University of Sheffield | Mechanical Engineering

MEng Mechanical Engineering

Developing an Open-Source Frequency Domain Modal Analysis Algorithm in Python

Adam AWADALLA

May 2025

Timothy J. Rogers

Report submitted to the University of Sheffield in partial fulfillment of the requirements for the degree of Master of Engineering

Word Count: 8001

Contents

1 Introduction **1**
1.1 Modal Analysis 1

2 Algorithms / development: **1**

3 Results **2**

A Derivation of p-LSCF modal parameter estimation method **5**
A.1 The right-matrix rational fractional model 5
A.2 Minimizing the sum of the squared residuals 6
A.3 Linearizing the error 6

1 Introduction

1.1 Modal Analysis

Report Story:
Structural Dynamics in engineering

- How studying structural mechanics has helped us have safer, lighter, greener structures.
- Modal Analysis, is regarded as **the** solution for linear structural dynamics.
- Maths of modal analysis:
  - EOM formulation.
  - Eigendecomposition of state matrices.
  - FRF in modal terms. (refer to plscf solution using that)
- This is in various industries, home appliances, aero, civil/structural, acoustics etc.
- Experimental Modal Analysis king.
- Curve-fitting methods for modal analysis.
- The need for algorithms in practice.

  Onto software usage in engineering contexts
- engineers consistently rely on software tools, this is great as it streamlines the important processes.
- Important aspects of software dev. :
  - logic/control flow
  - complexities and big O notation
  - unit testing and integration testing.
  - version control.
  - foss vs. prop. discuss why it's cheaper in long run, and better for everyone involved.
- Revisit the aims and objectives.

2 Algorithms / development:

maths-y bit

- lsce (brief)
- lscf (brief)
- plscf (important bits and refer to appendix for full derivation, use own notation and wording).
- Why the companion matrix solution works. $\mathbf{eig}(C(p)) = \lambda_i \rightarrow p(\lambda) = 0$

- here one must explain what poles are for a system.
- lsfd for modeshapes.

  software-y bit

- time complexity optimization. discuss that O notation doesn't always equal less time.
- unit testing, explain pytest stuff and fixturing and bla bla bla.
- using numpy (BLAS routines/subroutines)
- user facing code.
- formatting guidelines.

3 Results

- plscf on simulated modal data.
  - clean
  - noisy
  - slightly nonlinear data (mimo where $H_{ij} \approx H_{ji}$ but $H_{ij} \neq H_{ji}$)
  - high dofs.
- plscf on actual lab data.
- adam's plscf vs siemens lms polymax on same dataset.
- interpretation of stabilization diagram.

```python
import numpy as np

def _make_polynomial_basis_fcn(
    polynomial_order: int, frequency_vector: np.ndarray,
       sampling_frequency: float
):
    """Function that creates a Polynomial Basis Function matrix

    Args:
        polynomial_order (int): Order of the polynomial created.
           i.e. if 2 then P(x) = a0 * x^0  +  a1 * x^1 + a2 * x^2
        in the case of pLSCF, Omega(w) = P(e^{jw _delta t})

        frequency_vector (np.ndarray): vector of frequencies
           measured or simulated, can be hz or rads-1.
        MUST be either a row or column vector/1D Array

    Returns:
        Polynomial basis function matrix.
    """
```

```
18      # the polynomial basis function is actually the vandermonde
            matrix for the polynomials, A and B.
19      dt = 1 / sampling_frequency  # Sampling rate
20      s = np.exp(1.j*frequency_vector*dt) # this is the "x" in the
            polynomial
21      return np.vander(x=s,N=polynomial_order+1,increasing=True)
```

