Last revised : 27.08.2009

MATRIX FITTING TOOLBOX

for rational modeling from Y-parameter and S-parameter data

User's GUIDE and REFERENCE

version 1.0 for Matlab

Bjørn Gustavsen
SINTEF Energy Research
N-7465 Trondheim
NORWAY

e-mail: bjorn.gustavsen@sintef.no

Table of Contents

1. INTRODUCTION

This manual describes a collection of Matlab routines for rational multi-port modeling of symmetrical Y-parameter and S-parameter data in the frequency domain, including passivity enforcement. The output is a rational model on pole-residue form (1.1) and a corresponding state space model (1.2), both with stable poles.

$$\mathbf{Y}(s) = \sum_{m=1}^{N} \frac{\mathbf{R}_m}{s - a_m} + \mathbf{D} + s\mathbf{E}, \quad \mathbf{S}(s) = \sum_{m=1}^{N} \frac{\mathbf{R}_m}{s - a_m} + \mathbf{D} + s\mathbf{E} \tag{1.1}$$

$$\mathbf{Y}(s) = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D} + s\mathbf{E}, \quad \mathbf{S}(s) = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D} \tag{1.2}$$

- **`VFdriver.m`** identifies models (1.1) and (1.2) using the pole relocating Vector Fitting technique (VF) [1]. The applied strategy is to stack the upper triangle of an admittance matrix $\mathbf{Y}(s)$ (or scattering matrix $\mathbf{S}(s)$) into a single column which is next fitted by VF using a common pole set. The implementation includes relaxation of the nontriviality constraint [2] (improved convergence) as well as a fast implementation of the pole identification step [3] (reduced memory requirements and computation time).
- **`RPdriver.m`** perturbs the model so that it becomes passive and so that that $\mathbf{E}$ become positive real (Y-parameters). This is achieved by perturbing the eigenvalues of the residue matrices $\{\mathbf{R}_m\}$ and those of $\mathbf{D}$ and $\mathbf{E}$ while minimizing the change to the model's behavior. (Y-parameters: [5], S-parameters [7]). The passivity assessment is based on half-size test matrices (Y-parameters: [4], S-parameters [6]). **`RPdriver.m`** makes use of routine "quadprog" in the Matlab Optimization Toolbox.
- **`netgen_ATP.m`** exports the obtained rational model into the ATP simulation environment.

An overview of the procedure is shown in Fig. 1.1.

The program system has been tested on Matlab v7.5.0. All timing results are with a desktop computer running under Windows XP with a Pentium 2.5 GHz dual core CPU.

Download site (**matrix_fitting_toolbox_1.zip**):
http://www.energy.sintef.no/Produkt/VECTFIT/index.asp

Restrictions of use:

- Embedding any of (or parts from) the routines of the Matrix 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 me made as follows:
  - **`VFdriver.m`** and/or **`vectfit3.m`**: References [1],[2],[3]
  - **`RPdriver.m`** and/or **`FRPY.m`** applied to Y-parameters: [4],[5]
  - **`RPdriver.m`** and/or **`FRPS.m`** applied to S-parameters: [6],[7]

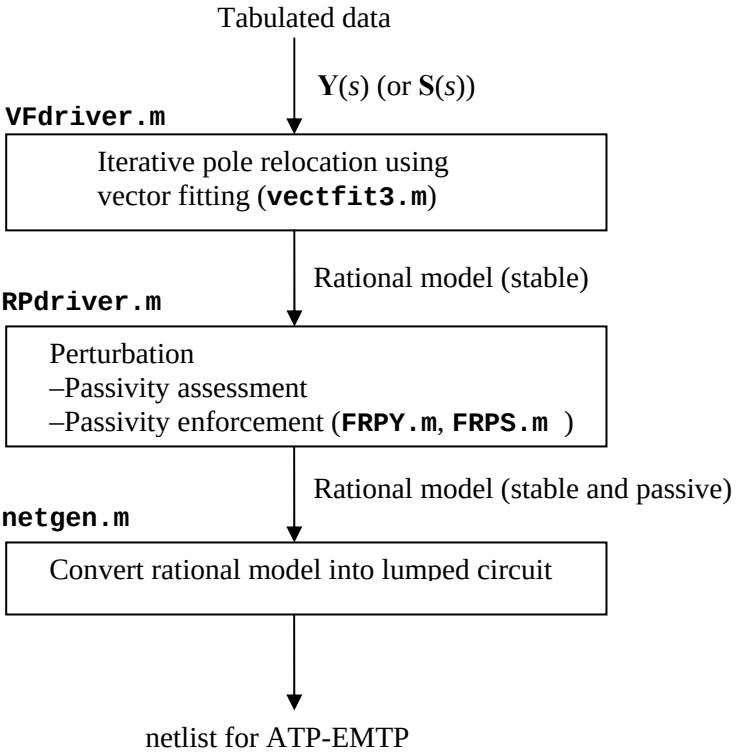


Fig. 1.1 Program overview

2. THE PACKAGE

2.1 Program Files

The package (**Matrix_Fitting_Toolbox_1.zip**) consists of the following files:

Documentation

user_manual_Matrix Fitting Toolbox.pdf	This document

Matlab routines:

VFdriver.m	Driver routine for rational fitting (calls **vectfit3.m**).
RPdriver.m	Driver routine for passivity enforcement
netgen_ATP.m	Generation of model for ATP simulation environment

Auxiliary routines:

vectfit3.m	**violextremaY.m**	**intercheig.m**	**fitcalcPRE.m**
FRPY.m	**violextremaS.m**	**interchsvd.m**	**fitcalcABCDE.m**
FRPS.m	**rot.m**	**pr2ss.m**	

Other files:

ex1_Y.m	Tutorial example (Y-parameters)
ex2_Y.m	Network equivalencing (Y-parameters)
ex3_Y.m	Network equivalencing (Y-parameters)
ex4_S.m	Network equivalencing (S-parameters)
ex2_Y.mat	s-domain data for **ex2_Y.m**
ex3_Y.mat	rational model for **ex3_Y.m**
ex4_S.mat	s-domain data for **ex4_S.m**
circuit.atp	ATP data file for example in Section 4.2.

Reference material (technical papers)

VF.pdf	[1] Standard Vector Fitting (VF)
VFrelaxed.pdf	[2] VF with relaxation
VFfast.pdf	[3] VF with fast implementation
passivitytestY.pdf	[4] Passivity test for Y-parameter model
passivityenforcementY.pdf	[5] Passivity enforcement for Y-parameter model
passivitytestS.pdf	[6] Passivity test for S-parameter model
passivityenforcementS.pdf	[7] Passivity enforcement for S-parameter model

2.2 Instalment

- Place all files in a common directory, e.g.

  ```
  c:\user\mtrxfit
  ```

- Include the directory in the Matlab search path,

  ```
  >>addpath c:\user\mtrxfit
  ```

3. FUNCTION CALL

3.1 VFdriver.m

The following function call with generate a pole-residue model for a data set (*s*, **H**(*s*)) with *n* ports and *Ns* frequency samples. (**H** can be Y- or S-parameter data).

```
[SER,rmserr,Hfit,opts2]=VFdriver(H,s,poles)
[SER,rmserr,Hfit,opts2]=VFdriver(H,s,poles,opts)
```

Input:

H : (n,n,Ns) 3D matrix holding the **H**-samples

s : (1,Ns) vector holding the frequency samples, $s = j\omega$ [rad/sec].

poles: (1,N) vector holding the initial poles (manual specification of initial poles). Use **opts** for automated specification.

opts is an optional structure that can be used for overriding defaults settings, and for requesting plots. (Example: `opts.N=30; opts.poletype='lincmplx'`).

Parameter	Purpose/Description	Default
	Automated generation of initial poles, taken as complex conjugate pairs that are distributed over the frequency band. Specify linear or logarithmic distribution, or a mix of the two. (Note: To invoke this option, specify the input array "poles" to be empty (poles=[])).	
N	Fitting order (integer)	–
poletype	Allowed values: 'lincmplx','logcmplx','linlogcmplx'	'lincmplx'
nu	Ratio between real and imag. part (pole=-nu*beta ± j*beta)	0.001
Niter1	n.o. VF iterations (fitting sum of matrix elements to obtain an improved pole set).	4
Niter2	n.o. VF iterations (fitting upper triangle of H)	4
weight	Array (Nc,Nc,Ns) containing user-defined weight for sample H(i,j,k) in least squares problem.	[]
weightparam	Automated generation of weight array. Used when opts.weight=[] =1 --> Same weight for all elements: weight(i,j,k)=1 =2 --> weight(i,j,k)=1./abs(H(i,j,k)) inverse weighting =3 --> weight(i,j,k)=1./sqrt(abs(H(i,j,k))) =4 --> weight(k)=1/norm(H(:,:,k)) inverse weighting =5 --> weight(k)=1/sqrt(norm(H(:,:,k)))	1
asymp	Control type of rational model =1 –-> D=0, E=0 =2 --> D~=0, E=0	2

	=3 --> D~=0, E~=0	
stable	Control handling of unstable poles =1 --> Will enforce stable poles poles by flipping any unstables into the left half plane	1
relaxed	=1 --> Will use VF with "relaxed" non-triviality constraint. (faster convergence)	1
plot	=1 --> magnitude plot of fitting result	1
logx	=1 --> plot using logarithmic freq. axis	0
logy	=1 --> plot using logarithmic y-axis	1
errplot	=1 --> include deviation (error) in plot	1
phaseplot	=1 --> additional plot of phase angles	1
screen	=1 --> will echo results to screen during fitting process.	1
cmplx_ss	=1 --> complex state space model (A,B,C,D,E) with diag. A. =0 --> real state space model with block-diagonal A.	1

Additional (advanced) parameters in structure **opts**:

Parameter	Purpose/Description	Default
remove_HFpoles	=1 --> Will remove poles at frequencies above factor_HF*s(end)	0
factor_HF	Is used only if remove_HFpoles=1	1.1
passive_DE	=1 --> Will enforce that D and E have positive eigenvalues	0
passive_DE_TOLD	Neg. eigenvals of D are made positive by this amount	1e-6
passive_DE_TOLE	Neg. eigenvals of E are made positive by this amount	1e-16

Output:

SER is a structure with the model on pole-residue form. For a model with *n* ports and *N* residue matrices, the dimensions are

```
SER.poles: (1,N)
SER.R:     (n,n,N)    (residue matrices)
SER.D:     (n,n)
SER.E:     (n,n)
```

The returned **SER** also holds matrices **A**,**B**,**C** for the associated state space model.

```
SER.A: (nN,nN)
SER.B: (nN,n)
SER.C: (n,nN)
```

rmserr : the resulting RMS-error of the fitting

Hfit : (n,n,Ns) 3D matrix holding the **H**-samples of the rational model

opts2 contains *all* options parameters, including default settings

3.2 RPdriver.m

The following function call will enforce passivity of a rational model generated by **VFdriver**,

```
[SER,Yfit,opts2]=RPdriver(SER,s);
[SER,Yfit,opts2]=RPdriver(SER,s,opts);
```

Input:
SER is a structure with the model on pole-residue form. The contents is the same as for VFdriver, see Section 3.1.
- Fields A,B,C are the input.
- Field SER.E is ignored when applied to S-parameters (opts.parametertype='S')

s : (1,Ns) A vector of frequency samples, $s=j\omega$. The perturbation seeks to minimize the change to $\mathbf{Y}(s)$ at the given samples.
opts Optional structure that can be used for overriding defaults settings, and for requesting plots.

