \frac{\prod_{j=1}^{n} \frac{s - z_j}{s - q_j}}{\prod_{j=1}^{n} \frac{s - p_j}{s - q_j}} = c_0 \frac{\prod_{j=1}^{n} s - z_j}{\prod_{j=1}^{n} s - p_j} \tag{2.34}$$

and obtain the model that actually best fits the data.
As already stated, the poles of this model equal the zeros of the VF weighting function. The next step is the pole relocation, performed as described in the previous section; we set:

$$q_j^{\nu+1} = p_j^\nu, \quad j = 1, ..., n \tag{2.35}$$

defining the set of poles to be used in the successive iteration.

As the number of iterations increases, the poles relocation converges to a particular set of poles such that:

$$\{q_j^{\nu+1}\} \approx \{q_j^{\nu}\} \tag{2.36}$$

When this happens, the poles and the zeros of the weighting function are approximately equal and the function becomes unitary for all the frequencies:

$$\xi^{\nu}(s) = \prod_{j=1}^{n} \frac{s - q_j^{\nu+1}}{s - q_j^{\nu}} \approx 1. \tag{2.37}$$

We argue that the final set of poles coincides with the dominant poles of the system under modeling.
Since the weighting function must approach unity as the poles converge, the related residues d_j must approach zero as the algorithm converges, so that the norm of the vector:

$$\boldsymbol{d}^{\nu} = (d_1^{\nu}, ..., d_n^{\nu}), \tag{2.38}$$

can be used as an index of convergence.
When convergence is reached, a post-processing step is performed to find the residues and the direct coupling of our model: we enforce the least square condition:

$$\breve{H}_k \approx R_0 + \sum_{j=1}^{n} \frac{R_j}{s_k - p_j}, \quad k = 1, ..., K \tag{2.39}$$

to obtain the finale model.
Since both VF and GSK are based on the compensation of the frequency bias induced by the Levy's method, they are formally equivalent; the difference relies in the method that is used to compensate the bias: while GSK applies an explicit weighting to the residuals, VF performs an implicit compensation based upon the poles relocation procedure. The implicit compensation guarantees the avoidance of numerical issues that can be induced by the SK and GSK algorithms and makes the VF algorithm perform better [20].

Algorithm 2.2 Basic VF algorithm

Require: frequency data $\{(s_k, \breve{H})\}_{k=1}^{K}$, set of initial poles $\{q_j\}_{j=1}^{n}$
  for $\nu = 1,2, ..., \nu_{max}$ **do**
    solve the LS problem $(\boldsymbol{\Phi}_0^{\nu} - \breve{\mathbf{H}}\boldsymbol{\Phi}_1^{\nu})\boldsymbol{x}_{\nu} \approx \boldsymbol{b}$
    extract the weighting function coefficients d_j^{ν} and build $\xi(s)$;
    compute the zeros z_j^{ν} of $\xi(s)$ solving the eigenvalue problem (2.28)
    relocate the poles: $q_j^{\nu+1} = z_j^{\nu}$ for $j = 1, ..., n$
  end for
  set $p_j = q_j^{(\nu_{max}+1)}$ for $j = 1, ..., n$;
  compute R_j for $j = 1, ..., n$ by solving (2.39) in LS sense.
  return the model $H(s)$

2.1.4 Consistency of Vector Fitting

In this section, we take a closer look to the consistency properties of the Vector Fitting Algorithm.
Suppose that the samples $\breve{H}(s_k)$ that we are given with are representative of a rational transfer function of order n of the form:

$$\breve{H}(s) = \alpha \frac{\prod_{j=1}^{n}(s-\zeta_j)}{\prod_{j=1}^{n}(s-p_j)}; \tag{2.40}$$

at each iteration we impose the approximation:

$$\xi(s)\breve{H}(s) \approx Q(s), \tag{2.41}$$

where $\xi(s)$ is again the vector fitting weighting function and $Q(s)$ is the same as in the approximation we imposed in the previous sections; here we denote their pole-zero form as:

$$\begin{aligned} Q(s) &= c_0 \frac{\prod_{j=1}^{n} s - w_j}{\prod_{j=1}^{n} s - q_j}, \\ \xi(s) &= \frac{\prod_{j=1}^{n} s - z_j}{\prod_{j=1}^{n} s - q_j}, \end{aligned} \tag{2.42}$$

then, our approximation is equivalent to:

$$\alpha \frac{\prod_{j=1}^{n}(s-\zeta_j)}{\prod_{j=1}^{n}(s-p_j)} \frac{\prod_{j=1}^{n}(s-z_j)}{\prod_{j=1}^{n}(s-q_j)} \approx c_0 \frac{\prod_{j=1}^{n}(s-w_j)}{\prod_{j=1}^{n}(s-q_j)} \tag{2.43}$$

where z_j and w_j are the zeros that result from the residues identification.
If we are so lucky to choose the order of the model equal to the order of the underlying system, as we have supposed in this case, the the least squares solution will force the left-hand side to share the same singularities of the right-hand one; for this reason, the zeros of the vector fitting weighting function z_j will coincide with the true poles of the system $\breve{H}(s)$.
Also the zeros w_j will be found such that they fit the zeros ζ_j of the system.
These facts show that if we choose the right order for the system, theoretically, the vector fitting converges in one single iteration; of course, numerical errors and roundoffs thwart the convergence of the algorithm in any practical implementation.

2.1.5 Convergence of Vector Fitting

By now, we have supposed that the pole relocation procedure comes up with a stable set of poles that are the dominant poles of the modeled system; formally, we can state that VF performs the following relocation:

$$\boldsymbol{q}_{\nu+1} = \mathbf{F}(\boldsymbol{q}_\nu), \tag{2.44}$$

where $\mathbf{F} : \mathbb{R}^n \to \mathbb{R}^n$ is a nonlinear function that takes the current vector of poles and returns the new poles for the successive iteration; convergence is reached if:

$$\boldsymbol{q}_\infty = \mathbf{F}(\boldsymbol{q}_\infty), \tag{2.45}$$

that is, if a fixed point is found.
Denoting with $\mathbf{J}$ the jacobian of the function $\mathbf{F}$, we can state that convergence is reached if, given a set $\Omega \subset \mathbb{R}^n : \boldsymbol{q}_0, \boldsymbol{q}_\infty \in \Omega$, the following statement holds:

$$\lambda_{max}(\mathbf{J}(\boldsymbol{q})) < 1 \quad \forall \boldsymbol{q} \in \Omega, \tag{2.46}$$

where $\lambda_{max}()$ is the maximum eigenvalue, i.e. the spectral radius.
Further, if

$$\lambda_{max}(\mathbf{J}(\boldsymbol{q}_\infty)) < 1, \tag{2.47}$$

then the fixed point is attractive and the algorithm will converge to it if one of the iterates $\boldsymbol{q}_\nu$ is sufficiently close to $\boldsymbol{q}_\infty$; conversely, if the spectral radius of the Jacobian computed at a fixed point is larger than one, then convergence is never reached to that point.
If all the fixed points are repelling, convergence cannot be achieved.
In practice, VF converges in the large majority of the cases, and this granted its adoption in several application areas; unfortunately, sometimes the pole relocation does not converge to a fixed point [25]. For this reason, we can think that a proof of the VF convergence, that is not available by now, will never be given.
In the cases for which the Vector Fitting does not return a stable solution, often the cause of the failure can be found in the lack of one of the following preconditions, on which the good behavior of the algorithm relies:

1. a proper implementation;
2. a suitable order n for the system and an appropriate choice of the starting poles (more on this in the following);
3. a feasible underlying system, that admits a rational approximation;
4. a set of data samples sufficient to characterize the actual system.

In order to check the convergence of the algorithm during its run, a variety of different methods can be applied.
For example, we can perform the convergence check by means of the *Hausdorff distance* [3], a metric particularly suitable to measure the distance between two different sets, that in our case are the two sets of poles returned by the algorithm at the end of two successive iterations.
We denote with A and B two generic sets and with a and b two arbitrary elements belonging to them; then, we define the following distances:

- Point-to-set distance,
$$d(a, B) = \min\{|a - b|, b \in B\}, \tag{2.48}$$
is the minimum distance between a single element of B and a prescribed element of A.

- One-sided distance between sets,
$$dist(A, B) = \max\{d(a, B), a \in A\}. \tag{2.49}$$

- Hausdorff distance,
$$Dist(A, B) = \max\{dist(A, B), dist(B, A)\}. \tag{2.50}$$

By definition, the Hausdorff distance is insensitive to the ordering of the elements of the two sets, that, in our case, are complex numbers representing the poles.
By denoting with Q_ν and $Q_{\nu+1}$ the poles related to successive iterations, the Hausdorff distance:
$$Dist(Q_\nu, Q_{\nu+1}), \tag{2.51}$$
must approach zero as the number of iterations increases.
A criterion to stop the algorithm when a satisfying level of poles refinement is achieved can be written as:
$$Dist(Q_\nu, Q_{\nu+1}) < \epsilon \cdot \max\{|q| : q \in Q_\nu\}, \tag{2.52}$$
where the threshold ϵ is chosen apriori.
The above test can be too restrictive in the case for which some poles are associated with very small residues, since their unachieved convergence is irrelevant for the quality of the final model; further, the test is not an assurance of the model quality, since the order n of the model could not be suitable for the application even if convergence is reached.
Another criterion to check the convergence of VF is performed by checking the convergence of the weighting function to unity. One can sample the values of the weighting function at the given frequency samples s_k, collect the obtained values into a vector
$$\boldsymbol{\xi}_\nu = (\xi^\nu(s_1), \xi^\nu(s_2), ..., \xi^\nu(s_k))^T, \tag{2.53}$$
and compare it with the target unitary vector, until a prescribed accuracy is reached; for example one can stop the iterations when:
$$||\mathbf{1} - \boldsymbol{\xi}_\nu|| < \epsilon \tag{2.54}$$
where ϵ is the prescribed degree of accuracy and $\mathbf{1}$ is a vector of ones.
We close this section with a remark about the convergence of vector fitting in presence of noisy data.
During each VF iteration the weighting function $\xi(s)$ that multiplies the data samples is

unknown, but it always approaches unity at high frequencies by construction; conversely, the value that the function assumes at low frequencies depends on the relation between the current set of poles and the optimal set of poles for the next iteration.
Consider the circumstance for which a starting pole at low frequency must be relocated to high frequency: in this case, as stated in the consistency analysis, the function will embed a factor:

$$f(s) = \frac{s + q_j^{\nu+1}}{s + q_j^{\nu}}, \quad |q_j^{\nu+1}| \gg |q_j^{\nu}|, \tag{2.55}$$

this implies that as the frequency approaches zero, the gain of the function will increase due to the frequency distance between $q_j^{\nu+1}$ and q_j^{ν}, reaching values far larger than one; as a consequence, if a low frequency noise is present in the data, it will be magnified by the weighting function.
The result is a loss of accuracy in the poles relocation condition that reduces the speed of the convergence.
The opposite case, that is when we want to move a pole from high frequencies to a lower band, is less critical, since in this case the weighting function will attenuate the low frequency components of the noise while it will not enhance the high frequencies ones.
We see that the asymmetry induced by the noise in the performance of the VF is obviously not desirable.

2.2 Practical Implementation Issues

We now poin out some practical aspects of the implementation of the VF algorithm that must be taken into account to achieve a satisfying macromodel. Some of them can be extended also to the implementation of the GSK algorithm and to the PSK algorithm that will be presented in the next chapter.

2.2.1 Causality, Stability and Realness

As already stated, the Vector Fitting algorithm allows us to have a major control over the poles of the final model with respect to other rational fitting algorithms; this implies that the physical characteristics of the model (with the exception of passivity) can be easily handled during the fitting procedure, since they rely on the poles of the model [19].
Using the VF, we can achieve the following properties for the model $H(s)$:

- realness: the model impulse response is real; in the frequency domain the condition is equivalent to $H(s^*) = H^*(s)$
- stability and causality: both are achieved when the model poles p_j have negative real part.

With reference to the VF model structure:

$$\breve{H}(s)_k = R_0 + \sum_{j=1}^{n} \frac{R_j}{s_k - p_j}, \tag{2.56}$$

then realness is guaranteed when $R_0 \in \mathbb{R}$ and:

1. Real poles are related with real residues:

$$p_j \in \mathbb{R} \rightarrow R_j \in \mathbb{R}. \tag{2.57}$$

2. Complex poles appears in complex conjugate couples and so do the related residues:

$$p_j = p_j^{'} + p_j^{''} \in \mathbb{C}, \quad p_j^{''} \neq 0 \rightarrow p_i = p_j^* = p_j^{'} - \mathrm{j}p_j^{''} \in \boldsymbol{q}, R_i = R_j^*. \tag{2.58}$$

We can enforce these conditions from the first iteration, considering the poles set $\{q_j^1\}$ whose elements appear as real or complex conjugate pairs.
For each frequency point avalaible in the data, at the iteration ν (the index will be omitted in the following) we enforce the condition:

$$\begin{aligned} &\psi_k^T \boldsymbol{x} \approx \breve{H}_k, \\ &\psi_k^T = \begin{pmatrix} 1 & \frac{1}{s_k - q_1} \dots \frac{1}{s_k - q_n} & \frac{-\breve{H}_k}{s_k - q_1} \dots \frac{-\breve{H}_k}{s_k - q_n} \end{pmatrix} \\ &,\boldsymbol{x} = (c_0 \quad c_1 \dots c_n \quad d_1, \dots d_n)^T. \end{aligned} \tag{2.59}$$

For every complex conjugate pair $q_j = q_{j+1}^*$ we want to obtain residues $d_j, d_{j+1}, c_j, c_{j+1}$ that make both the VF weighting function and the rational model satisfy the realness condition; then they must satisfy the conditions:

$$\begin{aligned} d_j = d_j' + \mathrm{j}d_j'' \rightarrow d_{j+1} = d_j^*, \\ c_j = c_j' + \mathrm{j}c_j'' \rightarrow d_{j+1} = c_j^*. \end{aligned} \tag{2.60}$$

To enforce this, we observe that:

$$\begin{aligned} \frac{d_j}{s - q_j} + \frac{d_j^*}{s - q_j^*} &= \left(\frac{1}{s - q_j} + \frac{1}{s - q_j^*}\right) d_j' + \left(\frac{\mathrm{j}}{s - q_j} - \frac{\mathrm{j}}{s - q_j^*}\right) d_j''; \\ \frac{c_j}{s - q_j} + \frac{c_j^*}{s - q_j^*} &= \left(\frac{1}{s - q_j} + \frac{1}{s - q_j^*}\right) c_j' + \left(\frac{j}{s - q_j} - \frac{j}{s - q_j^*}\right) c_j''; \end{aligned} \tag{2.61}$$

in these new representations, d_j', d_j'', c_j', c_j'' are all real unknowns; thus, for every complex conjugate pair of poles we can perform the substitution:

$$\begin{aligned} &(c_j, \quad c_{j+1})^T \leftarrow \left(c_j', \quad c_j''\right)^T, \\ &(d_j, \quad d_{j+1})^T \leftarrow \left(d_j', \quad d_j''\right)^T, \\ &\left(\frac{1}{s - q_j}, \quad \frac{1}{s - q_j^*}\right) \leftarrow \left(\frac{1}{s - q_j} + \frac{1}{s - q_j^*}, \quad \frac{\mathrm{j}}{s - q_j} - \frac{\mathrm{j}}{s - q_j^*}\right), \\ &\left(\frac{\breve{H}_k}{s - q_j}, \quad \frac{\breve{H}_k}{s - q_j^*}\right) \leftarrow \left(\frac{\breve{H}_k}{s - q_j} + \frac{\breve{H}_k}{s - q_j^*}, \quad \frac{\mathrm{j}\breve{H}_k}{s - q_j} - \frac{\mathrm{j}\breve{H}_k}{s - q_j^*}\right) \end{aligned} \tag{2.62}$$

we obtain a modified least square system whose rows read:

$$\hat{\boldsymbol{\psi}}_k^T \hat{\boldsymbol{x}} \approx \breve{H}_k \tag{2.63}$$

with $\hat{\boldsymbol{x}}$ real-valued unknown vector.
The successive step consists in splitting the regressor matrix and the right hand-side term in their real and imaginary parts, to obtain a real-valued least square system; we define:

$$\begin{aligned} \hat{\boldsymbol{\psi}}_k^{'T} &= Re\{\hat{\boldsymbol{\psi}}_k^T\}; \\ \hat{\boldsymbol{\psi}}_k^{''T} &= Im\{\hat{\boldsymbol{\psi}}_k^T\}; \\ \breve{H}'_k &= Re\{\breve{H}_k\}; \\ \breve{H}''_k &= Im\{\breve{H}_k\}; \end{aligned} \tag{2.64}$$

so that:

$$\begin{pmatrix} \hat{\boldsymbol{\psi}}_k^{'T} \\ \hat{\boldsymbol{\psi}}_k^{''T} \end{pmatrix} \hat{\boldsymbol{x}} \approx \begin{pmatrix} \breve{H}'_k \\ \breve{H}''_k \end{pmatrix}. \tag{2.65}$$

Collecting all the conditions to be enforced in a unique system we obtain:

$$\hat{\Psi} \hat{\boldsymbol{x}} \approx \hat{\boldsymbol{b}}. \tag{2.66}$$

The resulting system is real-valued, and so will be the unknowns, guaranteeing the presence of complex conjugate residues and, consequently, of the realness of the final model $H(s)$.
The splitting of real and imaginary parts must be performed for both the VF and the GSK in practical implementations.
During the pole relocation procedure, the poles are computed by means of an eigenvalue problem that involves complex matrices; this means that the resulting eigenvalues do not necessarily appear as complex conjugate pairs.
To avoid this problem, a transformation of the state space realization of the weighting function $\xi(s)$ must be performed to guarantee that the matrices involved in the eigenvalue problem are real-valued.
In particular, we build the state space matrix $\mathbf{A}$ at a generic iteration as:

$$\mathbf{A} = blkdiag\{\boldsymbol{q}\} \tag{2.67}$$

to avoid the presence of complex terms we can exploit the fact that:

$$\frac{d_j}{s - q_j} + \frac{d_j^*}{s - q_j^*} = \begin{pmatrix} d'_j & d''_j \end{pmatrix} \left[s\mathbf{I} - \begin{pmatrix} q'_j & q''_j \\ -q''_j & q'_j \end{pmatrix} \right]^{-1} \begin{pmatrix} 2 \\ 0 \end{pmatrix}, \tag{2.68}$$

from which we see that, in order to guarantee real entries in the matrix involved in the eigenvalue problem, one can perform the substitutions:

$$\begin{pmatrix} q_j & 0 \\ 0 & q_j^* \end{pmatrix} \leftarrow \begin{pmatrix} q'_j & q''_j \\ -q''_j & q'_j \end{pmatrix}, \quad (d_j \quad d_j^*) \leftarrow (d'_j \quad d''_j), \quad \begin{pmatrix} 1 \\ 1 \end{pmatrix} \leftarrow \begin{pmatrix} 2 \\ 0 \end{pmatrix}. \tag{2.69}$$

This substitutions can be easily performed to the alredy defined state space system through the application of a similarity transformation; in fact, given a nonsingular matrix $\mathbf{T}$, a generic scalar transfer function admits both the state space representations:

$$H(s) = \begin{pmatrix} \mathbf{A} & \boldsymbol{b} \\ \boldsymbol{c} & d \end{pmatrix}; \quad H(s) = \begin{pmatrix} \mathbf{TAT}^{-1} & \mathbf{T}\boldsymbol{b} \\ \boldsymbol{c}\mathbf{T}^{-1} & d \end{pmatrix} \tag{2.70}$$

that preserves the eigenvalues of the transfer matrix; in our case, the transformation matrix can be written as:

$$\mathbf{T} = blkdiag\{\mathbf{T}_i\} \tag{2.71}$$

where the matrices $\mathbf{T}_i$ are defined as:

$$\begin{cases} \mathbf{T}_i = 1 \quad if \quad q_i \in \mathbb{R}, \\ \mathbf{T}_i = \begin{pmatrix} 1 & 1 \\ j & -j \end{pmatrix} \quad if \quad q_i = q_{i+1}^* \in \mathbb{C}. \end{cases} \tag{2.72}$$

This trasformation grants that the eigenvalue problem will return real or complex conjugate eigenvalues.
The above procedure ensures the realness of the final model; to obtain a stable model, a very simple euristic procedure is used: every time an unstable pole is returned by the poles relocation procedure, we flip its real part to make it stable. This operation is necessary especially during the first iterations of the VF.

2.2.2 Order Selection and Initialization

The quality of the model obtained through the VF algorithm heavily relies on the prescribed order n imposed by the user. Even if no criterion is still avalaible to determine in advance the most suitable order, it is clear that it must be able to catch the dynamic of the data; for this reason, a lower bound for the order can be given as twice the number of resonant peaks shown by the data samples; starting from this lower bound, one can perform the identification more than once, increasing the order of the model until a satisfying approximation is reached.
The initial set of poles is commonly chosen following a heuristic strategy: given a frequency band of interest $\omega \in [0, \omega_{max}]$, the starting poles are set as complex conjugate pairs with imaginary parts linearly distributed in the frequency interval; their real parts should be such that the poles are weakly attenuated, that is:

$$q_{j-1,j} = q'_j \pm \mathrm{j}q''_j, \quad q''_j = \frac{j\omega_{max}}{n}, \quad q'_j = -\theta q''_j, \quad j = 2,4,...,n \tag{2.73}$$

with $\theta = 0.01$ or less.
It has been observed that this kind of distribution reduces the required number of relocation iterations and leads to a well-conditioned least square problem.
The strategy is slightly modified if the dynamic range of the data is particularly high in a large frequency band $[\omega_{min}, \omega_{max}]$; in this case a logarithmic distribution is preferable. With reference to the previous distribution, the substitution:

$$\begin{aligned} & q_j'' = \exp\{\alpha_{min} + (j-2)\frac{(\alpha_{min} - \alpha_{max})}{n-2}\} \quad j = 2,4,...,n \\ & \alpha_{min} = \ln(\omega_{min}), \quad \omega_{min} > 0 \\ & \alpha_{max} = \ln(\omega_{max}) \end{aligned} \tag{2.74}$$

we obtain a more suitable poles distribution.
If no resonant peaks occur in the data, then a logarithmic distribution of real poles can be chosen:

$$q_j'' = \exp\{\alpha_{min} + (j-1)\frac{(\alpha_{min} - \alpha_{max})}{n-1}\} \quad j = 1,2,...,n \tag{2.75}$$

2.2.3 Relaxed Normalization

The convergence issues related to the presence of noise in the data can be attenuated by means of a *relaxation* of the high frequency normalization of $\xi(s)$. Indeed one can make the direct coupling term of the weighting function d_0^ν to be a variable automatically tuned by the LS algorithm through the iterations.
We redefine the weighting function to be:

$$\xi^\nu(s) = d_0^\nu + \sum_{j=1}^{n} \frac{d_j^\nu}{s - q_j^\nu}, \tag{2.76}$$

and the poles relocation condition:

$$\left(d_0^\nu + \sum_{j=1}^{n} \frac{d_j^\nu}{s - q_j^\nu} \right) \breve{H}_k \approx c_0^\nu + \sum_{j=1}^{n} \frac{c_j^\nu}{s_k - q_j^\nu}, \tag{2.77}$$

With a consequential change of the LS system:

$$(\boldsymbol{\Phi}_0^\nu - \breve{\mathbf{H}}\boldsymbol{\Phi}_0^\nu)\boldsymbol{x}_\nu \approx \mathbf{0}, \tag{2.78}$$

and redefinition of the unknowns vector:

$$\boldsymbol{x}_\nu = (c_0^\nu, ..., c_n^\nu, d_0^\nu, d_1^\nu, ..., d_n^\nu)^T. \tag{2.79}$$

Without additional constraints, the LS will of course return a trivial all-zero solution.

Our aim is to make this solution infeasible, which can be achieved by requiring that the weigthing function $\xi(s)$ must not vanish. Among the possible strategies, the following has been observed to perform best: we impose the sum of the real parts of the weighting function over the avalaible data samples to be a nonzero constant, while preserving its value at unity when the algorithm converges. The constraint that we need is then:

$$\frac{1}{K}\sum_{k=1}^{K} Re\left\{ d_0^{\nu} + \sum_{j=1}^{n} \frac{d_j^{\nu}}{s_k - q_j^{\nu}} \right\} = 1. \tag{2.80}$$

We can impose this condition as an additional row of the LS problem.
It is a good idea to weight this row with a factor based on the magnitude of the data samples; a common choice is the energy of the data vector:

$$\alpha = \sqrt{\sum_{k=1}^{K} \left| \breve{H}_k \right|^2}, \tag{2.81}$$

then, defining:

$$\boldsymbol{v}_{\nu}^{T} = (1, v_{\nu,1} \quad \dots \quad v_{\nu,n}), \qquad v_{\nu,j} = \frac{1}{K}\sum_{k=1}^{K} Re\left\{ \frac{1}{s_k - q_j^{\nu}} \right\}, \tag{2.82}$$

we obtain the formulation:

$$\begin{pmatrix} \boldsymbol{\Phi}_0^{\nu} & -\breve{\mathbf{H}}\boldsymbol{\Phi}_0^{\nu} \\ \mathbf{0} & \alpha \boldsymbol{v}_{\nu}^{T} \end{pmatrix} \boldsymbol{x}_{\nu} \approx \begin{pmatrix} \mathbf{0} \\ \alpha \end{pmatrix}. \tag{2.83}$$

The resulting scheme is referred as *Relaxed Vector Fitting* [18].
Sometimes, it could be necessary to fix a lower and an upper bounds for the values of d_0^{ν}: after each iteration, if the numerical value of this unknown is unmanageable, the iteration can be repeated as a standard VF iteration with d_0^{ν} equal to the bound instead to one. The same procedure can be applied to a GSK scheme to improve its numerical robustness as well.

