User guide of Complex Vector Fitting toolbox

Implementation of Complex Vector Fitting for rational modeling of baseband S-parameter data

Version 1.0 for Matlab

Surrogate Modeling (SUMO) Group, IDLab, Department of Information Technology,

Ghent University – IMEC, Technologiepark-Zwijnaarde 15, B-9052 Ghent, Belgium

1 Introduction

Baseband equivalent signals and systems are widely used in the simulation of communication systems to simplify the modulation, demodulation, and filtering process [Jeruchim-06]. In this framework, the Complex Vector Fitting (CVF) modeling algorithm described in [Ye-19, Spina-19, Spina-20], allows one to efficiently and accurately perform frequency- and time-domain simulations of baseband systems. Indeed, starting from a set of frequency-dependent tabulated data describing the baseband scattering parameters of the system under study, the CVF modeling technique is able to build stable and passive continuous rational models in the form:

$$\mathbf{S}_b(s) = \sum_{k=1}^{K} \frac{\mathbf{R}_k}{s - p_k} + \mathbf{D} \quad (1)$$

where $\mathbf{S}_b$ are the baseband scattering parameters, $s = j2\pi f_b$ is the Laplace variable and f_b the baseband frequency samples, p_k are common poles, which can be either real or complex, and $\mathbf{R}_k$ are the corresponding residues, while $\mathbf{D}$ is a real matrix. Models in the form (1) can also be expressed via a suitable state-space representation:

$$\mathbf{S}_b(s) = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D} \quad (2)$$

to be used for frequency- and time-domain simulations.

This manual describes a collection of Matlab routines called Complex Vector Fitting toolbox for the rational modeling of multi-port, symmetrical baseband scattering parameter data via CVF. In particular, the computation of a CVF model is performed via two main functions:

- VFdriver_complex.m: it identifies models (1) and (2) using the pole relocating CVF technique [Ye-19], starting from a set of frequency-dependent tabulated data. The applied strategy is to stack the upper triangle of a scattering matrix into a single column which is next fitted by CVF using a common pole set.
- RPdriver_complex.m: it performs the passivity assessment and, eventually, enforcement of the computed CVF model. In particular, the passivity enforcement technique described in [Gustavsen-10] is adopted here, once suitably adapted for baseband systems.

The CVF modeling strategy is summarized in Fig. 1.

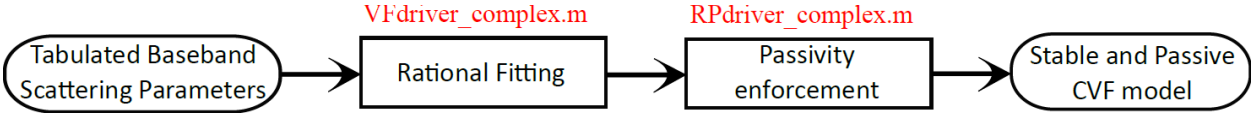

Fig. 1: Flowchart of the proposed Matlab routines.

It is important to remark that the model-building phase of CVF can leverage on several robust methodologies adopted by the Vector Fitting (VF) algorithm [Gustavsen-99]: the pole flipping scheme [Gustavsen-99], relaxed formulation [Gustavsen-06], and fast implementation based on QR decomposition [Deschrijver-08] used in VF can be directly adopted for CVF. Hence, the proposed implementation of the CVF modeling technique is based on the Matrix Fitting Toolbox, which offers a robust implementation of the VF algorithm and is available at

https://www.sintef.no/projectweb/vectorfitting/downloads/

The proposed code has been tested on Matlab 2018a.

Restrictions of use:

- Embedding any of (or parts from) the routines of the Complex Vector Fitting toolbox in a commercial software, or a software requiring licensing, is strictly prohibited. This applies to all routines, see Section 2.1.
- If the code is used in a scientific work, then reference should be made as follows:
    - Y. Ye, D. Spina, D. Deschrijver, W. Bogaerts, and T. Dhaene, "Time-domain compact macromodeling of linear photonic circuits via Complex Vector Fitting", Photonics Research, vol.: 7, issue: 7, pag.: 771-782, July 2019
    - D. Spina, Y. Ye, D. Deschrijver, W. Bogaerts and T. Dhaene, "Complex Vector Fitting toolbox: a software package for the modeling and simulation of general linear and passive baseband systems", *accepted for publication in Electronics Letters*, 2021
- The Matrix Fitting Toolbox is required for the code here provided to run correctly (see Section 2.2). However, the Matrix Fitting Toolbox is not included in this distribution: the Matrix Fitting Toolbox and related information are available at the website

  https://www.sintef.no/projectweb/vectorfitting/

  Please, verify on the latter website the necessary references to be made when using the Matrix Fitting Toolbox in a scientific work.

