Vector Fitting software

C. J. Smartt

November 13, 2018

1 Section

The vector fitting process introduced by Gustavsen and Semlyen [1] is a method for fitting rational function approximations to complex frequency domain transfer functions. This project provides an implementation of the vector fitting process. The software is written in Fortran and includes a Makefile to allow the software to be compiled under unix. Windows users may install Cygwin (which is effectively a unix type environment) and then compile the software under cygwin.

In addition to the software some simple examples are provided along with a script which may be used to run the examples and plot the results.

1.1 License

There are 2 license types used in the VECTOR_FIT project.

GPL - GNU General Public License v.3 See COPYING in root

LGPL - GNU Lesser General Public License v.3 See COPYING.LESSER in root

These licenses are applied as follows:

GPL: Applies to all source coded with the exception of the eispack library in /SRC/eispack.F90

LGPL Applies to the eispack library in /SRC/eispack.F90

2 Vector Fitting

The starting point is a set of complex frequency domain transfer function data, $f(s_i)$, provided at a discrete set of (complex) frequencies $s_i = j2\pi f_i, i = 1..M$ i.e. discrete samples of an underlying function $f(s)$. The vector fitting process generates a rational function approximation to $f(s)$ [1]. The rational function approximation to $f(s)$ is expressed in pole-residue format:

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

where c_n are the poles and a_n are the corresponding residues, d is a constant term and h is the coefficient of the term which is linear in s. The poles and residues are either real or come in complex conjugate pairs. The vector fitting process is an iterative procedure which provides estimates of the poles, residues, d and h such that the function $f'(s)$ approximates the original function $f(s)$.

Stage 1 of the process is pole identification. In order to start the process a set of starting poles, a'_n is specified. In reference [1] it is recommended that for functions with distinct resonant peaks the starting poles should be complex conjugate pairs with imaginary parts distributed linearly over the frequency range for which data $f(s_i)$ is available. i.e.

$$\begin{aligned} a'_n &= -\alpha + j\beta \\ a'_{n+1} &= -\alpha - j\beta \\ \alpha &= \tfrac{\beta}{100} \end{aligned} \tag{2}$$

For smooth functions real poles linearly or logarithmically spread over the frequency range for which data is available is recommended.

An unknown 'weighting function', $\sigma(s)$, is introduced where $\sigma(s)$ is approximated by a rational function $\sigma'(s)$. The product $\sigma(s)f(s)$ is also approximated as a rational function $(\sigma(s)f(s))_{fit}$. $\sigma'(s)$ has the same poles, a'_n, as $(\sigma(s)f(s))_{fit}$, thus we have:

$$\begin{bmatrix} (\sigma(s)f(s))_{fit} \\ \\ \sigma'(s) \end{bmatrix} = \begin{bmatrix} d + sh + \sum_{n=1}^{N} \frac{c_n}{s - a'_n} \\ \\ 1 + \sum_{n=1}^{N} \frac{c'_n}{s - a'_n} \sigma(s) \end{bmatrix} \tag{3}$$

A linear equation in the unknowns c_n, c'_n, d and h is obtained by multiplying the second equation by $f(s_i)$ and setting

$$(\sigma(s)f(s))_{fit} = \sigma'(s)f(s) \tag{4}$$

i.e.

$$d + s_i h + \sum_{n=1}^{N} \frac{c_n}{s_i - a'_n} = \left(1 + \sum_{n=1}^{N} \frac{c'_n}{s_i - a'_n}\right) f(s_i) \tag{5}$$

A set of M equations may be developed by writing equation 5 for all $i, i = 1..M$. Provided $M > 2N + 2$ we have an over determined system of linear equations which

may be solved for c_n, c'_n, d and h in a least squares sense, for example via the Morse Penrose pseudo inverse method [2].

Details of the formulation of the matrix equation may be found in Appendix A of [1].

From equation 4 we can write the rational function approximation, $f'(s)$, to $f(s)$ as

$$f'(s) = \frac{(\sigma(s)f(s))_{fit}}{\sigma'(s)} \tag{6}$$

If $(\sigma(s)f(s))_{fit}$ and $\sigma'(s)$ are written in pole-zero form taking note that the poles of both these functions are the same and are equal to the starting poles, a'_n then

$$(\sigma(s)f(s))_{fit} = h\frac{\prod_{n=1}^{N+1}(s - z_n)}{\prod_{n=1}^{N}(s - a'_n)} \tag{7}$$

$$\sigma'(s) = \frac{\prod_{n=1}^{N}(s - z'_n)}{\prod_{n=1}^{N}(s - a'_n)} \tag{8}$$

Thus equation 6 becomes

$$f'(s) = \frac{(\sigma(s)f(s))_{fit}}{\sigma'(s)} = \frac{h\prod_{n=1}^{N+1}(s - z_n)}{\prod_{n=1}^{N}(s - z'_n)} \tag{9}$$

i.e. the poles of $f'(s)$, the approximation to $f(s)$ are equal to the zeros of $\sigma'(s)$.

Appendix B of [1] states that the zeros of $\sigma'(s)$ may be found as the eigenvalues of the matrix $[H]$ where

$$[H] = [A] - (b)\,(c')^T \tag{10}$$

where $[A]$ is a diagonal matrix of starting poles, (b) is a column vector of ones and $(c')^T$ is a row vector of the residues, c'.

It may be the case that some of the poles found by this procedure may be unstable i.e. they lie in the right hand side of the s-plane. If this is the case then the poles are stabilised by reversing the sign of the real part of the pole.