Chapter 3

Fast Vector Fitting Algorithm for Multiport Systems

3.1 QR Factorization

The main results of the thesis project are the improvements of the performances of the PSK algorithm, both in terms of computational cost and of memory requirements; the result has been reached by means of the QR factorization. While most applications of this factorization regard orthogonalization, eigenvalues calculation, and solution of linear systems, we exploit its properties to carry out the decoupling of a least squares system whose regressors matrix exhibits a bordered block diagonal structure.
By bordered block diagonal we refer to a standard rectangular block diagonal matrix with the addition of a set of full columns on the right. The presence of these full columns entails a dependency between the optimal values of their related unknowns and the ones related to each diagonal block.
In the following we report a brief review of the theory behind the QR factorization and a practical implementation of its calculation when the matrices to be factorized share a subset of sequential columns.

3.1.1 Reduced Formulation of the QR Factorization

Given a rectangular matrix, the idea behind the QR factorization is to find an orthonormal set of sequential vectors that is able to span the same column spaces of A, and from which one can exactly reconstruct the starting matrix by means of linear combinations of the new orthonormal basis.
Formally speaking, given a matrix $\mathbf{A} \in \mathbb{C}^{m \times n}, m \geq n$ with full rank n, there exist two unique matrices, $\mathbf{Q} \in \mathbb{C}^{m \times n}$ and $\mathbf{R} \in \mathbb{C}^{n \times n}$ upper triangular such that:

$$\mathbf{A} = \mathbf{QR} \tag{3.1}$$

$$\mathbf{Q}^H \mathbf{Q} = \mathbf{I}_n \tag{3.2}$$

The following relation holds between the columns of A and the columns of Q:

$$span\left(\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_j\right) = span\left(\boldsymbol{a}_1, \boldsymbol{a}_2, \ldots, \boldsymbol{a}_j\right), \qquad j = 1,2,\ldots,n \tag{3.3}$$

Due to the upper triangular structure of R, the k-th column of A can written as a linear combination of the first k columns of Q by means of the non-zero coefficients of the k-th column of R; in formulas:

$$\boldsymbol{a}_1 = r_{11}\boldsymbol{q}_1 \tag{3.4}$$
$$\boldsymbol{a}_2 = r_{12}\boldsymbol{q}_1 + r_{22}\boldsymbol{q}_2 \tag{3.5}$$
$$\boldsymbol{a}_3 = r_{13}\boldsymbol{q}_1 + r_{23}\boldsymbol{q}_2 + r_{33}\boldsymbol{q}_3 \tag{3.6}$$
$$\vdots \tag{3.7}$$
$$\boldsymbol{a}_n = r_{1n}\boldsymbol{q}_1 + r_{2n}\boldsymbol{q}_2 + \ldots + r_{nn}\boldsymbol{q}_n \tag{3.8}$$

The most efficient algorithm for the computation of the QR decomposition, is based upon the Householder reflectors and requires $2n^2m - \frac{2}{3}n^3$ flops.

3.1.2 Full QR Factorization

The full QR factorization of $\mathbf{A} \in \mathbb{C}^{m\times n}, m \geq n$ adds a set of $m-n$ orthonormal columns to the already defined matrix $\mathbf{Q}$, trasforming it in a square unitary matrix; subsequently a matrix block of zeros is appended below the triangular matrix $\mathbf{R}$, that now becomes a $n \times m$ matrix. The new columns added to $\mathbf{Q}$ are all orthogonal to the range space of A and are then a basis for its null space.

3.1.3 Fast R Computation for Block-Sharing Matrices

Suppose we want to compute the $\mathbf{R}_n$ factors of a sequence of matrices of the following form:

$$\begin{aligned} \mathbf{A}_1 &= \left(\mathbf{B}, \mathbf{C}_1\right) \\ \mathbf{A}_2 &= \left(\mathbf{B}, \mathbf{C}_2\right) \\ &\vdots \\ \mathbf{A}_n &= \left(\mathbf{B}, \mathbf{C}_n\right) \end{aligned} \tag{3.9}$$

Instead of calculating all the $\mathbf{R}_n$ matrices from scratch, we can save a good number of computations in the following way. First we compute the QR factorization of $\mathbf{B}$ alone:

$$\mathbf{B} = \mathbf{Q}_B\mathbf{R}_B. \tag{3.10}$$

Then we perform a projection of every $\mathbf{C}_i$ matrix to the basis of $\mathbf{Q}_B$:

$$\hat{\mathbf{C}}_i = \mathbf{Q}_B^H\mathbf{C}_i. \tag{3.11}$$

At this point we perform the QR decomposition of the matrices:

$$\left(\mathbf{C}_i - \mathbf{Q}_B\hat{\mathbf{C}}_i\right) = \mathbf{Q}_{Pi}\mathbf{R}_{Pi}. \tag{3.12}$$

And we build the factors $\mathbf{R}_i$, relative to each $\mathbf{A}_i$ matrix, as follows

$$\mathbf{R}_i = \begin{pmatrix} \mathbf{R}_B & \hat{\mathbf{C}}_i \\ \mathbf{0} & \mathbf{R}_{Pi}, \end{pmatrix} \tag{3.13}$$

where $\mathbf{0}$ is a matrix of zeros with the same dimensions of $\mathbf{R}_{Pi}$.
The advantage of this method is that we can compute n times the QR factorization of the matrices $\mathbf{C}_i$, that are smaller than $\mathbf{A}_i$. Intermediate operations of matrix multiplication and matrix subtraction are still necessary.
With the hypothesis that the matrix blocks $\mathbf{B}$ and $\mathbf{C}_i$ are both of the same dimension $m \times n$, then the number of flops required to compute a single $\mathbf{R}_i$ is:

$$Fl_{QR_{Block}} = 2mn^2 - \frac{2}{3}n^3 + mn^2 + mn \tag{3.14}$$

3.2 Vector Fitting Improvement through QR Factorization

We now present the multiport formulation for the VF algorithm for the univariate model case. We do this in order to present the algebraic compression procedure that we will apply also to the PSK algorithm. Our main hypothesis is that all the responses of the transfer matrix share the same set of poles. This ensures that the pole relocation procedure of the vector fitting remains unchanged when we want to extend the algorithm formulation to the multiport case. Here we do not hightlight the fact that the poles are iteration dependent, so we drop the superscript ν. With this starting point we can write the model structure[1] [16]:

$$\mathbf{H}(s) = \mathbf{R}_0 + \sum_{l=1}^{n} \frac{\mathbf{R}_l}{s - p_l}, \tag{3.15}$$

where $\mathbf{R}_i \in \mathbb{C}^{P \times P}$, P is the number of ports of the system and p_l are the poles obtained by the poles relocation procedure.
The two steps of the vector fitting algorithm,*i.e.* the pole relocation and the residue identification, require the iterative solution of a least square problem whose dimension is determined by the model order, the number of frequency samples and the number of ports of the system.
To obtain the formulation of the LS problem, we recast the model structure into the form:

$$\mathbf{H}(s) = \frac{\mathbf{C}_0 + \sum_{l=1}^{n} \mathbf{C}_l \varphi_l(s)}{1 + \sum_{l=1}^{n} d_l \varphi_l(s)} \tag{3.16}$$

where $\mathbf{C}_l \in \mathbb{C}^{P \times P}$ and $\varphi_l(s)$ is in our case the l-th partial fraction. the model parameters are collected in one unknown vector:

$$\boldsymbol{x} = (\boldsymbol{c^1}; \boldsymbol{c^2}; \ldots; \boldsymbol{c^{P^2}}; \boldsymbol{d}) \tag{3.17}$$

[1] We assume without loss of generality that the number of input ports equals the number of output ports.

In this representation, every $\boldsymbol{c^i}$ vector stacks in column the entries of the matrices $\mathbf{C}_l$ with the same indices, while $\boldsymbol{d}$ embeds the parameters d_l.
Moreover, suppose that for the i-th response we are given K data samples, to be used to derive the model; then we collect these data in the vector:

$$\breve{\boldsymbol{h}}^i = \left(\breve{h}_1^i, \breve{h}_2^i, \ldots, \breve{h}_K^i\right)^T \tag{3.18}$$

With this notation, for each one of the P^2 transfer functions to be identified, we must enforce the linearized condition

$$\begin{gathered}\left(c_0^i + \sum_{l=1}^{n} \frac{c_l^i}{s_k - q_l}\right) - \left(1 + \sum_{l=1}^{n} \frac{d_l}{s_k - q_l}\right) \breve{h}_k^i \approx 0 \\ for \quad i = 1,2,\ldots,P^2 \quad , \quad k = 1,2,\ldots,K\end{gathered} \tag{3.19}$$

in least squares sense.
We notice that no theoretical difference occurs between the approximation above and the one we exposed in 2.31: the only distinction is that now the approximation must be imposed for every port response. With this observation it is easy to derive the structure of the LS problem associated to the computation of $\boldsymbol{x}$. By defining

$$\breve{\mathbf{H}}_i = diag\{\breve{\boldsymbol{h}}^i\}, \tag{3.20}$$

and with reference to $\boldsymbol{\Phi}_0$ and $\boldsymbol{\Phi}_1$ defined as in 1.27 and 2.18 respectively, the LS problem related to the fitting of a single response reads:

$$\begin{pmatrix}\boldsymbol{\Phi}_0 & -\breve{\mathbf{H}}_i\boldsymbol{\Phi}_1\end{pmatrix}\begin{pmatrix}\boldsymbol{c}^i \\ \boldsymbol{d}\end{pmatrix} \approx \breve{\boldsymbol{h}}^i. \tag{3.21}$$

When we are asked to identify all the P^2 responses with a common denominator, the global linear system for the multiport becomes:

$$\begin{pmatrix}
\boldsymbol{\Phi}_0 & \mathbf{0} & \mathbf{0} & \ldots & \mathbf{0} & -\breve{\mathbf{H}}_1\boldsymbol{\Phi}_1 \\
\mathbf{0} & \boldsymbol{\Phi}_0 & \mathbf{0} & \ldots & \mathbf{0} & -\breve{\mathbf{H}}_2\boldsymbol{\Phi}_1 \\
\mathbf{0} & \mathbf{0} & \boldsymbol{\Phi}_0 & \ldots & \mathbf{0} & -\breve{\mathbf{H}}_3\boldsymbol{\Phi}_1 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\mathbf{0} & \mathbf{0} & \mathbf{0} & \ldots & \boldsymbol{\Phi}_0 & -\breve{\mathbf{H}}_{P^2}\boldsymbol{\Phi}_1
\end{pmatrix}
\begin{pmatrix}
\boldsymbol{c}^1 \\ \boldsymbol{c}^2 \\ \boldsymbol{c}^3 \\ \vdots \\ \boldsymbol{c}^{P^2} \\ \boldsymbol{d}
\end{pmatrix}
\approx
\begin{pmatrix}
\breve{\boldsymbol{h}}^1 \\ \breve{\boldsymbol{h}}^2 \\ \breve{\boldsymbol{h}}^3 \\ \vdots \\ \breve{\boldsymbol{h}}^{P^2}
\end{pmatrix} \tag{3.22}$$

Notice that during the pole relocation phase, only the unknowns $\boldsymbol{d}$ are required at each iteration; anyway, since the systems are coupled, one must calculate all the unknowns at each iteration. This fact leads to an unnecessary waste in terms of memory and computations.

3.2.1 Least Square System Decoupling for Multiports

The system above can reach unmanageable dimensions when the number of ports or the order are very large; in fact, the dimensions of the regression matrix are $KP^2 \times$

$\big((n+1)\,P^2+n\big)$. Fortunately, we can notice that the non-zero blocks of the system are coupled only by the presence of the last columns; the idea, then, is to solve the problem only for the denominator's coefficients during the poles relocation step, being able to solve subsequently P^2 decoupled systems to find the remainder unknowns when we are ready to compute the residues. A suitable way to decouple the system can be found exploiting the properties of the economy size QR decomposition. The result is the so called "Fast Vector Fitting" (FVF) algorithm , [7], [16]. With this perspective, consider the LS matrix relative to a single decoupled response, and apply the QR factorization to it; the operation leads to

$$\Big(\boldsymbol{\Phi}_1 - \breve{\mathbf{H}}_i\boldsymbol{\Phi}_0\Big) = \mathbf{Q}_i\mathbf{R}_i = \mathbf{Q}_i\begin{pmatrix}\mathbf{R}_i^{11} & \mathbf{R}_i^{12}\\ \mathbf{0} & \mathbf{R}_i^{22}\end{pmatrix} \tag{3.23}$$

with $\mathbf{Q}_i \in \mathbb{C}^{K\times(2n+1)}$,$\mathbf{R}_i \in \mathbb{C}^{(2n+1)\times(2n+1)}$. Once $\mathbf{Q}_i$ has been computed, we can pre-multiply both sides of single port least square system by $\mathbf{Q}_i^H$ to obtain:

$$\begin{pmatrix}\mathbf{R}_i^{11} & \mathbf{R}_i^{12}\\ \mathbf{0} & \mathbf{R}_i^{22}\end{pmatrix}\begin{pmatrix}\boldsymbol{c}^i\\ \boldsymbol{d}\end{pmatrix} \approx \mathbf{Q}_i^H\breve{\boldsymbol{h}}^i = \begin{pmatrix}\boldsymbol{b}_1^i\\ \boldsymbol{b}_2^i\end{pmatrix}; \tag{3.24}$$

the last notation of the right hand side of the system emphasizes the fact that, after the projection, the reconstruction of the second half of the known terms depends only on the unknowns $\boldsymbol{d}$.
Since we are given with P^2 responses that share the same denominator, we apply the same procedure to each response individually, to collect all the left and right hand sides that we can exploit to find the best $\boldsymbol{d}$ vector in a least square sense. Defining:

$$\begin{pmatrix}\boldsymbol{b}_2^1\\ \boldsymbol{b}_2^2\\ \vdots\\ \boldsymbol{b}_2^{P^2}\end{pmatrix} = \boldsymbol{b}_2, \tag{3.25}$$

The resulting system is:

$$\begin{pmatrix}\mathbf{R}_1^{22}\\ \mathbf{R}_2^{22}\\ \vdots\\ \mathbf{R}_{P^2}^{22}\end{pmatrix}\boldsymbol{d} \approx \boldsymbol{b}_2. \tag{3.26}$$

The solution of this problem at each pole relocation step provides us the required $\boldsymbol{d}$ vector.

The advantage of this formulation is outstanding if we consider that the dimension of the new problem is $P^2n \times n$. In particular, the dependency of the regressor's dimension relies no more on the number of frequency samples K, which usually is far bigger than the number of poles of the denominator n.
Once the poles relocation step has been executed, the residues can be computed; we collect again the $n+1$ residues associated to each entry of the transfer matrix into the unknown residue vector $\boldsymbol{r}^i$, whose determination could be performed indipendently for each response; anyway, being the poles associated to the residues identical for all the

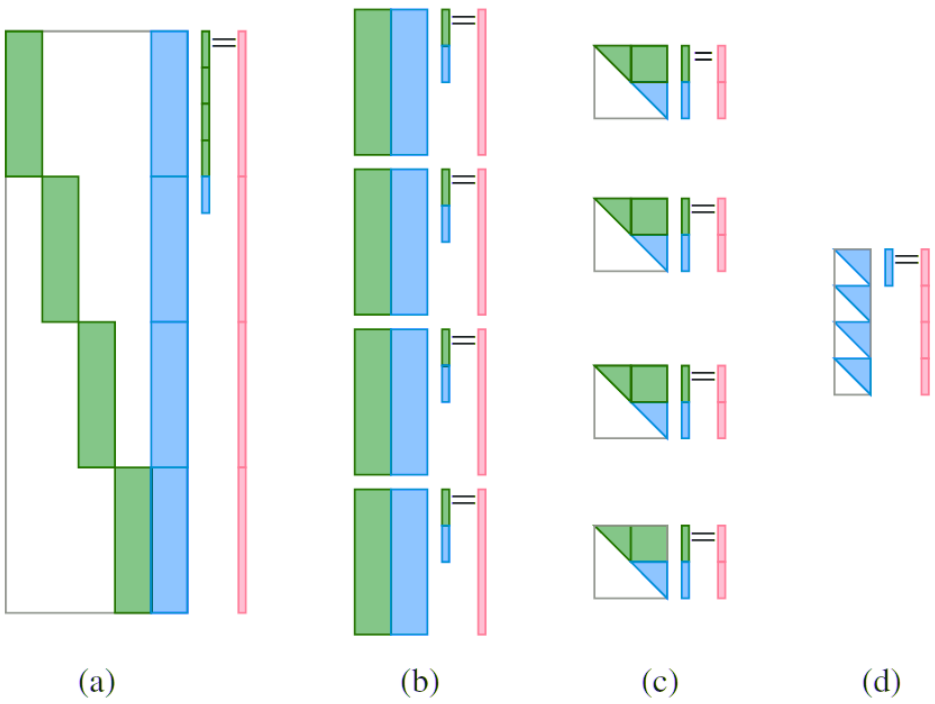


Figure 3.1: (a) The VF system for $P^2 = 4$. (b) Single responses systems. (c) Relative QR decompositions. (d) Compressed system for denominator's coefficients. ©2011

transfer functions, a more advantageous strategy is to collect the unknowns and the data samples in two matrices:

$$\mathbf{X} = \left(\boldsymbol{r}^1, \boldsymbol{r}^2, \ldots, \boldsymbol{r}^{P^2}\right), \qquad \mathbf{B} = \left(\breve{\boldsymbol{h}}^1, \breve{\boldsymbol{h}}^2, \ldots, \breve{\boldsymbol{h}}^{P^2}\right) \tag{3.27}$$

and solve the system:

$$\boldsymbol{\Phi}_0 \mathbf{X} \approx \mathbf{B} \tag{3.28}$$

To obtain all the residues. A QR factorization of $\boldsymbol{\Phi}_0$ can also be exploited to solve the system in an efficient way.

3.2.2 Vector Fitting Computational Complexity Reduction

With the theoretical background of the FVF at hand, the computational complexity reduction of the algorithm can be performed [7].
We first list the variables that will be involved in the calculations, together with their meaning:

- m: the number of rows of a matrix of interest.
- n: the number of columns of a matrix of interest

- p: the number of right-hand sides in a least square problem
- K: the number of frequency data samples (for simplicity equal for each response)
- H: the number of VF iterations required to relocate the poles
- L: the number of responses to be identified
- N: the number of poles of the model

Then, a computation of the flops required by the principal algebraic operations involved in the VF algorithm follows:

- QR Factorization:

$$Fl_{QR}(m,n) = \begin{cases} 2m^2n - \frac{2}{3}m^3 & \text{if } n \geqslant m \\ 2n^2m - \frac{2}{3}n^3 & \text{if } n < m \end{cases} \tag{3.29}$$

- Eigenvalue calculation for poles relocation:
  even if a precise number of flops cannot be computed due to the iterative nature of the solvers, a good estimate is:

$$Fl_{EIG}(m) \sim 10m^3 \tag{3.30}$$

- Multiple RHS least square problem:
  Given $\mathbf{A} \in \mathbb{R}^{m\times n}, \mathbf{B} \in \mathbb{R}^{m\times p}, \boldsymbol{x} \in \mathbb{R}^{n\times p}$, the solution is computed through three different steps:

  1. QR factorization of A:

$$\mathbf{A} = \mathbf{QR} \longrightarrow Fl_{QR}(m,n) \tag{3.31}$$

  2. Projection of $\mathbf{B}$:

$$\hat{\mathbf{B}} = \mathbf{Q}^H\mathbf{B} \longrightarrow Fl_{Proj}(m,n,p) = 2mnp \quad flops \tag{3.32}$$

  3. Solution of triangular systems:

$$\mathbf{RX} = \hat{\mathbf{B}} \longrightarrow n^2p \quad flops \tag{3.33}$$

  That require in total:

$$Fl_{MLS}(m,n,p) = Fl_{QR}(m,n) + 2mnp + n^2p \tag{3.34}$$

Now we have all the ingredients to compute an estimate of the flops required by the classical and the fast versions of the VF algorithm.
For the standard formulation we have:

$$\begin{aligned} Fl_{VF}(L,K,N,H) = {} & H \times Fl_{MLS}(2KL, ((N+1)L+N), 1) \\ & + H \times Fl_{EIG}(N) \\ & + Fl_{MLS}(2K, N+1, L), \end{aligned} \tag{3.35}$$

while for the fast formulation:

$$\begin{aligned} Fl_{FVF}(L,K,N,H) &= H \times L \times Fl_{QR}(2K,2N+1) \\ &+ H \times L \times Fl_{Proj}(2K,2N+1,1) \\ &+ H \times Fl_{MLS}((N+1)L+1,N+1,1) \\ &+ H \times Fl_{EIG}(N) \\ &+ Fl_{MLS}(2K,N+1,L) \end{aligned} \tag{3.36}$$

again we used $2K$ instead of K for the complex nature of the data involved in the calculations.
To derive an expression for the computational complexity of the two versions of the algorithm, we take advantage of the assumption $K \gg N$, which is obvious for any least square problem; additionally we know that the number of flops required by the QR factorization dominates over the other two terms involved in the solution of a multiple right-hand side least square problem, that is:

$$Fl_{MLS} \sim Fl_{QR} \tag{3.37}$$

This means that, asymptotically, the complexity of VF and of FVF converge to the complexity of the first terms of the upper equations; for the standard VF:

$$Fl_{VF} \simeq H \times Fl_{MLS}(2KL,((N+1)L+N),1) \propto HKL^3N^2, \tag{3.38}$$

for the fast vector fitting:

$$Fl_{FVF} \simeq H \times L \times Fl_{QR}(2K,2N+1) \propto HKLN^2. \tag{3.39}$$

The cubic dependency of the complexity with respect to the number of responses to be identified has disappeared after the compression of the least square system through the QR factorization; now the algorithm scalsaes linearly with the number of responses.
An additional flops saving is achieved by the FVF when it is formulated in the relaxed version; in such a case, the calculation of $\mathbf{R}_i$ can be performed faster as indicated in 1.1.3. Furthermore there is no need to transform the data samples since the right hand side of the decoupled systems is identically zero: the term $Fl_{Proj}(2K,2N+1,1)$ disappears in the computation of the flops and there is no need to store $\mathbf{Q}_i$.

3.2.3 Vector Fitting Memory Requirements Reduction

We now compute the memory requirements reduction induced by the fast formulation of the Vector Fitting.
Since the residues identification step shows no difference in terms of memory requirements for the two formulations, we will focus on the poles relocation procedure. For the classical formulation the storage of the full matrix

$$\begin{pmatrix} \mathbf{\Phi}_0 & \mathbf{0} & \mathbf{0} & ... & \mathbf{0} & -\breve{\mathbf{H}}_1\mathbf{\Phi}_1 \\ \mathbf{0} & \mathbf{\Phi}_0 & \mathbf{0} & ... & \mathbf{0} & -\breve{\mathbf{H}}_2\mathbf{\Phi}_1 \\ \mathbf{0} & \mathbf{0} & \mathbf{\Phi}_0 & ... & \mathbf{0} & -\breve{\mathbf{H}}_3\mathbf{\Phi}_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & ... & \mathbf{\Phi}_0 & -\breve{\mathbf{H}}_{P^2}\mathbf{\Phi}_1 \end{pmatrix}, \tag{3.40}$$

is required at every iteration of the poles relocation; as already stated, the matrix $\mathbf{A} \in \mathbb{R}^{KL\times((N+1)L+N)}$. If we consider that one must split the rows into their real and imaginary parts,the number of rows is multiplied by two; this leads to necessity to store the total number of elements:

$$E_{VF} = 2KL\left((N+1)L+N\right). \tag{3.41}$$

If the relaxed VF version is implemented, then a row and a column are added to the matrix $\mathbf{A}$ and:

$$E_{VF_{rel}} = 2\left(N+1\right)\left(L+1\right)\left(KL+1\right) \tag{3.42}$$

elements are required.
In the case of the fast VF, the matrix:

$$\mathbf{R_s} = \begin{pmatrix} \mathbf{R}_1^{22} \\ \mathbf{R}_2^{22} \\ \vdots \\ \mathbf{R}_L^{22} \end{pmatrix} \tag{3.43}$$

is sufficient to solve the least square problem of the pole relocation; since $\mathbf{R_s} \in \mathbb{C}^{LN\times N}$, the total number of elements to be store in the pole relocation phase is:

$$E_{FVF} = LN^2, \tag{3.44}$$

while, if a relaxed system is required, an additional unknown is to be found and the square matrix blocks dimensions increase by one, leading to:

$$E_{FVF_{rel}} = L\left(N+1\right)^2, \tag{3.45}$$

we can now compute the least squares compression factors for both the cases during the pole relocation phase (LSCF):

$$\begin{aligned} LSCF_{VF} &= \frac{E_{VF}}{E_{FVF}} = \frac{2KL\left((N+1)L+N\right)}{LN^2} \simeq \frac{2K\left(L+1\right)}{N} \\ LSCF_{VF_{rel}} &= \frac{E_{VF_{rel}}}{E_{FVF_{rel}}} = \frac{2\left(N+1\right)\left(L+1\right)\left(KL+1\right)}{L\left(N+1\right)^2} \simeq \frac{2K\left(L+1\right)}{N} \end{aligned} \tag{3.46}$$

approximately equal as one can expect.
Notice that to build the matrix $\mathbf{R_s}$, one must perform, for every response, the factorization:

$$\mathbf{G}_i = \left(\mathbf{\Phi}_1 - \breve{\mathbf{H}}_i\mathbf{\Phi}_0\right) = \mathbf{Q}_i\mathbf{R}_i, \tag{3.47}$$

so one must keep in memory, at least until the necessary computations are performed, the matrices required by the particular implementation of the algorithm. Anyway, whatever the implementation is, the intermediate matrices to be stored to compute the compressed system do not rely their dimensions on L: the number of ports influences the memory requirements only through $\mathbf{R_s}$, and does so linearly rather than quadratically.

Chapter 4

Multivariate Macromodels

This Chapter is co-authored by T.Bradde, M. De Stefano and A. Zanco.

