Institut Supérieur
de l'Aéronautique et de l'Espace

Research Project

ModalID
Modal Identification and Diagnosis User Guide

Author:
Elisa Bosco
Ankit Chiplunkar

Supervisor:
Dr. Joseph Morlier

June 27, 2012

Contents

1 Getting started

This document describes how to start using the **ModalID** toolbox for Matlab.It will provide an overview of all the functions used in **ModalID** and a tutorial, presenting an example to illustrate the use of this graphical interface.
Moreover this guide contains a brief introduction on the two modal analysis methods that are implemented in the toolbox. Anyway this document should not be considered as a textbook on modal analysis hence, for interested users, more in-depth examination is presented in the works cited in the bibliography, [1, 2].

2 About the ModalID toolbox

The past three decades have seen the development of several modal analysis software packages, starting from SDOF methods and leading to more efficient and general MDOF methods.

The development of this toolbox aims at an easy tool that allows to determine the modal parameters of a simple structure before and after its damage. This is followed by an analysis of the results in order to relate the change in the modal parameter to the level of degradation of the structure itself.

Up till this date two MIMO (multiple input/multiple output) identification methods have been implemented:

- the *Unified Matrix Ploynomial Method*
  This method makes the analysis in frequency domain on MIMO system; this method has been the major part of our contribution in this toolbox

- the *Least-Square Complex Exponential*;
  This is a time domain method. The data analysis achieved by this method is done making use of the codes available in the *EasyMod/EasyAnim* software package [3].

Several Matlab functions have been developed and used for various applications in structural dynamics:

- reading and writing of UFF (uniuversal file format) files,
- mode indicators (sum of FRFs, sum of FRF real part and sum of FRFs imaginary part) and their visualisation,

- MAC (modal assurance criterion) and modal collinearity for a comparison of two sets of analysis.

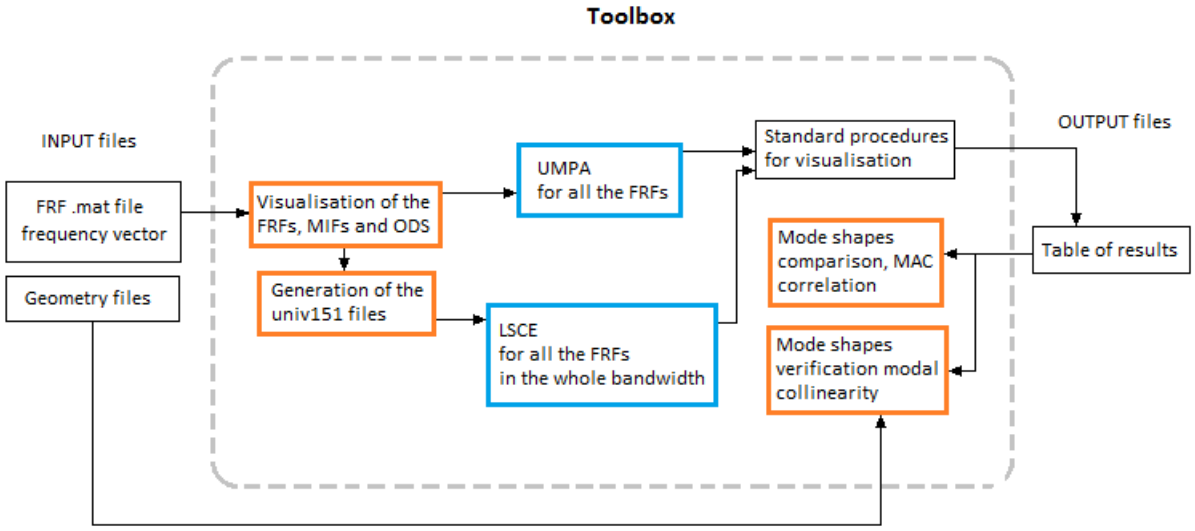

Figure 1: Schematic operating diagram of the toolbox ModalID

3 Installing ModalID

ModalID can be found as a RAR archive.

The archive is written to work on Matlab, therefore the archive should be extracted to a directory on the hard disk, e. g. on Windows OS:

```
C:\Programs\MATLAB\R2010a\toolbox\
```

After extracting the RAR archive the directory will contain different M-files.
Begin the toolbox by launching the *Main* M-file.
The **ModalID** directory must be addes to the MatLab path to make the toolbox functions available in MatLab:

- In MatLab, click on

  ```
  File, Set Path ...
  ```

- Click on

  `Add with Subfolders`

  and select the **ModalID** directory.