2 The software package

2.1 The code

The proposed implementation of CVF consists of the following files:

Main routines

- Create_Rat_Mod_CVF.m: Routine computing the CVF model (calls VFdriver_complex.m and RPdriver_complex.m).
- VFdriver_complex.m: Driver routine for rational fitting (calls vectfit3_complex.m).
- RPdriver_complex.m: Driver routine for passivity enforcement (calls violextremaS_complex.m and FRPS_complex.m).

Auxiliary routines

- vectfit3_complex.m: code computing the rational model.
- violextremaS_complex.m: code for passivity assessment.
- FRPS_complex.m: code for passivity enforcement.

Application Example

- Coupled_Micro_Main_File.m: main file to compute a CVF model starting from a set of tabulated baseband S-parameters. In particular, the VFdriver_complex.m and RPdriver_complex.m are called via the auxiliary function Create_Rat_Mod_CVF.m.

2.2 Installation instruction

This code requires the Matrix Fitting Toolbox to run correctly, which is not included in this distribution. The Matrix Fitting Toolbox is developed by Bjørn Gustavsen (email: Bjorn.Gustavsen@sintef.no) and it is available at https://www.sintef.no/projectweb/vectorfitting/downloads/
Once the Matrix Fitting Toolbox is installed,

- Download the Complex Vector Fitting toolbox.
- Place all files in a common directory, e.g. c:\user\cvfmodeling
- Include the directory in the Matlab search path.

2.3 Code description

The proposed implementation of CVF follows the same structure adopted by the Matrix Fitting Toolbox: the functions VFdriver.m and RPdriver.m of the Matrix Fitting Toolbox are modified into the functions VFdriver_complex.m and RPdriver_complex.m in order to model baseband systems, which have an asymmetric frequency response with regard to the positive and negative frequencies [Ye-19].

2.3.1 VFdriver_complex.m

This function generates a pole-residue model for a data set $(s, \mathbf{S}_b(s))$, with n ports and N_s frequency samples, and is called in Matlab as:

```
[SER,rmserr,Sbfit,opts2]=VFdriver_complex(Sb,s,poles,opts)
```

The input/output parameters of this function are described in details the following.

Input:

- `Sb`: 3D matrix of size (n, n, N_s) holding the baseband scattering parameters to be fitted.
- `s`: vector of the frequency samples of length N_s, where $s = j2\pi f_b$ is the Laplace variable and f_b the baseband frequency samples.
- `poles`: vector of length P holding the initial poles. The initial poles can be specified manually or chosen in an automated framework (default setting, see below).
- `opts`: it is an optional structure that can be used for overriding defaults settings and for requesting plots.

Output:

- `SER` is a data structure with the model expressed as poles-residues and state-space forms, where
  - `SER.poles`: vector of length P containing the poles.
  - `SER.R`: 3D matrix of size (n, n, P) representing the (n, n) residue matrices corresponding to each of the P poles.
  - `SER.D`: (n, n) matrix representing the constant term $\mathbf{D}$ in (1) and (2).
  - `SER.A` (nP, nP), `SER.B` (nP, n) and `SER.C` (n, nP): state-space matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ in (2).
  - `rmserr`: the resulting RMS-error of the fitting.
  - `Sbfit`: 3D matrix of size (n, n, N_s) with the model response computed for $s = j2\pi f_b$.
  - `opts2`: contains *all* the options parameters, including default settings.

The input and output parameters and the fitting options (described in the structure `opts`) for VFdriver_complex.m are the same as for VFdriver.m. The reader is referred to the user manual of the Matrix Fitting Toolbox (and the comments in the code of the function VFdriver_complex.m) for a detailed description. In the following, the differences between the two routines VFdriver_complex.m and VFdriver.m in terms of fitting options available to the users will be discussed. A detailed description of the CVF modeling approach is given in [Ye-19, Spina-19, Spina-20].