Output:
SER : The perturbed model, on pole residue form and on state space form.
Yfit : (n,n,Ns) 3D matrix holding the *Y*-samples (or *S*-samples) of the perturbed model (at freq. **s**)
opts2: Structure containing all options parameters, including default settings

Parameter	Purpose	Default
parametertype	'Y' --> Specifies Y-parameter model 'S' --> Specifies S-parameter model	'Y'
Niter_out	Max. n.o. iterations in outer loop in "Robust iterations"	10
Niter_in	Max. n.o. iterations in inner loop in "Robust iterations"	0
TOLGD	Y-parameter model: Negative egenvalues of G(s)=Re{Y(s)} and D are attempted to be made positive by an amount TOLGD S-parametermodel: Singular values of S(s) and D are attempted to be made smaller than unity positive by an amount TOLGD	1E-6
TOLE	Negative egenvalues of E are attempted to be made positive by an amount TOLE. (Y-parameter model)	1E-12
cmplx_ss	=1 --> perturbed statespace model on complex form =0 --> perturbed statespace model on real form	1
weightparam	Automated generation of weight array for LS part of contraint problem. (Applies to both Y- and S-parameters) =1 --> Same weight for all elements: weight(i,j,k)=1	1

	=2 --> weight(i,j,k)=1./abs(Y(i,j,k)) inverse weighting =3 --> weight(i,j,k)=1./sqrt(abs(Y(i,j,k))) =4 --> weight(k)=1/norm(Y(:,:,k)) inverse weighting =5 --> weight(k)=1/sqrt(norm(Y(:,:,k)))	
weightfactor	Least Squares weight for out-of band auxiliary samples	1e-3
colinterch	=1 --> Will recover the correct sequence of eigenvalues (singular values) during passivity assessement	1
outputlevel	=1 --> Detailed output to command window =0 --> Essential output to command window	1
plot.s_pass	Array of frequency samples (jw). If provided, eigenvalues of G(s) / singular values of S(s) will during iterations be plotted at these frequencies.	
plot.xlim	If provided, the plot of eig(G(s))/sing(S) will be limited in frequency [Hz] to this band. Syntax: [xlow xhigh]	
plot.ylim	If provided, the plot of eig(G(s)) will be limited in range to this band. Syntax: [ylow yhigh]	

3.3 netgen_ATP.m

Auxiliary routine that exports the rational model into a data file for ATP [8]. The data file contains the branch cards of an equivalent electric circuit. The file can be imported into an ATP data file using the $INCLUDE feature of ATP. The calculation of the network is explained in [9]. Compared to the earlier version in mtrxfit.zip, the current version makes use of two more digits in the representation of the circuit elements.

The (six-character) node names in the ATP-circuit are
`X____1, X____2, X____3, etc`

where '`X`' is a user provided node name, given in input variable NOD.

`netgen_ATP(SER,NOD,fname)`

`SER` Structure holding the rational model, as produced by `VFdriver` or `RPdriver`
`NOD` Single character
`fname` File name where data is to be written.

Example:
```
NOD='A';
fname ='RLC_ATP.txt';
netgen_ATP(SER,NOD,fname);
```

4. USER'S GUIDE

4.1 Hints and advice

VFdriver.m

- `poletype`. This parameter is used for automated generation of an initial pole set. The choice of this parameter depends on the nature of the frequency response to be fitted. If the poles (resonances) appear to be
  - linearly distributed in frequency, choose `'lincmplx'`;
  - logarithmically distributed in frequency, choose `'logcmplx'`;

  Some times (for instance when fitting a transformer response), the poles appear to be logarithmically distributed (and real) at low frequencies, and linearly distributed (and complex) at high frequencies. In such cases, it may be best to choose `'linlogcmplx'`;
- `passive_DE`. Setting this parameter will request `VFdriver` to enforce **D** and **E** of the rational model to be positive definite, meaning that the model is asymptotically passive. If the resulting model is to be submitted to `RPdriver` for a final passivity correction, it is essential that **D** (Y-parameter case) is non-singular since the passivity check by the Hamiltonian matrix requires to compute the inverse of **D**. In the case of S-parameters, we require that **D**+**I** and **D**–**I** are non-singular.

RPdriver.m

- `Niter_out`. This parameter defines the number of iterations for the *outer* loop in Section 5.2.3. The default value is 4 but it may be necessary to increase this value to get rid of all passivity violations.
- `Niter_in`. This parameter defines the number of iterations for the *inner* loop in Section 5.2.3. The default value is three but it may be necessary to increase this value if the passivity enforcement keeps creating new violations.
- `TOLGD`. The routine tries to enforce negative minima of eig(**G**(s)) to be positive by this amount (as well as eigenvalues of **D**). The performance of the routine is dependent on the selected value: using a too small value (close to zero) will often result in an increase of the required number of iterations (due to the nonlinearity of the problem), and **D** may become nearly singular. On the other hand, using a too large value will cause a too large perturbation of the model.

Inaccurate fitting result

If an accurate fitting result cannot be obtained (**`VFdriver.m`**) no matter what order you try, then there is probably something wrong with your frequency response. A rational function in the frequency domain has a real and imaginary part, which are related in a "special way". This means that not all functions are fittable; they have to be "physical". So the first requirement is that that the frequency response (s,**Y**(s)) is not corrupted in some way. The default setting for

VFdriver is to enforce stable poles (`opts.stable=1`). Try to allow unstable poles (`opts.stable=0`) and see if the problem goes away.

Divergence of passivity enforcement

Observe what happens to “Max. violation” in the Matlab command window during iterations. If there is a tendency of divergence, specify a non-zero value for `opts.Niter_in` in order to enable the inner iteration loop. It may also help to increase the value of `TOLGD`.

Also, if the model is created by `VFdriver` using `passive_DE=1`, make sure that `passive_DE_TOLD` is positive and non-zero. Try to increase the value of this parameter.

Crash in passivity enforcement

If you specify `opts.asymp=1` in the call to **`VFdriver.m`**, you will get a model with **D**=0. In that case, **D** is singular and so the passivity assessment will crash in `RPdriver.m`.

4.2 Example 1: Electrical circuit

File: **`ex1_Y.m`**
In this small example is demonstrated the usage of `VFdriver` as well as the generation of simulation models for ATP and EMTP-RV.

4.2.1 Data case

Consider the electrical network in Fig. 4.2.1 (quantities given in units [Ω], [H], [F]). We wish to calculate a black box model of the equivalent with respect to terminals 1 and 2 when only the frequency response at the terminals are known.

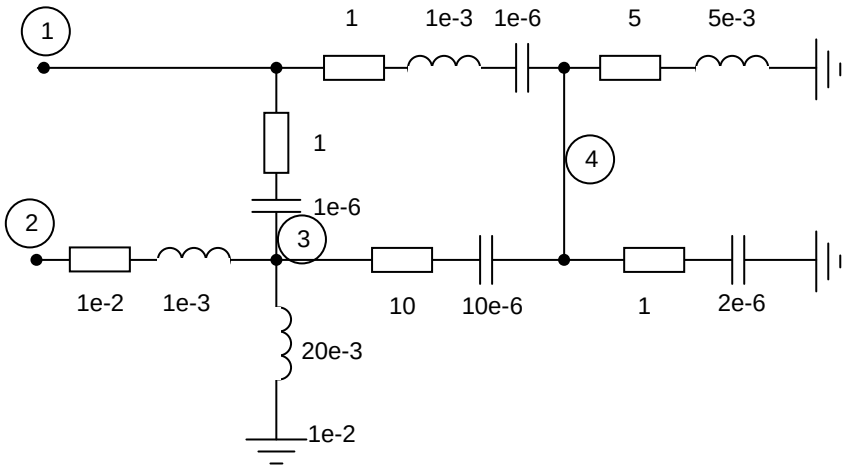


Fig. 4.2.1 Electric circuit

```
% This file is part of the Matrix Fitting Toolbox, v1.
% Filename: ex1_Y.m
% Package: Matrix_Fitting_Toolbox_1.zip.
% Programmed by B. Gustavsen. October 08, 2008.
%
clear all

Ns=501;    %Number of frequency samples
Nc=2;      %Size of Y (after reduction)
bigY=zeros(Nc,Nc,Ns);
Y=zeros(4,4);
s=2*pi*i*logspace(1,5,Ns);

%Component values:
R1=1;    L1=1e-3; C1=1e-6;
R2=5;    L2=5e-3;
R3=1;    C3=1e-6;
L4=1e-3;
R4=1e-2; L5=20e-3;
R6=10;   C6=10e-6;
R7=1;    C7=2e-6;

%Building Y, reduction:
```

```
for k=1:Ns
  sk=s(k);
  y1=1/( R1+sk*L1+1/(sk*C1) );
  y2=1/( R2+sk*L2 );
  y3=1/( R3+1/(sk*C3));
  y4=1/( R4+sk*L4);
  y5=1/(sk*L5);
  y6=1/( R6+1/(sk*C6) );
  y7=1/( R7+1/(sk*C7) );

  Y(1,1)= y1+y3;
  Y(2,2)= y4;
  Y(3,3)= y3 +y4 +y5 +y6;
  Y(4,4)= y1 +y2 +y6 +y7;

  Y(1,3)=-y3; Y(1,4)=-y1;
  Y(2,3)=-y4;
  Y(3,1)=-y3; Y(3,2)=-y4; Y(3,4)=-y6;
  Y(4,1)=-y1; Y(4,3)=-y6;

  %Eliminating nodes 3 and 4:
  Yred=Y(1:2,1:2)-Y(1:2,3:4)*Y(3:4,3:4)^(-1)*Y(3:4,1:2);
  bigY(:,:,k)=Yred;
  %bigY(:,:,k)=diag(diag(Yred));
end

%================================================
%=           POLE-RESIDUE FITTING               =
%================================================
opts.N=8;              %Order of approximation. (Is used when opts.poles=[]).
opts.poletype='logcmplx';   %Will use logarithmically spaced, complex poles. (Is
used when opts.poles=[]).
poles=[]; %[] -->N initial poles are automatically generated as defined by
opts.startpoleflag
[SER,rmserr,bigYfit,opts2]=VFdriver(bigY,s,poles,opts); %Creating state-space model
and pole-residue model
```

4.2.2 Running the example

Running **`ex1_Y.m`** from Matlab's command window gives the output to screen shown below.

```
>> ex1_Y
-----------------S T A R T-------------------------
****Stacking matrix elements (lower triangle) into single column ...
****Calculating improved initial poles by fitting column sum ...
   Iter 1
   Iter 2
   Iter 3
   Iter 4
****Fitting column ...
   Iter 1
   Iter 2
   Iter 3
   Iter 4
Elapsed time is 0.198631 seconds.
****Transforming model of lower matrix triangle into state-space model of full
matrix ...
****Generating pole-residue model ...
****Plotting of results ...
-------------------E N D--------------------------
****Creating equivalent circuit for ATP...
>>
```

The result in Fig. 4.2.2 shows that the approximation is highly accurate as the deviation is close to machine precision. (The plot is automatically generated).