- Save the path and close the dialog window.

4 Terms of use

ModalID is a free software; you can redistribute it and/or modify it.
Scientific publications presenting results obtained with **ModaID** must include a proper reference:

[1] G. Kouroussis, L. Ben Fekih, C. Conti, O. Verlinden, EasyMod: A MatLab/SciLab toolbox for teaching modal analysis, *Proceedings of the 19th International Congress on Sound and Vibration*, Vilnius (Lithuania), July 9-12, 2012.

5 Theoretical Introduction

This section deals with a quick overview on the two methods of analysis utilized by the toolbox **ModlaID**.

5.1 LSCE - Least Square Complex Esponential

Least Square Complex Exponential is a time domain modal analysis method. It explores the relationship between the IRF of a MDOF system and its complex poles and residues through a complex exponential. By establishing the analytical links between the two, we can construct an AR model. The solution of this model leads to the establishment of a polynomial whose roots are the complex roots of the system. Having estimated the roots (alias the natural frequencies and damping ratios), the residues can be derived from the AR model for mode shapes. The IRF can be derived from the inverse Fourier transform of an FRF or from random decrement process. The LSCE method begins with the transfer function of a MDoF system, follows its inverse Laplace transform to get the IRF.

$$h_{ij}(t) = \sum_{k=1}^{2N} (A_{ij})_r e^{s_k t} \tag{1}$$

The IRF may be sampled at a series of equally spaced time intervals.

$$h_k = \sum_{k=1}^{2N} (A_{ij})_r z_r^k \quad (k = 0, 1, \ldots, 2N) \quad z_r^k = e^{s_r k \Delta}$$

All these samples are real value data, although the residues $(A_{ij})_r$ and the roots s_r are complex quantities. The next step is to estimate the roots and residues from the sampled data. This solution is aided by the conjugacy of the roots. Mathematically, this means that z_r are roots of a polynomial with only real coefficients:

$$\beta_0 + \beta_1 z_r + \beta_2 z_r^2 + \cdots + \beta_{2N-1} z_r^{2N-1} + \beta_{2N} z_r^{2N} = 0 \tag{2}$$

The coefficients can be estimated from the samples of the IRF data. since there are $2N + 1$ equalities in the IRF equation, we can multiply each equality with a corresponding coefficient β and add all equalities together to form the following equation:

$$\sum_{k=0}^{2N} \beta_k h_k = \sum_{r=1}^{2N} (A_{ij})_r \sum_{k=0}^{2N} \beta_k z_r^k \tag{3}$$

We know that the right hand side is going to be zero when z_r is a root of the polynomial equation 2. This will lead us to a simple relationship between the coefficients β and the IRF samples, namely:

$$\sum_{k=0}^{2N} \beta_k h_k = 0 \tag{4}$$

This equation offers a numerical way of estimating the β coefficients. In equation 2 we can assign β_{2N} to be one. Taking a set of $2N$ samples of IRF, one linear equation is formed 4. Taking $2N$ sets of $2N$ samples of IRF, a set of $2N$ linear equations is drawn:

$$\begin{bmatrix} h_0 & h_1 & h_2 & \cdots & h_{2N-1} \\ h_1 & h_2 & h_3 & \cdots & h_{2N} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ h_{2N-1} & h_{2N} & h_{2N+1} & \cdots & h_{4N-2} \end{bmatrix} \begin{Bmatrix} \beta_0 \\ \beta 1 \\ \vdots \\ \beta_{2N-1} \end{Bmatrix} = \begin{Bmatrix} h_{2N} \\ h_{2N+1} \\ \vdots \\ h_{4N-1} \end{Bmatrix} \tag{5}$$