In the previous chapter, we assumed that the system under modeling is characterized by a fixed (yet unknown) physical structure. In many situations, however, this hypothesis is not the most suitable: some physical parameters of the system could be design objectives or could be intrinsically uncertain due to production process tolerances. A parametric macromodel is able to reproduce the system behaviour for all the possible values that the varying parameters assume within a prescribed range. This possibility proves to be extremely useful in many fields of the design process, from the optimization of the design variables, to the simulation of worst-case scenarios induced by the physical realization of the structure. Typical examples regard the role of temperature in electronic devices, the geometrical parameters of an interconnect, the linearization point of a non-linear device, and many more.
The construction flow of a parametric macromodel requires the knowledge of the input-output behavior for a discrete number of values within the range that the parameters can span; once those data are collected and processed, the interpolation algorithm returns a closed form description of the system within the entire range of variation.
In this case, the input-output data must be representative of the model behavior within all the range of values assumed by each parameter; in particular, consider the case in which the model is required to depend on a number ρ of design parameters. Then, for the i-th parameter we can denote its variation range as

$$\Theta_i = \left[\theta^i_{min}, \theta^i_{max}\right] \quad for \quad i = 1,2,..,\rho \quad . \tag{4.1}$$

Thus, the global parameter domain can be defined as:

$$\Theta = \Theta_1 \times \Theta_2 \times ..., \times \Theta_\rho \quad \subseteq \mathbb{R}^\rho. \tag{4.2}$$

A point in Θ is uniquely identified by its projections along the parameters axes. To keep the notation compact, this point is denoted as

$$\vartheta_{\mathbf{m}} = (\vartheta_{m_1}, \ldots, \vartheta_{m_\rho})^T \tag{4.3}$$

where $\mathbf{m}$ is a multi-index $\mathbf{m} = [m_1, \ldots, m_\rho]$.
To synthetize a parametric macromodel, a set of M points in the parameter domain Θ is

defined to be representative of the parametric system response; for each of these points, we collect K frequency samples of the transfer functions associated with the underlying system. The resulting dataset reads:

$$\breve{\mathbf{H}}_{k,m} = \breve{\mathbf{H}}(s_k, \theta_m) \quad for \quad k = 1,2,...,K \quad m = 1,2,...,M \quad , \tag{4.4}$$

If, as it is common, we collect data at real frequencies ω_k, our goal is to obtain a model:

$$\mathbf{H}(\mathrm{j}\omega, \theta) \approx \breve{\mathbf{H}}(\mathrm{j}\omega, \theta) \quad for \quad \theta \in \Theta, \quad \omega \in [\omega_{min}, \omega_{max}] \tag{4.5}$$

While the structure of an univariate model is supposed to be a rational function of the Laplace variable, we are free to cast the dependence of the model on the parameters in a larger set of possible structures: a variety of basis functions can be used to fit the data. The thesis project is particularly focused on the investigation of issues related to the construction of precise and reliable parametric macromodels, for which many open problems still exist.

4.1 Parametric Model Formulation

Approximating the true system response $\breve{\mathbf{H}}(s_k, \vartheta_k)$ in a suitable macromodel form is fundamental to include the curve fitting result in system-level simulations using standard circuit solver such as SPICE. Several mathematical structures are available: the identification algorithm efficiency, in frequency and time domain, is affected by this choice. Moreover, all the formulations may suffer from ill-conditioning depending on the parameter-dependent basis choice.
Therefore, considering a P-ports multivariate macromodel of a generic LTI system, we adopt the so-called *Parameterized Sanathanan-Koerner (PSK)* [38], [37], [10], [9], [17] form

$$\mathbf{H}(s;\vartheta) = \frac{\mathbf{N}(s,\vartheta)}{\mathsf{D}(s,\vartheta)} = \frac{\sum_{n=0}^{\bar{n}} \sum_{\ell=1}^{\bar{\ell}} \mathbf{R}_{n,\ell}\, \xi_\ell(\vartheta)\, \varphi_n(s)}{\sum_{n=0}^{\bar{n}} \sum_{\ell=1}^{\bar{\ell}} r_{n,\ell}\, \xi_\ell(\vartheta)\, \varphi_n(s)}. \tag{4.6}$$

We remark that the model numerator and denominator are constructed by linear combination of suitable basis functions: it is straightforward to prove that if the basis functions $\varphi_n(s)$ are rational, the model indicated in (4.6) is a rational function $\forall\vartheta$.
In particular, we denoted with $\bar{n}$ the frequency basis order and with $\bar{\ell}$ the cardinality of the parameter-dependent basis function. To maintain the notation compact, we define a multi-index $\ell = (\ell_1, \ldots, \ell_\rho)$, if $\rho > 1$.
Both the numerator and denominator coefficients are guaranteed real-valued: they are indicated with $\mathbf{R}_{n,\ell} \in \mathbb{R}^{P\times P}$ and $r_{n,\ell} \in \mathbb{R}$, respectively, in (4.6). We can simplify the model expression presented in (4.6), gathering the parameter information

$$\mathbf{H}(s;\vartheta) = \frac{\mathbf{N}(s,\vartheta)}{\mathsf{D}(s,\vartheta)} = \frac{\sum_{n=0}^{\bar{n}} \mathbf{R}_n(\vartheta)\, \varphi_n(s)}{\sum_{n=0}^{\bar{n}} r_n(\vartheta)\, \varphi_n(s)}, \tag{4.7}$$

where

$$\mathbf{R}_n(\vartheta) = \sum_{\ell_N=1}^{\bar{\ell}_N} \mathbf{R}_{n,\ell_N}\, \xi_{\ell_N}(\vartheta) \quad r_n(\vartheta) = \sum_{\ell_D=1}^{\bar{\ell}_D} r_{n,\ell_D}\, \xi_{\ell_D}(\vartheta) \tag{4.8}$$

are the numerator and denominator model coefficients, respectively.
Note that a different parameter-dependent basis order for numerator ($\bar{\ell}_N$) and denominator ($\bar{\ell}_D$) polynomials is possible, as specified in (4.8). Without loss of generality, in the following we will set $\bar{\ell}_N = \bar{\ell}_D = \bar{\ell}$.
The model structure presented before is completely general with respect to the input data set $\breve{\mathbf{H}}(s_k, \vartheta_k)$ representation (scattering, admittance or impedance).

4.1.1 Parameter-Dependent Basis Functions

The variations induced by the external parameters $\vartheta \in \Theta$ are embedded in the model structure (4.6) through the parameter-dependent basis functions $\xi_\ell(\vartheta)$. These basis functions must be selected carefully because upon this choice depends on the fitting accuracy. The literature offers several sets of functions, which are characterized by their own numerical properties.
In the following we will consider only one external parameter ($\rho = 1$).
One important point for our further observations is the (commonly used) procedure of improving the numerical conditioning of fitting algorithms by the normalization of the polynomials argument within $[-1,\, 1]$. In particular, we compute the normalized parameter value $\tilde{\theta}$ as:

$$\tilde{\theta} = -1 + 2 \cdot \frac{\vartheta - \vartheta_{min}}{\vartheta_{max} - \vartheta_{min}} \,. \tag{4.9}$$

The problem conditioning will direct affect the parameter-dependent basis choice.
We now provide several examples of the available choices for the parameter-dependent basis.

Monomials

The simplest polynomial function that could be used to capture the parameter evolution is defined as the standard monomials basis functions [39]

$$\xi_\ell(\vartheta) = \vartheta^\ell, \tag{4.10}$$

where $\ell = 0, ..., \bar{\ell}$ (as defined in (4.6)) and $\bar{\ell}$ is the basis order.
We provide a numerical example, realizing a third-order basis as showed in Fig.4.1. This represents the most intuitive case for parameter-dependent basis definition, but this sort of basis function usually leads to the construction of an *ill-conditioned* fitting problem.

Chebychev Polynomials

We introduce here a new set of basis functions, namely the orthogonal polynomials [1], [29]. From [6], we know that any orthogonal polynomial can be expressed with the recurrence formula that reads:

$$\xi_{\ell+1}(\vartheta) = (\alpha_\ell \vartheta + \beta_\ell)\xi_\ell(\vartheta) + \delta_{\ell-1}\xi_{\ell-1}(\vartheta). \tag{4.11}$$

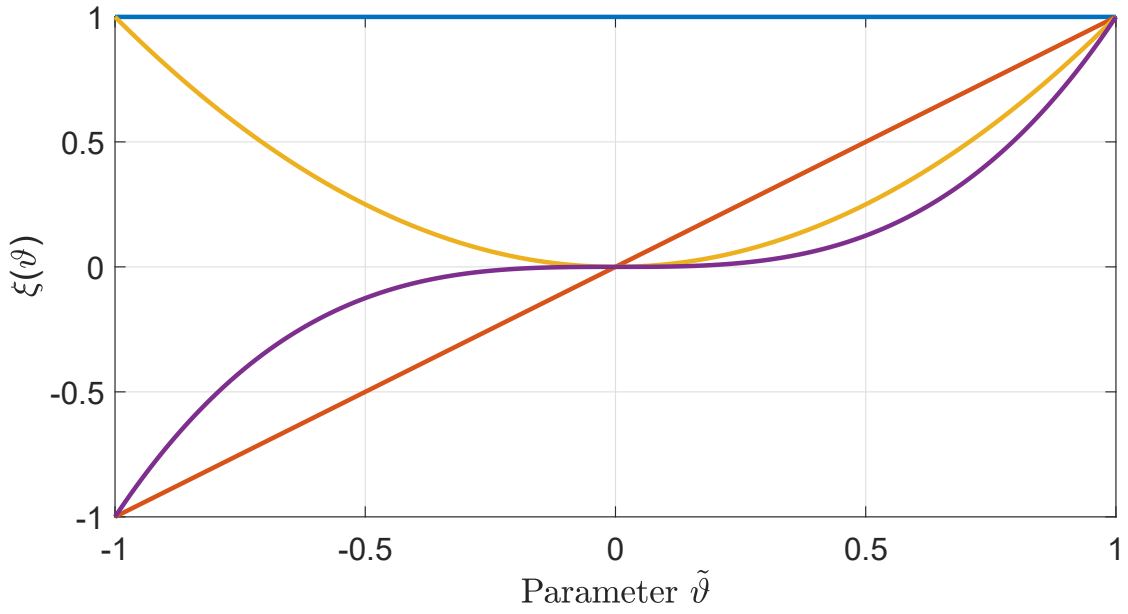


Figure 4.1: Monomials parameter-dependent basis evolution for $\ell = 0,1,2,3$.

In the following we will extensively use Chebychev polynomials, a special class of orthogonal polynomials, for which the expansion coefficients α, β, δ are equal to:

$$\alpha_0 = 1\,, \qquad \beta_0 = 0\,, \qquad \delta_0 = 0 \qquad \ell = 1\,, \tag{4.12}$$

$$\alpha_\ell = 2\,, \qquad \beta_\ell = 0\,, \qquad \delta_\ell = -1 \qquad \forall \ell \geq 1\,. \tag{4.13}$$

It is well known that the basis functions defined as before present very favourable numerical properties, which lead to a well-conditioned (and easy manageable) fitting problem. In this case, the regressor matrix created using such basis shows a reasonable condition number.
We denote the Chebychev polynomials of the first kind basis functions $\xi_\ell(\vartheta) = T_\ell(\vartheta)$ (see [4] and [11]) as:

$$T_\ell(\vartheta) = \cos[\ell \cos^{-1}(\vartheta)]\,, \quad \vartheta \in [-1,\, 1]\,, \quad \ell = 0, ..., \bar{\ell}; \tag{4.14}$$

which is equivalent to the standard expression

$$T_\ell(\cos t) = \cos(\ell t)\,, \quad t \in [0,\, 2\pi]\,, \quad \ell = 0, ..., \bar{\ell}. \tag{4.15}$$

An example of the fourth order Chebychev polynomials (first kind) is reported in Fig. 4.2.

Fourier Series

In order to guarantee a parameterization from a smooth function, when ϑ implies periodic variations, with $\vartheta \in [0,\, 2\pi]$ (e.g. the external parameter is an angle), as discussed in [15], we can define a parameter-dependent basis function as the standard Fourier basis in the

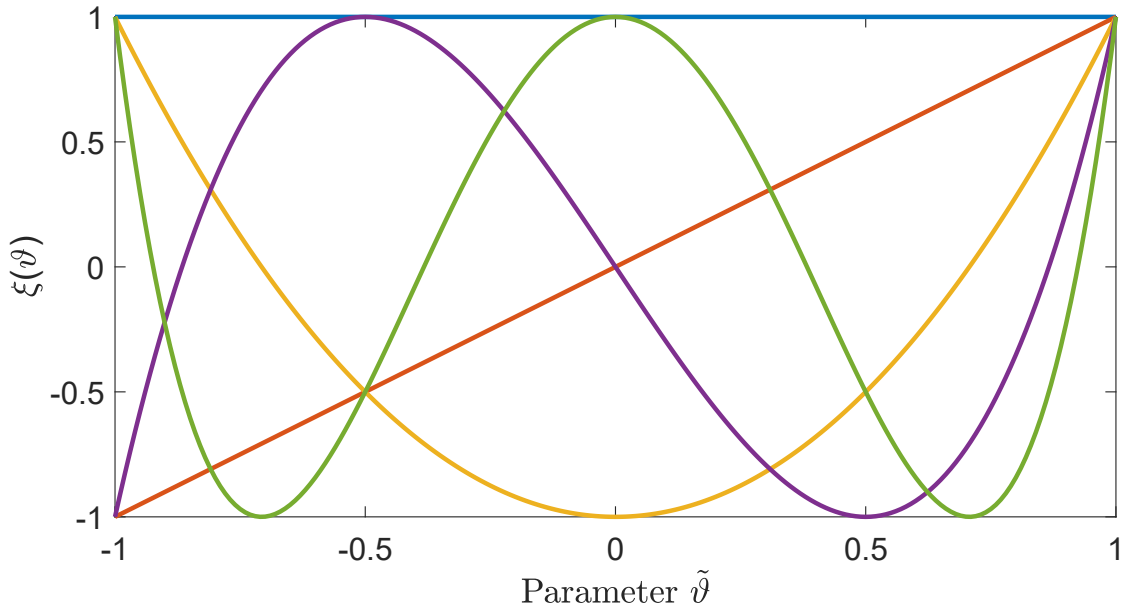


Figure 4.2: Chebychev parameter-dependent basis evolution for $\ell = 0,1,2,3$

trigonometric form

$$\xi_\ell(\vartheta) = \begin{cases} 1, & \ell = 0 \\ \cos(\lceil \ell/2 \rceil \vartheta), & \ell = 1,3,5,\ldots \\ \sin(\lceil \ell/2 \rceil \vartheta), & \ell = 2,4,6,\ldots \end{cases} \tag{4.16}$$

where the argument of $\lceil \cdot \rceil$ is rounded to the nearest larger integer. Figure 4.3 provides a numerical example for the first five terms of the Fourier basis ($\ell = 0,...,4$).

4.1.2 State Space and Descriptor Forms

We now present the state-space and descriptor realizations of a parameter-dependent LTI system, starting from the pole-residue form of the model $\mathbf{H}(s;\vartheta)$. As in the univariate case, also for a multivariate model this representation is appropriate to describe the properties of the model in algebraic form.

State Space Realizations

Following the procedure reported in Section 1.4.2, we can realize a parameter-dependent macromodel equivalent state-space description. Recalling the pole-residue model form of (1.35) and embedding the parameter dependency ϑ, the extension is straightforward.
In fact, considering the model of (4.7), with $\varphi_n(s)$ defined as the partial-fraction basis

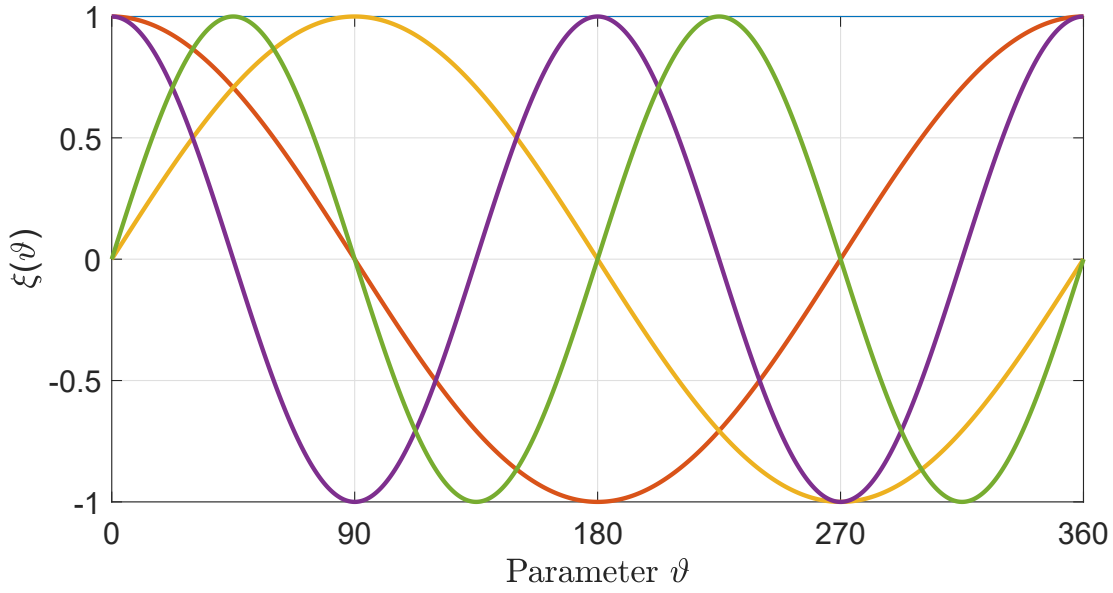


Figure 4.3: First four terms (ℓ = 0,1,2,3) of the Fourier parameter-dependent basis, through the parameter range $\vartheta \in [0°,360°]$. The polynomials arguments is normalized within $[-1,1]$ using the variable range.

with a prescribed set of real and complex poles q_n (see 1.2.1), we can write

$$\mathbf{N}(s,\vartheta) = \mathbf{R}_0(\vartheta) + \sum_{n=1}^{\bar{n}} \frac{\mathbf{R}_n(\vartheta)}{s-q_n} \tag{4.17}$$

$$\mathsf{D}(s,\vartheta) = r_0(\vartheta) + \sum_{n=1}^{\bar{n}} \frac{r_n(\vartheta)}{s-q_n}, \tag{4.18}$$

which allows us to construct the two separate state-space realizations for $\mathbf{N}(s,\vartheta)$ and $\mathsf{D}(s,\vartheta)$ as

$$\mathbf{N}(s,\vartheta) \leftrightarrow \left(\begin{array}{c|c} \mathbf{A}_0 & \mathbf{B}_0 \\ \hline \mathbf{C}_1(\vartheta) & \mathbf{D}_1(\vartheta) \end{array}\right) \tag{4.19}$$

$$\mathsf{D}(s,\vartheta)\mathbb{I}_P \leftrightarrow \left(\begin{array}{c|c} \mathbf{A}_0 & \mathbf{B}_0 \\ \hline \mathbf{C}_2(\vartheta) & \mathbf{D}_2(\vartheta) \end{array}\right), \tag{4.20}$$

where

$$\begin{aligned} \mathbf{A}_0 &= \text{blkdiag}\{\mathbf{A}_{0r}, \mathbf{A}_{0c}\} \\ \mathbf{B}_0^{\mathsf{T}} &= [\mathbf{B}_{0r}^{\mathsf{T}}, \mathbf{B}_{0c}^{\mathsf{T}}] \\ \mathbf{C}_1(\vartheta) &= \begin{bmatrix} \mathbf{R}_1(\vartheta) & \cdots & \mathbf{R}_{\bar{n}}(\vartheta) \end{bmatrix} \\ \mathbf{C}_2(\vartheta) &= \begin{bmatrix} \mathbb{I}_P\, r_1(\vartheta) & \cdots & \mathbb{I}_P\, r_{\bar{n}}(\vartheta) \end{bmatrix} \\ \mathbf{D}_1(\vartheta) &= \mathbf{R}_0(\vartheta) \\ \mathbf{D}_2(\vartheta) &= \mathbb{I}_P\, r_0(\vartheta). \end{aligned} \tag{4.21, 4.22}$$

with

$$\begin{aligned}
\mathbf{A}_{0r} =& \text{blkdiag}\{q_n \mathbb{I}_P\}_{n=1}^{\bar{n}_r} \\
\mathbf{A}_{0c} =& \text{blkdiag}\left\{ \begin{bmatrix} p'_n \mathbb{I}_P & p''_n \mathbb{I}_P \\ -p''_n \mathbb{I}_P & p'_n \mathbb{I}_P \end{bmatrix} \right\}_{n=1}^{\bar{n}_c} \\
\mathbf{B}_{0r} =& \left[1,\ldots,1\right]^{\mathsf{T}} \otimes \mathbb{I}_P \\
\mathbf{B}_{0c} =& \left[2,0,\ldots,2,0\right]^{\mathsf{T}} \otimes \mathbb{I}_P
\end{aligned} \tag{4.23}$$

Following the steps described in [37], we finally obtain the (compact) model state-space realization by the cascade of expression (4.19) as:

$$\mathbf{H}(s,\vartheta) = \mathbf{N}(s,\vartheta)(\mathsf{D}(s,\vartheta)^{-1}\mathbb{I}_P) \leftrightarrow \left(\begin{array}{c|c} \mathbf{A}_0 - \mathbf{B}_0\mathbf{D}_2^{-1}(\vartheta)\mathbf{C}_2(\vartheta) & \mathbf{B}_0\mathbf{D}_2^{-1}(\vartheta) \\ \hline \mathbf{C}_1(\vartheta) - \mathbf{D}_1(\vartheta)\mathbf{D}_2^{-1}(\vartheta)\mathbf{C}_2(\vartheta) & \mathbf{D}_1(\vartheta)\mathbf{D}_2^{-1}(\vartheta) \end{array} \right). \tag{4.24}$$

We recall [37] for more details.

Descriptor Forms

Recalling to the descriptor representation (1.57) of Section 1.4.2, we now define its parameter-dependent form [37] to (4.24).
The descriptor matrices, which now depend on the external parameter ϑ, are

$$\begin{aligned}
\mathbf{E} &= \begin{pmatrix} \mathbb{I}_N & \mathbf{0}_{N,P} \\ \mathbf{0}_{P,N} & \mathbf{0}_{P,P} \end{pmatrix} \quad \mathbf{A}(\vartheta) = \begin{pmatrix} \mathbf{A}_0 & \mathbf{B}_0 \\ \mathbf{C}_2(\vartheta) & \mathbf{D}_2(\vartheta) \end{pmatrix} \\
\mathbf{C}(\vartheta) &= \begin{pmatrix} \mathbf{C}_1(\vartheta) & \mathbf{D}_1(\vartheta) \end{pmatrix} \qquad \mathbf{B} = \begin{pmatrix} \mathbf{0}_{N,P} \\ -\mathbb{I}_P \end{pmatrix}
\end{aligned} \tag{4.25}$$

with $\mathbf{0}_{J,K}$ null matrix of size $J \times K$. The other matrices of (1.58) denote the state-space realization of the model numerator $\mathbf{N}(s,\vartheta)$, described by the set $\{\mathbf{A}_0, \mathbf{B}_0, \mathbf{C}_1(\vartheta), \mathbf{D}_1(\vartheta)\}$, and the (extended) denominator $\mathsf{D}(s,\vartheta)\mathbb{I}_P$, described by $\{\mathbf{A}_0, \mathbf{B}_0, \mathbf{C}_2(\vartheta), \mathbf{D}_2(\vartheta)\}$, which are exactly the same elements of (4.19).
The model expression of (4.6) is equivalent to

$$\mathbf{H}(s;\vartheta) = \mathbf{C}(\vartheta)(s\mathbf{E} - \mathbf{A}(\vartheta))^{-1}\mathbf{B}\,, \tag{4.26}$$

as detailed in [37] and [14].

Chapter 5

Parameterized Macromodeling

5.1 Parameterized Sanathanan-Koerner

In this section an extension of the GSK algorithm aimed to return a parameterized macromodel is presented. It exploits both the results obtained for the VF and GSK; the resulting algorithm is named Parameterized Sanathanan-Koerner (PSK) iteration [37].
The scalar formulation of the algorithm is presented here, while the multiport formulation is explained in the next chapter.
We start by clarifying some aspects of the parameterized macromodeling that justify the model structure on which PSK is based.

5.1.1 Poles Parameterization

The final goal of a parameterized macromodel is to describe the dynamics of the underlying structure in a closed form with respect to an arbitrary number of physical parameters whose value can vary within a well defined range.
Since we are making no assumptions about how the parameters influence the system dynamics, the most general formulation for a parmeterized macromodel is such that both poles and residues are parameter-dependent; a fully parameterized macromodel would thus read:

$$\mathbf{H}(s,\theta) = \sum_{j=1}^{n} \frac{\mathbf{R}_j(\theta)}{s - p_j(\theta)} + \mathbf{R}_0(\theta), \tag{5.1}$$

where we denote with θ all the parameters on which the model depens on.
This structure, that is the most desirable, is unfortunately ill-defined in the majority of the cases; indeed, the dependency of the poles on the parameters is generally non smooth: two different real poles can collide and generate a complex conjugate pair and, viceversa, complex conjugate poles can collide and fork in two distinct real poles. An example of this fact is the well known RLC oscillator, whose behavior strongly depends on the damping factor, that is fixed by the value of the resistance R. Even the modeling of a second order system with a single parameter would consequently lead to an ill-posed problem.
The behavior of the poles under a parameter variation suggests that an explicit parameterization of them is to be avoided in order to obtain a precise and smooth description

of the model in the parameter range; the alternative is to find a model structure that implicitly parameterizes the poles behavior.
We opt for a fully implicit and global parameterization of the model: in this formulation we preserve the model structure used for the GSK algorithm and we let the partial fractions coefficients of both numerator and denominator be parameter-dependent. The identification procedure is achieved through the PSK algorithm.