Once the new poles have been calculated, the residues in $f'(s)$ may be found by solution of equation 12. A set of M linear equations in the unknowns d, h and c_n is obtained by evaluating quation 12 at each frequency, s_i, $i = 1..M$ and using the known poles, a_n, of $f'(s)$:

$$d + s_i h + \sum_{n=1}^{N} \frac{c_n}{s_i - a_n} = f(s_i) \tag{11}$$

Again we have an overdetermined system of equations which may be solved in a least squares sense for d, h and c_n.

The two stage process may be iterated using the newly found poles as the new starting poles in the process. This iteration converges rapidly.

2.1 Choice of model order

An important aspect to generating an effective function fit is the choice of model order. The model order should be chosen to be high enough so as to accurately reproduce the input data. At some point it is usually found that increasing the order of the approxmation no longer gives a significant improvement in the accuracy of the model fit. This point unually determines the optimum model order

3 Vector fitting Software

The software may be compiled in a unix type operating system by going into the SRC/ directory and using the command

make

The makefile may need to be altered for different compilers or to apply specific compilation flags for example. The Makefile in SRC/ is designed to use the f95 gfortran compiler. The executable file, Vfit, is moved to the bin directory.

3.1 Running Vfit

The complex frequency domain data which Vfit fits a model to should be in an ascii file with three columns: frequency, real part of the function at this frequency, imaginary part of function at this frequency. Data for each frequency should be on a sperate line. An example file is shown below:

```
1000.00000       8.50000009E-02  -159.154678
1059.56042       8.50000009E-02  -150.208176
1122.66772       8.50000009E-02  -141.764648
1189.53442       8.50000009E-02  -133.795685
1260.38281       8.50000009E-02  -126.274742
    .                .                .
    .                .                .
    .                .                .
79340968.0       8.50000009E-02   21.4340954
84066512.0       8.50000009E-02   22.7109394
89073600.0       8.50000009E-02   24.0638466
94378816.0       8.50000009E-02   25.4972954
100000000.       8.50000009E-02   27.0161057
```

The vector fitting process is run with the command

Vfit

A number of command line arguments may be added to configure the vector fitting process. These are as follows:

```
const:    include the constant term (d) in the fit function
noconst:  do not include the constant term
sterm:    include the s term (h) in the fit function
nosterm:  do not include the s term (h) in the fit function
logf:     use logarithmically spaced starting poles
nologf:   use linearly spaced starting poles
cmplx:    use complex starting poles
nocmplx:  use real starting poles
```

The default is to include both the constant (d) term and the s term (h) and to use linearly spaced complex poles.

For example to use logarithmically spaced, real starting poles use the command

Vfit logf nocmplx

When Vfit is run the user is propted for the file name for the input frequency domain data. Once this is read successfully the user is propted for the model order to fit and then the number of iterations of the fitting process to perform.

The output of the Vfit program are the following:

1. A file Vfit.filter containing the pole-residue representation of the input function for the specified model order. The format of the function uses a normalised angular frequency where the angular frequency normalisation (wnorm) is based on the maximum frequency in the input data. The function is therefore of the form

$$f'(s) = d + \frac{s}{wnorm}h + \sum_{n=1}^{N} \frac{c_n}{\frac{s}{wnorm} - a_n} \tag{12}$$

At the end of the file is the (normalised) frequency range and number of samples in the input data.

```
Vfit filter output
           1  # order
   628318530.71795857       # wnorm
   8.5000000899999470E-002  # d
   27.017695590661077       # h
                  pole (a)                                 residue (c)
-1.12842870712061E-021 0.00000000000000   1.59154949383951E-003 0.00000000000000
            wnorm_min             wnorm_max                  nw
1.0000000000000001E-005   1.0000000000000000              200
```

This output format is compatible with the network_synthesis process [**?**] which may be used to generate Spice models for passive frequency dependent impedance functions. The combination of vector fitting and network synthesis processes may be run efficiently using the script in the Spice_impedance_synthesis project [4].

As well as the function fit output, a data file (function_data.fout) containing both the input data and the function fit data at the discrete frequencies specified in the input file is produced. This is an ascii file and the format of the data is:

```
frequency      Re{f}     Im{f}     |f|      Re{f_fit)     Im{f_fit}     |f_fit|
```

The data in this file may be plotted by gnuplot for example to show a comparison between the input data and the model fit (see section 4.

In addition to the output files, the output to the screen shows the initial poles, the error at each iteration and the final function fit coefficients.

```
         initial poles (w)             intitial poles (f)
 ( -4712.38916    ,  0.00000000 ) ( -750.000000  ,  0.00000000 )

 Iteration:           1  error=   1.8246864240496829E-006
 Iteration:           2  error=   1.8246864240496765E-006

 d=   8.50000009E-02
```

```
h=   4.29999965E-08
              pole (a)                       residue (c)
( -7.09012655E-13,  0.00000000 ) (  1000000.06  ,  0.00000000 )
```

4 Examples

In the directory TEST_DATA/ there are three example datasets, a gnuplot file to plot the comparison between input data and function fit data along with a script to run Vfit and plot the comparison for a given model order and number of iterations.

The script to run the examples is run as follows:

run_Vfit filename model_order max_iternations

Examples:

1. Data from the impedance of a 'realistic' capacitor model consisting of a R L C series combination, R=85 mohms, L=43nH, C=1uF

run_Vfit CAPACITOR_CJS_Z.CSV 1 2

2. Measured impedance data for port 1 of a band pass filter with short circuit port 2.

run_Vfit BPF_SHORT_Z.CSV 12 4

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] W.H. Press, S.A. Teukolsky, W. T. Vetterling, B. P. Flannery *Numerical Recipes in C* Second Edition, Cambridge University Press, 1992.

[3] *https://github.com/chrissmartt/network_synthesis*

[4] *https://github.com/chrissmartt/Spice_impedance_synthesis*