The selection of IRF data samples can vary provided that the h elements in each row are evenly spaced in sampling and sequentially arranged. No two rows have identical h elements. The number of rows in equation 5 can exceed the number of β coefficients for the least-square solutions.
With the known β coefficients, equation2 can be solved to yield the z_r roots. These roots are related to the system complex natural frequencies s_r. Since the complex natural frequencies s_r are determined by the undamped natural frequencies ω_r and damping ratios ζ_r, as shown below:

$$s_r = -\zeta_r \omega_r + j\omega_r \sqrt{1-\zeta_r^2} \tag{6}$$

$$s_r^* = -\zeta_r \omega_r - j\omega_r \sqrt{1-\zeta_r^2} \tag{7}$$

we can derive the natural frequency and damping ratio of the rth mode as:

$$\omega_r = \frac{1}{\Delta}\sqrt{lnz_r lnz_r^*} \tag{8}$$

$$\zeta_r = \frac{-ln(z_r z_r^*)}{2\omega_r \Delta} \tag{9}$$

To determine the mode shapes of the system from the IRF data, we can write:

$$\begin{bmatrix} 1 & 1 & \cdots & 1 & \\ z_1 & z_2 & z_3 & \cdots & z_{2N} \\ \vdots & \vdots & \vdots & \vdots & \\ z_1^{2N-1} & z_2^{2N-1} & \cdots & z_{2N}^{2N-1} & \end{bmatrix} \begin{Bmatrix} (A_{ij})_1 \\ (A_{ij})_2 \\ \vdots \\ (A_{ij})_{2N} \end{Bmatrix} = \begin{Bmatrix} h_0 \\ h_1 \\ \vdots \\ h_{2N-1} \end{Bmatrix} \tag{10}$$

The solution to this set of linear equations will yield the residues. The above analysis describes the main thrust of the LSCE method and its execution.

5.2 UMPA - Unified Matrix Complex Polynomial Approach

The Unied Matrix Polynomial Approach (UMPA) is an historical attempt to place most commonly used modal parameter estimation algorithms within a single educational framework. It is a frequency domain MDOF mthod for extracting the modal parameters of a system. To understand its formulation, the polynomial model used for frequency response functions is considered.

$$H_{pq}(\omega_i) = \frac{X_p(\omega_i)}{F_q(\omega_i)} = \frac{\sum_{k=0}^{n} \beta_k (j\omega)^k}{\sum_{k=0}^{m} \alpha_k (j\omega)^k} \tag{11}$$

Rewriting this model for a general multiple input, multiple output case and stating it in terms of frequency response functions:

$$[\sum_{k=0}^{m} (j\omega)^k [\alpha_k]][H_{pq}(\omega_i)] = [\sum_{k=0}^{n} (j\omega)^k [\beta_k]] \tag{12}$$

This model in the frequency domain is the AutoRegressive with eXogenous inputs (ARX(m,n)) model that corresponds to the AutoRegressive (AR) model in time domain for the case of free decay or impulse response data:

$$\sum_{k=0}^{m}[\alpha_k]h_{pq}(t_{i+k}) = 0 \tag{13}$$

The general matrix polynomial model concept recognizes that both time and frequency domain models generate functionally similar matrix polynomial models. This model which describes both domains is thus termed as Unified Matrix Polynomial Approach (UMPA).

5.2.1 Low Order Frequency Domain Algorithm

Lower order, frequency domain algorithms are basically UMPA based models that generate first or second order matrix coefficient polynomials. Starting with the multiple input, multiple output frequency response model a second order matrix polynomial model is formed.

$$[\sum_{k=0}^{m}(j\omega)^k[\alpha_k]][H_{pq}(\omega_i)] = [\sum_{k=0}^{n}(j\omega)^k[\beta_k]] \tag{14}$$

for order $m = 2$

$$[[\alpha_2](j\omega_i)^2 + [\alpha_1(j\omega_i) + [\alpha 0][H(\omega_i] = [\beta_1(j\omega_i)] + [\beta_0] \tag{15}$$

This basic equation can be repeated for several frequencies and the matrix polynomial coefficients can be obtained using either $[\alpha_2]$ or $[\alpha_0]$ normalization .

$[\alpha_2]$ Normalization

$$\begin{bmatrix}[\alpha_0] & [\alpha_1] & [\beta_0] & [\beta_1]\end{bmatrix}_{N_O x 4N_i} \begin{bmatrix}(j\omega_i)^0[H(\omega_i)] \\ (j\omega_i)^1[H(\omega_i)] \\ -(j\omega_i)^0[I] \\ -(j\omega_i)^1[I]\end{bmatrix}_{4N_O x N_i} = -(j\omega_i)^0[H(\omega_i)]_{N_O x N_i} \tag{16}$$

These coefficients are then used to form a companion matrix and eigenvalue decomposition can be applied to estimate the modal parameters.