Both CVF and VF adopt pole-residue models formed by real and complex poles having a negative real part, in order to guarantee the stability of the model. However, the complex poles and the corresponding residues computed via VF must always occur in complex conjugate pairs, but this condition does not hold for CVF: this has an impact on the pole-selection strategy to choose the initial poles and the model type, as described below.

<u>Choice of the initial poles</u>

The initial poles can be specified by the user in the vector `poles` or can be chosen automatically by the CVF toolbox, with a similar strategy as the one adopted in the Matrix Fitting Toolbox.

In particular, the Matrix Fitting Toolbox implements three automated sampling strategies to select the initial poles, via the input parameter `opts.poletype`. The initial poles are chosen as complex conjugate pairs in the frequency range of interest that are

- linearly spaced → `opts.poletype='lincmplx'`
- logarithmically spaced → `opts.poletype='logcmplx'`
- a mixture of linear and logarithmically spaced → `opts.poletype='linlogcmplx'`

The VFdriver_complex.m function offers the same options, even if the poles are not complex conjugate pairs anymore. Hence, the relation among the real α and imaginary β part of the initial poles for CVF is

$$p_k = \alpha_k + j\beta_k; \quad \alpha_k = -c|\beta_k|$$

where c is a constant value representing the ratio of the real and imaginary part of the poles (default value 0.001), and the imaginary part of the initial poles for CVF can be chosen as

- linearly spaced → `opts.poletype='lincmplx'`
- logarithmically spaced → `opts.poletype='logcmplx'`
- a mixture of linear and logarithmically spaced → `opts.poletype='linlogcmplx'`

in the frequency range of interest $[2\pi f_{min}; 2\pi f_{max}]$.

Figure 2 shows an example of the distribution of the real and imaginary part of the initial poles when `opts.poletype='lincmplx'`. The different trend between the imaginary and real part is due to the following constraint: the real part of all poles must assume negative values in order for the CVF model to be stable.

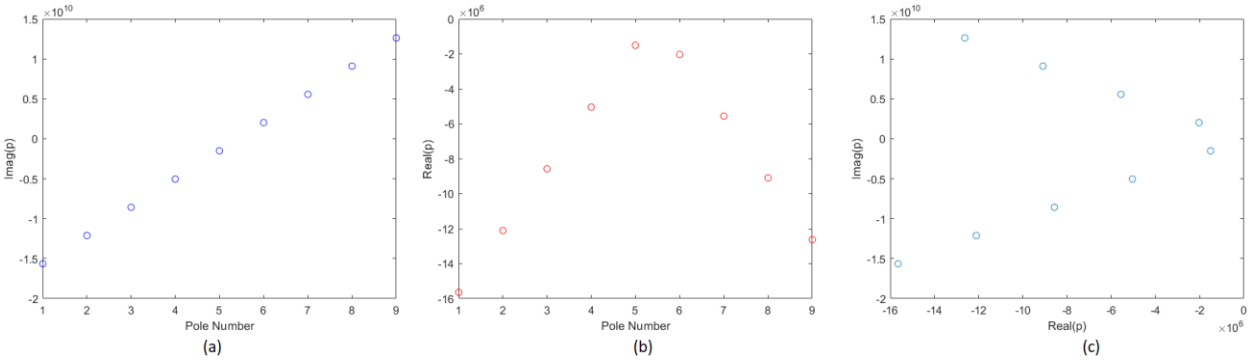

Fig. 2: imaginary part (a) and real part (b) of the initial poles. Location of the initial poles in the complex plane (c).

A complete study on the influence of the distribution of the initial poles on the accuracy of the CVF model has not yet been presented in the literature. The function VFdriver_complex.m assumes linearly spaced poles as default choice.