Fig. 4.2.2 Generated plot of rational fitting

In addition to the parameters specified in the input opts, there several other parameters that are set internally in **`VFdriver.m`**. All settings that were used are returned in structure `opts2`. (Note that the parameters that were manually overridden in **`ex1_Y.m`** are listed first (`N`, `poletype`). Using input array `opts`, all default settings can be changed, see Section 3.1.

```
>> opts2
                 N: 8
          poletype: 'logcmplx'
                nu: 1.0000e-003
            Niter1: 4
            Niter2: 4
            weight: []
       weightparam: 1
             asymp: 2
            stable: 1
           relaxed: 1
              plot: 1
              logx: 1
              logy: 1
           errplot: 1
         phaseplot: 1
            screen: 1
          cmplx_ss: 1
    remove_HFpoles: 0
         factor_HF: 1.1000e+000
        passive_DE: 0
      passive_TOLD: 1.0000e-006
      passive_TOLE: 1.0000e-016
```

Pole-residue model:

```
>> SER.poles
ans =
 -4.7619e-001
 -1.2876e+005
 -1.0229e+003 +3.5994e+003i
 -1.0229e+003 -3.5994e+003i
 -2.2888e+003 +1.8044e+004i
 -2.2888e+003 -1.8044e+004i
 -1.0116e+003 +3.8290e+004i
 -1.0116e+003 -3.8290e+004i

>> SER.R
ans(:,:,1) =
 -1.7930e-014 -2.1595e-007
 -2.1595e-007  4.7619e+001
ans(:,:,2) =
 -1.0019e+004 -5.3834e+002
 -5.3834e+002 -2.8926e+001
ans(:,:,3) =
  2.1958e-001 +2.2124e-001i  3.7368e+000 +1.9303e+000i
  3.7368e+000 +1.9303e+000i  5.5986e+001 +9.2914e+000i
ans(:,:,4) =
  2.1958e-001 -2.2124e-001i  3.7368e+000 -1.9303e+000i
  3.7368e+000 -1.9303e+000i  5.5986e+001 -9.2914e+000i
ans(:,:,5) =
  1.8288e+002 +6.5934e+001i -2.6855e+002 -7.6004e+001i
 -2.6855e+002 -7.6004e+001i  3.9227e+002 +8.1793e+001i
ans(:,:,6) =
  1.8288e+002 -6.5934e+001i -2.6855e+002 +7.6004e+001i
 -2.6855e+002 +7.6004e+001i  3.9227e+002 -8.1793e+001i
```

```
ans(:,:,7) =
  1.3290e+002 +1.8383e+001i  7.5653e+001 +9.8304e-001i
  7.5653e+001 +9.8304e-001i  4.2401e+001 -4.7459e+000i
ans(:,:,8) =
  1.3290e+002 -1.8383e+001i  7.5653e+001 -9.8304e-001i
  7.5653e+001 -9.8304e-001i  4.2401e+001 +4.7459e+000i

>> SER.D
ans =
  8.3333e-002  1.6229e-017
  1.6229e-017 -9.0221e-018

>> SER.E
ans =
     0     0
     0     0
```

State-space model:

The state-space model has a diagonal **A** with complex-valued **A** and **C**, as requested by parameter `opts.cmplx_ss=1`.

```
>> SER.A
ans =
   (1,1)    -4.7619e-001
   (2,2)    -1.2876e+005
   (3,3)    -1.0229e+003 +3.5994e+003i
   (4,4)    -1.0229e+003 -3.5994e+003i
   (5,5)    -2.2888e+003 +1.8044e+004i
   (6,6)    -2.2888e+003 -1.8044e+004i
   (7,7)    -1.0116e+003 +3.8290e+004i
   (8,8)    -1.0116e+003 -3.8290e+004i
   (9,9)    -4.7619e-001
  (10,10)   -1.2876e+005
  (11,11)   -1.0229e+003 +3.5994e+003i
  (12,12)   -1.0229e+003 -3.5994e+003i
  (13,13)   -2.2888e+003 +1.8044e+004i
  (14,14)   -2.2888e+003 -1.8044e+004i
  (15,15)   -1.0116e+003 +3.8290e+004i
  (16,16)   -1.0116e+003 -3.8290e+004i

>> SER.B

ans =

     1     0
     1     0
     1     0
     1     0
     1     0
     1     0
     1     0
     1     0
     0     1
     0     1
     0     1
     0     1
     0     1
     0     1
     0     1
     0     1


>> (SER.C).'
ans =
 -1.7930e-014              -2.1595e-007
 -1.0019e+004              -5.3834e+002
  2.1958e-001 +2.2124e-001i  3.7368e+000 +1.9303e+000i
  2.1958e-001 -2.2124e-001i  3.7368e+000 -1.9303e+000i
  1.8288e+002 +6.5934e+001i -2.6855e+002 -7.6004e+001i
  1.8288e+002 -6.5934e+001i -2.6855e+002 +7.6004e+001i
  1.3290e+002 +1.8383e+001i  7.5653e+001 +9.8304e-001i
```

```
  1.3290e+002 -1.8383e+001i  7.5653e+001 -9.8304e-001i
 -2.1595e-007                4.7619e+001
 -5.3834e+002               -2.8926e+001
  3.7368e+000 +1.9303e+000i  5.5986e+001 +9.2914e+000i
  3.7368e+000 -1.9303e+000i  5.5986e+001 -9.2914e+000i
 -2.6855e+002 -7.6004e+001i  3.9227e+002 +8.1793e+001i
 -2.6855e+002 +7.6004e+001i  3.9227e+002 -8.1793e+001i
  7.5653e+001 +9.8304e-001i  4.2401e+001 -4.7459e+000i
  7.5653e+001 -9.8304e-001i  4.2401e+001 +4.7459e+000i
```

The **D** and **E** matrices are the same as for the pole-residue model.

With `opts.cmplx_ss=0`, a real-only state-space model is produced,

```
   (1,1)    -4.7619e-001
   (2,2)    -1.2876e+005
   (3,3)    -1.0229e+003
   (4,3)    -3.5994e+003
   (3,4)     3.5994e+003
   (4,4)    -1.0229e+003
   (5,5)    -2.2888e+003
   (6,5)    -1.8044e+004
   (5,6)     1.8044e+004
   (6,6)    -2.2888e+003
   (7,7)    -1.0116e+003
   (8,7)    -3.8290e+004
   (7,8)     3.8290e+004
   (8,8)    -1.0116e+003
   (9,9)    -4.7619e-001
  (10,10)   -1.2876e+005
  (11,11)   -1.0229e+003
  (12,11)   -3.5994e+003
  (11,12)    3.5994e+003
  (12,12)   -1.0229e+003
  (13,13)   -2.2888e+003
  (14,13)   -1.8044e+004
  (13,14)    1.8044e+004
  (14,14)   -2.2888e+003
  (15,15)   -1.0116e+003
  (16,15)   -3.8290e+004
  (15,16)    3.8290e+004
  (16,16)   -1.0116e+003


>> SER.B

ans =

     1     0
     1     0
     2     0
     0     0
     2     0
     0     0
     2     0
     0     0
     0     1
     0     1
     0     2
     0     0
     0     2
     0     0
     0     2
     0     0

>> (SER.C).'

ans =

 -1.7930e-014 -2.1595e-007
 -1.0019e+004 -5.3834e+002
  2.1958e-001  3.7368e+000
  2.2124e-001  1.9303e+000
```

```
 1.8288e+002 -2.6855e+002
 6.5934e+001 -7.6004e+001
 1.3290e+002  7.5653e+001
 1.8383e+001  9.8304e-001
-2.1595e-007  4.7619e+001
-5.3834e+002 -2.8926e+001
 3.7368e+000  5.5986e+001
 1.9303e+000  9.2914e+000
-2.6855e+002  3.9227e+002
-7.6004e+001  8.1793e+001
 7.5653e+001  4.2401e+001
 9.8304e-001 -4.7459e+000
```

4.2.3 Generation of simulation model for ATP

ex1_Y.m calls **netgen_ATP.m** :

```
NOD='A';
fname ='RLC_ATP.txt';
netgen_ATP(SER,NOD,fname);  %Creating branch-cards for ATP
```

This produces a file `RLC_ATP.txt` with the following contents:

```
$VINTAGE,1
C <BUS1><BUS2><BUS3><BUS4><   OHM        ><   milliH     ><   microF     >
C
$VINTAGE,1
C <BUS1><BUS2><BUS3><BUS4><   OHM        ><   milliH     ><   microF     >
C
C (1,1)
  A____1                    1.20000000e+001
  A____1                                                 3.00000000e-010
  A____1                   -2.20505113e+006-4.63060716e+009
  A____1                   -1.21959492e+001-9.47194746e-002
  A____1A 3__1              3.76645764e+002 1.26376926e+002
  A 3__1                   -2.27028923e+003
  A 3__1                                                 4.71374555e-001
  A____1A 5__1             -2.57338708e+001-5.83599210e+000
  A 5__1                   -1.14565657e+004
  A 5__1                                                -5.19129996e-001
  A____1A 7__1              1.09495943e+001 2.39745382e+000
  A 7__1                   -1.39356194e+003
  A 7__1                                                 2.82071301e-001
C (1,2)
  A____1A____2             -8.60370102e+015
  A____1A____2                                          -1.00000000e-010
  A____1A____2              2.20505112e+006 4.63060713e+009
  A____1A____2              2.39175279e+002 1.85754765e+000
  A____1A 3_12             -3.85654189e+002-1.33803037e+002
  A 3_12A____2              2.62569679e+003
  A 3_12A____2                                          -4.55368132e-001
  A____1A 5_12              1.37691254e+001 1.86184206e+000
  A 5_12A____2             -2.32353531e+002
  A 5_12A____2                                           1.52736719e+000
  A____1A 7_12             -9.97399817e+000-6.60910817e+000
  A 7_12A____2             -1.88522825e+004
  A 7_12A____2                                          -1.03186254e-001
C (2,2)
  A____2                    3.25513438e+015
  A____2                                                 3.00000000e-010
  A____2                    9.99999910e-003 2.09999971e+001
  A____2                   -2.26979329e+002-1.76282818e+000
  A____2A 3__2              1.42263095e+001 8.37207251e+000
  A 3__2                    3.23949426e+002
  A 3__2                                                 8.90531215e+000
  A____2A 5__2              1.26628586e+001 4.04153239e+000
  A 5__2                    9.12933873e+002
  A 5__2                                                 7.58320546e-001
  A____2A 7__2             -8.84566299e-001 4.23533375e+000
  A 7__2                    2.78476100e+003
  A 7__2                                                 1.60882715e-001
$VINTAGE,0
```

Nodes A____1 and A____2 represent the terminals of the equivalent.

This circuit is conveniently imported into ATP-EMTP using the $INCLUDE statement imbedded in an ATP data file.

Consider the situation that a unit step voltage is applied to terminal 1 and we wish to calculate the resulting current flowing into terminal 2, see Fig. 4.2.3.

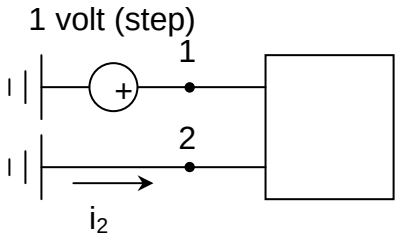


Fig. 4.2.3 ATP-simulation

An ATP-file for the simulation case in Fig. 4.2.3 is shown below, given in file `circuit.atp`. (A 1 μΩ resistor is used for the current measuremet). Note that the equivalent electrical network is imported using the $INCLUDE statement, and that the node names are A____1 and A____2. (To run this ATP-case, be sure to update the path statement in circuit.atp).

```
BEGIN NEW DATA CASE
C
C  Example described in user_manual.pdf
C
C     ======================================================
C     =   File:   circuit.atp                              =
C     =   Version 1.0                                      =
C     =   Last revised: 19.03.2002                         =
C     =   Programmed by: Bjorn Gustavsen,                  =
C     =   SINTEF Energy Research, N-7465 Trondheim, NORWAY =
C     =   This file is part of the "matrixfitter-package"  =
C     ======================================================
C
$DUMMY, XYZ000
C
C
C FIRST MISCELLANEOUS DATA CARD
$DUMMY, XYZ000
C FIRST MISCELLANEOUS DATA CARD
C  dT  >< Tmax >< Xopt >< Copt >
1.000E-6    .005
C
C SECOND MISCELLANEOUS DATA CARD
C    1-8    9-16   17-24   25-32   33-40   41-48   49-56   57-64   65-72   73-80
C  PRINT    PLOT NETWORK   PR.SS  PR.MAX   I PUN   PUNCH    DUMP   MULT. DIAGNOS
C 0=EACH  0=EACH   0= NO   0= NO   0= NO   0= NO   0= NO    INTO  ENERG.   PRINT
C K=K-TH  K=K-TH   1=YES   1=YES   1=YES   1=YES   1=YES    DISK STUDIES    0=NO
   10000       1       0       0                                   1
C
C BRANCHES
C 3456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
C    3-8  9-14 15-20 21-26 27-32 33-38 39-44
C   NODE NAMES   REFERENCE  RES.  IND.  CAP.               (OUTPUT IN COLUMN 80)
C                   BRANCH          MH    UF                                 I=   1
C <BUS1><BUS2><BUS3><BUS4>   OHM   OHM  UMHO                                 V=   2
C                                                                           I.V  3
        A____2             1.E-6                                                 1
$INCLUDE C:\user\mtrx_fitter_new\RLC_ATP.txt
C
BLANK CARD TERMINATING BRANCH CARDS
C SWITCH CARDS
C 3456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
C    3-8  9-14     15-24     25-34     35-44     45-54     55-64     65-74
C                                                 (OUTPUT OPTION IN COLUMN 80)
C   NODE NAMES                              IE FLASHOVER   SPECIAL   REFERENCE
C                TIME TO   TIME TO        OR   VOLTAGE   REQUEST SWITCH-NAME
C   BUS1  BUS2     CLOSE      OPEN   NSTEP                  WORD  BUS5  BUS6
C
BLANK CARD TERMINATING SWITCH CARDS
```

```
C SOURCE CARDS
C 34567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
C COLUMN 1.2: TYPE OF SOURCE 1   17.(E.G. 11-13 ARE RAMP FUNCTIONS. 14 = COSINE)
C COLUMN 9.10: 0=VOLTAGE SOURCE.  1=CURRENT SOURCE
C    3-8       11-20     21-30     31-40     41-50     51-60     61-70     71-80
C   NODE   AMPLITUDE FREQUENCY TO IN SEC   AMPL-A1   TIME-T1   T-START    T-STOP
C   NAME                IN HZ      DEGR             SECONDS   SECONDS   SECONDS
14A____1         1.0     0.001        0.                           0.         1.
C
BLANK CARD TERMINATING SOURCE CARD
C NODE VOLTAGE OUTPUT
C 34567890123456789012345678901234567890123456789012345678901234567890123456789012345678901234567890
C    3-8  9-14 15-20 21-26 27-32 33-38 39-44 45-50 51-56 57-62 63-68 69-74 75-80
C   BUS1  BUS2  BUS3  BUS4  BUS5  BUS6  BUS7  BUS8  BUS9 BUS10 BUS11 BUS12 BUS13
C  A____1A____2
C  NBY1A NBY1B NBY1C NBY6A NBY6B NBY6C NEUT  NBY6G GENA  GENB  GENC
BLANK CARD TERMINATING OUTPUT CARDS
BLANK CARD ENDING PLOT CARDS
BEGIN NEW DATA CASE
```