5.1.2 The Parameterized Sanathanan Koerner Iteration

The PSK algorithm in its scalar formulation, allows us to obtain a parametric model of the form:

$$H(s;\vartheta) = \frac{N(s,\vartheta)}{D(s,\vartheta)} = \frac{\sum_{n=0}^{\bar{n}} \sum_{\ell=1}^{\bar{\ell}} R_{n,\ell}\, \xi_\ell(\vartheta)\, \varphi_n(s)}{\sum_{n=0}^{\bar{n}} \sum_{\ell=1}^{\bar{\ell}} r_{n,\ell}\, \xi_\ell(\vartheta)\, \varphi_n(s)}, \tag{5.2}$$

where $R_{n,\ell} \in \mathbb{R}$ and $r_{n,\ell} \in \mathbb{R}$ are the model coefficients. In the majority of the cases the frequency-dependent functions $\varphi_n(s)$ are the partial fractions defined in 1.2.1, while $\xi_\ell(\vartheta)$ are suitable basis functions, usually the Chebycev polynomials, appropriate to describe the model dependency on the parameters.
As already stated in Section 4.1, in the general case $\ell = (\ell_1, \ldots, \ell_\rho)$ is a multi-index whose dimension depends on the number ρ of parameters.
We pointout that the quantity

$$\bar{\ell} = \prod_{i=1}^{\rho} \bar{\ell}_i. \tag{5.3}$$

represents the total number of basis functions used to embed the parameter-dependent behaviour. We will assume without loss of generality that both the denominator and the numerator are constructed with the same number of basis functions.
The PSK algorithm iteratively updates the model coefficients $R_{n,\ell}$ and $r_{n,\ell}$ until a desired precision is achieved or the values of the estimates stabilize: the strategy is the same used for the GSK: the frequency bias induced by the linearization of the residuals is compensated through the values of the denominator of the model obtained at the previous iteration, computed at the fitting points. The main difference here is that, being the model parameter-dependent, so will be the residuals and the cost function.
We start from a dataset of avalaible points in the frequency-parameter domain:

$$\breve{H}_{k,m} = \breve{H}(s_k;\vartheta_m), \quad k = 1,\ldots,K,\ m = 1,\ldots,M, \tag{5.4}$$

where we implicitly state that for every value of the parameter ϑ_m the same number of frequency samples K of the related model are collected. At each iteration the residuals are defined as

$$r^{\nu}_{k,m}(\boldsymbol{x}_\nu) = \breve{H}_{k,m} - H(s_k;\vartheta_m), \quad for \quad k = 1,2,\ldots,K \quad , \quad m = 1,2,\ldots,M. \tag{5.5}$$

where now the unknowns vector is:

$$\begin{aligned} \boldsymbol{x}_\nu &= (\boldsymbol{c}_\nu, \boldsymbol{d}_\nu)^T, \\ \boldsymbol{c}_\nu &= (R^{\nu}_{0,1}, R^{\nu}_{0,2}, \ldots, R^{\nu}_{0,\bar{\ell}}, \ldots, R^{\nu}_{\bar{n},1}, R^{\nu}_{\bar{n},2}, \ldots, R^{\nu}_{\bar{n},\bar{\ell}})^T \\ \boldsymbol{d}_\nu &= (r^{\nu}_{0,1}, r^{\nu}_{0,2}, \ldots, r^{\nu}_{0,\bar{\ell}}, \ldots, r^{\nu}_{\bar{n},1}, r^{\nu}_{\bar{n},2}, \ldots, r^{\nu}_{\bar{n},\bar{\ell}})^T; \end{aligned} \tag{5.6}$$

here $\boldsymbol{c}_\nu$ and $\boldsymbol{d}_\nu$ contain the numerator and denominator coefficients at the current iteration. Our aim is to minimize the quantity $\|\boldsymbol{r}(\boldsymbol{x}_\nu)\|$, i.e. the norm of the vector which collects all the residuals.
We linearize the residuals and compensate for the bias defining the modified residuals:

$$e^{\nu}_{k,m}(\boldsymbol{x}_\nu) = \frac{N(s_k;\vartheta_m;\boldsymbol{x}_\nu) - D^{\nu}(s_k;\vartheta_m;\boldsymbol{x}_\nu)\breve{H}_{k,m}}{D^{\nu-1}(s_k;\vartheta_m;\boldsymbol{x}_{\nu-1})} \quad for \quad k = 1,2,\dots,K \quad , \quad m = 1,2,\dots,M. \tag{5.7}$$

At each iteration we want to minimize the quantity $\|\boldsymbol{e}(\boldsymbol{x}_\nu)\|$, the norm of the vector collecting all the modified residuals. Again we minimize the eucledean norm, i.e. we minimize the cost function:

$$J_{PSK}(\boldsymbol{x}_\nu) = \sum_{k=1}^{K}\sum_{m=1}^{M}\left|\frac{N(s_k;\vartheta_m;\boldsymbol{x}_\nu) - D^{\nu}(s_k;\vartheta_m;\boldsymbol{x}_\nu)\breve{H}_{k,m}}{D^{\nu-1}(s_k;\vartheta_m;\boldsymbol{x}_{\nu-1})}\right|^2, \tag{5.8}$$

where the notation is the same used for the description of the GSK iteration. We now define some matrices to formulate the least squares system that minimizes $J_{PSK}(\boldsymbol{x}_\nu)$.
We here place emphasis on the matrices dimensions, since they are crucial to perform the complexity analysis of the next chapter.
First, we define the matrix:

$$\boldsymbol{\Phi}(\vartheta) = \begin{pmatrix} \xi_0(\vartheta)...\xi_{\bar{\ell}}(\vartheta) & \frac{\xi_0(\vartheta)}{s_1-q_1}...\frac{\xi_{\bar{\ell}}(\vartheta)}{s_1-q_1} & ... & \frac{\xi_0(\vartheta)}{s_1-q_{\bar{n}}}...\frac{\xi_{\bar{\ell}}(\vartheta)}{s_1-q_{\bar{n}}} \\ \xi_0(\vartheta)...\xi_{\bar{\ell}}(\vartheta) & \frac{\xi_0(\vartheta)}{s_2-q_1}...\frac{\xi_{\bar{\ell}}(\vartheta)}{s_2-q_1} & ... & \frac{\xi_0(\vartheta)}{s_2-q_{\bar{n}}}...\frac{\xi_{\bar{\ell}}(\vartheta)}{s_2-q_{\bar{n}}} \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ \xi_0(\vartheta)...\xi_{\bar{\ell}}(\vartheta) & \frac{\xi_0(\vartheta)}{s_K-q_1}...\frac{\xi_{\bar{\ell}}(\vartheta)}{s_K-q_1} & ... & \frac{\xi_0(\vartheta)}{s_K-q_{\bar{n}}}...\frac{\xi_{\bar{\ell}}(\vartheta)}{s_K-q_{\bar{n}}} \end{pmatrix}, \tag{5.9}$$

which collects the partial fraction basis, evaluated at each frequency for which we have a data sample; each element of the partial fractions is combined with all the elements of the parameters basis.
For simplicity we will suppose that for each parameter value exploited for the solution of the least squares problem, the same number of frequency samples is given; thus:

$$\boldsymbol{\Phi}(\vartheta) \in \mathbb{C}^{K\times(\bar{n}+1)\bar{\ell}}. \tag{5.10}$$

Second, the matrix

$$\breve{\mathbf{H}}(\vartheta) = diag\{\breve{H}_{1,\vartheta},\breve{H}_{2,\vartheta},\breve{H}_{3,\vartheta},...,\breve{H}_{K,\vartheta}\}, \quad \breve{\mathbf{H}}(\vartheta) \in \mathbb{C}^{K\times K} \tag{5.11}$$

collects in its diagonal all the data frequency samples referred to a given parameter value.

Third, the PSK weights are also distinguished with respect to the parameter value at which the denominator is computed:

$$\mathbf{W}^{\nu-1}(\vartheta) = diag\left\{\frac{1}{D^{\nu-1}(s_1,\vartheta)}, \frac{1}{D^{\nu-1}(s_2,\vartheta)}, ..., \frac{1}{D^{\nu-1}(s_k,\vartheta)}\right\}, \tag{5.12}$$

also in this case $\mathbf{W}^{\nu-1}(\vartheta) \in \mathbb{C}^{K\times K}$.
With these three ingredients we can formulate the LS problem for the PSK iteration, that will exhibit the following structure:

$$\begin{pmatrix} \mathbf{W}^{\nu-1}(\vartheta_1)\mathbf{\Phi}(\vartheta_1), & \mathbf{W}^{\nu-1}(\vartheta_1)\breve{\mathbf{H}}(\vartheta_1)\mathbf{\Phi}(\vartheta_1) \\ \mathbf{W}^{\nu-1}(\vartheta_2)\mathbf{\Phi}(\vartheta_2), & \mathbf{W}^{\nu-1}(\vartheta_2)\breve{\mathbf{H}}(\vartheta_2)\mathbf{\Phi}(\vartheta_2) \\ \mathbf{W}^{\nu-1}(\vartheta_3)\mathbf{\Phi}(\vartheta_3), & \mathbf{W}^{\nu-1}(\vartheta_3)\breve{\mathbf{H}}(\vartheta_3)\mathbf{\Phi}(\vartheta_3) \\ & \vdots \\ \mathbf{W}^{\nu-1}(\vartheta_M)\mathbf{\Phi}(\vartheta_M), & \mathbf{W}^{\nu-1}(\vartheta_M)\breve{\mathbf{H}}(\vartheta_M)\mathbf{\Phi}(\vartheta_M) \end{pmatrix} \boldsymbol{x}_\nu \approx \mathbf{0}; \tag{5.13}$$

where $\mathbf{0} \in \mathbb{C}^{KM,1}$ is a column of zeros. Note that the number of rows doubles for the presence of complex data in case the system is formulated in real algebra.
To avoid the non trivial solution a final row and its relative right-hand side term must be added. They can be computed in the same way as the one that appears in the relaxed version of the VF algorithm, with the difference that this time the denominator depends also on the parameter basis.
In order to simplify the notation, we rename the matrix blocks that appear in the regressor and write:

$$\mathbf{\Gamma}^\nu = \begin{pmatrix} \mathbf{W}^{\nu-1}(\vartheta_1)\mathbf{\Phi}(\vartheta_1) \\ \mathbf{W}^{\nu-1}(\vartheta_2)\mathbf{\Phi}(\vartheta_2) \\ \mathbf{W}^{\nu-1}(\vartheta_3)\mathbf{\Phi}(\vartheta_3) \\ \vdots \\ \mathbf{W}^{\nu-1}(\vartheta_M)\mathbf{\Phi}(\vartheta_M) \end{pmatrix}, \quad \mathbf{\Xi}^\nu = \begin{pmatrix} \mathbf{W}^{\nu-1}(\vartheta_1)\breve{\mathbf{H}}(\vartheta_1)\mathbf{\Phi}(\vartheta_1) \\ \mathbf{W}^{\nu-1}(\vartheta_2)\breve{\mathbf{H}}(\vartheta_2)\mathbf{\Phi}(\vartheta_2) \\ \mathbf{W}^{\nu-1}(\vartheta_3)\breve{\mathbf{H}}(\vartheta_3)\mathbf{\Phi}(\vartheta_3) \\ \vdots \\ \mathbf{W}^{\nu-1}(\vartheta_M)\breve{\mathbf{H}}(\vartheta_M)\mathbf{\Phi}(\vartheta_M) \end{pmatrix} \tag{5.14}$$

and formulate again the the system as follows, adding the non triviality constraint:

$$\begin{pmatrix} \mathbf{\Gamma}^\nu, & \mathbf{\Xi}^\nu \\ \mathbf{0}, & \alpha\boldsymbol{v}_\nu^T \end{pmatrix} \boldsymbol{x}_\nu \approx \begin{pmatrix} \mathbf{0} \\ \alpha \end{pmatrix}. \tag{5.15}$$

The iterations are performed until a given degree of accuracy is reached or the maximum number of allowed iterations is reached.
A remark on the selection of the proper partial fractions concludes the discussion: since the poles are implicitly parameterized by the PSK, the partial fractions to be used as frequency-dependent basis are chosen by fixing the parameters to a prescribed value, performing a vector fitting identification on the univariate resulting model, and using the output set of poles to define the partial fractions.
A natural choice is to use the "central" response, the one that occurs for the mean value

of the parameters (or, equivalently, the one nearest to the mean value among the avalaible data).

Algorithm 5.3 Basic Parameterized Sanathanan-Koerner

Require: frequency-parameter data $\{(s_k, \theta_m, \breve{H})\}_{k,m=1}^{K,M}$,order $\bar{n}$ for the frequency basis, orders $\bar{\ell}_i$ for the parameters basis, threshold ϵ
perform the VF algorithm for the univariate model
set $\boldsymbol{x}_0 = \mathbf{0}$
for $\nu = 1,2,...,\nu_{max}$ **do**
build the matrices $\boldsymbol{\Gamma}^\nu, \boldsymbol{\Xi}^\nu$ using the poles obtained through the VF
solve the LS problem 5.15
if $||\boldsymbol{x}_\nu - \boldsymbol{x}_{\nu-1}|| < \epsilon||\boldsymbol{x}_\nu||$ *or* $\nu = \nu_{max}$ **then**
break
end if
end for
return $H(s, \theta)$

5.2 PSK for Multiports and Fast PSK

The PSK algorithm formulation for multiports allows us to obtain a parametric model of the form:

$$\mathbf{H}(s;\vartheta) = \frac{\mathbf{N}(s,\vartheta)}{D(s,\vartheta)} = \frac{\sum_{n=0}^{\bar{n}} \sum_{\ell=1}^{\bar{\ell}} \mathbf{R}_{n,\ell}\, \xi_\ell(\vartheta)\, \varphi_n(s)}{\sum_{n=0}^{\bar{n}} \sum_{\ell=1}^{\bar{\ell}} r_{n,\ell}\, \xi_\ell(\vartheta)\, \varphi_n(s)}, \tag{5.16}$$

where $\mathbf{R}_{n,\ell} \in \mathbb{R}^{P\times P}$ and $r_{n,\ell} \in \mathbb{R}$ are the model coefficients and, in the majority of the cases $\varphi_n(s)$ are the partial fractions basis. Again, $\xi_\ell(\vartheta)$ are suitable basis functions, usually the Chebychev polynomials, that interpolate the parametric macromodel behaviour over the parameter space Θ.
The extension of the PSK to a multiport system with common denominator is trivial and traces the same reasoning involved for the VF; also in this case we will drop the superscript ν for the unknown vector of coefficients, implicitly stating that they are iteration-dependent. First we define the unknown vector

$$\boldsymbol{x} = (\boldsymbol{c}^1, \boldsymbol{c}^2, \ldots, \boldsymbol{c}^{P^2}, \boldsymbol{d})^T, \tag{5.17}$$

with this representation, every $\boldsymbol{c}^i$ vector collects in column the entries of the matrices $\mathbf{R}_{n,\ell}$ with the same indices, while $\boldsymbol{d}$ contains the parameters of the denominator. To identify each sample of the dataset, we will use a the following notation, similar to the one reported in 3.18: every sample:

$$\breve{h}^i_{k,m}, \quad for \quad k = 1,2,\ldots,K \quad , \quad m = 1,2,\ldots,M \quad , \quad i = 1,2,\ldots,P^2 \tag{5.18}$$

refers to the k-th frequency sample of the i-th response, collected for the m-th value of the parameters.

For each response we define the PSK residual quantity:

$$e^i_{k,m} = \frac{N^i(s_k;\vartheta_m) - D^\nu(s_k;\vartheta_m)\breve{h}^i_{k,m}}{D^{\nu-1}(s_k;\vartheta_m)} \qquad (5.19)$$
$$for \quad k = 1,2,\ldots,K \quad , \quad m = 1,2,\ldots,M \quad , \quad i = 1,2,\ldots,P^2,$$

where the dependency of all the quantities on the vector $\boldsymbol{x}$ is implicit. At each iteration we want to minimize the quantity $\|\boldsymbol{e}\|^2$, where

$$\boldsymbol{e} = \begin{pmatrix} \boldsymbol{e}^1 \\ \boldsymbol{e}^2 \\ \vdots \\ \boldsymbol{e}^{P^2} \end{pmatrix} \qquad (5.20)$$

where each $\boldsymbol{e}^i$ is defined as in (5.7) and is relative to the i-th port response. To minimize the eucledean norm of this vector we must minimize the cost function:

$$J_{MPSK}(\boldsymbol{x}) = \sum_{i=1}^{P^2}\sum_{k=1}^{K}\sum_{m=1}^{M}\left|\frac{N^i(s_k,\vartheta_m) - D^\nu(s_k,\vartheta_m)\breve{h}^i_{k,m}}{D^{\nu-1}(s_k,\vartheta_m)}\right|^2. \qquad (5.21)$$

Consequently the global linear system for the multiport PSK becomes:

$$\begin{pmatrix} \boldsymbol{\Gamma} & \mathbf{0} & \mathbf{0} & ... & \mathbf{0} & \boldsymbol{\Xi}_1 \\ \mathbf{0} & \boldsymbol{\Gamma} & \mathbf{0} & ... & \mathbf{0} & \boldsymbol{\Xi}_2 \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\Gamma} & ... & \mathbf{0} & \boldsymbol{\Xi}_3 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & ... & \boldsymbol{\Gamma} & \boldsymbol{\Xi}_{P^2} \end{pmatrix} \begin{pmatrix} \boldsymbol{c}^1 \\ \boldsymbol{c}^2 \\ \boldsymbol{c}^3 \\ \vdots \\ \boldsymbol{c}^{P^2} \\ \boldsymbol{d} \end{pmatrix} \approx \mathbf{0}, \qquad (5.22)$$

where the terms $\boldsymbol{\Gamma}$ are defined as in (5.14) with the difference that we dropped the dependency on ν; each $\boldsymbol{\Xi}_i$ is equivalent to the one of equation (5.14), but in this case it is built with respect to the i-th port response, i.e:

$$\boldsymbol{\Xi}_i = \begin{pmatrix} \mathbf{W}^{\nu-1}(\vartheta_1)\breve{\mathbf{H}}^i(\vartheta_1)\boldsymbol{\Phi}(\vartheta_1) \\ \mathbf{W}^{\nu-1}(\vartheta_2)\breve{\mathbf{H}}^i(\vartheta_2)\boldsymbol{\Phi}(\vartheta_2) \\ \mathbf{W}^{\nu-1}(\vartheta_3)\breve{\mathbf{H}}^i(\vartheta_3)\boldsymbol{\Phi}(\vartheta_3) \\ \vdots \\ \mathbf{W}^{\nu-1}(\vartheta_M)\breve{\mathbf{H}}^i(\vartheta_M)\boldsymbol{\Phi}(\vartheta_M) \end{pmatrix} \qquad (5.23)$$

with

$$\breve{\mathbf{H}}^i(\vartheta) = diag\{\breve{h}^i_{1,\vartheta}, \breve{h}^i_{2,\vartheta}, \breve{h}^i_{3,\vartheta}, ..., \breve{h}^i_{K,\vartheta}\}. \qquad (5.24)$$

The regressor in this case, for practical purposes, has dimensions $2KMP^2 \times \left((\bar{n}+1)\bar{\ell}\left(P^2+1\right)\right)$ (from now on, we will imply that the LS are formulated in real algebra).
The structure that is shown above is the same that we found for the multiport VF; is thus natural to exploit the same compression procedure that we achieved with the QR factorization also for this system.

5.2.1 Least Square System Decoupling for Multiports

The procedure of decoupling for the resolution of the PSK system follows the same steps of the one related to the Vector Fitting. The main difference is that while in the VF it's sufficient to find only the unknowns related to the denominator in the H iterations of the poles relocation, for every PSK iteration we must find all the parameters related to denominator and numerators. Therefore, we compute the QR factorizations:

$$\mathbf{P}_i = (\mathbf{\Gamma}, \mathbf{\Xi}_i) = \mathbf{Q}_i \mathbf{R}_i = \mathbf{Q}_i \begin{pmatrix} \mathbf{R}_i^{11} & \mathbf{R}_i^{12} \\ \mathbf{0} & \mathbf{R}_i^{22} \end{pmatrix} \tag{5.25}$$

in this case $\mathbf{Q}_i \in \mathbb{C}^{2KM \times 2(\bar{n}+1)\bar{\ell}}$,$\mathbf{R}_i \in \mathbb{C}^{2(\bar{n}+1)\bar{\ell} \times 2(\bar{n}+1)\bar{\ell}}$.
We observe that the $\mathbf{\Gamma}$ matrix is always the same for every factorization, so that the computation of $\mathbf{R}_i$ can be performed as indicated in Section 3.1.3
Once all the $\mathbf{R}_i^{22}$ blocks has been computed, one solves the system:

$$\begin{pmatrix} \mathbf{R}_1^{22} \\ \mathbf{R}_2^{22} \\ \vdots \\ \mathbf{R}_{P^2}^{22} \end{pmatrix} \boldsymbol{d} \approx \mathbf{0}. \tag{5.26}$$

Adding the non-triviality constraint row and relative RHS term.
Once the system has been solved, the numerator unknowns are computed through the solution of a multiple right-hand sides least squares of the form:

$$\mathbf{\Gamma X} \approx \mathbf{B}, \tag{5.27}$$

with:

$$\mathbf{X} = (\boldsymbol{c}^1, \boldsymbol{c}^2, \ldots, \boldsymbol{c}^{P^2}), \tag{5.28}$$

and:

$$\mathbf{B} = (-\mathbf{\Xi}_1 \boldsymbol{d}, -\mathbf{\Xi}_2 \boldsymbol{d}, \quad \ldots \quad , -\mathbf{\Xi}_{P^2} \boldsymbol{d}) \tag{5.29}$$

One can keep in memory the QR factorization already performed over $\mathbf{\Gamma}$ to solve the system efficiently. We will refer to this new version of the PSK algorithm as fast PSK (FPSK) by analogy with the relative version of the Vector Fitting.

5.2.2 PSK Computational Complexity Reduction

To compute the computational complexity of PSK and FPSK, we will adopt the same notation used in 3.2.2; we remark that:

- N is here the number of partial fraction basis functions exploited to build the model.
- $\bar{\ell}$ is the number of all the possible combinations of the parameters basis functions.
- F is the number of frequency samples collected, assumed equal for every parameters value and for every response.

The algebraic operations involved in the PSK iteration are the same involved in the VF iteration, except for the calculation of the eigenvalues, so we will refer to the flops calculation of the previous section and of 1.1.3.
The number of flops necessary for the PSK algorithm is (we neglect the non triviality constraint since irrelevant):

$$Fl_{PSK}\left(L,K,M,N,\bar{\ell},H\right)=H\times Fl_{MLS}\left(2KML,(\bar{n}+1)\bar{\ell}\,(L+1)\,,1\right) \tag{5.30}$$

To compute the flops required by the FPSK algorithm (in the relaxed form), remind that we perform once per cycle the QR factorization:

$$\mathbf{\Gamma}=\mathbf{Q}_{\Gamma}\mathbf{R}_{\Gamma} \tag{5.31}$$

and L times the block QR factorization, for every response. The number of flops required for FPSK is:

$$\begin{aligned} Fl_{FPSK}\left(L,K,M,N,\bar{\ell},H\right) &= H\times Fl_{QR}\left(2KM,(\bar{n}+1)\bar{\ell}\right)\\ &+H\times L\times Fl_{QR_{block}}\left(2KM,(\bar{n}+1)\bar{\ell}\right)\\ &+H\times Fl_{MLS}\left(L(\bar{n}+1)\bar{\ell},(\bar{n}+1)\bar{\ell},1\right)\\ &+H\times Fl_{MLS}\left(2KM,(\bar{n}+1)\bar{\ell},L\right) \end{aligned} \tag{5.32}$$

Here we make the assumption that $KM>(\bar{n}+1)\bar{\ell}$, and proceed with an asymptotic estimation of the flops required by the two versions of the algorithm. For the standard PSK, the evaluation is trivial and leads to:

$$Fl_{PSK}\propto HKM(\bar{n}+1)^2\bar{\ell}^2L^3, \tag{5.33}$$

While for the FPSK:

$$Fl_{FPSK}\propto HKM(\bar{n}+1)^2\bar{\ell}^2L, \tag{5.34}$$

Also in this case the dependency of the number of flops on the number of responses switches from cubic to linear. We observe that this improvement is more relevant in the PSK than how it is in the VF, since $HKM(\bar{n}+1)^2\bar{\ell}^2$ can assume very large values in parametric modelling problems.
If we want to mantain certain coefficients of the denominator fixed, we must consider that the L QR factorizations that are necessary must be computed over a matrix with (approximately) twice the number of columns with respect to the one that we considered here; moreover, the application of the matrices $\mathbf{Q}_i^H$ to the data samples becomes necessary; it leads to a slight increment of the number of required flops.

5.2.3 PSK Memory Requirements Reduction

We end the discussion about the advantages of the FPSK considering the memory requirements reduction that it allows. For the standard PSK the system matrix to keep in memory contains:

$$E_{PSK}=2KML(\bar{n}+1)\bar{\ell}\,(L+1) \tag{5.35}$$

elements. The new formulation requires the storage of the compressed system matrix necessary to compute the denominators coefficients:

$$\mathbf{R_s} = \begin{pmatrix} \mathbf{R}_1^{22} \\ \mathbf{R}_2^{22} \\ \vdots \\ \mathbf{R}_L^{22} \end{pmatrix}, \mathbf{R_s} \in \mathbb{C}^{L(\bar{n}+1)\bar{\ell}\times(\bar{n}+1)\bar{\ell}} \tag{5.36}$$

that contains:

$$E_{R_s} = L(\bar{n}+1)^2\bar{\ell}^2 \tag{5.37}$$

elements.
During a single iteration,to solve for the numerator coefficients, one must also store the matrix $\mathbf{\Gamma} \in \mathbb{C}^{2KM\times(\bar{n}+1)\bar{\ell}}$; also its QR factorization is exploited to calculate the blocks of $\mathbf{R}_s$, so, in general, the factorization of $\mathbf{\Gamma}$ should be kept in memory if we want to speed up the computations. Besides the possible implementations of the algorithm, we consider the worst case, that is the one in which we keep in memory the three matrices $\mathbf{Q}_L$, $\mathbf{R}_L$ and $\mathbf{R_s}$, when we are about to compute the last block $\mathbf{R}_L^{22}$ and we compute the QR factorization of $(\mathbf{\Gamma}, \mathbf{\Xi}_L)$, from scratch. In this scenario the memory required by the algorithm is:

$$E_{FPSK_{wc}} = E_{R_s} + 4KM(\bar{n}+1)\bar{\ell} + 4(\bar{n}+1)^2\bar{\ell}^2 \tag{5.38}$$

Also in this worst case, we see that the memory requirements are influenced by the number of responses linearly, with a rate of grow much smaller than the one found for the classical formulation.