Consider the Jacobian matrix, for a system with $n_{outputs} = 3$:

$$J = \begin{pmatrix} X_1 & 0 & 0 & Y_1 \\ 0 & X_2 & 0 & Y_2 \\ 0 & 0 & X_3 & Y_3 \end{pmatrix} \tag{3.1}$$

$$J^H J = \begin{pmatrix} X_1^H X_1 & 0 & 0 & X_1^H Y_1 \\ 0 & X_2^H X_2 & 0 & X_1^H Y_2 \\ 0 & 0 & X_2^H X_2 & X_1^H Y_3 \\ X_1^* Y_1^T & X_2^* Y_2^T & X_3^* Y_3^T & Y_1^H Y_1 + Y_2^H Y_2 + Y_3^H Y_3 \end{pmatrix} \tag{3.2}$$

if:

$$R_o = Re(X_o^H X_o)$$
$$S_o = Re(X_o^H Y_o)$$
$$T_o = Re(Y_o^H Y_o)$$

$$2Re(J^H J)\theta = 2Re \begin{pmatrix} X_1^H X_1 & 0 & 0 & X_1^H Y_1 \\ 0 & X_2^H X_2 & 0 & X_1^H Y_2 \\ 0 & 0 & X_2^H X_2 & X_1^H Y_3 \\ X_1^* Y_1^T & X_2^* Y_2^T & X_3^* Y_3^T & Y_1^H Y_1 + Y_2^H Y_2 + Y_3^H Y_3 \end{pmatrix} \begin{pmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \alpha \end{pmatrix} \tag{3.3}$$

$$= 2 \begin{pmatrix} R_1\beta_1 + S_1\alpha \\ R_2\beta_2 + S_2\alpha \\ R_3\beta_3 + S_3\alpha \\ S_1^T\beta_1 + S_2^T\beta_2 + S_3^T\beta_3 + (T_1 + T_2 + T_3)\alpha \end{pmatrix} \tag{3.4}$$

which corresponds to the solutions in the normal equations in terms of alpha and beta.

$$l^{LS}(\theta) = tr\{\theta^T Re(J^H J)\theta\} \tag{3.5}$$

from vector calculus, if

$$\mathbf{f} = \mathbf{x^T B x} \tag{3.6}$$

then,

$$\frac{\partial \mathbf{f}}{\partial \mathbf{x}} = 2\mathbf{B}\mathbf{x} \tag{3.7}$$

then the normal equation for polymax should be:

$$\frac{\partial l^{LS}}{\partial \theta} = 2Re(J^H J)\theta \tag{3.8}$$

A Derivation of p-LSCF modal parameter estimation method

A.1 The right-matrix rational fractional model

The polyreference least-squares complex frequency domain method employs a right matrix fractional model to fit MIMO Frequency Response Function measurements into a set of rational polynomial transfer functions:

$$[H(\omega)] = [N(\omega)][D(\omega)]^{-1} \tag{A.1}$$

Such that $H(\omega) \in \mathbb{C}^{N_{outputs} \times N_{inputs}}$ is the FRF matrix, where $D(\omega) \in \mathbb{C}^{N_{inputs} \times N_{inputs}}$, is the denominator matrix polynomial, and $N(\omega) \in \mathbb{C}^{N_{outputs} \times N_{inputs}}$, is the numerator matrix polynomial. The rows corresponding to each output o in the FRF matrix can be represented as such:

$$\langle H_o(\omega) \rangle = \langle N_o(\omega) \rangle \, [D(\omega)]^{-1} \tag{A.2}$$

The row vector numerator polynomial for the o^{th} output, and the denominator matrix polynomial are defined in terms of a polynomial basis function, $\Omega(\omega)$, and their respective polynomial coefficients, β and α as such:

$$\langle N_o(\omega) \rangle = \sum_{r=1}^{p} \Omega_r(\omega) \, \langle \beta_{or}(\omega) \rangle \tag{A.3}$$

$$[D(\omega)] = \sum_{r=1}^{p} \Omega_r(\omega) [\alpha_r] \tag{A.4}$$

With the polynomial basis function $\Omega_r(\omega) = e^{j\omega\Delta tr}$. Although not initially obvious as polynomials with conventional form $p(x) = c_0 + c_1 x + c_2 x^2 + \cdots + c_n x^n$, the basis functions are expressed in the s-domain where $s = e^{j\omega\Delta t}$. The polynomial coefficients, $\alpha_r \in \mathbb{R}^{N_{inputs} \times N_{inputs}}$ and $\beta_{or} \in \mathbb{R}^{1 \times N_{inputs}}$, are assembled into matrix form:

$$\beta_o = \begin{pmatrix} \beta_{o0} \\ \beta_{o1} \\ \beta_{o2} \\ \cdots \\ \beta_{op} \end{pmatrix} \in \mathbb{R}^{(p+1) \times N_{inputs}} \tag{A.5}$$

$$\alpha = \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \cdots \\ \alpha_p \end{pmatrix} \in \mathbb{R}^{N_{inputs} * (p+1) \times N_{inputs}} \tag{A.6}$$

$$\theta = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \cdots \\ \beta_{N_o} \\ \alpha \end{pmatrix} \in \mathbb{R}^{(N_{outputs} + N_{inputs})(p+1) \times N_{inputs}} \tag{A.7}$$

A.2 Minimizing the sum of the squared residuals

The collection of both sets of coefficients into one variable θ, makes performing the least squares problem simpler, in a sense, as it becomes the one unknown in this least squares model. As typical in any fitting method, one must minimize the error between the model and the real or measured value. The nonlinear least-squares error for:
Measured FRF: $\hat{H}_o(\omega_k)$
Model FRF: $H_o(\omega_k)$
is weighted such that:

$$\epsilon_o^{NLS}(\theta, \omega_k) = w_o(\omega_k)(H_o(\omega_k) - \hat{H}_o(\omega_k)) \tag{A.8}$$

Where $\epsilon_o^{NLS} \in \mathbb{C}^{1\times N_{inputs}}$, $w_o(\omega_k)$ is a scalar weighing function which captures the variation and deviation between multiple inputs on the same measurement point, and $\forall k = 0, 1, 2, \ldots, N_{frequency}$. Said weighing function is typically denoted by

$$w_o(\omega_k) = \frac{1}{\sqrt{\mathbf{var}[H_o(\omega_k)]}} \tag{A.9}$$

(See [reference for weighted linear regressions] for more information on weighted least squares.) One can then define the nonlinear cost function as the sum of the error "squared", (hermitian inner product), over the data points, in this case, spectral lines and outputs;

$$l^{NLS}(\theta) = \sum_{o=1}^{N_{out}} \sum_{k=1}^{N_f} \mathbf{tr}\{(\epsilon_o^{NLS}(\theta, \omega_k))^H \epsilon_o^{NLS}(\theta, \omega_k)\} \tag{A.10}$$

In this equation, $\mathbf{tr}\{\bullet\}$ denotes the trace of a matrix, also known as the sum of diagonal elements, and $\bullet^H$ denotes the Hermitian (conjugate) transpose. The trace operator is used as the trace of a product of 2 matrices $\mathbf{A} \in \mathbb{C}^{m\times n}$ and $\mathbf{B} \in \mathbb{C}^{m\times n}$ will equal the sum of each individual element in $\mathbf{A}$ with the individual elements of $\mathbf{B}$. This provides a sum of all square residuals/errors in the cost function.

$$\mathbf{tr}\{\mathbf{A}^H\mathbf{B}\} = \mathbf{tr}\{\mathbf{A}\mathbf{B}^H\} = \mathbf{tr}\{\mathbf{B}^H\mathbf{A}\} = \mathbf{tr}\{\mathbf{B}\mathbf{A}^H\} = \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}b_{ij} \tag{A.11}$$

A.3 Linearizing the error

One can then obtain the polynomial coefficients through minimizing the cost function in A.10, by setting the derivative $\frac{\partial l^{NLS}}{\partial \theta}$ equal to zero, however a nonlinear cost function will yield nonlinear derivative equations, (typically called normal equations in linear regression). A subsequent linearization of the cost function can approximate (suboptimally) the least squares problem, this is achieved through right multiplying the cost function with denominator polynomial $\mathbf{D}$. This gives a linear error:

$$\begin{aligned}\epsilon_o^{LS}(\omega_k, \theta) &= w_o(\omega_k)(N_o(\omega_k, \beta_o) - \hat{H}_o(\omega_k)D(\omega_k, \alpha)) \\ &= w_o(\omega_k)\sum_{r=0}^{p}(\Omega_r(\omega_k)\beta_{or} - \Omega_r(\omega_k)\hat{H}_o(\omega_k)\alpha_r)\end{aligned} \tag{A.12}$$