5.2.2 High Order Frequency Domain Algorithm

A high order model has $m > 2$ and it is usually used when the spatial domain is under-sampled. In this case the matrix of coefficients$[\alpha_k]$ is going to be $N_i x N_i$ and $[\beta_k]$ is going to be $N_i x N_O$, when $N_i < N_O$. Therefore normalizing respect to $[\alpha_m]$ we get the following linear matrix equation:

$$\begin{bmatrix}[\alpha_0] & [\alpha_1] & \cdots & [\alpha_{m-1}] & [\beta_0] & \cdots & [\beta_{m-2}]\end{bmatrix} \begin{bmatrix} (j\omega_i)^0[H(\omega_i)] \\ (j\omega_i)^1[H(\omega_i)] \\ \vdots \\ (j\omega_i)^{m-1}[H(\omega_i)] \\ -(j\omega_i)^0[I] \\ \vdots \\ -(j\omega_i)^{m-2}[I] \end{bmatrix} = -(j\omega_i)^m[H(\omega_i)] \tag{17}$$

To solve for the coefficient matrices, an overdeterminated set of equations is generated by evaluating equation 17 at a number of frequencies. the scaled identity matrix is $N_O x N_O$.
The system poles are the eigenvalues of the companion matrix $[C]$ formed with the $[\alpha_k]$ coefficient matrices.

$$[C] = \begin{bmatrix} -[\alpha_{m-1}] & -[\alpha_{m-2}] & -[\alpha_{m-3}] & \cdots & \cdots & -[\alpha_2] & -[\alpha_1] & -[\alpha_0] \\ [I] & [0] & [0] & \cdots & \cdots & [0] & [0] & [0] \\ [0] & [I] & [0] & \cdots & \cdots & [0] & [0] & [0] \\ [0] & [0] & [I] & \cdots & \cdots & [0] & [0] & [0] \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ [0] & [0] & [0] & \cdots & \cdots & [0] & [0] & [0] \\ [0] & [0] & [0] & \cdots & \cdots & [0] & [I] & [0] \\ [0] & [0] & [0] & \cdots & \cdots & [0] & [I] & [0] \end{bmatrix} \tag{18}$$

6 Functions - By format

As already done in the **EasyMod** software this toolbox makes use of files called *universal files*. Their format is standard in the field of vibration/dynamic experimentation. In fact they are defined as data files containing measurement, analysis, units or geometry[1] under ASCII format. They have the following structure:

[1]The extensions commonly used are .UFF, .UF or .UNV .

bbbb-1	
bbxxx	⟵ for FRF data, 15 and 82 for geometry, 164 for the units, ...
...	
...	} related data in the appropriate format
...	
bbbb-1	(b: *blank space*)

Table 1: Universal files structure

The main advantage of this format is that all commercial software packages can, in principle, import or export these files. The files supported by **ModalID** are:

- the **58** file giving a selected FRF (or time history or coherence),
- the **151** file which is a head file and brings together all the others

Methods implemented in **ModalID** are presented as functions to be described in the next section. Don't hesitate to use the `help` command on *M*atlab.
These functions work with variables of various type. The most usual ones are:

- real numbers as the frequency step ,the number of experimental nodes, frequencies or modal parameters;
- vectors as those related to the frequency, the time and the local parameters;
- a set of vectors like the matrix containing all the FRFs to analyze, or its equivalent in time domain;
- strings for file names;
- structures like `infoFRF` or `infoMode` which are defined as:

`infoFRF`	`response`	response node
	`dir_response`	response direction
	`excitation`	excitation node
	`dir_excitation`	excitation direction
	`infoMode`	(array of) structure related to the modal analysis

`infoMODE`	`frequencyk`	natural frequency
	`etak`	loss factor
	`Bijk`	Modal constant

7 Functions - By category

This section enumerates all the **ModalID** functions. Additional information can be obtained by using the `help` function in *MatLab*.

7.1 General functions

`[varargin]=modal_analysis_GUI(varargout)`: Main function that creates the graphical interface.

`plot_FRF(ft,FHt)`: Plots the Frequency Response Function and the Mode indicator function in the same graph.

`ods_real(ft,FHt)`: Plots the operational deflective shape (Real part of the FRF).

`ods_imag(ft,FHt)`: Plots the operational deflective shape (Imaginary part of the FRF).

`[FRF] = FHttoFRF(g,Ni,No)`: Converts the FRF matrix into a usable form

`unv55write(infoMODE,filename,ind_complex)`: This function writes the information about the modal parameters in a 55 type UFF file.