At the end of `ex1_Y.m`, the theoretical setp response is calculated:

```
% Plotting step response:
NN=length(SER.A) ; I=ones(NN,1);
t=(0:1e-5:5e-3); Nt=length(t);
for k=1:Nt
  if opts2.cmplx_ss==1
    y=SER.C*diag( diag(SER.A).^(-1).*(exp(diag(SER.A).*t(k))-I) )*SER.B +SER.D;
  else
    y=SER.C*( (SER.A)^(-1))*((expm((full(SER.A)).*t(k))-diag(I)) ) *SER.B +SER.D;
  end
  yy(k)=y(2,1);
end
figure(4),plot(1000*t,yy);
xlabel('Time [ms]'); ylabel('Current [A]');
```

Fig. 2.4 Compares the theoretical solution with the ATP simulation result.

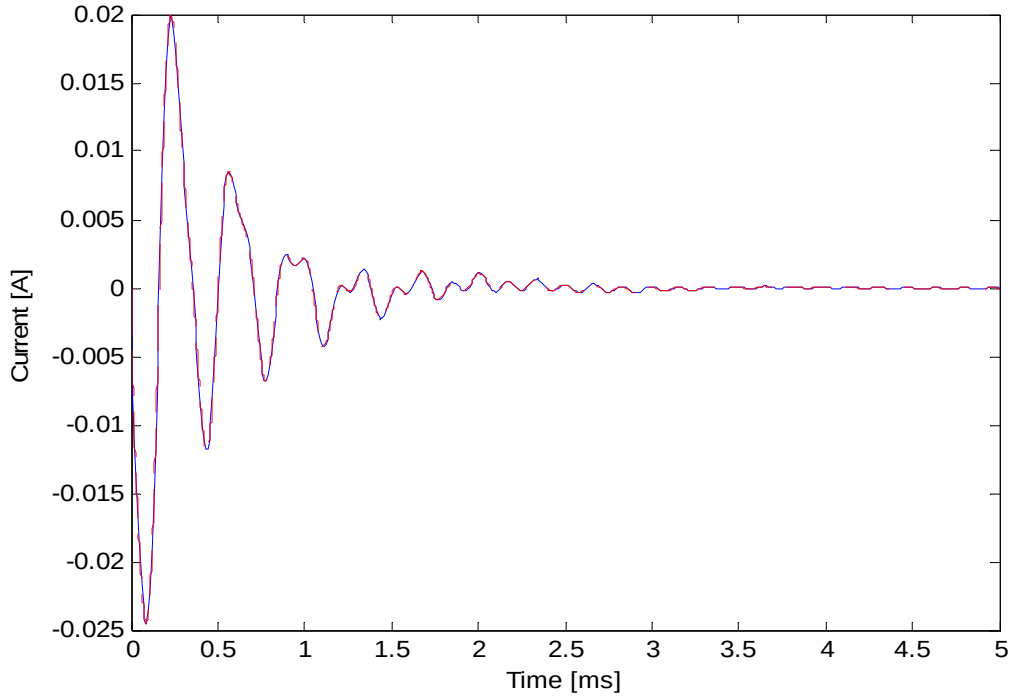


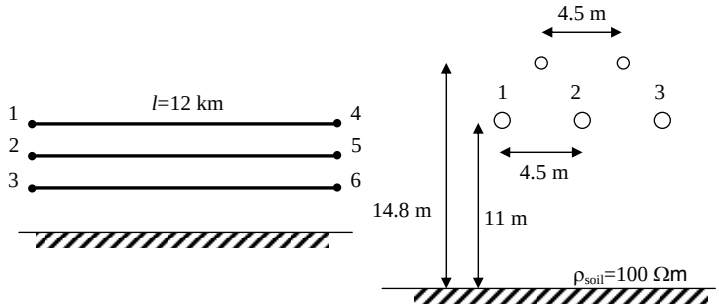
Fig. 4.2.4 Step voltage response. Blue: theoretical, red: ATP simulation.

4.3 Example 2: Network equivalencing

File: **ex2_Y.m**
In this second example we will demonstrate both rational fitting (**VFdriver.m**) and passivity enforcement **(RPdriver.m**).

4.3.1 Generation of frequency domain data

We consider a three-conductor overhead line of length 12 km, see Fig. 4.3.1. The admittance matrix **Y** is computed with respect to one line end with other line end open circuited.



Phase wires: Rdc=0.121 Ω/km, d=21.66 mm
Ground wires: Rdc=0.359 Ω/km, d=12.33 mm

Fig. 4.3.1. 132 kV overhead line

On top of **ex2_Y.m**, the example case is loaded from file,

```
load fdne %-->s, bigY
```

4.3.2 Rational fitting

In **ex2_Y.m** , the following call creates the a 50th order rational model of the 3×3 Y.

```
%================================================
%=           POLE-RESIDUE FITTING               =
%================================================
opts.N=50 ;%           %Order of approximation.
opts.poletype='linlogcmplx'; %Mix of lin. spaced and log. spaced poles
opts.weightparam=5; %5 --> weighting with inverse magnitude norm
opts.Niter1=7;    %Number of iterations for fitting sum of elements (fast!)
opts.Niter2=4;    %Number of iterations for matrix fitting
opts.asymp=2;      %Fitting includes D
opts.logx=0;       %=0 --> Plotting is done using linear abscissa axis
poles=[];

[SER,rmserr,bigYfit,opts2]=VFdriver(bigY,s,poles,opts);
```

```
>> ex2_Y
-----------------S T A R T--------------------------
****Stacking matrix elements (lower triangle) into single column ...
****Calculating improved initial poles by fitting column sum ...
   Iter 1
   Iter 2
   Iter 3
   Iter 4
   Iter 5
   Iter 6
   Iter 7
****Fitting column ...
   Iter 1
   Iter 2
   Iter 3
   Iter 4
Elapsed time is 0.649053 seconds.
****Transforming model of lower matrix triangle into state-space model of full
matrix ...
****Generating pole-residue model ...
****Plotting of results ...
-------------------E N D--------------------------
```

The magnitude plot is shown in Fig. 4.3.2. The order is in this case slightly too low so that an inaccurate fitting results locally.

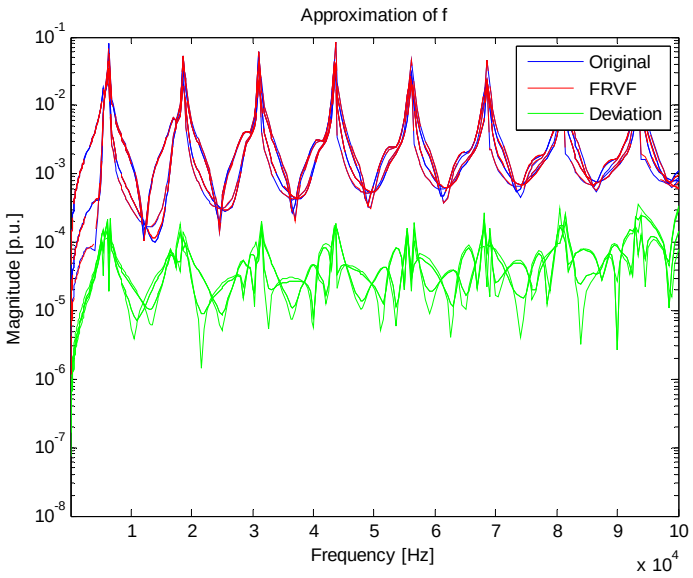


Fig. 4.3.2. Rational fitting

4.3.3 Passivity enforcement

The inaccurate fitting results in that the obtained model is non-passive. In **ex2_Y.m** , the following enforces passivity,

```
%================================================
%=           Passivity Enforcement              =
%================================================
clear opts;
opts.parametertype='Y';
opts.plot.s_pass=2*pi*i*linspace(0,2e5,1001).';
opts.plot.ylim=[-2e-3 2e-3];

[SER,bigYfit_passive,opts3]=RPdriver(SER,s,opts);
```

This gives an output to the screen as shown below. It is seen that passivity is enforced in 8.6 sec. The total time was reduced to 5.7 sec by removing plotting, achieved by commenting out as follows,

```
%opts.plot.s_pass=2*pi*i*linspace(0,2e5,1001).';
%opts.plot.ylim=[-2e-3 2e-3];
```

```
-----------------S T A R T--------------------------
*** Y-PARAMETERS ***

  [ 1  0 ]
  Passivity Assessment:
    N.o. violating intervals: 3
    Max. violation, eig(G) : -0.0035493
    Max. violation, eig(D) : -0.00010945
    Max. violation, eig(E) :  None
  Passivity Enforcement...
    Building system equation (once)...
    Done
Optimization terminated.

  [ 2  0 ]
  Passivity Assessment:
    N.o. violating intervals: 1
    Max. violation, eig(G) : -4.2799e-005
    Max. violation, eig(D) :  None
    Max. violation, eig(E) :  None
  Passivity Enforcement...
Optimization terminated.

-->Passivity was successfully enforced.
   Max. violation, eig(G) :  None
   Max. violation, eig(D) :  None
   Max. violation, eig(E) :  None
Time summary:
   Passivity assessment : 0.16619 sec
   Passivity enforcement: 1.8029 sec
   Total: 1.9661 sec
------------------E N D--------------------------
```

In the output dialogue,

- The bracket values [x y] denoted the iteration count for the outer (x) and inner (y) loops of the "robust iteration" (Section 5.2.3). In this case, the inner loop iterations have been disabled (`opts.Niter_in=0`; default value).
- The **D** has initially negative eigenvalues. The passivity enforcement takes care of this, making the violating eigenvalue positive by an amount 1*E*–6, as requested by `opts.TOLGD=1e-6;`

```
>> eig(SER.D)
ans =
  1.0000e-006
  1.0470e-003
  1.2376e-003
```

Thee eigenvalues of **G**=real(**Y**)) are plotted during the iterations, since the parameter `opts.plot.s_pass` was provided in the call. Fig. 4.3.3 shows the final result. It is observed that the eigenvalues are all positive after the passivity enforcement step.