5.3 Implementation of the FPSK Algorithm

In this chapter we present our implementation of the FPSK algorithm for multiport systems. The algorithm is implemented in MATLAB and allows a variety of choices for the user, in order to guarantee a certain degree of adaptability during the modeling phase. A qualitative description of our implementation is given, making use of a basic flowchart. Before entering into the details of the implementation, we present some different modeling strategies that have not been exposed already from a theoretical point of view. Here we assume that the data points that must be fitted are collected at real frequencies as it is usual in practical implementations.

5.4 Minimization of the Relative Error

The modeling of parametric multiport system requires the reconstruction of families of transfer functions characterized by different dynamic ranges. For example, filters or reactive components are designed to attenuate the signal components related to some prescribed frequency bands, while mantaining or amplifying others; transmission lines can exhibit weak coupling effects that require the modeling of transfer functions with very small magnitudes in the entire bandwidth of interest. In these situations, the minimization of the absolute error performed by the LS can result in a lack of accuracy of

the fitting in the regions with higher attenuating behaviour. To overcome this issue, one can modify the cost function of the LS problem in such a way that the residuals are normalized with respect to the value of the data-point to which they refer; doing this, every data-point assumes the same importance in the evaluation of the cost function and the fitting procedure makes no difference with respect to the magnitude of the data.
Without loss of generality, let us consider the cost function associated to a generic univariate modeling procedure applied to a scalar (one-port) system; the cost function reads:

$$J(\boldsymbol{x}) = \sum_{k=1}^{K} \left| H(\mathrm{j}2\pi f_k; \boldsymbol{x}) - \breve{H}(\mathrm{j}2\pi f_k) \right|^2 = \sum_{k=1}^{K} \delta_k^2(\boldsymbol{x}) \tag{5.39}$$

where δ_k denotes the k-th residual quantity. This cost function depends on the decision vector $\boldsymbol{x}$. We can define a new residual defined as:

$$\delta_k^*(\boldsymbol{x}) = \frac{\delta_k(\boldsymbol{x})}{|\breve{H}(\mathrm{j}2\pi f_k)|^\beta}, \tag{5.40}$$

where with β we denote a parameter that controls the influence of the weighting over the residuals. The minimization of these new residuals guarantees that the error is minimized with respect to the magnitude of the related data-point; the corresponding cost function can be written as:

$$J_{rel}(\boldsymbol{x}) = \sum_{k=1}^{K} \left| \frac{H(\mathrm{j}2\pi f_k; \boldsymbol{x}) - \breve{H}(\mathrm{j}2\pi f_k)}{|\breve{H}(\mathrm{j}2\pi f_k)|^\beta} \right|^2 = \sum_{k=1}^{K} \delta_k^{*2}(\boldsymbol{x}). \tag{5.41}$$

With respect to our multiport parameterized case, the modified cost function reads:

$$J_{rel}(\boldsymbol{x}) = \sum_{i=1}^{P^2} \sum_{k=1}^{K} \sum_{m=1}^{M} \left| \frac{N^i(\mathrm{j}2\pi f_k; \vartheta_m; \boldsymbol{x}) - D^\nu(\mathrm{j}2\pi f_k; \vartheta_m; \boldsymbol{x})\, \breve{h}_{k,m}^i}{D^{\nu-1}(\mathrm{j}2\pi f_k; \vartheta_m; \boldsymbol{x}) |\breve{h}_{k,m}^i|^\beta} \right|^2, \tag{5.42}$$

where the data $\breve{h}_{k,m}^i$ are defined as in (5.18). The minimizaton of this cost function is achieved with a modification of the LS problem that returns the coefficients of the PSK at each iteration; in fact, we can define a port-dependent weight matrix:

$$\begin{aligned} &\mathbf{W}_{rel}^i = (diag\{|\breve{h}_{1,1}^i|^\beta, \ldots, |\breve{h}_{K,1}^i|^\beta, |\breve{h}_{1,2}^i|^\beta, \ldots, |\breve{h}_{K,2}^i|^\beta, \ldots, |\breve{h}_{1,M}^i|^\beta, \ldots, |\breve{h}_{K,M}^i|^\beta\})^{-1}, \\ &for \quad i = 1,2,\ldots,P^2, \end{aligned} \tag{5.43}$$

and a global weighting matrix for the whole system:

$$\mathbf{W}_{rel} = blkdiag\{\mathbf{W}_{rel}^1, \mathbf{W}_{rel}^2, \ldots, \mathbf{W}_{rel}^{P^2}\}; \tag{5.44}$$

by left-multipliyng the regressor and the right-side of a LS system by this matrix, then the relative error will be minimized.
Note that when the FPSK algorithm is implemented in the relaxed formulation, the weighting can be applied to each decoupled regressor defined in (5.25), to obtain the proper weighted matrix

$$\mathbf{P}_i^{rel} = \mathbf{W}_{rel}^i \mathbf{P}_i, \tag{5.45}$$

of which the QR factorization will be computed.

5.5 Frequency Mask

Several applications require the macromodels to catch the system behaviour with superior precision in some prescribed frequency bands rather than in others. For example, power systems, electronic systems for telecomunications and all the other applications that are intended to work at a prescribed frequency, must be modeled with high accuracy in that bandwidth. Also the precision of the model at the DC point is crucial for several applications, in particular for those which rely on the linearization around a given operating point. For these reasons, one can tune the fitting algorithm to maximize the model's precision at given frequency points. The goal can be achieved, again, by means of a modification of the cost function, giving more importance to the residuals associated to particular frequencies. To this aim, we can define a frequency mask:

$$w(f) : \mathbb{R}_0^+ \to \mathbb{R}^+ \tag{5.46}$$

that, given a value of frequency, returns a value that can be used to weight the related residual in the cost function: the greater is the value of the frequency mask, the greater will be the precision achievable at the corresponding frequency. By sampling the frequency mask at the frequencies at which the frequency data are collected, we obtain the vector:

$$\boldsymbol{w}_{freq} = (w(f_1), w(f_2), \ldots, w(f_K))^T. \tag{5.47}$$

With these samples we can modify the cost function and write:

$$J_{freq}(\boldsymbol{x}) = \sum_{i=1}^{P^2} \sum_{k=1}^{K} \sum_{m=1}^{M} \left| w(f_k) \left[\frac{N^i(\mathrm{j}2\pi f_k; \vartheta_m; \boldsymbol{x}) - D^\nu(\mathrm{j}2\pi f_k; \vartheta_m; \boldsymbol{x})\, \breve{h}^i_{k,m}}{D^{\nu-1}(\mathrm{j}2\pi f_k; \vartheta_m; \boldsymbol{x})} \right] \right|^2. \tag{5.48}$$

To minimize this cost function in LS sense within the PSK algorithm, with the hypothesis of relaxed formulation, we can modify the regressor in (5.25). We define the matrices

$$\begin{aligned} &\mathbf{F} = diag\{\boldsymbol{w}_{freq}\}, \\ &\mathbf{W}^*_{freq} = \mathbb{I}_{M \times P^2} \otimes \mathbf{F}; \end{aligned} \tag{5.49}$$

left-multiplication of the regressor in (5.22) by $\mathbf{W}^*_{freq}$ ensures that the desired frequency mask will be applied to the fitting precision.
For what concerns the FPSK and our implementation, we do not need the matrix $\mathbf{W}^*_{freq}$, but we can use a smaller matrix:

$$\mathbf{W}_{freq} = \mathbb{I}_M \otimes \mathbf{F}; \tag{5.50}$$

with this matrix we can modify all the decoupled regressors defined in (5.25) as follows:

$$\mathbf{P}_i^{freq} = \mathbf{W}_{freq} \mathbf{P}_i \tag{5.51}$$

before to compute the related QR factorizations.

5.6 Fixed Denominator Coefficients

During the identification procedure, one may decide to fix the value of some prescribed denominator coefficient. It is for example the case described in Section 2.2.3, when one wants to fix the value of d_0. For practical purposes, when one implements the FPSK algorithm, it is sufficient to modify the decoupled systems:

$$\mathbf{P}_i \begin{pmatrix} \boldsymbol{c}^i \\ \boldsymbol{d} \end{pmatrix} \approx \mathbf{0} \tag{5.52}$$

by moving the regressor columns associated to fixed denominator coefficients on the right-hand side of the LS problem, multiplied by the respective coefficient value. This ensures that the remaining coefficients are optimized with respect to the values that we want to impose to the fixed ones. We observe that doing this, the imposition of fixed denominator coefficients can be handled together with any modification of the cost function (i.e.minimization of the relative error or frequency masking) in a straightforward way: it is sufficient to weight the matrices $\mathbf{P}_i$ before building the non-zero right-hand sides.

5.7 Enforcing a Positive Real Denominator

If we want to guarantee the global stability of the final macromodel, we can constraint the LS problem (5.25) in order to guarantee that the for every value of the parameter space the denominator of the model is positive real. In order to do this, since the final poles of the model are implicitly parameterized by the parameter-reated basis functions, one must enforce the condition of positive-realness through all the parameter space. A positive lower bound for the values assumed by the real part of the denominator is imposed by means of a set of constraints added to the LS system, whose solution is found by means of a convex optimization procedure. The constraints are related to the points in the parameter space for which fitting data-points are avalaible and to additional points obtained by means of an adaptive sampling procedure that collects the points for which the denominator of the model at the previous iteration is not positive real.
When the model is returned by the FPSK algorithm, a post-processing step is performed to guarantee that the final model is PR: the coefficients of the denominator are perturbed in order to ensure the stability and a new set of coefficients for the numerator is computed again to obtain the optimal fitting with respect to the new denominator. For further details about this procedure see [36].
We point out that when we apply the set of constraints that guarantees a PR denominator, the non-triviality constraint necessary for the relaxed formulation of the FPSK can be avoided, since the trivial solution is ruled out by the presence of the condition of positive real denominator.

5.8 Input and Output Descriptions

The algorithm we propose is driven by the following set of inputs:

1. **Data**: a MATLAB structure in which the data for the identification are collected. Usually the data coming from full-wave solvers or physical measurements are provided in TOUCHSTONE files, so a preprocessing step is required to cast them in the desired multi-dimensional array form. One must embed in the Data structure also the information about the parameters, including the values at which the frequency responses are collected, the physical units of the numeric values, the model representation and which set of responses must be used for the identification rather than for the validation.

2. **Order of the model**: a number that defines the order of the model, that is, the number of its poles. We will indicate it with $\bar{n}$, as in the previous chapter.

3. **Parameters basis degree for the numerator**: specifies the number of basis functions that will be used to fit the numerator dependency on the parameter variation. The basis functions for the frequency domain are the partial fractions.

4. **Parameters basis degree for the denominator**: specifies the number of basis functions that will be used to fit the denominator dependency on the parameter variation.

5. **Basis choice**: allows to chose one among the following basis to interpolate the parameter-dependent behaviour: monomials, Chebychev polynomials, harmonic functions.

6. **Options**: a MATLAB structure embedding all the options that will drive the algorithm.

As output, the algorithm returns the following:

1. **Model**: the macromodel returned by the modeling algorithm.

2. **Fitting and validation errors**: the differences between the data and the model, in relative and absolute sense.

3. **Errors**: possible errors encountered by the algorithm.

5.9 Modeling Options

The modeling procedure of parameterized multiports is achievable through the FPSK algorithm in the basic form exposed in chapter two; anyway, often the modeling procedure can be performed best with a series of case-dependent slight modifications of the problem formulation.
For this reason, the code we built is controlled by a set of options, to be embedded in a MATLAB structure, that customize the algorithm in the most proper way according to the application. The user is allowed to set the following options:

- **Target accuracy**: sets the accuracy required by the user to the macromodel. The iterations of the FPSK stop if the overall error between the data and the model is less than the prescribed value. If the user does not express any preference,

than the target accuracy (indicated in the following as ϵ) is set to 0.001. (Syntax: *Options.TargetAccuracy* = ...)

- **Maximum number of iterations** : sets the limit for the number of iterations to be performed by the algorithm; if the maximum is reached, the algorithm stops and the model is returned as it is at the last iteration. If the user does not express any preference, then the maximum number of iterations, (indicated in the following as $\bar{N}$) is set to 10. (Syntax: *Options.MaxIterations* = ...)

- **Fixed basis poles**: allows to impose a fixed set of poles for the partial fraction basis that will be used during the identification procedure. They are defined as in Section 2.2.2, with linear distribution. If the user does not require a set of fixed basis poles, than a VF estimation of the poles is performed as explained in Section 5.1.2. Note that a set of fixed basis poles would coincide with the actual poles of the model only if it is univariate (Syntax: *Options.FixedBasisPoles* = 1/0)

- **Relaxed Normalization**: if this option is set to one, than both the preliminary VF and the FPSK iterations will be solved using the relaxed formulation of the least squares problem. This is also the standard choice for the algorithm: if the user does not express other preferences, than the relaxed formulation will be used. (Syntax: *Options.RelaxedNormalization* = 1/0)

- **Fixed denominator coefficients**: this options is mutually exclusive with the one related to the relaxed normalization. It allows the user to fix the value of certain coefficients of the denominator series expansion to a fixed value. (Syntax: *Options.FixedDenCoeff* = $[d_0, d_1, d_2, ..., d_n]$*; insert nan for the coefficients that must not be fixed*)

- **Minimization of the relative error**: this option allows to identify the model in such a way that the relative error between the model and the data is minimized instead of the absolute error. This grants the possibility to recontruct responses with high dynamic range. If the user does not express his preference, this option is set to zero and the absolute error is minimized instead. This option has two companion options that regulate its influence on the cost function modification:

  1. **Weight power**: this parameter regulates the influence of the weighting in the least squares problem; it coincides with the exponent β used in Section 5.4. If the user does not express any preference, it will be set to 1.(Syntax: *Options.weightPower* = ...)
  2. **Weight saturation** : this parameter defines a saturation threshold for the values of the weights applied to the least squares problem in order to avoid numerical issues; the threshold value is the inverse of this parameter.
     If the user does not express any preference, it will be set to 10^{-8}.(Syntax: *Options.weightDeltaMin* = ...)

  (Syntax: *Options.MinimizeRelError* = 1/0)

- **Frequency masking**: if this option is activated, then all the least squares problems solved during the modeling procedure will be weighted in such a way that a

prescribed frequency range will assume more importance with respect to the others; as a consequence, the fitting of the model versus the data will be better in that frequency band with respect to the others. This option is set by means of two specifications that regulate its impact on the cost function modification:

1. **Frequency range**: it defines the frequency range whose fitting will be enhanced. The possible choices are DC point, low frequencies, middle frequencies, high frequencies (defined automatically by the algorithm on the basis of the frequency data points). The current implementation of the algorithm provides a basic piecewise-costant frequency mask: the mask value is equal to one for the frequencies outside the chosen frequency range and equal to a constant value greater than one for the frequencies inside the chosen frequency range.
2. **Frequency weight**: It is the constant value assumed by the frequency mask within the prescribed frequency range whose fitting must be enhanced. If no preference is expressed by the user, it will be set to 20.(Syntax: *Options.freqWeight* = ...)

(Syntax: *Options.FreqMask='LowPass'/'HighPass','MidFreq','EnhanceDC'*)

- **Return a positive real denominator**: if this option is chosen, then the constrained identification explained in Section 5.7 will be performed. The choice of this option automatically implies a relaxed formulation of the algorithm and, consequently, can not be chosen together with the presence of fixed denominator coefficients. This option has a set of companion options that drive the imposition of the stability in the desired way:

  1. **Solver selection**: it allows the user to choose one among the avalaible solver that can be used to solve the constrained least squares problem. The possible choices are:

     (a) CVX solver;

     (b) MyIPM: a solver provided by Prof. S. Grivet-Talocia;

     (c) lsqlin: native MATLAB solver.

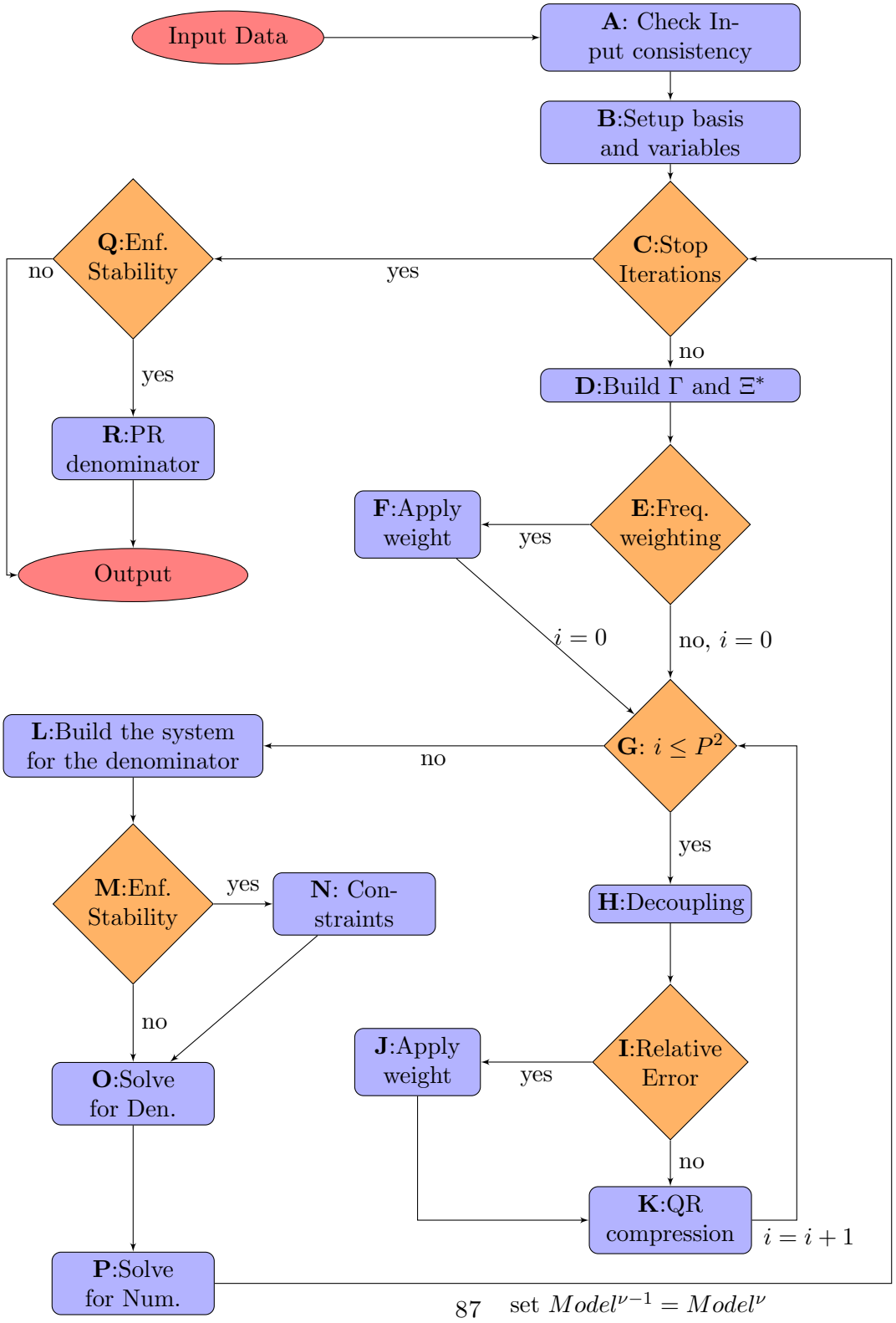
     (Syntax: *Options.solverSelection* = 0/1/2/3; 1,2,3 follows the above ordering; 0 makes the code check sequentially the presence of the solvers listed above, following the hierarchy induced by the ordering.)
  2. **Maximum iterations of denominator perturbation**: it defines the maximum number of cycles that can be performed in the post-processing step for the ensurance of a positive real denominator. If no preference is expressed by the user, then this limit is set to 10. (Syntax: *Options.MaxIterPR* = ...)
  3. **Predictive constraints imposition**: if this option is set to 1, then an adaptive method for the definition of the constraints is performed. This ensures that the process of perturbation of the denominator converges to a solution. To enable this option, one must explicitly set it to 1, otherwise a standard perturbation scheme will be applied. (Syntax: *Options.Predictive* = 1/0)

(Syntax: *Options.Predictive* = 0/1)

5.10 Flowchart and Algorithm Description

We now explain the structure of the implemented algorithm in detail. We will make use of a flowchart whose blocks will be denoted with capital letters in order to be identified and explained in depth one by one.
Some consideration can be useful to understand the organization of the code. We are implementing the FPSK with a number of options on which the structure of the LS systems to be solved at each iteration depends; in particular, since FPSK relies on an individual processing of the port-dependent responses, we organized the code in such a way that all the port-dependent operations are performed inside a single loop (the block **G** of Figure 5.1), while all the operations that are not port-dependent are performed before this loop. In particular, the two different weighting procedures, the minimization of the relative error and the application of a frequency mask, are handled differently: with the hypothesis that every port responses share the same number of frequency samples K, the application of the frequency weighting is not port-dependent and will be applied first; on the contrary, the minimizaton of the relative error is based upon a data-dependent weighting, that must be performed port-by-port, separately.
In the following, we labeled each block with a name that is representative of the main operation performed at each stage; nevertheless, each block is representative of a major number of operations with respect to the one we highlighted. Of those operations, we describe the most important from a theoretical point of view, while we neglect some technical implementation details that are not relevant for the understanding of the algorithm structure.

Figure 5.1: Flowchart of the implemented FPSK algorithm; each block is labeled with a capital letter and is explained in the following.



A: the first stage of the algorithm is intended to make sure that the inputs given to the algorithm are consistent; in particular, here we first check that the sum of the cardinalities of the parameter basis for numerator and denominator does not exceed the number of samples in the parameter space, in order to guarantee the consistency of the LS problem; if it is not, a warning is returned to the user. Second, we check the Options structure and initialize to the default values all the fields that have not been speciefied by the user.

B: in the second stage we setup the basis functions and the necessary variables. In particular, the number of input-output ports of the system, the dimensions of the data set, the samples necessary for the validation, the model representation. Also the FPSK weights are initialized to unity to perform the first iteration.
For the frequency-dependent basis, we chose the partial fractions: the poles are chosen by means of a vector fitting iteration performed on the univariate model obtained fixing the parameters values to the mean values of every single parameter interval. The vector fitting is performed by an external function that inherits all the options that influences both PSK and VF, in order to obtain a consistent final estimate. Alternatively, if the option "FixedBasisPoles" is set to 1, then a set of linearly distributed basis poles is generated, as explained in Section 2.2.2.
The basis functions for the parameters are initialized: if a normalization of the parameters domain is required (for example, for the Chebychev polynomials basis), then the extrema of the parameters intervals are extracted from the input Data structure.
Once initialized, the basis functions are computed at the fitting points of the domain and kept in memory to be used when necessary.

C: this decision block handles the FPSK iterations: it is implemented through a *while* loop that stops under two conditions: the iteration number exceeds the maximum number of iterations $\bar{N}$ prescribed by the relative option, or, the accuracy of the model is superior to the prescribed accuracy ϵ.

D: here we build the matrices $\mathbf{\Gamma}$ and Ξ^*. With Ξ^* we denote the matrix:

$$\Xi^* = \begin{pmatrix} \mathbf{W}^{\mu-1}(\vartheta_1)\mathbf{\Phi}(\vartheta_1) \\ \mathbf{W}^{\nu-1}(\vartheta_2)\mathbf{\Phi}(\vartheta_2) \\ \mathbf{W}^{\nu-1}(\vartheta_3)\mathbf{\Phi}(\vartheta_3) \\ \vdots \\ \mathbf{W}^{\nu-1}(\vartheta_M)\mathbf{\Phi}(\vartheta_M) \end{pmatrix}, \tag{5.53}$$

defined with the same notations used in Section 5.1.2.
Since these two matrices are related only to the samples of the basis and to the values of the denominator at the previous iteration, they are equal for every port and are used as a starting point for the computation of the port-dependent regressor matrices $\mathbf{P}_i$.

E,F: at this point we check if the user wants to apply a frequency mask to tune the accuracy of the fitting problem. If the answer is yes, then we pass through the block **F**, where the weights $\mathbf{W}_{freq}$ are applied to the matrices Γ and Ξ^* as described in section

5.5.
Two mutually exclusive operations are performed now: if we require a PR denominator, then the necessary constraints are computed, otherwise, the non triviality constraint is computed.
If the minimization of the relative error is not required, then all the blocks $\mathbf{\Gamma}$ of the decoupled regressors $(\mathbf{\Gamma}, \mathbf{\Xi}_i)$ are equal and a QR factorization of the matrix Γ is performed, in order to compute faster the $\mathbf{R}_i^{22}$ matrices, as indicated in Section 3.1.3:

$$\mathbf{\Gamma} = \mathbf{Q}_\Gamma \mathbf{R}_\Gamma. \tag{5.54}$$

This operation is not possible if the optimization is performed with respect to the relative error, since the weights to be applied are port-dependent and the regressors $\mathbf{P}_i$ do not share the same block $\mathbf{\Gamma}$.

G: this loop is intended to perform all the QR decompositions that we need to build the regressors for the computation of the denominator coefficients $\boldsymbol{d}$ in (5.26). For every iteration of the loop the factorization of the decoupled regressor $\mathbf{P}_i$ is computed and the regressor of (5.26) is built adding the current $\mathbf{R}_i^{22}$ block below the previous ones.