Model type

The Matrix Fitting Toolbox allows one to compute models in the form

$$\mathbf{H}(s) = \sum_{i=1}^{N} \frac{\mathbf{R}_i}{s - p_i} + \mathbf{D} + s\mathbf{E} \quad (3)$$

where $\mathbf{H}(s)$ represents the transfer function of the system under study, which can be expressed as impedance, admittance or scattering parameters. The input parameter `opts.asymp` decides the model type:

- `opts.asymp=1` --> D=0, E=0 ('Strictly proper')
- `opts.asymp=2` --> D~=0, E=0 ('Proper')
- `opts.asymp=3` --> D~=0, E~=0 ('Improper')

The proposed implementation of the CVF modeling approach does not allow for an improper model, and only the options

- `opts.asymp=1` --> D=0, E=0 ('Strictly proper')
- `opts.asymp=2` --> D~=0, E=0 ('Proper') (Default value)

are valid.

Additionally, the equivalent state-space representation of (3) can be computed as real-valued (matrix $\mathbf{A}$ is block-diagonal) or complex valued (matrix $\mathbf{A}$ is diagonal). This is possible since the complex poles of a

VF model are in complex conjugate pairs. The user can decide the model type via the input parameter `opts.cmplx_ss`:

- `opts.cmplx_ss=1` generates complex state space model with diagonal $\mathbf{A}$
- `opts.cmplx_ss=0` generates real-only state space model with block-diagonal $\mathbf{A}$

Rational models computed via CVF are always complex-valued by construction, since the complex poles are not computed as complex conjugate pairs. Hence, the parameter `opts.cmplx_ss` is assumed equal to 1 by default, and it is the only admissible value.

Plotting fitting results

The results of the fitting process can be plotted by the function VFdriver_complex.m using the same options as VFdriver.m. The figures generated by the function depend on the input parameters

- `opts.plot, opts.logx, opts.logy, opts.errplot, opts.phaseplot`

which are described in the user manual of the Matrix Fitting Toolbox. Only one difference is to be remarked: if the user chooses to a logarithmic axis for the frequency values, only the modeling results at positive frequencies will be displayed. Since baseband scattering parameters have also components at negative frequency values (by definition), choosing this option is discouraged. By default, the function VFdriver_complex.m will plot the results adopting a linear axis for the frequency values. Fitting results are always plotted in figure 101 for the magnitude and figure 102 for the phase.

2.3.2 RPdriver_complex.m

This function performs the passivity assessment and, eventually, enforcement of a CVF model. The reader is referred to [Ye-18, Ye-19] for a detailed discussion on the passivity conditions for baseband signals and systems.

```
[SER,Sbfit,opts2]=RPdriver_complex(SER,s,opts)
```

Input

- `SER`: structure computed by VFdriver_complex.m containing the CVF model.
- `s`: vector holding the frequency samples of length N_s, $s = j2\pi f_b$ is the Laplace variable and f_b the baseband frequency samples.
- `opts`: optional structure that can be used for overriding defaults settings and for requesting plots.

Output

- `SER`: data structure containing the CVF model, where the passivity violations are corrected (for both poles/residues and state-space forms).
- `Sbfit`: 3D matrix of size (n, n, N_s) with the model response computed for $s = j2\pi f_b$.
- `opts2`: contains *all* the options parameters, including default settings.

The passivity assessment is based on the computation of the eigenvalues of the Hamiltonian matrix. The main differences with respect to VF models are as follows:

- The Hermitian operator is used in the Hamiltonian matrix rather than the transpose operator, since the CVF state-space matrices are complex [Ye-19, Spina-20].
- As a result, half-size passivity tests such as the one in [Gustavsen-08] cannot be adopted and the eigenvalues of the entire Hamiltonian matrix must be computed.

The CVF model passivity is enforced by perturbation of the residue matrix eigenvalues. In the proposed software package, the passivity enforcement implemented in the functions RPdriver.m and FRPS.m of the Matrix Fitting Toolbox is suitably modified, since complex poles are not computed as complex conjugate pairs in CVF. However, the input and output parameters and the passivity assessment/enforcement options (described in the structure `opts`) for RPdriver_complex.m are the same as for RPdriver.m. The reader is referred to the user manual of the Matrix Fitting Toolbox for a detailed description. From a user perspective, there are only two relevant differences:

- The CFV modeling technique described in [Ye-19, Spina-19, Spina-20] is applicable only to scattering parameters, while the function RPdriver.m can also be adopted to model admittance parameters.
- Additionally, state-space representations computed via CVF are always complex-valued, while real-valued ones are also admissible for VF. Hence, the input parameters `opts.parametertype` and `opts.cmplx_ss` can assume only one value:
  - `opts.parametertype='S'` → Data is expressed as scattering parameters.
  - `opts.cmplx_ss=1` → The rational model to be perturbed is a complex-valued one.