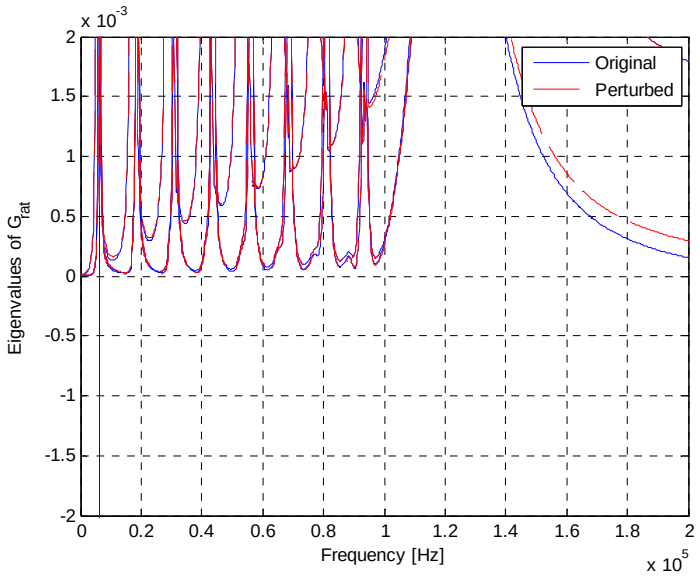


Fig. 4.3.3 Eigenvalues of **G**(*s*).

You can check on all settings that were used in the passivation process via the returned output structure `opts3`. (These values can be changed via the input structure `opts`).

```
>>opts3 =
          method: 'FRP'
    parametertype: 'Y'
        Niter_out: 10
         Niter_in: 0
            TOLGD: 1.0000e-006
             TOLE: 1.0000e-012
        cmplx_ss: 1
      weightparam: 1
     weightfactor: 1.0000e-003
       colinterch: 1
      outputlevel: 1
```

At the end of the file (**`ex2_Y.m`**), the original model is compared with the perturbed model (at samples defined by array *s* in call to **`RPdriver.m`**), see Fig. 4.3.4. It is seen that the change to the elements of **Y** is very small (within the fitting band), and that the deviation is "parallel" to the magnitude curves. In the right panel is shown the same result when setting `opts.method='FRP'`. It is seen that the deviation curves are now much "flatter".

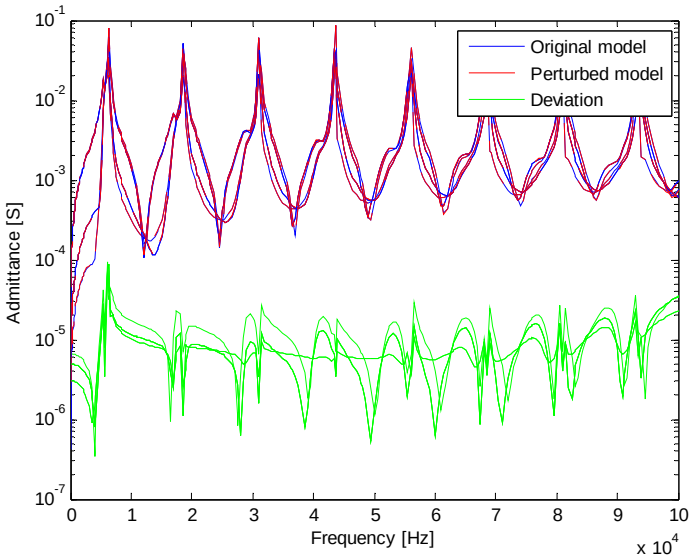


Fig. 4.3.4 The effect of perturbation on the admittance matrix, **Y**

The amount of information to the Matlab command window can be reduced by setting parameter "`opts.outputlevel`" to zero:

```
opts.outputlevel=0;

-----------------S T A R T--------------------------
*** Y-PARAMETERS ***
    Max. violation  : -0.0035493
Optimization terminated.
    Max. violation  : -4.2799e-005
Optimization terminated.
-->Passivity was successfully enforced.
-------------------E N D----------------------------
```

4.4 Example 3: Network equivalencing of transmission line

File: **ex3_Y.m**

Here, we demonstrate **RPdriver** for example ex4b.m in QPpassive.zip. This is a six-port equivalent of the transmission line in Fig. 4.3.1, but with a line length of 45 km. The fitting range is 10 Hz to 10 kHz using 30 pole-residue terms.

```
clear all

load ex3    %--> s, SER

%================================================
%=           Passivity Enforcement              =
%================================================
opts.Niter_in=2;
opts.Niter_out=20;
opts.parametertype='Y';
opts.plot.s_pass=2*pi*i*linspace(0,3e4,1001).';
opts.plot.ylim=[-2e-3 2e-3];
[SER2,bigYfit_passive,opts2]=RPdriver(SER,s,opts);


>> ex3_Y
-----------------S T A R T--------------------------
*** Y-PARAMETERS ***

  [ 1  0 ]
  Passivity Assessment:
    N.o. violating intervals: 3
    Max. violation, eig(G) : -0.0053053 @ 11991.6316 Hz
    Max. violation, eig(D) :  None
    Max. violation, eig(E) :  None
  Passivity Enforcement...
    Building system equation (once)...
    Done
Optimization terminated.

  [ 1  1 ]
  Passivity Assessment:
  Passivity Enforcement...
Optimization terminated.

  [ 1  2 ]
  Passivity Assessment:
  Passivity Enforcement...
Optimization terminated.

  [ 2  0 ]
  Passivity Assessment:
    N.o. violating intervals: 1
    Max. violation, eig(G) : -7.8331e-006 @ 11913.9947 Hz
    Max. violation, eig(D) :  None
    Max. violation, eig(E) :  None
  Passivity Enforcement...
Optimization terminated.

  [ 2  1 ]
  Passivity Assessment:
  Passivity Enforcement...
Optimization terminated.

  [ 2  2 ]
```

```
  Passivity Assessment:
  Passivity Enforcement...
Optimization terminated.

  [ 3  0 ]
  Passivity Assessment:

-->Passivity was successfully enforced.
   Max. violation, eig(G) :  None
   Max. violation, eig(D) :  None
   Max. violation, eig(E) :  None
Time summary:
   Passivity assessment : 0.5389 sec
   Passivity enforcement: 4.2076 sec
   Total: 5.0695 sec
-------------------E N D----------------------------
```

The amount of information to the Matlab command window can be reduced by setting parameter "`opts.outputlevel`" to zero:

```
opts.outputlevel=0;
```

```
>> ex3_Y
-----------------S T A R T--------------------------
*** Y-PARAMETERS ***
    Max. violation  : -0.0053053
Optimization terminated.
Optimization terminated.
Optimization terminated.
    Max. violation  : -7.8331e-006
Optimization terminated.
Optimization terminated.
Optimization terminated.
-->Passivity was successfully enforced.
-------------------E N D----------------------------
>>
```

- Fig. 4.4.1 shows plot of the eigenvalues of **G**(*s*), as generated by **`RPdriver.m`**. It is seen that the perturbation makes the eigenvalues positive.
- Fig. 4.4.2 shows the plot of the elements of **Y**(*s*), as generated by **`ex3_Y.m`**. It is seen that the change to the elements is very small inside the fitting band (10 Hz – 10 kHz).

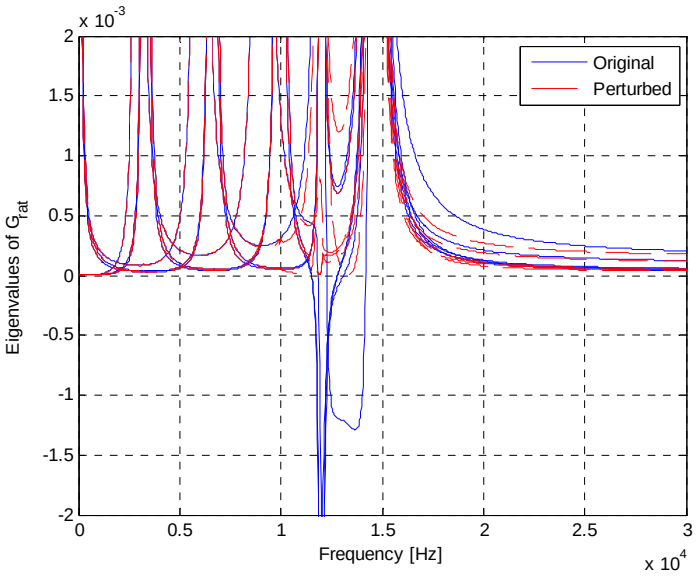


Fig. 4.4.1 Eigenvalues of **G**(*s*).

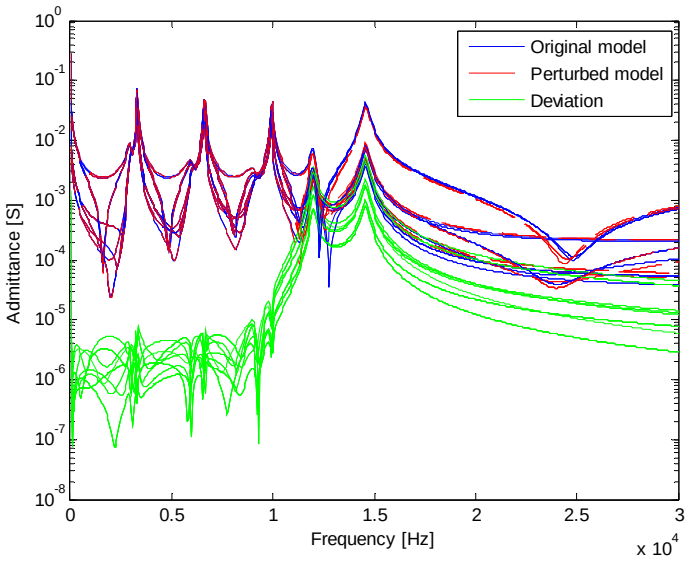


Fig. 4.4.2 The effect of perturbation on the admittance matrix, **Y**(s)

4.5 Example 4: Network equivalencing: S-parameters

This is the same example as in Section 4.2 (Eample 2), but with the Y-parameters converted to S-parameters before carrying out the rational fitting. The example loads the S-parameters and calculates a rational model using VFdriver. In the subsequent passivity enforcement, no inner-loop iterations are used.

Fig. 4.5.1 shows that the singular values are enforced to be smaller than unity. The change to the model's behavior is small as seen in Fig. 4.5.2.