`infoMODE = unv55read(filename,No)`: This function reads the 55 type UFF file containing the information about the modal parameters.

`unv58write(H,numin,dir_excitation,numout,dir_response,fmin,finc,filename)`: This function writes the information of a FRF in a 58 type UFF file.

`[H,freq,infoFRF] = unv58read(filename)`: This function reads the 58 type UFF file containing the information of a FRF.

`h = mif(G)`: Computes the modr indicator function.

`gen_files_univ`: Generates the universal files for the geometry taken into account.

`[receptance, mobility, inertance] = gen_FRF(M, C, K, numin, numout, ft)`: Generates the FRF starting from the matrices of mass damping and stiffness.

`[INDICATOR] = indicator_mode(H)`: Generates a structure that contains the indicators of the modes of the system.

`[cursor1, cursor2] = plot_indicators (INDICATOR, ft)`: Creates a interactive plot of the mode indicators.

7.2 LSCE Method

`[lsce_res,infoFRF,infoMODE] = lsce(H,freq,infoFRF,MaxMod,prec1,prec2,num)`: This function implements the Least Square Exponential method for extracting the modal parameters from the system analyzed.

`[FTEMP,XITEMP,TESTXI,FNMOD,XIMOD] = rec(FN,X,i,FMAX,FTEMP,XITEMP,TESTXI,FNMOD,XIMOD`
This function applies the comparison between the frequency and damping values obtained in the current step and the ones of the previous step.

`[H,Ni,nddl,Np,dt] = AnMatIR(mat)`: Analysis of the impulse response matrix.

`[MatIRs] = gen_resp_impul(H,ff,str_array)`: This file computes the impulse response of the system.

`[G] = MatSur2(H,No,Ni,Nt,p)`: This file computes the overdetermined matrix of the system.

`[L,z] = PbValPp(x,Ni,p)`: This function solves the eigenvalue problem in a specific case.

`[lsce_res,Y] = releve(FTEMP,XITEMP,TESTXI,num)`: This function returns as an output the modal parameters into a tabular form.

7.3 UMPA Method

`[varargout]=UMPA(ft,FHt,MaxMod,prec1,prec2, num)`: This function implements the Unified Matrix Polynomial Method for a High Order system.

`[nat_freq,InvCond,err,epsilon] = low_order_UMPA(Ni,No,H,freq,m,num)`: Tis function implements the Unified Matrix Polynomial Method for a Low Order

system.

`stabdiag_UMPA(ftemp,xitemp,testxi,FMAX,MaxMod,H,f)`: This function displays the stabilization chart.

`[UMPA_res] = releve_UMPA(FTEMP,XITEMP,TESTXI,num)`: This function returns as an output the modal parameters into a tabular form.

`[wmod,ximod] = dedoubl_UMPA(w,xi)`:This function suppresses the redundant values of frequency and damping vectors while conserving the links between these two vectors.

`[ftemp,xitemp,testxi,FNMOD,XIMOD] = rec_UMPA(fn,xi,N, FMAX,ftemp,xitemp,testxi,FNMOD,XIMOD,prec1,prec2)`: This function applies the comparison between the frequency and damping values obtained in the current step and the ones of the previous step.

`[e,InvCond,err,epsilon] = Solver_1(Ni,No,W,WI,FRF,m,num)`: This function generates the linear system of the UMPA method for High order.

7.4 Validation method

This function calculates the modal assurance criterion (MAC) of two modal parameters sets and plots, if necessary, the associated chart.

8 Example

A three degree of freedom system with known mass spring and damper, is taken into account in order to make an illustrative example.

$$[M] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, in\ kg \tag{19}$$

$$[C] = \begin{bmatrix} 40 & 0 & 0 \\ 0 & 40 & 0 \\ 0 & 0 & 40 \end{bmatrix}, in\ Ns/m \tag{20}$$

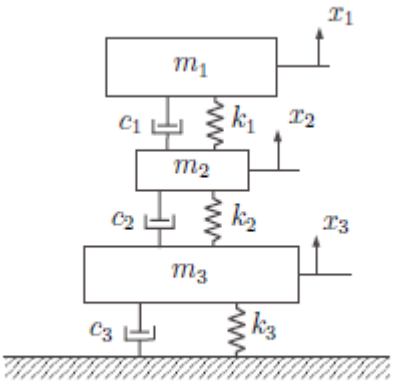

Figure 2: 3-dof discrete model

$$[K] = \begin{bmatrix} 237315 & -161000 & 0 \\ -161000 & 398315 & -161000 \\ 0 & -161000 & 398315 \end{bmatrix}, in\ N/m \tag{21}$$