Stacking the error in terms for all spectral lines in one matrix $E_o^{LS}(\theta) \in \mathbb{C}^{N_f\times N_i n}$:

$$E_o^{LS}(\theta) = \begin{pmatrix} \epsilon_o^{LS}(\omega_1, \theta) \\ \epsilon_o^{LS}(\omega_2, \theta) \\ \epsilon_o^{LS}(\omega_3, \theta) \\ \cdots \\ \epsilon_o^{LS}(\omega_{N_f}, \theta) \end{pmatrix} = \begin{pmatrix} X_o & Y_o \end{pmatrix} \begin{pmatrix} \beta_o \\ \alpha \end{pmatrix} \tag{A.13}$$

Here, new variables $\mathbf{X}$ and $\mathbf{Y}$ are introduced:

$$X_o = \begin{pmatrix} w_o(\omega_1)\Big(\Omega_0(\omega_1) + \Omega_1(\omega_1)\dots\Omega_p(\omega_1)\Big) \\ w_o(\omega_2)\Big(\Omega_0(\omega_2) + \Omega_1(\omega_2)\dots\Omega_p(\omega_2)\Big) \\ \vdots \\ w_o(\omega_{N_f})\Big(\Omega_0(\omega_{N_f}) + \Omega_1(\omega_{N_f})\dots\Omega_p(\omega_{N_f})\Big) \end{pmatrix} \in \mathbb{C}^{N_f \times (p+1)} \tag{A.14}$$

$$Y_o = \begin{pmatrix} -w_o(\omega_1)\Big(\Omega_0(\omega_1) + \Omega_1(\omega_1)\dots\Omega_p(\omega_1)\Big) \otimes \hat{H}_o(\omega_1) \\ -w_o(\omega_2)\Big(\Omega_0(\omega_2) + \Omega_1(\omega_2)\dots\Omega_p(\omega_2)\Big) \otimes \hat{H}_o(\omega_2) \\ \vdots \\ -w_o(\omega_{N_f})\Big(\Omega_0(\omega_{N_f}) + \Omega_1(\omega_{N_f})\dots\Omega_p(\omega_{N_f})\Big) \otimes \hat{H}_o(\omega_{N_f}) \end{pmatrix} \in \mathbb{C}^{N_f \times N_{in}(p+1)} \tag{A.15}$$

Where $\otimes$ is the Kronecker product. In these equations, $\mathbf{X}$ is used to capture the frequency content of the least squares problem, and $\mathbf{Y}$ is used to capture both the frequency content and the measured response data. Using these matrices, one can reconstruct the nonlinear cost function into one that is linear:

$$\begin{aligned} l^{LS}(\theta) &= \sum_{o=1}^{N_{out}} \sum_{k=1}^{N_f} \mathbf{tr}\{(\epsilon_o^{LS}(\omega_k,\theta))^H \epsilon_o^{LS}(\omega_k,\theta)\} \\ &= \sum_{o=1}^{N_{out}} \mathbf{tr}\Big\{(E_o^{LS}(\theta))^H E_o^{LS}(\theta)\Big\} \\ &= \sum_{o=1}^{N_{out}} \mathbf{tr}\left\{\begin{pmatrix}\beta_o^T & \alpha^T\end{pmatrix}\begin{pmatrix}X_o^H \\ Y_o^H\end{pmatrix}\begin{pmatrix}X_o & Y_o\end{pmatrix}\begin{pmatrix}\beta_o \\ \alpha\end{pmatrix}\right\} \end{aligned} \tag{A.16}$$