For a detailed screen output, change "`opts.outputlevel`" from 0 to 1.

```
%================================================
%=           Passivity Enforcement              =
%================================================
clear opts;
opts.plot.s_pass=2*pi*i*linspace(0,2e5,1001).';
opts.plot.ylim=[0.95 1.05];
opts.Niter_in=0;
opts.parametertype='S';
opts.outputlevel=0; %Min. output to screen

[SER,bigYfit_passive,opts3]=RPdriver(SER,s,opts);

>> ex4_S


-----------------S T A R T--------------------------
*** S-PARAMETERS ***
    Max. violation  : 0.014094
Optimization terminated.
    Max. violation  : 0.00027043
Optimization terminated.
    Max. violation  : 4.1078e-005
Optimization terminated.
    Max. violation  : 7.5435e-006
Optimization terminated.
    Max. violation  : 4.4734e-007
Optimization terminated.
->Passivity was successfully enforced.
------------------E N D-------------------------
Elapsed time is 5.681933 seconds.
>>
```

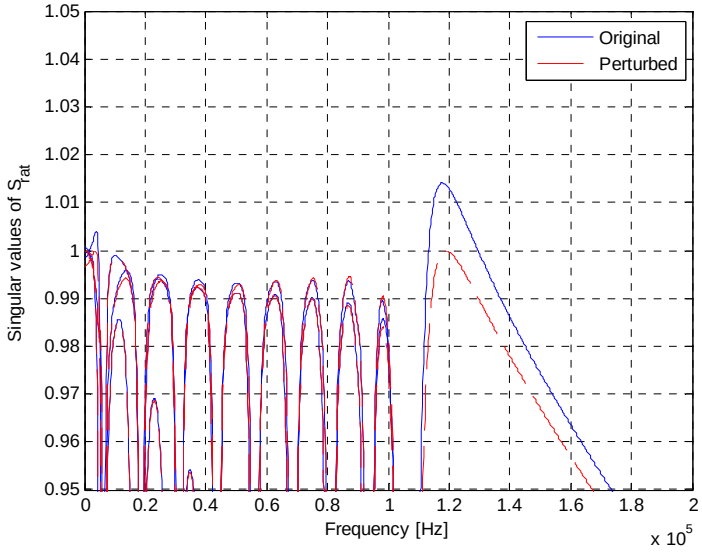


Fig. 4.5.1 Singular values of **S**

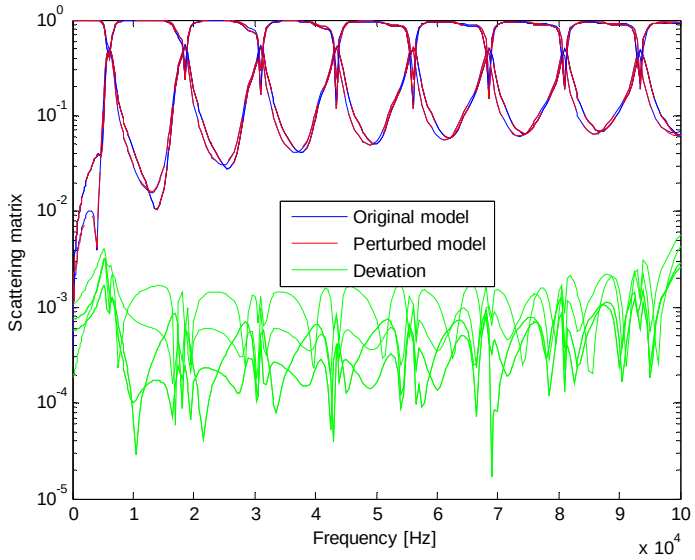


Fig. 4.5.2 The effect of perturbation on **S**

5. REFERENCE FOR COMPUTATIONAL APPROACH

5.1 Pole-residue modeling by Vector Fitting

5.1.1 General

Y-parameters

Given a symmetric admittance matrix **Y** as function of frequency (s,$\mathbf{Y}(s)$), the objective is to calculate a pole-residue model (5.1) which approximates ("fits") the original data as close as possible. (**D** and **E** are possibly zero)

$$\mathbf{Y}(s) \cong \mathbf{Y}_{rat}(s) = \sum_{m=1}^{N} \frac{\mathbf{R}_m}{s - a_m} + \mathbf{D} + s\mathbf{E} \tag{5.1}$$

In the case of a physical system, the following properties apply

1. **Y** is a symmetrical matrix. Hence, $\{\mathbf{R}_m\}$, **D**, and **E** are symmetric
2. **D** and **E** are real matrices
3. The poles and residues are either real or come in complex conjugate pairs (causality requirement)
4. The poles are in the left half plane
5. The model is passive, i.e. it cannot generate energy. For the symmetrical model, this implies that

$$eig(\mathrm{Re}\{\mathbf{Y}_{rat}(s)\}) > 0 \tag{5.2}$$

$$eig(\mathbf{D}) > 0 \tag{5.3}$$

6. The capacitance matrix **E** is positive, i.e.

$$eig(\mathbf{E}) > 0 \tag{5.4}$$

Unfortunately, there is no efficient method available that can fit (5.1) accurately while at the same time satisfy requirements 1-6. In the matrix fitter package, the approach is to first fit (5.1) while satisfying 1-4. Requirements 5-6 are subsequently enforced by perturbation of model parameters, see Section 5.2.

S-parameters

In the case of S-parameters, we are interested in calculating the (symmetrical) model

$$\mathbf{S}(s) \cong \mathbf{S}_{rat}(s) = \sum_{m=1}^{N} \frac{\mathbf{R}_m}{s - a_m} + \mathbf{D} \tag{5.5}$$

Similarly as with the admittance model, we require the poles to be stable and the poles/residues to be real or complex conjugate. The passivity requirement is however, different, now being related to the singular value decomposition,

$$\mathbf{S}(s) = \mathbf{U}(s)\boldsymbol{\Sigma}(s)\mathbf{V}^H(s) \tag{5.6}$$

where $\boldsymbol{\Sigma}$ is a diagonal matrix which contains the singular values, $\sigma_1(s)$... $\sigma_n(s)$. Passivity of the model entails that all singular values are smaller than unity, i.e.

$$\sigma_i(s) < 1, i = 1 \ldots n \tag{5.7}$$

Similarly as with Y-parameters, the passivity criterion is enforced by perturbing the model's parameters, see Section 5.2.

5.1.2 Standard Vector Fitting

The rational fitting is done via routine **`VFdriver.m`**. The approach is to stack the upper triangle of **Y** (or **S**) into a single vector of elements (**h**(s)) that is subjected to rational fitting by Vector Fitting (VF) [1], as implemented in **`vectfit3.m`**.

VF is an iterative technique which *relocates* an initial pole set to better positions by solving a linear least squares problem. VF can also be viewed as a reformulation of the Sanathanan-Koerner (SK) iteration [10]. In VF, the polynomial basis has been replaced by a partial fractions basis, and the explicit weighting has been replaced by pole relocation. This gives VF the advantage of better conditioning, it allows to enforce stable poles by pole flipping, and it arrives directly at the pole-residue model. An additional advantage is that the initial poles can often be favourably specified, thereby leading to relatively few iterations. (With SK-iterations, there is no such flexibility since the initial weighting is uniform).

The workings of VF is in the following explained for the fitting of a scalar (single-element) function, $h(s)$,

$$h(s) \cong \sum_{m=1}^{N} \frac{r_m}{s - a_m} + d + se \tag{5.8}$$

A set of initial poles $\{a_m\}$is first specified. With the initial poles as known quantities, the linear problem (5.9) is solved as an overdetermined linear least squares problem.

$$\overbrace{(\sum_{m=1}^{N}\frac{\tilde{r}_m}{s-a_m}+1)}^{\sigma(s)}h(s) \cong \sum_{m=1}^{N}\frac{r_m}{s-a_m}+d+se \tag{5.9}$$

By writing each of the two sums of partial fractions in (5.6) as a product of zeros over poles, it can be shown [1] that the poles for $h(s)$ must be equal to the zeros of $\sigma(s)$. These zeros are calculated by solving the eigenvalue problem (5.10) [11, Appendix C] where **A** is a diagonal matrix holding the poles $\{a_m\}$, **b** is a column of ones, and **c** holds the residues $\{\tilde{r}_m\}$.

$$\{a_m\} = eig(\mathbf{A}-\mathbf{b}\cdot\mathbf{c}^T) \tag{5.10}$$

Repeated application of (5.9) and (5.10) relocates the initial pole set to better positions. Convergence implies in $\{\tilde{r}_m = 0\}$ in (5.9). In practice, one will usually terminate the iterations before the convergence is complete. The final residues are calculated by solving (5.8) with known poles. During the iterations, unstable poles may occasionally occur. Any unstable pole is flipped into the left half plane by inverting the sign of the real part, thereby guaranteeing a rational model (5.8) with *stable poles* only. (The pole flipping is requested by setting parameter `opts.stable=1`, which is the default setting)

The pole identification and subsequent residue identification both lead to solving an overdetermined problem of the form

$$\mathbf{A}\mathbf{x} \cong \mathbf{b} \tag{5.11}$$

A transformation of variables is used to ensure that complex poles and residues come in conjugate pairs. In the actual implementation, the conditioning of **A** is improved by scaling its columns to unit length. The preferred solver is (sparse) QR decomposition with rank revealing column pivoting. (**`vectfit3.m`** solves the equation using the "\" operator in Matlab).

The initial poles to be specified for the first VF iteration consists of weakly attenuated conjugate pairs that are distributed over the frequency band of interest,

$$a_n = -\alpha + j\beta \,,\, a_{n+1} = -\alpha - j\beta \tag{5.12a}$$

$$\alpha = \nu \cdot \beta \tag{5.12b}$$

The factor ν in (5.9b) is usually taken as 0.01 or smaller. This choice of initial poles ensures a well-conditioned system matrix **A** in (5.8).

For the fitting of the matrix problem (5.1), the upper triangle of the matrix elements are stacked into a single vector and subjected to fitting by VF. This results in a symmetrical,

rational model with a common pole set, i.e. a pole-residue model. The details are shown in the Closure of [1].

5.1.3 Relaxed Vector Fitting

Equation (5.9) has been normalized by setting one coefficient to unity. It was found that this normalization can seriously impair the pole relocation process when fitting noisy responses. This problem was overcome by the *relaxed* formulation [2] where the fixed coefficient (unity) is replaced with an unknown coefficient,

$$\overbrace{(\sum_{m=1}^{N}\frac{\tilde{r}_m}{s-a_m}+\tilde{d})}^{\sigma(s)}h(s)\cong\sum_{m=1}^{N}\frac{r_m}{s-a_m}+d+se \tag{5.13}$$

As normalization, an additional row is introduced in the least squares (LS) problem (N_s is the number of samples).

$$\mathrm{Re}\{\sum_{k=1}^{Ns}(\sum_{m=1}^{N}\frac{\tilde{r}_m}{s_k-a_m}+\tilde{d})\}=N_s \tag{5.14}$$

This equation is given a LS weighting in relation to the size of *h* by

$$weight=\left\|w(s)\cdot h(s)\right\|_2/N_s \tag{5.15}$$

The new poles are now calculated as

$$\{a_m\}=eig(\mathbf{A}-\mathbf{b}\cdot\tilde{d}^{-1}\cdot\mathbf{c}^T) \tag{5.16}$$

It is remarked that the new constraint simply imposes that that integral of $\sigma(s)$ is non-zero, without fixing any coefficients. This gives improved convergence compared to usage of (5.9), where $\sigma(s)$ is enforced to approach unity at infinite frequency. In particular, the original VF formulation is biased to relocating poles from high frequencies towards low frequencies.

5.1.4 Fast implementation

In the pole identification and residue identification steps, the system equation **Ax**=**b** is solved via QR-decomposition,

$$\mathbf{A}=\mathbf{QR} \tag{5.17a}$$

$$\mathbf{x}\cong\mathbf{R}\setminus(\mathbf{Q}^{\mathrm{T}}\mathbf{b}) \tag{5.17b}$$

In the case of multi-port systems, the solution of the pole identification step (5.9) or (5.13) can be time consuming and memory demanding. This problem is overcome using the fast implementation in [3] which recognizes that we only need to calculate the free variables associated with σ. This allows to build a smaller, compact system matrix by exploring the

block-structure of the system matrix. Each block is solved for independently via QR-decomposition, leading to a new system matrix that has as many columns as there are free variables in σ.

To see this, consider the fitting of two elements. The system matrix for the pole-identification step has structure (5.18) and we are only interested in calculating $\mathbf{x}_3$.

$$\begin{bmatrix} \mathbf{A} & 0 & \mathbf{B}_1 \\ 0 & \mathbf{A} & \mathbf{B}_2 \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \end{bmatrix} \cong \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix} \tag{5.18}$$

We first consider the first equation

$$\begin{bmatrix} \mathbf{A} & \mathbf{B}_1 \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_3 \end{bmatrix} \cong \mathbf{b}_1 \tag{5.19}$$

Applying QR-decomposition gives

$$\begin{bmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} \\ 0 & \mathbf{R}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_3 \end{bmatrix} \cong \mathbf{Q}^T \begin{bmatrix} \mathbf{b}_1^1 \\ \mathbf{b}_1^2 \end{bmatrix} = \begin{bmatrix} \mathbf{y}_1^1 \\ \mathbf{y}_1^2 \end{bmatrix} \tag{5.20}$$

where superscripts 1 and 2 denote upper and lower partition of the vector, respectively.