The frequency range is $[0\ Hz;\ 200\ Hz]$ and the number of samples is imposed to 400.
First of all the FRF matrix is generated with a simple *MatLab* code.

```
clear all
close all
clc

M=eye(3,3);       % Mass matrix in kg

C=40*eye(3,3);  % Damping matrix in Ns/m

K=[237315 -161000 0;
    -161000 398315 -161000;
    0 -161000 398315];              % Stiffness matrix in N/m

% frequency vector our frequency range is [0 - 200] Hz;
ft=[200/400:200/400:200];

[receptance, mobility, inertance] = gen_FRF(M, C, K, 1, 1, ft);
unv58write(inertance,1,3,1,3,0,200/400,'3DL_H11.unv');
[receptance, mobility, inertance] = gen_FRF(M, C, K, 1, 2, ft);
unv58write(inertance,1,3,2,3,0,200/400,'3DL_H21.unv');
[receptance, mobility, inertance] = gen_FRF(M, C, K, 1, 3, ft);
unv58write(inertance,1,3,3,3,0,200/400,'3DL_H31.unv');
```

```
% Now we create the FRF matrix
[H11, ft, infoFRF(1)]=unv58read('3DL_H11.unv');
[H21, ft, infoFRF(2)]=unv58read('3DL_H21.unv');
[H31, ft, infoFRF(3)]=unv58read('3DL_H31.unv');
H=[H11, H21, H31];

% Natural frequencies of the undamped system
freq=(eig(inv(M)*K)).^0.5/2/pi;
```

Here the `eig` function of *MatLab* permits to determine the eigenvalue of the equivalent undamped system. This is used as a preliminary test as the computations with the *LSCE* method and with the *UMPA* method should show the same results.

$$Det(M^{-1}K - \lambda I) = 0 \tag{22}$$

$$\omega_i = \frac{\sqrt{(\lambda_i)}}{2\pi} \tag{23}$$

It is to be noted that for the LSCE analysis it is necessary to write the `.unv` files to create the data structure for the analysis. Then the FRF can be displayed in the various ways allowed by *MatLab*.

```
% Visualisation of the various layouts
figure
subplot(4,3,1)
plot(ft, 20*log10(abs(H11)));
xlabel('Frequency [Hz]'),ylabel('H_{11}');
subplot(4,3,2)
plot(ft, 20*log10(abs(H21)));
xlabel('Frequency [Hz]');ylabel('H_{21}');
subplot(4,3,3)
plot(ft, 20*log10(abs(H31)));
xlabel('Frequency [Hz]'),ylabel('H_{31}');

subplot(4,3,4)
plot(real(H11),imag(H11));
xlabel('H_{11} [real]');ylabel('H_{11} [imag]');
subplot(4,3,5)
plot(real(H21),imag(H21));
xlabel('H_{21} [real]');ylabel('H_{21} [imag]');
subplot(4,3,6)
plot(real(H31),imag(H31));
xlabel('H_{31} [real]');ylabel('H_{31} [imag]');
```

```
22
subplot(4,3,7)
24 plot(ft,real(H11));
xlabel('Frequency [Hz]');ylabel('H_{11} [Real]');
26 subplot(4,3,8)
plot(ft,real(H21));
28 xlabel('Frequency [Hz]');ylabel('H_{21} [Real]');
subplot(4,3,9)
30 plot(ft,real(H31));
xlabel('Frequency [Hz]');ylabel('H_{31} [Real]');
32
subplot(4,3,10)
34 plot(ft,imag(H11));
xlabel('Frequency [Hz]');ylabel('H_{11} [Imag]');
36 subplot(4,3,11)
plot(ft,imag(H21));
38 xlabel('Frequency [Hz]');ylabel('H_{21} [Imag]');
subplot(4,3,12)
40 plot(ft,imag(H31));
xlabel('Frequency [Hz]');ylabel('H_{31} [Imag]');
```