H: inside the loop we first obtain the current $\mathbf{\Xi}_i$ matrix from the $\mathbf{\Xi}^*$ matrix:

$$\begin{aligned} &\mathbf{\Xi}_i = \breve{\mathbf{H}}_i \mathbf{\Xi}^*, \quad with \\ &\breve{\mathbf{H}}_i = blkdiag\{\breve{\mathbf{H}}_i(\vartheta_1), \breve{\mathbf{H}}_i(\vartheta_2), ..., \breve{\mathbf{H}}_i(\vartheta_M)\}; \end{aligned} \tag{5.55}$$

here the matrices $\breve{\mathbf{H}}_i(\vartheta_1), \breve{\mathbf{H}}_i(\vartheta_2), ..., \breve{\mathbf{H}}_i(\vartheta_M)$ are defined as in (5.24).

I,J: the decision block **I** checks if the minimization of the relative error option has been chosen by the user. If it is so, then we move to block **J**, where the weights are applied to return a new matrix to be factorized:

$$\mathbf{P}_i^{rel} = \mathbf{W}_{rel}^i \mathbf{P}_i. \tag{5.56}$$

The matrices

$$\mathbf{\Gamma}_i = \mathbf{W}_{rel}^i \mathbf{\Gamma} \tag{5.57}$$

are kept in memory to be used later, for the computation of the numerator coefficients.
If the absolute error must be minimized the block **J** is ignored and the matrix to be factorized in the next block is simply $\mathbf{P}_i$.

K: now the QR factorizations are performed: with respect to the chosen options, a variety of cases must be distinguished:

- There are fixed denominator coefficients: in this case, before to perform the factorization, the regressor columns associated to the fixed coefficients are multiplied by the prescribed coefficients and are used to build the right-hand side of the system related to the i-th port, the vector $\boldsymbol{b}^i$, that is saved in memory to be used later. The presence of a non-zero right-hand side forces us to compute both the factors of the QR decomposition, since every $\boldsymbol{b}^i$ vector must be projected over the basis defined by $\mathbf{Q}_i$; for this reason, here the QR factorizations must be performed from scratch. At the end of this stage the matrices $\mathbf{Q}_i, \mathbf{R}_i$ relative to the i-th port will be avalaible.

- If the relaxed formulation of the FPSK is chosen, then the $\mathbf{Q}_i$ factors must not be computed, since the LS system associated to the denominator coefficients is homogeneous. If the minimization of the relative error is required, than the factorization of $\mathbf{P}_i^{rel}$ must be performed from scratch, otherwise, we can exploit the matrices $\mathbf{Q}_\Gamma, \mathbf{R}_\Gamma$ computed in the block **F** and formula (3.1.3) to compute $\mathbf{R}_i^{22}$.

Every time a factor $\mathbf{R}_i$ is computed, the relative block $\mathbf{R}_i^{22}$ is extracted and used to build iteratively the regressor in equation (5.26).

L: the LS problem for the computation of the denominator coefficients is now set up: the regressor is given as output of the previous stage, so we need to setup the right-hand side term. If the relaxed formulation is used, the system is homogeneous, the right-hand side term is the null vector and the resulting LS problem equals the one in (5.26). If, otherwise, there are fixed denominator coefficients, then the right-hand side term must be built. For every vector $\boldsymbol{b}_i$, we apply the projection

$$\mathbf{Q}_i^H \boldsymbol{b}_i = \begin{pmatrix} \boldsymbol{b}_1^i \\ \boldsymbol{b}_2^i \end{pmatrix} \tag{5.58}$$

and we extract the vector $\boldsymbol{b}_2^i$, for which we want to enforce the approximation

$$\mathbf{R}_i^{22} \boldsymbol{d}^* \approx \boldsymbol{b}_2^i \tag{5.59}$$

where $\boldsymbol{d}^*$ denotes the denominator coefficients that are not fixed. The resulting system reads:

$$\begin{pmatrix} \mathbf{R}_1^{22} \\ \mathbf{R}_2^{22} \\ \vdots \\ \mathbf{R}_{P^2}^{22} \end{pmatrix} \boldsymbol{d}^* \approx \begin{pmatrix} \boldsymbol{b}_2^1 \\ \boldsymbol{b}_2^2 \\ \vdots \\ \boldsymbol{b}_2^{P^2} \end{pmatrix} \tag{5.60}$$

M,N,O: At this stage the algorithm checks if the user requires a globally stable macromodel; if yes, then the constraints that grant a positive real denominator are added to the system, the operations needed to perform the constrained LS are executed and the denominator coefficients are found; otherwise the non-triviality constraint computed at step **E** is added as the last row of system (5.26) and the denominator coefficients are found by means of a standard LS problem.

P: once the denominator coefficients are avalaible we can proceed with the computation of the numerator unknowns. Also in this stage, a variety of cases must be distinguished.

- If we are minimizing the absolute error, then the system to be solved is the one we defined in (5.27); if there are no fixed denominator coefficients, then we already computed the matrices $\mathbf{Q}_\mathbf{\Gamma}$, $\mathbf{R}_\mathbf{\Gamma}$ in block **E** and we can exploit them to solve the system efficiently. If it is not so, then the QR factorization of $\mathbf{\Gamma}$ is not avalaible and one can perform a LU factorization of $\mathbf{\Gamma}$ to solve the system.

- If we are minimizing the relative error, then for each response we must solve the system

$$\boldsymbol{\Gamma}_i \boldsymbol{c}^i \approx \boldsymbol{b}^i_{num}, \tag{5.61}$$

where $\boldsymbol{c}^i$ is the vector that embeds the numerator coefficients relative to the i-th response and $\boldsymbol{b}^i_{num}$ is defined has

$$\boldsymbol{b}^i_{num} = -\boldsymbol{\Xi}_i \boldsymbol{d}. \tag{5.62}$$

Also in this case, since a QR factorization of the matrices $\boldsymbol{\Gamma}_i$ is not avalaible, we can perform a LU factorization of the regressors to solve the systems efficiently.

At the end of this stage, the model coefficients are avalaible and the model at the current iteration is used to compute the FPSK weights for the successive iteration.

Q,R: If the stop conditions of the FPSK are satisfied, then we check if the model denominator is required to be positive real. If it is so, then the final stage of denominator perturbation is performed, as explained in [36]. Otherwise, no modification is applied to the model.

Output: The outputs of the algorithm are returned to the user.

Chapter 6

Test Cases and Results

Here we present a set of numerical results achieved by our algorithm. We will focus on the numeric precision of our estimate, reported in terms of residual errors, and we will provide illustrative plots, where we show the fitting of the obtained models versus the data; further, we will compare the execution time of our FPSK algorithm with respect to the standard PSK. Every example is designed in order to stress the algorithm in some particular aspect: highly variational frequency-parameter behavior, high dynamic range, large number of ports, multiparametric case; for the results regarding the imposition of stability whereas an unconstrained identification would not guarantee it, see [36]. The examples are presented together with a physical description of the structure under modeling or with a basic undelying circuit description if the data were not provided with a full characterization of the system.
All the examples are driven by data collected in scattering representation, so we will refer to each single response of transfer matrices with the symbol $S(i, j)$.
If not differently specified, the test were performed on a laptop with 8 GB of memory and a two cores processor running at 2,6 GHz clock speed.

6.1 PCB Interconnect Over a Slotted Reference Plane

With this example we present a structure that requires a high cardinality of both frequency and parameter basis in order to obtain a satisfactory fitting.

6.1.1 Structure Description

The structure we are modeling is depicted in figure 6.1; it is a microstrip running over a dielectric substrate with an underlying ground plane. The microstrip current return path in the ground plane is interrupted by a rectangular slot, placed at distance d from the center of the reference plane. This distance is the design variable parameter of our problem.

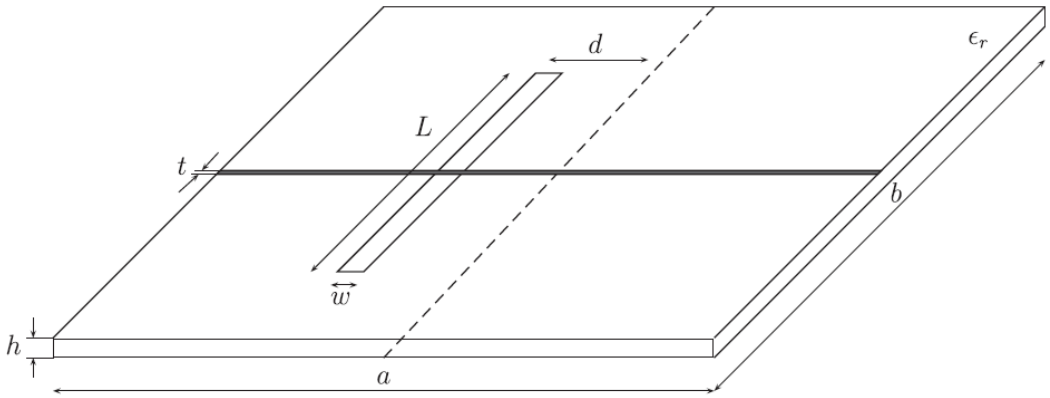


Figure 6.1: Schematic representation of the structure under modeling.

The geometric (fixed) parameters of the structure are:

- $a = 100$ mm;
- $b = 100$ mm;
- $\epsilon_r = 4.7$;
- $t = 0.035$ mm;
- $w = 0.12$ mm;
- $h = 0.3$ mm;
- $L = 25$ mm.

We are interested in the transmissions and the reflections over the microstrip terminations, so we will build a two-ports macromodel.

6.1.2 Results

To build the macromodel we started with a dataset obtained by means of 15 different electromagnetic full-wave simulations, each one performed for a different value of the design parameter, the slot offset, varying from a minimum of $\theta_{min} = 1$ mm to a maximum of $\theta_{max} = 25$ mm. Every simulation returned a set of 1858 scattering samples in the frequency domain, with $f_{min} = 0$ Hz and $f_{max} = 10$ GHz. The total number of samples was therefore 27870.
The order of the model has been set to $\bar{n} = 34$, while the cardinality of the parameter

basis (the Chebychev polynomials) is 11 for the numerator and 4 for the denominator. The total number of unknowns returned by the FPSK is then:

$$N_{coeff} = P^2 \times (\bar{n}+1) \times \bar{\ell}_{num} + (\bar{n}+1) \times \bar{\ell}_{den} = 1680 \tag{6.1}$$

We ran the algorithm to minimize the absolute error of the model versus the data, performing 10 iterations of the FPSK scheme; the problem was unconstrained, so the final model is not guaranteed to be stable. The mean time required for each iteration is:

$$T_{FPSK} \approx 6.8377s, \tag{6.2}$$

While a single iteration of standard PSK algorithm required:

$$T_{PSK} \approx 543.84s, \tag{6.3}$$

To compute the fitting error we applied the following criterion: we computed the errors for each response i and for each sample θ_m of the parameter,

$$\begin{aligned} AbsError_m^i &= \sqrt{\frac{1}{K}\sum_k^K \left| H^i(\mathrm{j}2\pi f_k, \theta_m) - \breve{h}_{k,m}^i \right|^2} \\ RelError_m^i &= \sqrt{\frac{1}{K}\sum_k^K \left| \frac{H^i(\mathrm{j}2\pi f_k, \theta_m) - \breve{h}_{k,m}^i}{\left|\breve{h}_{k,m}^i\right|} \right|^2}, \end{aligned} \tag{6.4}$$

and, for each port, we considered the maximum among all the M resulting errors

$$\begin{aligned} AbsError^i &= \max_m AbsError_m^i \\ RelError^i &= \max_m RelError_m^i. \end{aligned} \tag{6.5}$$

We obtained the following results:

Response	Absolute Error	Relative Error
$S(1,1)$	5.62×10^{-3}	1.15×10^{-2}
$S(1,2)$	3.86×10^{-3}	6.22×10^{-3}
$S(2,1)$	3.86×10^{-3}	6.21×10^{-3}
$S(2,2)$	5.40×10^{-3}	1.10×10^{-2}

figures 6.2 and 6.3 report the plots of the resulting model versus the data for two particular responses.

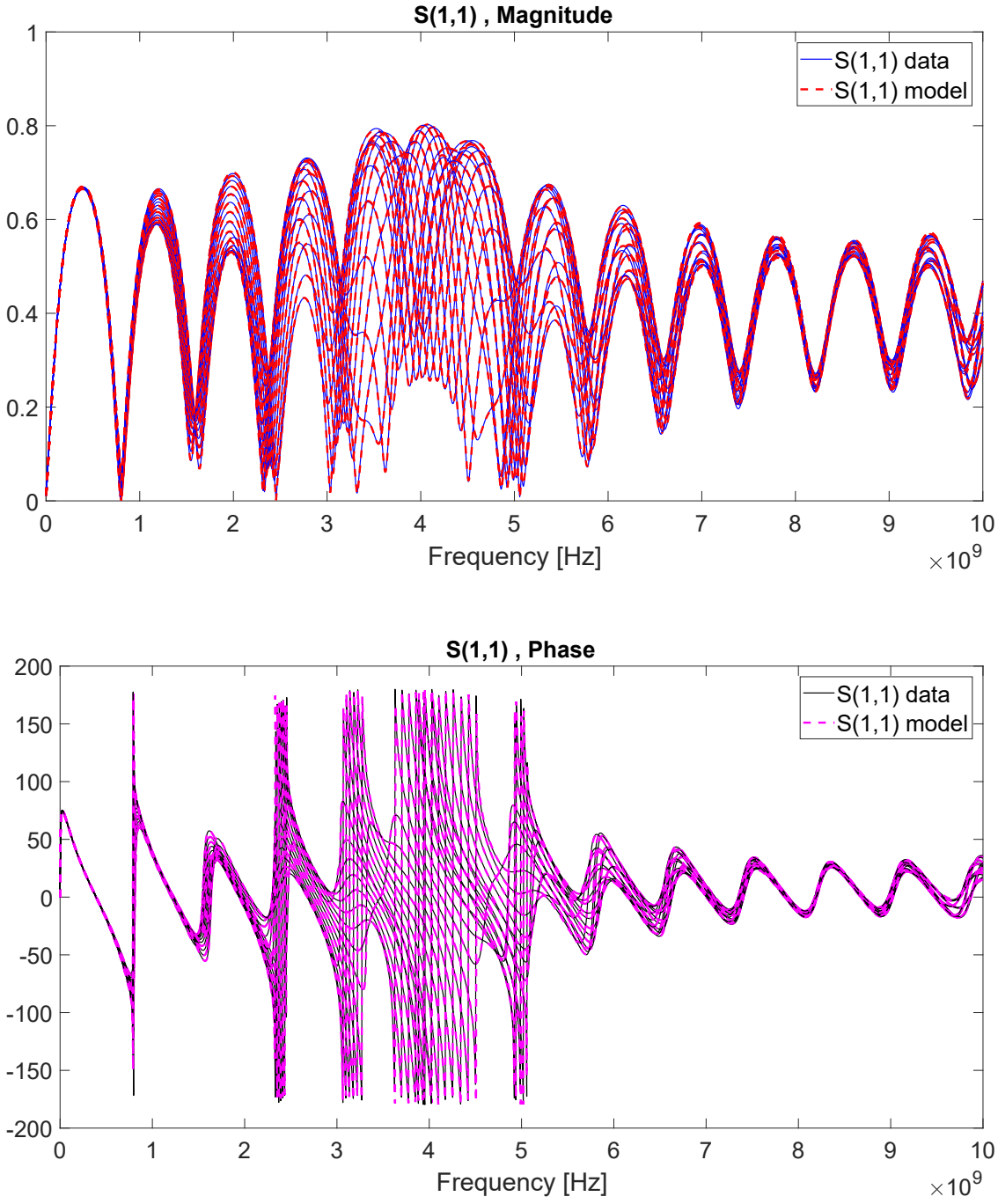


Figure 6.2: Magnitude and phase fitting of the element $S(1,1)$ of the transfer matrix

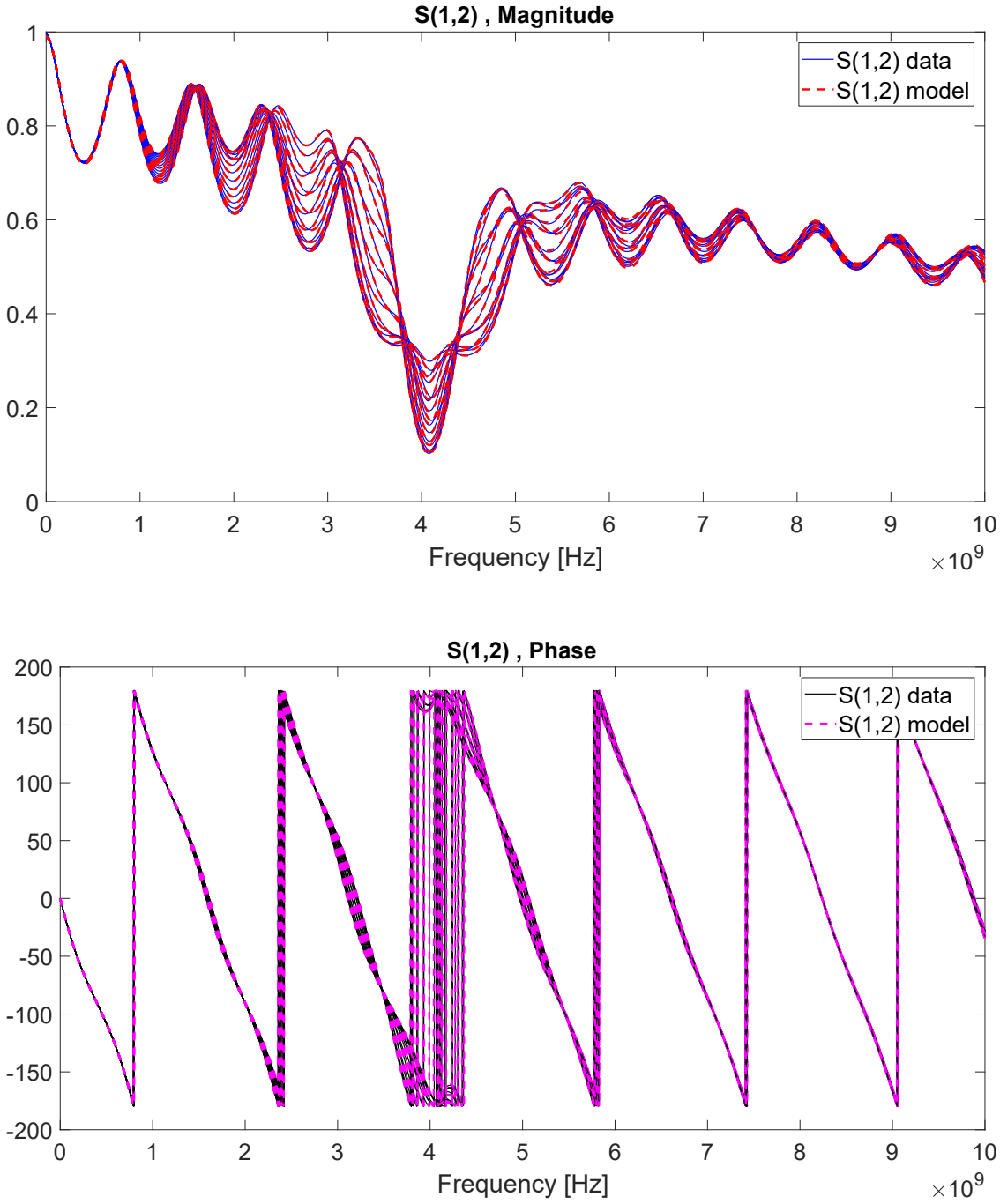


Figure 6.3: Magnitude and phase fitting of the element $S(1,2)$ of the transfer matrix

6.2 Coupled Transmission Lines

With this example we will compare the scaling of PSK and FPSK algorithms with respect to the number of ports of the system. Further, we will test the minimization of relative

error strategy.

6.2.1 Structure Description

The structure under modeling is a set of J differential pairs of parallel wires, reported in figure 6.4, disposed one next to the others, with the following characteristics:

- wires lenght=10cm;
- radius of the conductors $r_w = 0.5$ mm;
- radius of the dielectric insulator $r_d = 0.8$ mm;
- relative permittivity $\epsilon_r = 4.2$;
- distance between the wires center $D = 1.61$ mm.

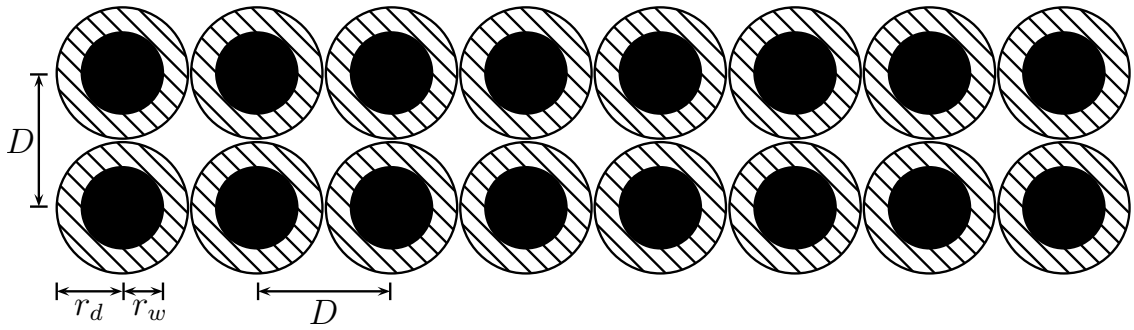


Figure 6.4: Schematic representation of the structure under modeling.

Every differential pair is considered as a decoupled transmission line for a length $L - L_c$, while over the length L_c the conductors are coupled in a $2J$ multiconductor transmission line. The length L_c is our free design parameter and its value will range between a minimum of $\theta_{min} = 20$ mm to a maximum of $\theta_{max} = 40$ mm. We simulated the system with a full-wave solver for J ranging from 1 to 18, obtaining a set of 9 structures with a number of ports that ranges from 2 to 36.

6.2.2 Results

We will test the scaling law of PSK and FPSK with respect to the number of ports ; further, we will compare the macromodel obtained for $J = 6$ with respect to the data after 7 iterations of the FPSK algorithm. To run the test, we used a server with 24 GB of memory, with a clock frequency of 2.2 GHz, in order to be able to run the PSK algorithm, which is particularly demanding in terms of memory requirements.
In figure 6.5 we report the execution time of a single iteration of PSK and FPSK as the number of ports P of the system increases. We can see how the linear scaling of FPSK with respect to the number of port responses is confirmed by the experiment; we note also that the native MATLAB solver exploited for the execution of the PSK

iteration performs better than what we expected: this is because of the sparsity pattern of the LS regressor associated to the PSK iteration, that is exploited by the internal matlab routines; anyway, we were not able to idenitfy models with more than 4 ports due to the memory requirements of PSK. We also see how the FPSK iteration with the minimization of the relative error requires more computational effort due to the necessity to calculate all the required QR factorizations indipendently, as opposed to the absolute error minimization case, which requires only a single QR factorization.

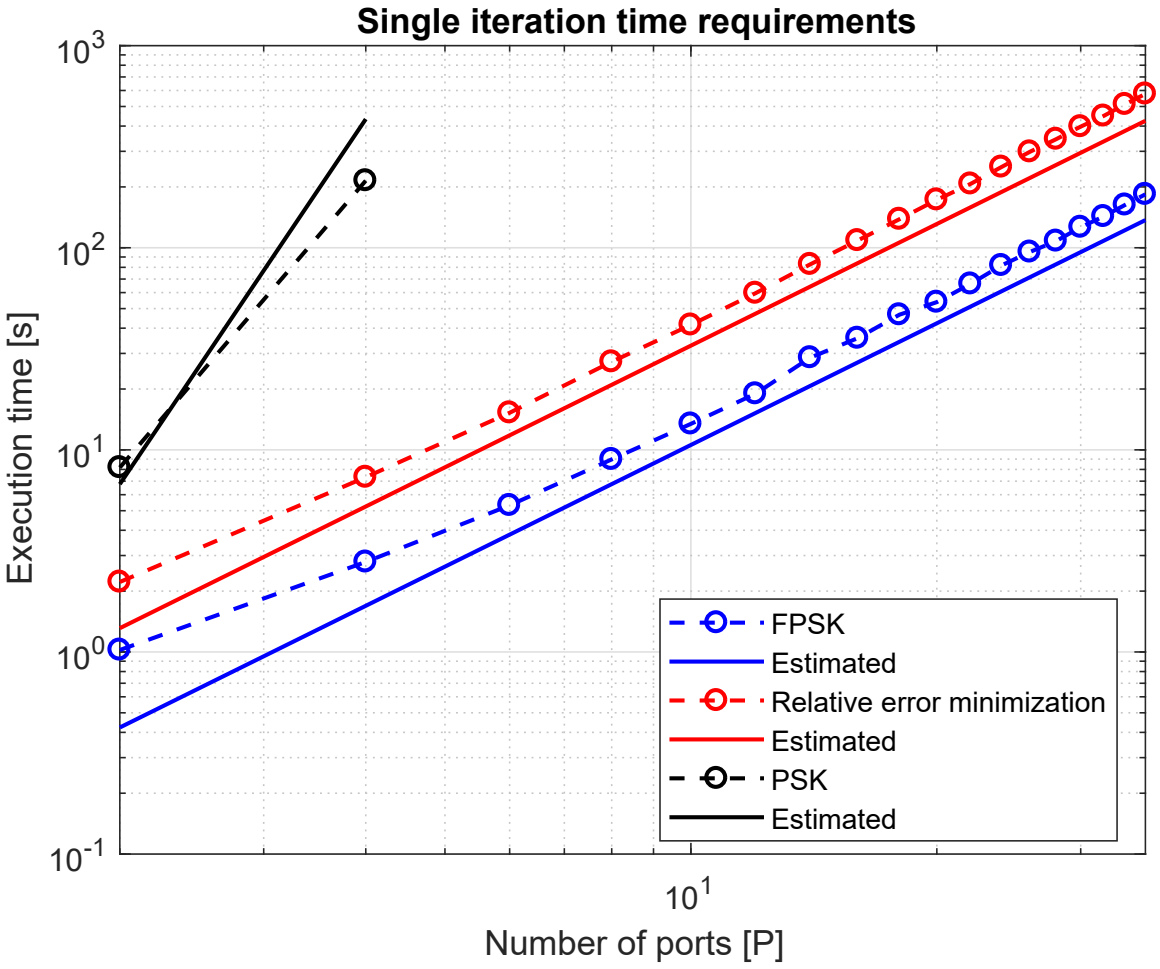


Figure 6.5: Time requirements for a single PSK and FPSK iteration. The FPSK algorithm requirements are reported for both absolute and relative error minimization.