From this we get

$$\mathbf{R}_{22}\mathbf{x}_3 \cong \mathbf{y}_1^2 \tag{5.21}$$

Equation (5.21) is built for all equations (block-rows). When fitting a vector of *n* elements we thus get (5.22). In order to improve the numerical conditioning, the columns of the new system matrix are scaled to unit length before solving by (5.17).

$$\begin{bmatrix} \mathbf{R}_{22,1} \\ \mathbf{R}_{22,2} \\ \vdots \\ \mathbf{R}_{22,n} \end{bmatrix} \mathbf{x}_3 \cong \begin{bmatrix} \mathbf{y}_1^2 \\ \mathbf{y}_2^2 \\ \vdots \\ \mathbf{y}_n^2 \end{bmatrix} \tag{5.22}$$

5.1.5 Expansion into State Space model

The pole-residue model (5.1) can be directly converted into the form

$$\mathbf{Y}_{rat} = \mathbf{C}(s\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D} + s\mathbf{E} \tag{5.23}$$

The expansion process is straightforward as shown in [9]. **A** is a diagonal matrix that holds the poles $\{a_m\}$, repeated as many times as **Y** has columns. **C** holds the elements of the residue matrices $\{\mathbf{R}_m\}$. **B** is a selector matrix containing ones and zeros that associate each input to a separate block (column set) in **A** and **C**.

The building of **A**,**B**,**C** is done as shown in Fig. 5.1 for a third-order, two-port case. It is seen that **C** is established by copying the columns of the **R**-matrices into the appropriate locations in **C**. **B** is a selector matrix that associated the inputs to blocks (columns) of **A** and **C**. It is noted that has **A** is diagonal with the poles repeated as many times as there ports.

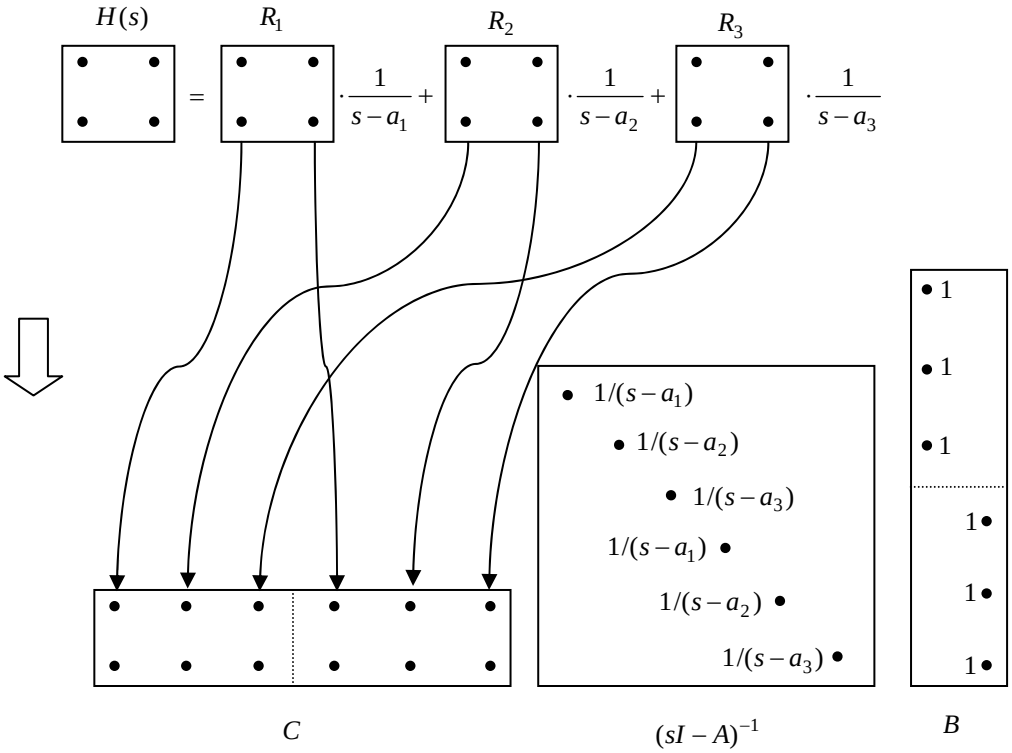


Fig. 5.1. Converting (5.1) into a state-space model

The state-space model can be converted into a real-only model (via a similarity transformation) as follows [XXX].

$$\hat{\mathbf{A}} = \begin{bmatrix} \mathrm{Re}\{a\} & \mathrm{Im}\{a\} \\ -\mathrm{Im}\{a\} & \mathrm{Re}\{a\} \end{bmatrix}, \quad \hat{\mathbf{c}} = \begin{bmatrix} \mathrm{Re}\{\mathbf{c}\} & \mathrm{Im}\{\mathbf{c}\} \end{bmatrix} \tag{5.24}$$

$$\hat{\mathbf{b}} = \begin{bmatrix} 2\,\mathrm{Re}\{\mathbf{b}^T\} \\ -2\,\mathrm{Im}\{\mathbf{b}^T\} \end{bmatrix} = \begin{bmatrix} 2\mathbf{b}^T \\ \mathbf{0}^T \end{bmatrix} \tag{5.25}$$

5.2 Passivity enforcement by Residue Perturbation (Y-parameters)

5.2.1 Passivity assessment via Singularity Test Matrix

Frequency bands with passivity violations can easily be detected by computing the eigenvalues of Re{**Y**} as function of frequency. The presence of a negative eigenvalue at any frequency point means that the model is non-passive at that frequency. For a symmetrical model, this gives the passivity criterion

$$eig(\mathrm{Re}\{\sum_{m=1}^{N}\frac{\mathbf{R}_m}{s-a_m}+\mathbf{D}+s\mathbf{E}\}=eig(\mathbf{G}_{rat}(s))>0,\ \forall s \tag{5.26}$$

In practice, however, passivity violations are often very local in nature and are easily missed out in a sweep over discrete frequencies. In order to pinpoint Fortunately, an algebraic criterion exists which allows to precisely detect frequency boundaries of passivity violations.

The first step is to expand the pole-residue model (5.1) into a real-only state-space model (5.20)-(5.22). From **A**,**B**,**C**,**D**, the Singularity Test Matrix **S** [6] is formed.

$$\mathbf{P}_Y=\mathbf{A}(\mathbf{B}\mathbf{D}^{-1}\mathbf{C}-\mathbf{A}) \tag{5.27}$$

$\mathbf{P}_Y$ gives, via the subset of its positive-real eigenvalues ω^2 the frequencies ω where $\mathbf{G}_{rat}$ becomes singular and these define crossover frequencies where an eigenvalue of $\mathbf{G}_{rat}$ changes sign [6]. (Note that (5.27) is only applicable for symmetrical models, $\mathbf{Y}=\mathbf{Y}^{\mathrm{T}}$ [16])

In the case that **D** is singular, the following transformation of variables is introduced [12] before applying (5.27).

$$\bar{\mathbf{A}}=\mathbf{A}^{-1},\quad \bar{\mathbf{B}}=-\bar{\mathbf{A}}\mathbf{B},\quad \bar{\mathbf{C}}=\mathbf{C}\bar{\mathbf{A}},\quad \bar{\mathbf{D}}=\mathbf{D}-\mathbf{C}\bar{\mathbf{A}}\mathbf{B} \tag{5.28}$$

With (5.27), the positive-real eigenvalues of $\mathbf{P}_Y$ gives the (squared) *inverse* crossover frequencies ω^{-2}.

Note that the test matrix $\mathbf{P}_Y$ is only half the size of the Hamiltonian matrix that has traditionally been used for passivity assessment. Therefore, usage of $\mathbf{P}_Y$ is about eight times faster for large cases, and it more convenient to apply since the need for eigenvalue thresholding is avoided.

In the present work we detect bands of violations via $\mathbf{P}_Y$ as follows:

1. Establish a sorted list of frequencies from the subset of positive real eigenvalues ω of $\mathbf{P}_Y$.

$$\boldsymbol{\omega}=\{\omega_1,\omega_2,\ldots\omega_n\} \tag{5.29}$$

2. Evaluate the criterion (5.26) at the midpoint between frequencies.

$$s = \{\frac{\omega_1 + \omega_2}{2}, \ldots, \frac{\omega_{n-1} + \omega_n}{2}\} \tag{5.30}$$

3. If passivity is violated at the sample $(\omega_i + \omega_{i+1})/2$ in (5.26), then the band $[\omega_i\ \omega_{i+1}]$ defines a band of passivity violation.
4. In addition, the criterion (5.23) is evaluated at samples $\omega_1/2$ and $2\omega_n$ to check if passivity is violated between 0 and ω_1, and between ω_n and infinite frequency.

5.2.2 Fast Residue Perturbation (FRP)

As shown in Section 5.1, application of VF to the data leads to a pole-residue model on the form

$$\mathbf{Y}(s) = \sum_{m=1}^{N} \frac{\mathbf{R}_m}{s - a_m} + \mathbf{D} + s\mathbf{E} \tag{5.31}$$

Matrices $\{\mathbf{R}_m\}$, **D**, and **E** are assumed to be symmetrical.

Using the ideas in [13],[5], passivity is enforced by perturbing the elements of the residue matrices, $\{\mathbf{R}_m\}$ and **D**-matrix. In addition, **E** is enforced to be positive definite (has positive eigenvalues). This leads to the constrained optimization problem

$$\mathbf{\Delta Y} = \sum_{m=1}^{N} \frac{\Delta\mathbf{R}_m}{s - a_m} + \Delta\mathbf{D} + s\Delta\mathbf{E} \cong \mathbf{0} \tag{5.32a}$$

$$eig(\mathrm{Re}\{\mathbf{Y} + \sum_{m=1}^{N} \frac{\Delta\mathbf{R}_m}{s - a_m} + \Delta\mathbf{D}\}) > \mathbf{0} \tag{5.32b}$$

$$eig(\mathbf{D} + \Delta\mathbf{D}) > \mathbf{0} \tag{5.32c}$$

$$eig(\mathbf{E} + \Delta\mathbf{E}) > \mathbf{0} \tag{5.32d}$$

The first part (5.32a) minimizes the change to the admittance matrix elements while the second part (5.32b) enforces that the perturbed model meets the passivity criterion (5.1). The third (5.32c) and fourth (5.32d) parts enforce that **D** and **E** become positive definite,

The implementation of (5.32) leads to the form (5.33) where $\Delta\mathbf{x}$ holds the perturbed elements of $\{\mathbf{R}_m\}$, **D**, and **E**. This problem is solved using Quadratic Programming (QP).

$$\min_{\Delta\mathbf{x}} \frac{1}{2}(\Delta\mathbf{x}^T\ \mathbf{A}_{sys}{}^T \mathbf{A}_{sys}\ \Delta\mathbf{x}) \tag{5.33a}$$

$$\mathbf{B}_{sys}\ \Delta\mathbf{x} < \mathbf{c} \tag{5.33b}$$

Matrix $\mathbf{A}_{sys}$ is block diagonal while $\mathbf{B}_{sys}$ is full but with a few rows.

The number of free variables is reduced by individually diagonalizing the residue matrices $\{\mathbf{R}_m\}$and the **D** and **E** matrix, and perturbing only their eigenvalues [5], leading to Fast Residue Perturbation (RP). First order perturbation of a residue matrix gives

$$\frac{\mathbf{R}_m + \Delta\mathbf{R}_m}{s - a_m} \approx \frac{\mathbf{T}_m(\Lambda_{\mathbf{R}_m} + \Delta\Lambda_{\mathbf{R}_m})\mathbf{T}_m^T}{s - a_m} \tag{5.34}$$

Thus,

$$\Delta\mathbf{R}_m = \mathbf{T}_m \, \Delta\mathbf{\Gamma}_{Rm} \, \mathbf{T}_m^{-1} \tag{5.35a}$$

$$\Delta\mathbf{D} = \mathbf{T}_D \, \Delta\mathbf{\Gamma}_D \, \mathbf{T}_D^{-1} \tag{5.35b}$$

$$\Delta\mathbf{E} = \mathbf{T}_E \, \Delta\mathbf{\Gamma}_E \, \mathbf{T}_E^{-1} \tag{5.35c}$$

This leads to a full but much smaller $\mathbf{A}_{sys}$ (and $\mathbf{B}_{sys}$) in (5.33). For instance, with n ports and N poles, the number of free variables in (5.33) is reduced from $M=(n(n+1))N/2$ to $M=nN$ (when utilizing the symmetry). The solving of (5.33) is done using Quadratic Programming by Matlab routine **`quadprog.m`**. The reduction of problem size leads to large savings in computation time since the complexity of the core operations in QP is $O(M^3)$. In the case of complex conjugate residue matrices, the real and imaginary parts are diagonalized separately.