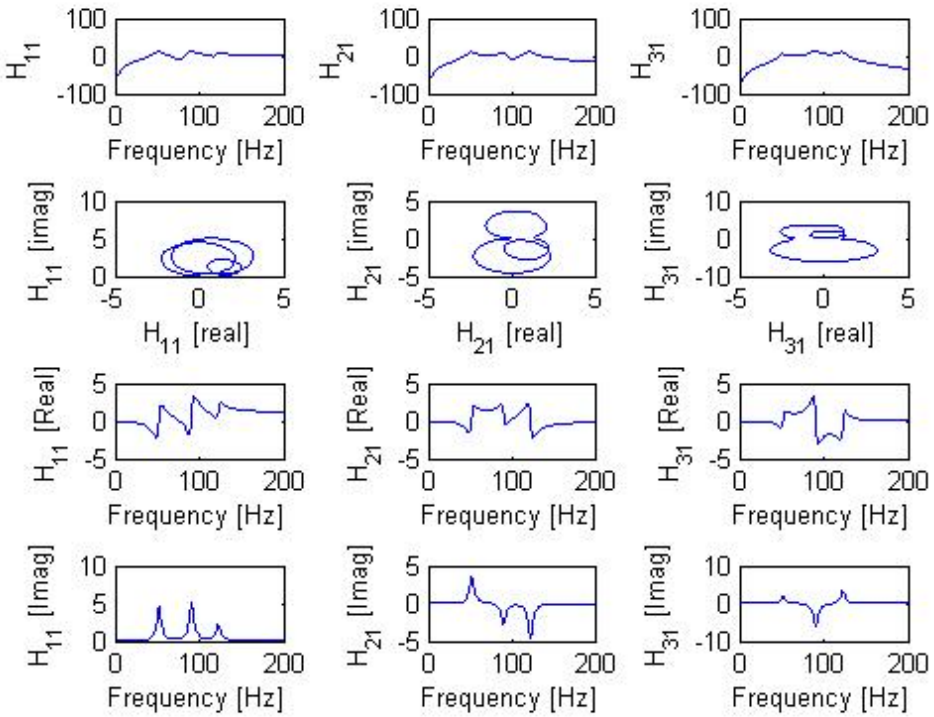

Figure 3: FRFs in various formats

Before using an identification method, it is often interesting to use mode indicators, for checking the local range of frequency to be analyzed.

```
% now we make the check on the local frequency range
[INDICATOR] = indicator_mode(H);
%[cursor1,cursor2] = plot_indicators(INDICATOR,ft);

figure()
subplot(3,1,1)
plot(ft,20*log10(INDICATOR.ISUM))
grid on
xlabel('Frequency [Hz]');
ylabel('Indicator I_{Sum} [dB]');
subplot(3,1,2)
plot(ft,20*log10(INDICATOR.ISRe))
grid on
```

```
xlabel('Frequency [Hz]');
ylabel('Indicator I_{Sum Real} [dB]');
subplot(3,1,3)
plot(ft,20*log10(INDICATOR.ISIm))
grid on
xlabel('Frequency [Hz]');
ylabel('Indicator I_{Sum Imag} [dB]');
```