To show the quality of the resulting macromodels we consider the case $J = 6$, that is modeled by means of a 12 ports system. To obtain the macromodel, we used a dataset of 11 scattering responses, sampled in the frequency domain with $f_{min} = 1$ Hz and $f_{max} = 5$ GHz; each of them embeds 500 frequency samples and is representative of the model behaviour as the coupling length L_c varies from θ_{min} to θ_{max}. The total number of samples is then 5500. The order chosen for the macromodel is $\bar{n} = 37$, while the

cardinality of the Chebychev polynomials used to interpolate the parameter-dependent behaviour is $\bar{\ell}_{num} = 6$ for the numerator and $\bar{\ell}_{den} = 5$ for the denominator; consequently, we are looking for $N_{coeff} = 33022$ unknown coefficients.
In figures 6.6 to 6.8 we show the results of the identification for a transmission, a reflection and a near-end crosstalk responses (S(7,1), S(1,1) and S(2,1) respectively). In figure 6.9, we show how the minimization of the relative error can enhance the fitting of weak responses: the far-crosstalk response S(7,4) is modeled with both absolute and relative error minimization to compare the results.

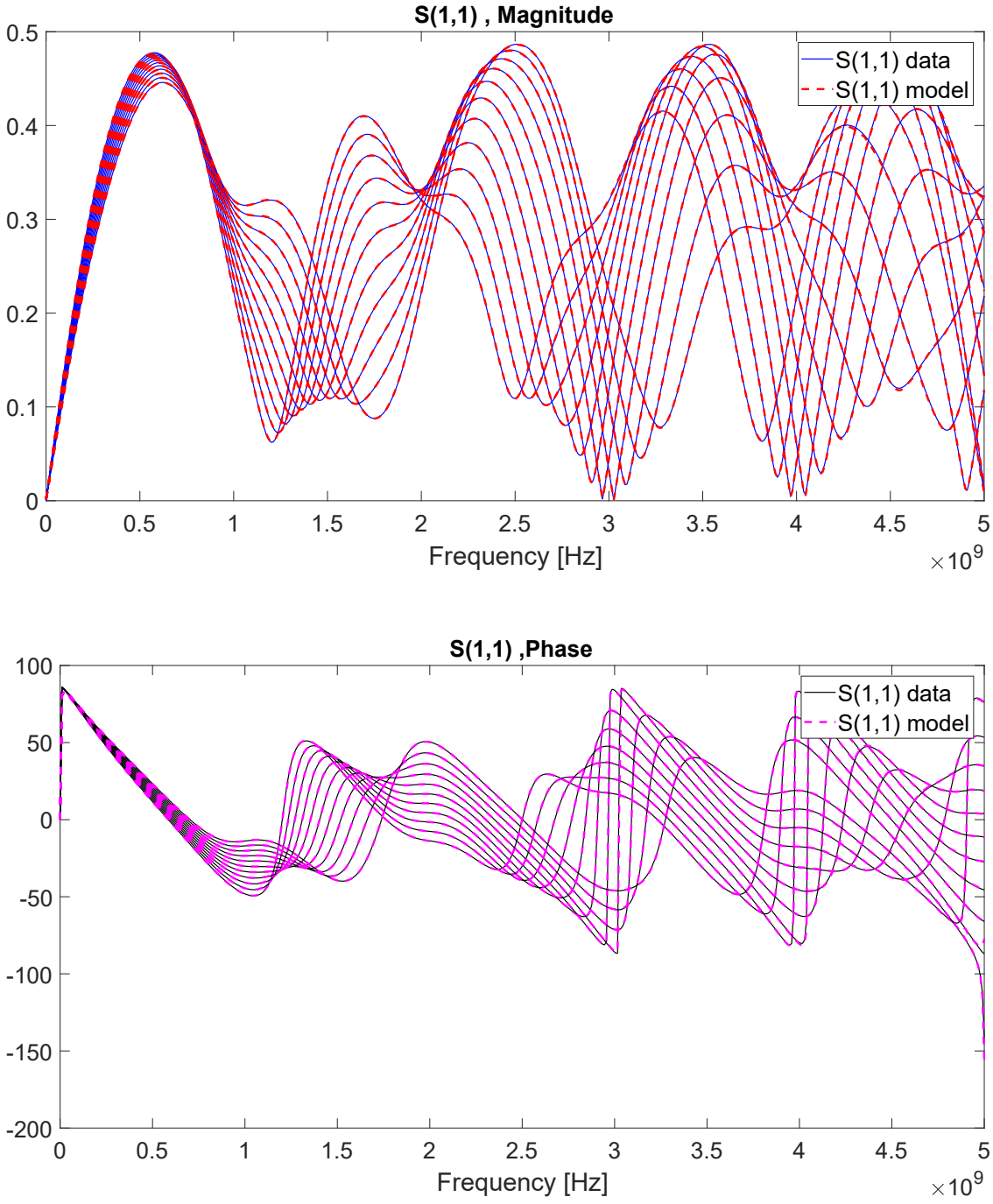


Figure 6.6: Magnitude and phase fitting of the element $S(1,1)$ of the transfer matrix; the absolute and the relative errors, computed as in (6.5) are 9.46×10^{-4} and 2.94×10^{-3} respectively.

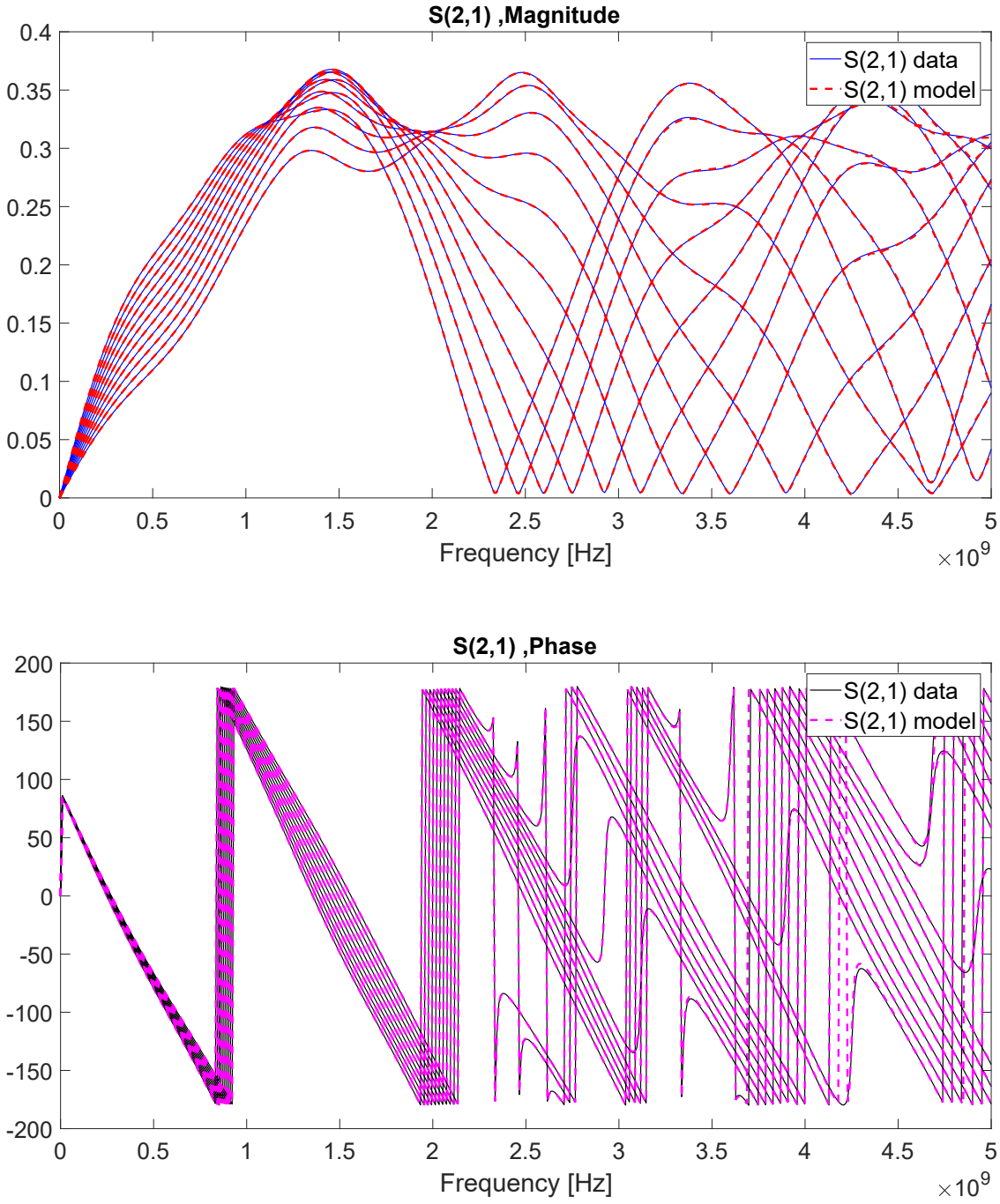


Figure 6.7: Magnitude and phase fitting of the element $S(2,1)$ of the transfer matrix; the absolute and the relative errors, computed as in (6.5) are 6.78×10^{-4} and 2.82×10^{-3} respectively.

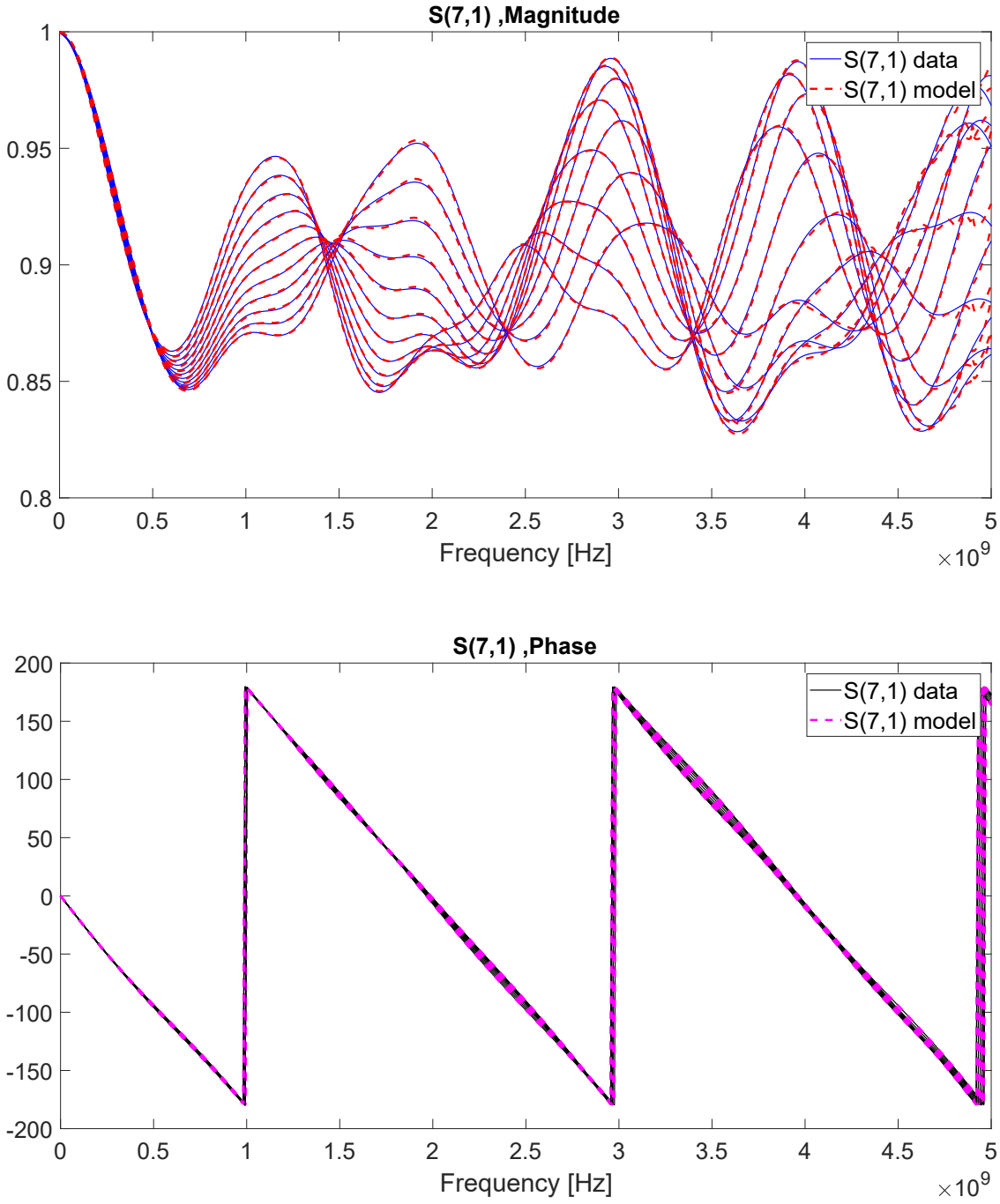
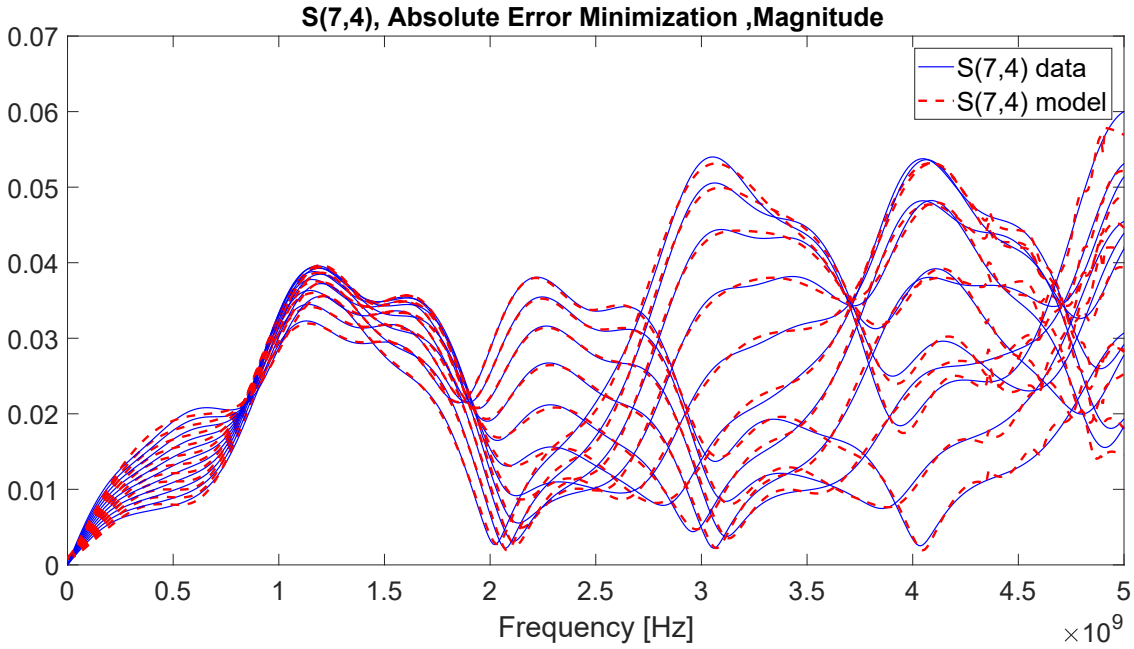


Figure 6.8: Magnitude and phase fitting of the element $S(7,1)$ of the transfer matrix; the absolute and the relative errors, computed as in (6.5) are 1.54×10^{-3} and 1.71×10^{-3} respectively.



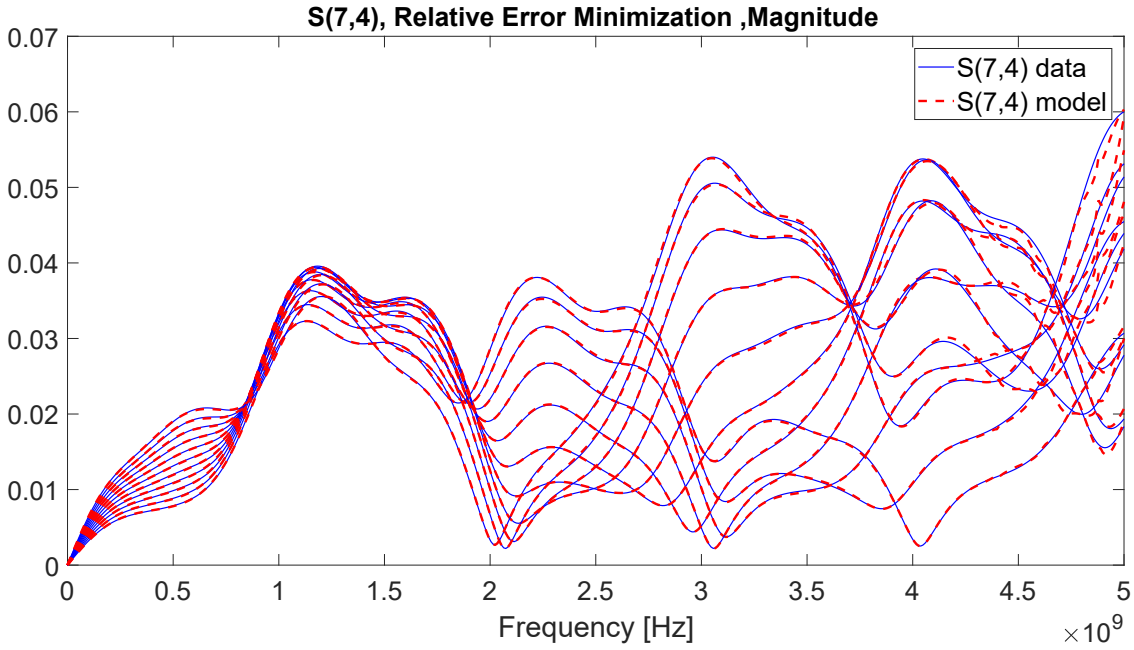


Figure 6.9: A comparison of the Magnitude fitting of a far-crosstalk performed with the two strategies of absolute and relative error minimization; with the first: $AbsErr = 9.88 \times 10^{-4}, RelErr = 3.26 \times 10^{-2}$; with the second: $AbsErr = 5.14 \times 10^{-4}, RelErr = 2.024 \times 10^{-2}$.

6.3 Via with Residual Stub

With this test-case, we demonstrate a frequency weighting applied to the macromodeling procedure; in particular,the main objective is to enhance the precision of the model at the DC point.

6.3.1 Structure Description

This structure (depicted in figure 6.10) is a via connecting a microstrip line and a stripline; the metallization process that is performed to create the via, running from top to bottom, generates a residual stub that is not necessary in order to guarantee the efficacy of the via and is likely to represent a source of issues for the signal integrity. For this reason, we want to parameterize the model behavior with respect to the stub height h, in order to run simulations, find an optimal value for the parameter and adjust the stub height through the backdrilling procedure.

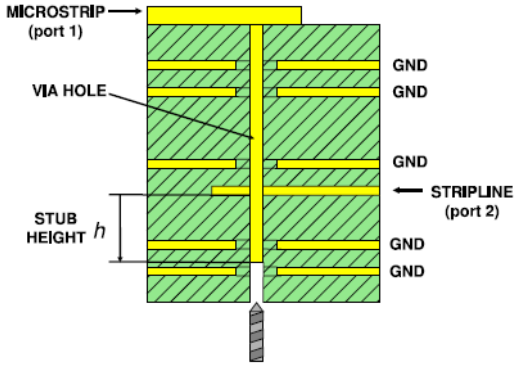


Figure 6.10: Schematic representation of the structure under modeling;©IEEE 2008

6.3.2 Results

The FPSK in this case was driven by a dataset composed of 10 different frequency responses obtained by means of a full-wave solver that simulates the system behavior for 10 different values of the stub height, ranging from $\theta_{min} = 0$ um to $\theta_{max} = 716$ um; every frequency response contains 1001 samples, with $f_{min} = 0$ Hz and $f_{max} = 40$ GHz. We left 4 frequency responses for the validation, so the total number of samples used for the fitting is 6060.
The order of the model has been set to $\bar{n} = 13$, while the cardinality of the parameter basis (the Chebycev polynomials), equal for the numerator and the denominator, is 3; the total number of unknown coefficients is then 210. We ran a single iteration of the algorithm with and without the enhancement of the DC point precision. Our frequency mask was defined as:

$$w(f) = \begin{cases} w(0) = 100 \\ w(f) = 1 \quad \forall f > 0 \end{cases} \tag{6.6}$$

The time required by the iteration of FPSK is:

$$T_{FPSK} \approx 0.97s, \tag{6.7}$$

while for the standard PSK we need

$$T_{PSK} \approx 4.57s. \tag{6.8}$$

The errors of the model without the DC point enhancement, computed as in (6.5), are:

Response	Absolute Error (Validation)	Relative Error (Validation)	Absolute Error (Fitting)	Relative Error (Fitting)
$S(1,1)$	1.09×10^{-2}	3.82×10^{-2}	1.24×10^{-2}	2.44×10^{-2}
$S(1,2)$	8.78×10^{-3}	1.09×10^{-3}	6.66×10^{-3}	8.93×10^{-3}
$S(2,1)$	8.72×10^{-3}	1.08×10^{-2}	7.00×10^{-3}	9.44×10^{-3}
$S(2,2)$	1.20×10^{-2}	4.10×10^{-2}	8.04×10^{-3}	1.40×10^{-2}

while for the model with higher DC precision:

Response	Absolute Error (Validation)	Relative Error (Validation)	Absolute Error (Fitting)	Relative Error (Fitting)
$S(1,1)$	1.16×10^{-2}	4.09×10^{-2}	1.32×10^{-2}	2.62×10^{-2}
$S(1,2)$	8.57×10^{-3}	1.07×10^{-2}	6.47×10^{-3}	8.67×10^{-3}
$S(2,1)$	8.51×10^{-3}	1.05×10^{-2}	6.72×10^{-3}	9.06×10^{-3}
$S(2,2)$	1.20×10^{-2}	4.11×10^{-2}	7.99×10^{-3}	1.45×10^{-2}

In figures 6.11 and 6.13 we report a graphical result of two responses with particular emphasis on the DC point precision.

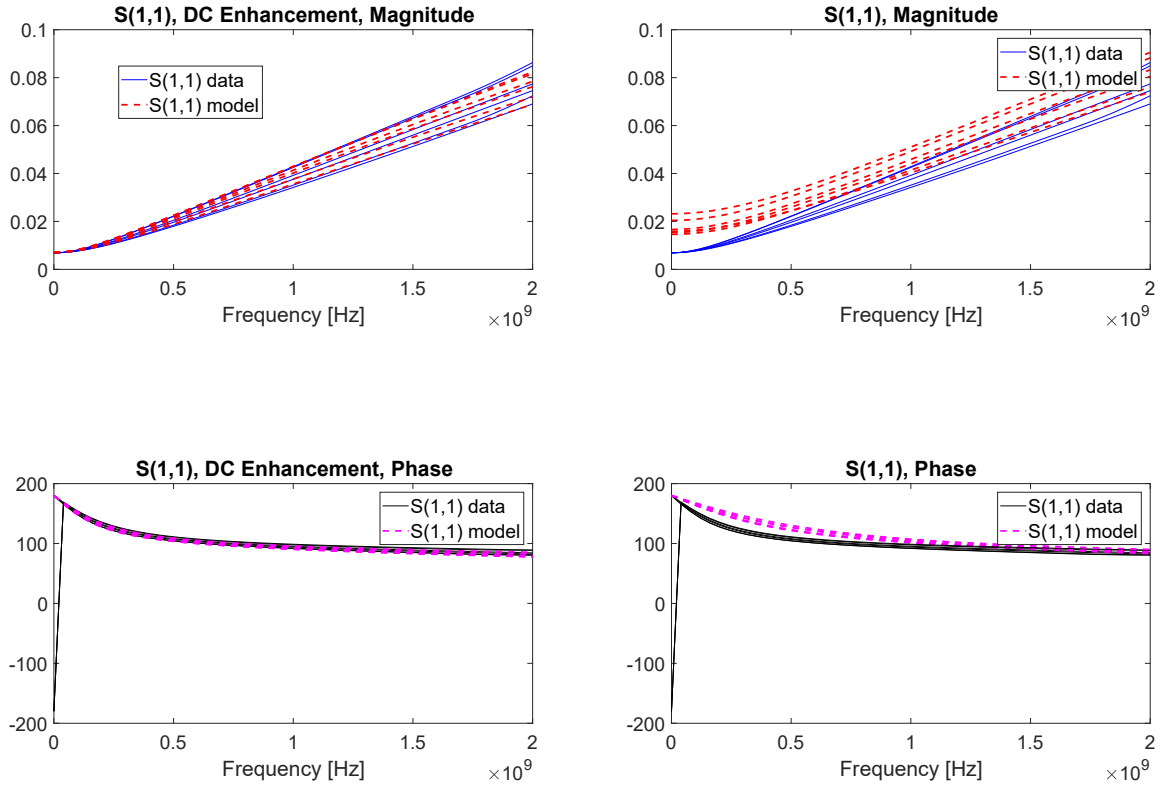


Figure 6.11: Magnitude and phase fitting of the $S(1,1)$ element of the transfer matrix; left panel: with DC precision enhancement; right panel: without DC precision enhancement.