If one defines a *Jacobian* matrix $\mathbf{J} \in \mathbb{C}^{N_f N_{out} \times (N_{in}+N_o ut)(p+1)}$ for the problem as such:

$$\mathbf{J} = \begin{pmatrix} X_1 & 0 & \dots & 0 & Y_1 \\ 0 & X_2 & \dots & 0 & Y_2 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & X_{N_{out}} & Y_{N_{out}} \end{pmatrix} \tag{A.17}$$

The cost function can be represented as:

$$l^{LS}(\theta) = \mathbf{tr}\{\theta^T \mathbf{J}^H \mathbf{J}\theta\} \tag{A.18}$$

To obtain real values of θ, one must place a constraint on the cost function such that:

$$l^{LS}(\theta) = \mathbf{tr}\{\theta^T Re(\mathbf{J}^H \mathbf{J})\theta\} \tag{A.19}$$

Where the Gramian matrix of $\mathbf{J}$ can be represented in terms a set of variables, $\mathbf{R}$,$\mathbf{S}$ and $\mathbf{T}$:

$$Re(\mathbf{J}^H \mathbf{J}) = \begin{pmatrix} R_1 & 0 & \dots & 0 & S_1 \\ 0 & 0 & \dots & 0 & S_2 \\ \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & R_{N_{out}} & S_{N_{out}} \\ S_1^T & S_2^T & \dots & S_{N_{out}}^T & \sum_{o=1}^{N_{out}} T_o \end{pmatrix} \tag{A.20}$$

in which:

$$R_o = Re(X_o^H X_o) \tag{A.21}$$

$$S_o = Re(X_o^H Y_o) \tag{A.22}$$

$$T_o = Re(Y_o^H Y_o) \tag{A.23}$$

The cost function can then be minimized in terms of α and β to find the best least-squares fit:

$$\frac{\partial l^{LS}(\theta)}{\partial \beta_o} = 2(R_o \beta_o + S_o \alpha) = 0 \qquad \forall O = 1, 2, \ldots, N_{out} \tag{A.24}$$

$$\frac{\partial l^{LS}(\theta)}{\partial \alpha} = 2 \sum_{o=1}^{N_{out}} (S_o^T \beta_o + T_o \alpha) \tag{A.25}$$

Giving normal equations of this least squares problem in terms of the wanted polynomial coefficients, one can also assemble those normal equations into 1 equation:

$$\frac{\partial l^{LS}(\theta)}{\partial \theta} = 2Re(\mathbf{J}^H \mathbf{J})\theta = 0 \tag{A.26}$$

The denominator coefficients α are used to obtain the poles and the modal participation factors, which are sufficient information for the constructing a stabilization diagram. Hence, one can further reduce the normal equations by setting:

$$\beta_o = R_o^{-1} S_o \alpha \tag{A.27}$$

This yields the reduced normal equation:

$$\left\{ 2 \sum_{o=1}^{N_{out}} (T_o - S_o^T R_o^{-1} S_o) \right\} \alpha = 0 \qquad \mathbf{M}\alpha = 0 \tag{A.28}$$

For a non-trivial solution to the normal equation, a constraint is set on α, where:

$$\alpha_p = \mathbf{I}_{N_{in}} \tag{A.29}$$

The rest of the denominator coefficients are then found using:

$$\mathbf{M}(1 : N_{in} * p, 1 : N_{in} * p) \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \cdots \\ \alpha_{p-1} \end{pmatrix} = \mathbf{M}(1 : N_{in} * p, N_{in} * p + 1 : N_{in} * (p + 1)) \tag{A.30}$$

The least-squares estimate for α is then:

$$\hat{\alpha}_{LS} = \begin{Bmatrix} \alpha \\ \mathbf{I}_{N_{in}} \end{Bmatrix}$$

This makes the denominator polynomial $\mathbf{D}$ a *monic* polynomial. When fitting theoretical models to experimental data, the difficulty does not typically lie in the mathematical framework enabling the modelling process. Instead, the challenge is in constructing a model that provides physically meaningful insights. Given that the goal of many modal parameter estimation methods is to find a set of poles which As apparent, it is straightforward to find the polynomial coefficients using measured data.