Note: The implementation does not include the modal weighting described in [5].

5.2.3 Reducing the number of constraints

Since the computation time needed for the perturbation is strongly dependent on the number of constraints (rows in $\mathbf{B}_{sys}$), we use as few constraints as possible. Violating frequency bands that share common border frequencies are joined together into a single band. This is shown in Fig. 1 for and example with 3 bands, b_1, b_2, b_5. Within each concatenated band, the eigenvalues of Re{**Y**(s)} are calculated by frequency sweeping while removing any "artificial" eigenvalue switchovers by the switching back procedure described in [14]. Within each band, the global minimum of all violating eigenvalues are included in the constraint equation, as indicated by the two black dots in Fig. 5.1. The passivity enforcement routine will then bring the two minima up to the zero line by an amount c_1 and c_2. Because of the linearization between the eigenvalues and the free variables, the value for c_1 and c_2 is chosen slightly higher than the vertical distance to the zero line by an amount *tol*. This reduces the number of iterations needed for removing all passivity violations.

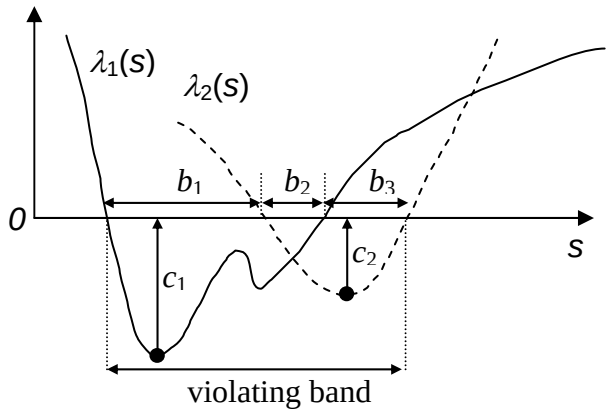


Fig. 5.2 Sample selection for passivity enforcement

5.2.4 Robust iterations

One problem with perturbation techniques is that passivity enforcement at selected frequency samples can result in that new violations arise at other frequencies, and divergence can result. In order to tackle this problem, the “robust iterations” [5] is used, see Fig. 5.3 is used. An inner loop is introduced where the solution is discarded if new violations are detected, and the problem is solved again with additional constraints added at the new frequencies of violating minima. That way, one prevents the new violations from appearing. (Matrices **D** and **E** are removed from the perturbation process as soon as they become positive definite).

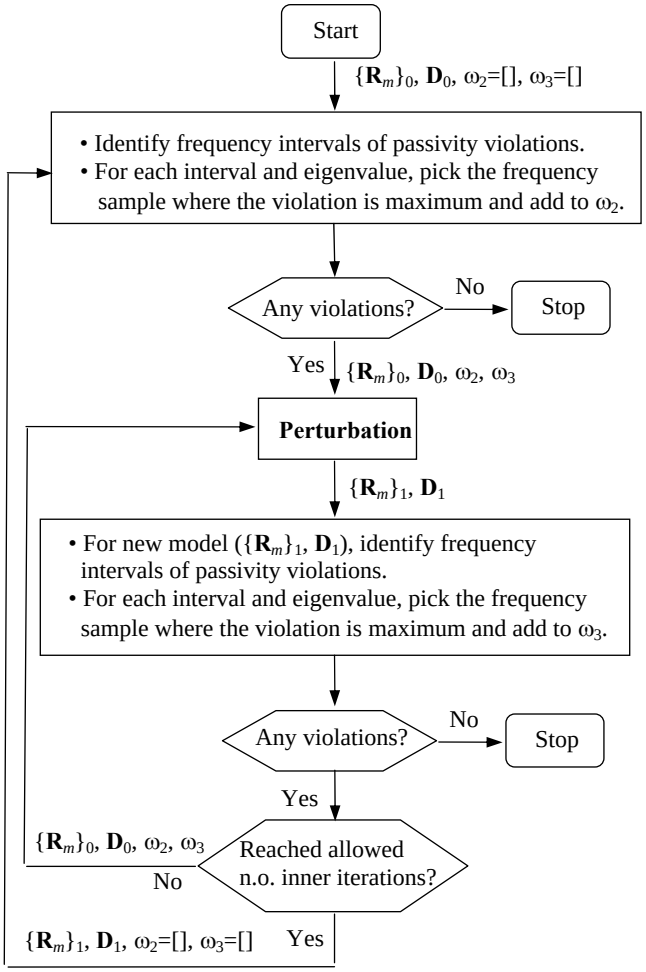


Fig. 5.3 Robust iterations

5.3 Passivity enforcement by Residue Perturbation (S-parameters)

The implemented passivity enforcement scheme for S-parameter models is similar to that for Y-parameters. The following explains the main differences.

5.3.1 Passivity assessment

The passivity assessment is now done using the test matrix [7]

$$\mathbf{P}_S = (\mathbf{A} - \mathbf{B}(\mathbf{D} - \mathbf{I})^{-1}\mathbf{C})(\mathbf{A} - \mathbf{B}(\mathbf{D} + \mathbf{I})^{-1}\mathbf{C}) \tag{5.36}$$

$\mathbf{P}_S$ gives, via the subset of its negative-real eigenvalues $-\omega^2$, the crossover frequencies $j\omega$ where the singular values of **S** are unity. In the case that (**D**+**I**) or (**D**–**I**) is singular, the transformation (5.28) is applied [].

5.3.2 Passivity enforcement

The constraint matrix is obtained by expressing the singular values of $\boldsymbol{S}$ as the eigenvalues of the augmented matrix $\boldsymbol{H}$ (5.37) which leads to the augmented problem (5.38).

$$\boldsymbol{H} = \begin{bmatrix} 0 & \boldsymbol{S}^H \\ \boldsymbol{S} & 0 \end{bmatrix} \tag{5.37}$$

$$\begin{bmatrix} \boldsymbol{V} & \boldsymbol{V} \\ \boldsymbol{U} & -\boldsymbol{U} \end{bmatrix}^{-1} \begin{bmatrix} 0 & \boldsymbol{S}^H \\ \boldsymbol{S} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{V} & \boldsymbol{V} \\ \boldsymbol{U} & -\boldsymbol{U} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Sigma} & 0 \\ 0 & -\boldsymbol{\Sigma} \end{bmatrix} \tag{5.38}$$

Inverting the left factor in (5.38) gives (5.39), and retaining the partition associated with **Σ** gives (5.40), which in compact form is written as (5.41). Finally, the relation between perturbed singular values of **S** and the elements of **H** (5.37) becomes as shown in (5.42). See [8] for the full details.

$$\frac{1}{2}\begin{bmatrix} \boldsymbol{V}^{-1} & \boldsymbol{U}^{-1} \\ \boldsymbol{V}^{-1} & -\boldsymbol{U}^{-1} \end{bmatrix} \begin{bmatrix} 0 & \boldsymbol{S}^H \\ \boldsymbol{S} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{V} & \boldsymbol{V} \\ \boldsymbol{U} & -\boldsymbol{U} \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Sigma} & 0 \\ 0 & -\boldsymbol{\Sigma} \end{bmatrix} \tag{5.39}$$

$$\frac{1}{2}\begin{bmatrix} \boldsymbol{V}^{-1} & \boldsymbol{U}^{-1} \end{bmatrix} \begin{bmatrix} 0 & \boldsymbol{S}^H \\ \boldsymbol{S} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{V} \\ \boldsymbol{U} \end{bmatrix} = \boldsymbol{\Sigma} \tag{5.40}$$

$$\boldsymbol{QHT} = \boldsymbol{\Sigma} \tag{5.41}$$

$$\Delta\sigma_i(\boldsymbol{S}) = \Delta\lambda_i(\boldsymbol{H}) = \frac{\boldsymbol{q}_i^T \, \Delta\boldsymbol{H} \, \boldsymbol{t}_i}{\boldsymbol{q}_i^T \, \boldsymbol{t}_i}, \; i = 1...n \tag{5.42}$$

6. REFERENCES

[1] B. Gustavsen and A. Semlyen, "Rational approximation of frequency domain responses by Vector Fitting", *IEEE Trans. Power Delivery*, vol. 14, no. 3, pp. 1052-1061, July 1999.

[2] B. Gustavsen, "Improving the pole relocating properties of vector fitting", *IEEE Trans. Power Delivery*, vol. 21, no. 3, pp. 1587-1592, July 2006.

[3] 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 and Wireless Components Letters*, vol. 18, no. 6, pp. 383-385, June 2008.

[4] A. Semlyen and B. Gustavsen, "A half-size singularity test matrix for fast and reliable passivity assessment of rational models", *IEEE Trans. Power Delivery*, vol. 24, no. 1, pp. 345-351, January 2009.

[5] B. Gustavsen, "Fast passivity enforcement for pole-residue models by perturbation of residue matrix eigenvalues", *IEEE Trans. Power Delivery*, vol. 23, no. 4, pp. 2278-2285, October 2008.

[6] B. Gustavsen and A. Semlyen, "Fast passivity assessment for S-parameter rational models via a half-size test matrix", *IEEE Trans. Microwave Theory And Techniques*, vol. 56, no. 12, pp. 2701-2708, December 2008.

[7] B. Gustavsen, "Fast passivity enforcement for S-parameter models by perturbation of residue matrix eigenvalues", *IEEE Trans. Advanced Packaging*, accepted.

[8] http://www.eeug.org/

[9] B. Gustavsen, "Computer code for rational approximation of frequency dependent admittance matrices", *IEEE Trans Power Delivery*, vol. 17, no. 4, pp. 1093-1098, October 2002.

[10] C.K. Sanathanan and J. Koerner, "Transfer function synthesis as a ratio of two complex polynomials", *IEEE Trans. Automatic Control*, vol. 8, pp. 56-58, 1963.

[11] B. Gustavsen and A. Semlyen, "Simulation of transmission line transients using vector fitting and modal decomposition", IEEE Trans. Power Delivery, vol. 13, no. 2, pp. 605-614, April 1998.

[12] R.N. Shorten, P. Curran, K. Wulff, and E. Zeheb, "A note on spectral conditions for positive realness of transfer function matrices", *IEEE Trans. Automatic Control*, vol. 53, no. 5, June 2008.

[13] B. Gustavsen, and A. Semlyen, "Enforcing passivity for admittance matrices approximated by rational functions", *IEEE Trans. Power Systems*, vol. 16, no. 1, pp. 97-104, Feb. 2001

[14] L.M. Wedepohl, H.V. Nguyen, and G.D. Irwin, "Frequency-dependent transformation matrices for untransposed transmission lines using Newton-Raphson method", *IEEE*

Trans. Power Delivery, vol. 11, no. 3, pp. 1538-1546, August 1996.

[15] D. Deschrijver and T. Dhaene, "Modified half-size test matrix for robust passivity assessment of S-parameter macromodels", *IEEE Microwave and Wireless Components Letters*, vol. 19, no. 5, pp. 263-265, May 2009.

[16] B. Gustavsen and A. Semlyen, "On passivity tests for unsymmetrical models", *IEEE Trans. Power Delivery*, vol. 24, no. 3, pp. 1739-1741, July 2009.

7. ACKNOWLEDGEMENT

The development of this v1.0 of the Matrix Fitter Toolbox was supported by KMB project "Electric power systems for subsea processing and transportation of oil and gas", which is financed by the Norwegian Research Council Petromaks programme and industry partners.

Aker Solutions

Compagnie Deutsch

FMC Technologies

FRAMO

Nexans Norway

Norwegian Research Council

Oceaneering

Petrobras

Prysmian

Siemens

StatoilHydro

Total

Vetco Gray