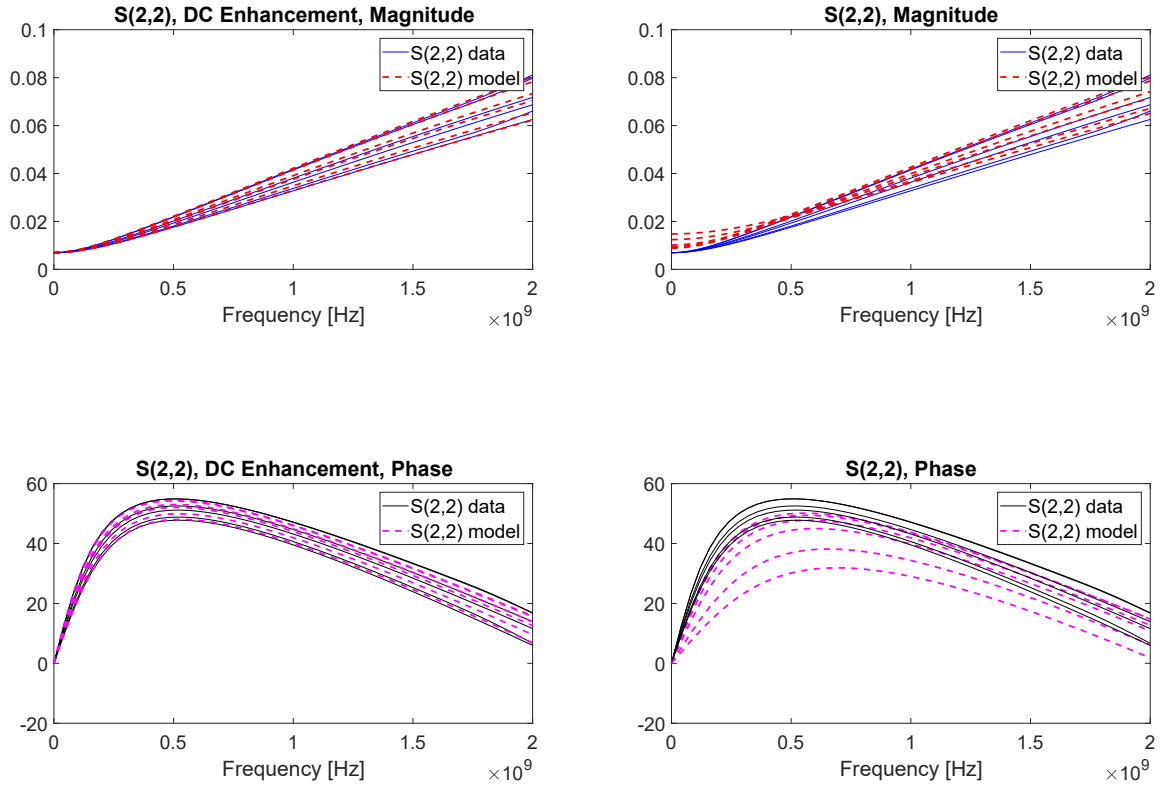


Figure 6.12: Magnitude and phase fitting of the $S(2,2)$ element of the transfer matrix; left panel: with DC precision enhancement; right panel: without DC precision enhancement.

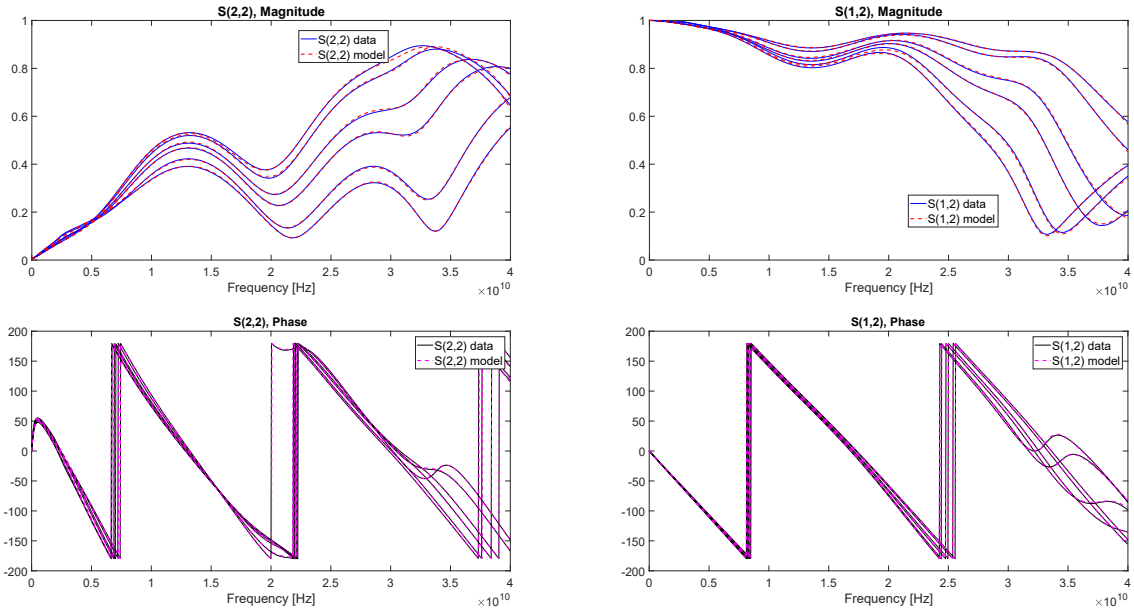


Figure 6.13: Full plot of elements $S(1,2)$ and $S(2,2)$ of the transfer matrix.

6.4 Low Noise Amplifier

With this example we will test the algorithm in a multivariate case, modeling a system depending on two different parameters.

6.4.1 Structure Description

We are now modeling a low noise amplifier (LNA), a circuit devoted to amplify very weak signals with the aim to keep the signal-to-noise ratio as large as possible. This component is usually exploited as a first stage amplification circuit of signals coming from antennas and is ubiquitous in electronics systems for telecommunications. The data were provided by Intel (courtesy of Dr. Pietro Brenner ans Dr.Gianni Signorini, Intel Munich) and the design parameters to be embedded in the macromodel are the common mode voltage V_{CM} at the differential amplifier input port and the suppy voltage V_{DD}. The data provided was representive of a 6 ports system.

6.4.2 Results

Our dataset is composed of 35 different frequency responses collected through a full-wave simulator in scattering representation; each one is composed of 235 data samples, with $f_{min} = 0$ Hz and $f_{max} = 1$ GHz, and is collected for a different combination of the two design parameters, the bias voltage V_{DD} (ranging from $\theta^1_{min} = 0.9$ V to $\theta^1_{max} = 1.2$ V) and the common mode voltage V_{CM} (ranging from $\theta^2_{min} = 0.4$ V to $\theta^2_{max} = 0.6$ V). The total number of frequency samples is then 296100.
The order of the model has been set to $\bar{n} = 16$, while the cardinality of the parameter

basis (again Chebychev polynomials),for both numerator and denominator, is $\bar{\ell}_1 = 3$ to catch the dependency on the bias voltage and $\bar{\ell}_2 = 2$ to catch the dependency on the common mode voltage. The total number of unknowns is then 3774. Due to the highly attenuating behavior of most of the responses, we minimized the relative error during the modeling procedure (minimization of the absolute error led to a badly scaled problem that caused the failure of the algorithm).
We performed 10 iterations for a total runtime of 166.8 s; the single iteration time requirement is then:

$$T_{FPSK} \approx 16.7s. \tag{6.9}$$

A single iteration of PSK required:

$$T_{PSK} \approx 84.18s \tag{6.10}$$

and resulted in the failure of the algorithm due to the impossibility to minimize the relative error in the earlier implementation of PSK (which we did not care modifying since outperformed by the new FPSK algorithm). The maximum errors over all the responses, computed as in (6.5), are $AbsError^{Max} = 1.97 \times 10^{-4}$ and $RelError^{Max} = 1.91 \times 10^{-2}$. Figure 6.14 report some plots of the model versus the data with particular emphasis in the frequency bands with largest variations.

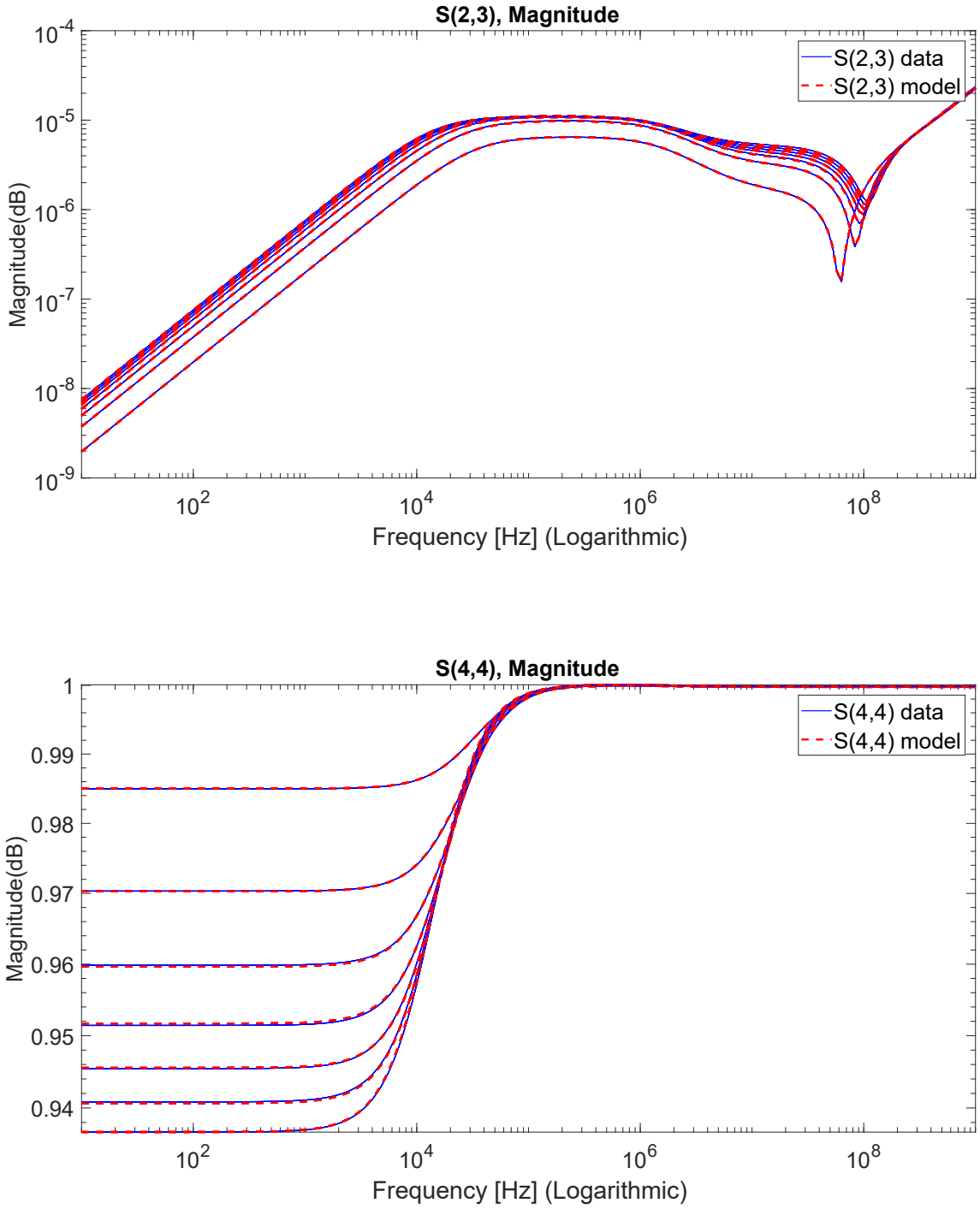


Figure 6.14: Magnitude of responses $S(2,3)$, $S(4,4)$ in dB. For the first: absolute error equal to 7.04×10^{-8}, relative equal to 1.06×10^{-2}. For the second: absolute error equal to 1.93×10^{-4}, relative error equal to 1.97×10^{-4}. We note how the precision of high dynamic range models is guaranteed by the minimization of the relative error.

6.5 Other examples

Here we present in a tabulated form the results achieved by our algorithm over other test cases, summarizing the main informations and comparing the execution time of FPSK and standard PSK, for a single iteration; the difference between the results of the two algorithms is equal to the 13-th decimal digit (at least). All the test are performed minimizing the absolute error; the algorithm stops after a maximum of ten iterations if the prescribed accuracy ($\epsilon = 0.001$) is not reached before. In some cases, the number of parameters samples were not sufficient to perform a validation test without downgrading the fitting performances; in such cases we denote the validation test with the symbol NaN to pointout that no validation was performed and that all the avalaible samples were used for the fitting. We report the results in a table, where each test is identified with the number of the following enumeration:

1. Input capacitor; the plates side-length is the free parameter.
2. S-Shaped microstrip; the middle-segment is the free parameter.
3. Multi-layer integrated inductor: an inductor with 1.5 turns, placed in a multi-layer substrate, with square outline; the square side length is the free parameter.
4. Integrated inductor, 1.5 turns: an integrated inductor with square outline; the square side length is the free parameter.
5. Integrated inductor, 2 turns: an integrated inductor with square outline; the square side length is the free parameter.
6. Microstrip filter with double folded stub: a microstrip band-stop filter whose stub length is the free parameter.
7. Transmission line with embedded discontinuity: a transmission line with a lumped RLC discontinuity. The value of the capacitance C is the free parameter, ranging from 0.1 to 10 pF.
8. The structure is identical to the previous one; the value of the capacitance C is the free parameter, that in this case ranges from 0.1 to 1 pF.
9. The structure is identical to the previous one; the value of the capacitance C in this case ranges from 1 to 10 pF.
10. The structure is identical to the previous one; now we let the inductance L vary from 10pH to 1nH.
11. Printed Circuit Board interconnect; (Courtesy of Prof. Christian Schuster and Dr. Jan Preibisch, Technische Universitat Hamburg-Harburg, Hamburg, Germany). For details see [14, 17, 32, 43]. A geometric design variable is the free parameter.
12. The same as the previous example; differs in the range of variation of the free parameter.

13. The structure is the same described in Section 6.1; in this case we changed the range of variation of the slot offset.

14. The structure is the same described in Section 6.1; now we fix the slot offset length to 0 and we leave the slot length L as a free parameter.

15. Again as in Section 6.1, with the slot offset fixed to 25 mm and the slot length L considered as the free parameter.

Test	$\bar{n}$	$\bar{\ell}_N$	$\bar{\ell}_D$	P^2	Fitting points	Max Abs. Err. (Val)	Max Rel. Err. (Val)	Max Abs. Err. (Fit)	Max Rel. Err. (Fit)	FPSK (s)	PSK (s)
(1)	4	5	2	4	3820	8.61×10^{-3}	1.31×10^{-2}	7.47×10^{-3}	1.75×10^{-2}	0.19	0.62
(2)	16	5	4	4	3600	NaN	NaN	1.71×10^{-3}	5.53×10^{-2}	0.28	1.01
(3)	8	4	3	4	13356	2.06×10^{-3}	3.37×10^{-3}	2.01×10^{-3}	2.56×10^{-3}	0.31	1.19
(4)	6	4	3	4	13356	6.85×10^{-4}	9.20×10^{-4}	7.49×10^{-4}	1.02×10^{-3}	0.22	0.81
(5)	6	4	3	4	13356	2.40×10^{-3}	3.19×10^{-3}	1.86×10^{-3}	2.49×10^{-3}	0.22	0.81
(6)	10	3	3	4	13200	2.98×10^{-3}	4.38×10^{-3}	1.86×10^{-3}	2.73×10^{-3}	0.31	1.44
(7)	18	2	2	4	24000	7.03×10^{-4}	3.89×10^{-3}	7.72×10^{-4}	4.27×10^{-3}	0.58	3.10
(8)	18	2	2	4	24000	5.43×10^{-4}	2.54×10^{-3}	7.33×10^{-4}	2.80×10^{-3}	0.57	3.12
(9)	18	2	2	4	24000	6.28×10^{-4}	1.12×10^{-3}	7.31×10^{-4}	1.49×10^{-3}	0.51	3.08
(10)	18	2	2	4	24000	1.01×10^{-3}	2.09×10^{-3}	1.58×10^{-3}	2.69×10^{-3}	0.56	3.11
(11)	50	5	3	4	16000	8.14×10^{-3}	1.33×10^{-2}	2.08×10^{-3}	7.97×10^{-3}	1.36	25.41
(12)	4	5	3	4	14000	8.14×10^{-4}	3.30×10^{-3}	8.92×10^{-4}	3.42×10^{-3}	0.84	9.64
(13)	48	6	3	4	66888	NaN	NaN	3.81×10^{-2}	7.80×10^{-2}	4.76	144.27
(14)	30	6	3	4	66888	NaN	NaN	8.19×10^{-3}	1.32×10^{-2}	3.79	65.05
(15)	29	5	3	4	66888	NaN	NaN	4.38×10^{-3}	7.26×10^{-3}	3.50	52.43

Chapter 7

Conclusions and Further Improvements

With this thesis work we improved the state of the art of the parameterized macromodeling framework: the extraction of parametric macromodels of large multiport structures is now possible in a much faster and reliable way, and can be carried out on a common laptop, whereas the earlier standard algorithm of choice required server-like resources in terms of computational cost and memory requirements. The achievement of a linear scaling of the complexity with respect to the number of responses to be processed allows to consider the possibility to model a much wider class of electromagnetic structures of practical interest for modern industrial design processes. Further, the issues that arise during the fitting of high dynamic range systems has been addressed and the solution have been incorporated in a flexible modeling tool, which also offers a starting point for the development of a more sofisticated frequency-dependent fitting precision strategies. The present implementation is suitable to be improved, with minor modifications, in order to achieve better results; in the following we propose some of the ideas that will be part of our future improvements efforts.

Improve the Flexibility of the Frequency Weighting

The proposed algorithm offers a basic tool for the enhancement of the fitting precision within prescribed frequency bands. A more sofisticated strategy can be exploited in order to achieve better results: the frequency mask can be built making use of the common techniques that are widely used in the filter design framework, adding the possibility to specify a larger number of design specifications in order to obtain a more flexible tuning of the fitting precision. Such techniques are well known and can be easily incorporated in our MATLAB software.

Parallelization of the Code

The structure of the FPSK algorithm is particularly suitable to apply a parallel computing strategy that can boost the algorithm speed, processing simultaneously more than one response: the operations performed in the loop that decouples the port responses are all independent and can therefore be performed in parallel with minor modifications of the code, following the same strategies proposed in [7] for the parallelization of the VF algorithm.

Compute the Numerator only once

The decoupling strategy carried out within the FPSK algorithm allows to compute the denominator of the model alone, without the need to find the numerator coefficients; since the iterations of the FPSK differs one from each other only for the updating of the weights, defined as the inverse of the denominator, it is possible to compute only the denominator of the model at each iteration until its estimate stabilizes, according to some prescribed criterium; when this happens, the linearization bias is reduced to the minimum achievable and the numerator can be computed, with respect to the linearized residuals.

Bibliography

[1] M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions*. Dover Publications, 1968.

[2] B. Anderson and S. Vongpanitlerd. Network analysis and synthesis: A modern systems theory approach, eaglewood cli s, 1973.

[3] M. Barnsley. *Fractals Everywhere*. Academic Press, 1993.

[4] J. P. Boyd. *Chebyshev and Fourier spectral methods*. Courier Corporation, 2001.

[5] S. Boyd, V. Balakrishnan, and P. Kabamba. A bisection method for computing the h norm of a transfer matrix and related problems. *Mathematics of Control, Signals and Systems*, 2(3):207–219, 1989.

[6] T. Chihara. An introduction to orthogonal polynomials (gordon and breach science publishers, new york, ny). Technical report, ISBN 0-677-04150-0, 1978.

[7] A. Chinea and S. Grivet-Talocia. On the parallelization of vector fitting algorithms. *IEEE Transactions on Components, Packaging and Manufacturing Technology*, 1(11):1761–1773, Nov 2011.

[8] J. E. Dennis and R. B. Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*, volume 16. Society for Industrial and Applied Mathematics, 1996.

[9] F. Ferranti, L. Knockaert, and T. Dhaene. Guaranteed passive parameterized admittance-based macromodeling. *IEEE Transactions on Advanced Packaging*, 33(3):623–629, 2010.

[10] F. Ferranti, L. Knockaert, and T. Dhaene. Passivity-preserving parametric macromodeling by means of scaled and shifted state-space systems. *IEEE Transactions on Microwave Theory and Techniques*, 59(10):2394–2403, 2011.

[11] A. Gil, J. Segura, and N. M. Temme. *Numerical methods for special functions*, volume 99. Siam, 2007.

[12] G. H. Golub and C. F. Van Loan. *Matrix computations*, volume 3. Johns Hopkins Univ Pr, 1996.

[13] S. Grivet-Talocia. Passivity enforcement via perturbation of hamiltonian matrices. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 51(9):1755–1769, 2004.

[14] S. Grivet-Talocia. A perturbation scheme for passivity verification and enforcement of parameterized macromodels. *IEEE Transactions on Components, Packaging and Manufacturing Technology*, 7(11):1869–1881, 2017.

[15] S. Grivet-Talocia and E. Fevola. Compact parameterized black-box modeling via

fourier-rational approximations. *IEEE Transactions on Electromagnetic Compatibility*, 59(4):1133–1142, 2017.

[16] S. Grivet-Talocia and B. Gustavsen. *Passive macromodeling: Theory and applications*, volume 239. John Wiley & Sons, 2015.

[17] S. Grivet-Talocia and R. Trinchero. Behavioral, parameterized, and broadband modeling of wired interconnects with internal discontinuities. *IEEE Transactions on Electromagnetic Compatibility*, 60(1):77–85, 2018.

[18] B. Gustavsen. Improving the pole relocating properties of vector fitting. *Power Delivery, IEEE Transactions on*, 21(3):1587–1592, july 2006.

[19] B. Gustavsen and A. Semlyen. Rational approximation of frequency domain responses by vector fitting. *Power Delivery, IEEE Transactions on*, 14(3):1052–1061, jul 1999.

[20] W. Hendrickx and T. Dhaene. A discussion of "rational approximation of frequency domain responses by vector fitting". *Power Systems, IEEE Transactions on*, 21(1):441–443, feb. 2006.

[21] N. J. Higham. *Accuracy and stability of numerical algorithms*. SIAM, Philadelphia, PA, 1996.

[22] T. Kailath. *Linear systems*. Prentice-Hall Englewood Cliffs, NJ, 1980.

[23] R. Kalman. On a new characterization of linear passive systems. 1964.

[24] P. Lancaster and M. Tismenetsky. *The theory of matrices: with applications*. Academic Press, 1985.

[25] S. Lefteriu and A. C. Antoulas. On the convergence of the vector-fitting algorithm. *Microwave Theory and Techniques, IEEE Transactions on*, 61(4):1435–1443, 2013.

[26] E. C. Levy. Complex curve fitting. *IRE Trans. on Automatic Control*, 4:37–44, 1959.

[27] L. Ljung. *System Identification: Theory for the User*. Prentice Hall, 1999.

[28] D. G. Luenberger. *Optimization by vector space methods*. Wiley-Interscience, 1997.

[29] A. F. Nikiforov, V. B. Uvarov, and R. P. Boas. *Special functions of mathematical physics*. Birkhäuser, 1988.

[30] J. Nocedal and S. J. Wright. *Numerical optimization*. Springer verlag, 1999.

[31] A. V. Oppenheim and A. S. Willsky. *Signals and systems*. Prentice Hall, Englewood Cliffs, NJ: Prentice Hall, 1983.

[32] J. Preibisch, T. Reuschel, K. Scharff, J. Balachandran, B. Sen, and C. Schuster. Exploring efficient variability-aware analysis method for high-speed digital link design using pce. *DesignCon, Jan*.

[33] C. Sanathanan and J. Koerner. Transfer function synthesis as a ratio of two complex polynomials. *Automatic Control, IEEE Transactions on*, 8(1):56–58, jan 1963.

[34] C. Sanathanan and J. Koerner. Transfer function synthesis as a ratio of two complex polynomials. *IEEE transactions on automatic control*, 8(1):56–58, 1963.

[35] C. Scherer and S. Weiland. Linear matrix inequalities in control. *Lecture Notes, Dutch Institute for Systems and Control, Delft, The Netherlands*, 3, 2000.

[36] M. D. Stefano. Automated generation of stable bias-dependent small-signal behavioural macromodels for circuit-level simulation. Master's thesis, Politecnico di Torino, Italy,(TO), 2018.

[37] P. Triverio, S. Grivet-Talocia, and M. S. Nakhla. A parameterized macromodeling strategy with uniform stability test. *IEEE Transactions on Advanced Packaging*, 32(1):205–215, 2009.

[38] P. Triverio, M. Nakhla, and S. Grivet-Talocia. Parametric macromodeling of multiport networks from tabulated data. In *Electrical Performance of Electronic Packaging, 2007 IEEE*, pages 51–54. IEEE, 2007.

[39] C. F. Van Loan. Matrix computations (johns hopkins studies in mathematical sciences), 1996.

[40] M. Wohlers and E. Beltrami. Distribution theory as the basis of generalized passive-network analysis. *IEEE Transactions on Circuit Theory*, 12(2):164–170, 1965.

[41] M. R. Wohlers, N. Declaris, and H. G. Booker. Lumped and distributed passive networks: a generalized and advanced viewpoint. 1969.

[42] D. Youla, L. Castriota, and H. Carlin. Bounded real scattering matrices and the foundations of linear passive network theory. *IRE Transactions on Circuit Theory*, 6(1):102–124, 1959.

[43] A. Zanco, S. Grivet-Talocia, T. Bradde, and M. De Stefano. Multivariate macromodeling with stability and passivity constraints. In *Signal Propagation on Interconnects, 2018. SPI 2018. 22th IEEE Workshop on*. IEEE, 2018.