These two options are the default ones and cannot be modified in order for the code to work correctly.

3 Application Example

One application example is provided with this software package in the file "Coupled_Micro_Main_File.m". The system under study is formed by three coupled microstrips, described in [Medico-19] and shown in Fig. 3. The conductors have length l = 5 cm and width w = 120 µm. The spacing between the microstrips is s_1 = 200 µm and s_2 = 100 µm. The substrate is a Roger RT/duroid 5880 with relative permittivity ϵ = 2.2 and thickness h = 127 µm. The scattering parameters of the microstrips are simulated in Advanced Design System (ADS, Keysight Technologies) in the frequency range [30; 90] GHz for 601 linearly-spaced frequency samples and are stored in the Matlab dataset "3Coupled_micros_Initial_Data.mat". The corresponding baseband S-parameters are obtained by shifting the scattering parameters computed via ADS around the central frequency f_c = 57 GHz. Hence, the baseband S-parameters are defined in the frequency range [-27; 33] GHz: this choice illustrates the ability of CVF to model baseband S-parameters defined for different positive and negative frequency values. The problem under study is quite challenging, since the frequency range considered is relatively large (60 GHz) and the baseband S-parameters have a dynamic behavior (see Fig. 4a). However, CVF is able to compute an accurate, stable and passive model with 69 poles, with a maximum absolute error among the data and the model response of less than 50 dB (see Fig. 4b).

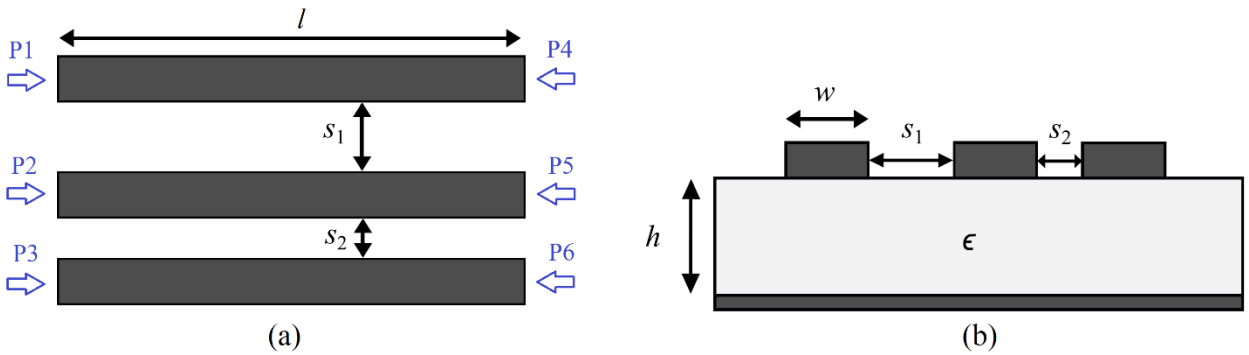

Fig. 3: (a) Coupled microstrips with the ports indicated in blue. (b) Cross section.

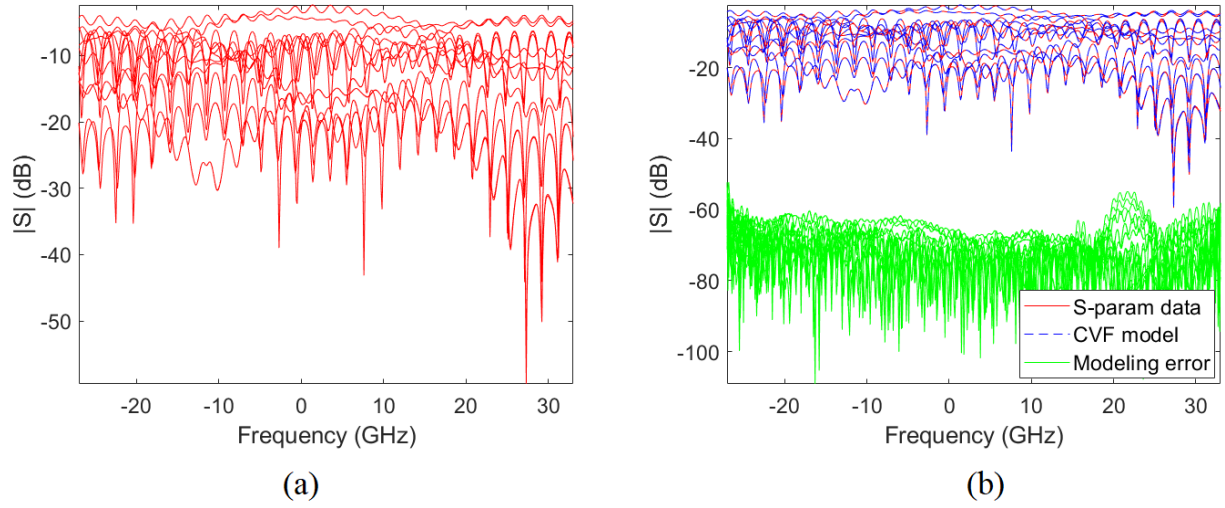

Fig. 4: (a) Baseband S-parameters. (b) CVF modeling results: baseband S-parameters (red line), CVF model response (blue dashed line), modeling error (green line).

References

[Jeruchim-06] M. C. Jeruchim, P. Balaban, and K. S. Shanmugan, Simulation of Communication Systems: Modeling, Methodology and Techniques (Springer, 2006).

[Ye-19] Y. Ye, D. Spina, D. Deschrijver, W. Bogaerts, and T. Dhaene, "Time-domain compact macromodeling of linear photonic circuits via Complex Vector Fitting", Photonics Research, vol.: 7, issue: 7, pag.: 771-782, July 2019

[Spina-19] Y. Ye, D. Spina, D. Deschrijver, W. Bogaerts and T. Dhaene, "Efficient Time-Domain Modeling and Simulation of Passive Bandpass Systems", Proceedings of International Conference on Electromagnetics in Advanced Applications (ICEAA), Granada, Spain, 9-13 September 2019

[Spina-20] D. Spina, Y. Ye, D. Deschrijver, W. Bogaerts and T. Dhaene, "Complex Vector Fitting toolbox: a software package for the modeling and simulation of general linear and passive baseband systems", *submitted to Electronics Letters*, 2020

[Gustavsen-99] B. Gustavsen and A. Semlyen, "Rational approximation of frequency domain responses by vector fitting," IEEE Trans. Power Delivery 14, 1052–1061 (1999).

[Gustavsen-06] B. Gustavsen, "Improving the pole relocating properties of vector fitting," IEEE Trans. Power Delivery 21, 1587–1592 (2006).

[Deschrijver-08] D. Deschrijver, M. Mrozowski, T. Dhaene, and D. De Zutter, "Macromodeling of multiport systems using a fast implementation of the vector fitting method," IEEE Microwave Compon. Lett. 18, 383–385 (2008).

[Gustavsen-10] B. Gustavsen, “Fast passivity enforcement for S-parameter models by perturbation of residue matrix eigenvalues,” IEEE Trans. Adv. Packag. 33, 257–265 (2010).

[Ye-18] Y. Ye, D. Spina, Y. Xing, W. Bogaerts, T. Dhaene, “Numerical modeling of a linear photonic system for accurate and efficient time-domain simulations”, Photonics Research, vol.: 6, issue: 6, pag.: 560 - 573, June 2018

[Gustavsen-08] B. Gustavsen and A. Semlyen, “Fast passivity assessment for S-parameter rational models via a half-size test matrix”, IEEE Trans. Microwave Theory Tech. 56, 2701–2708 (2008).

[Medico-19] R. Medico, D. Spina, D. Vande Ginste, D. Deschrijver and T. Dhaene, “Machine-Learning-Based Error Detection and Design Optimization in Signal Integrity Applications”, IEEE Transactions on Components, Packaging and Manufacturing Technology, vol. 9, no. 9, pp. 1712-1720, Sept. 2019