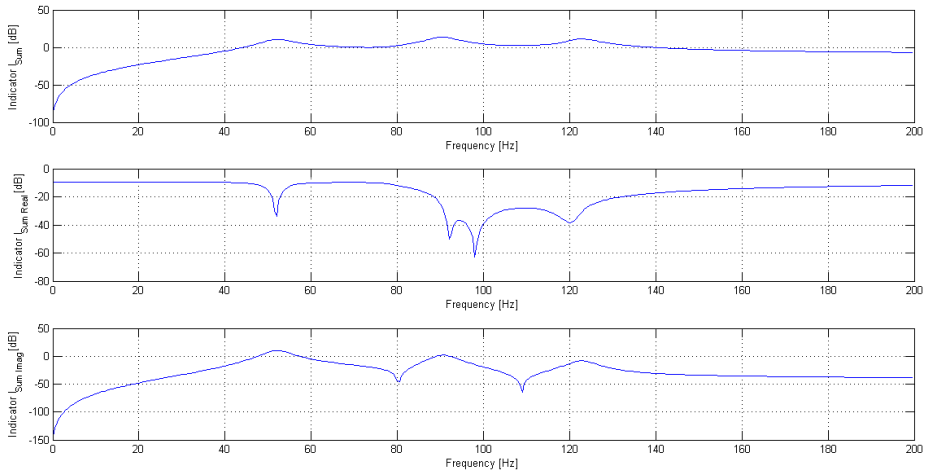

Figure 4: Mode indicators

Now it is the time to test the two implemented methods. The LSCE method is implemented first.

```
% now we begin the indentification step, with the LSCE method
[RES,infoFRF,infoMODE] = lsce_MID(H,ft,infoFRF,40,1/100,1/100,5);

%Mode shapes visualization
figure()
lsce_mode_shape=[real(infoMODE.Bijk(3,:)); real(infoMODE.Bijk(6,:));
    ...
     real(infoMODE.Bijk(9,:))];

% Geometry
Nodes=[1 0 0 1;2 0 0 2;3 0 0 3];

subplot(3,1,1)
```

```
stem(Nodes(:,1),lsce_mode_shape(:,1))
xlabel('Node Position')
ylabel('Mode Shape - First Mode')
subplot(3,1,2)
stem(Nodes(:,1),lsce_mode_shape(:,2))
xlabel('Node Position')
ylabel('Mode Shape - Second Mode')
subplot(3,1,3)
stem(Nodes(:,1),lsce_mode_shape(:,3))
xlabel('Node Position')
ylabel('Mode Shape - Third Mode')
axis([1 3 -4 4]);
```

Following are the graphs with the information obtained with the LSCE method.

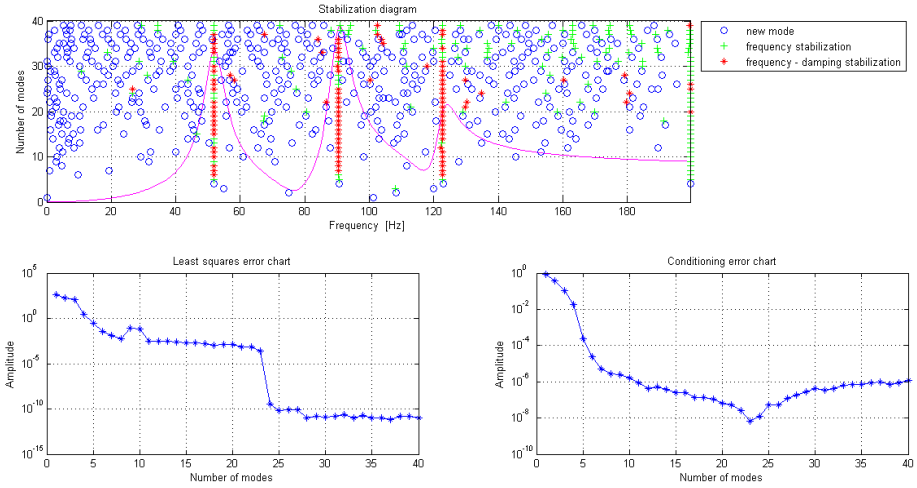

Figure 5: Example of information provided by the LSCE method

Following are the graphs with the mode shapes obtained with the LSCE method.

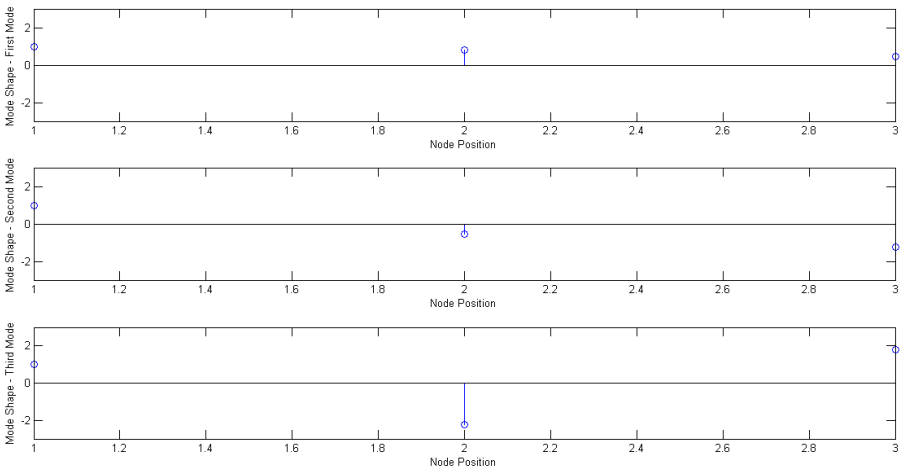

Figure 6: Mode shapes visualization in MatLab

Then it is the time of the computation with the UMPA method.

```
% the second way to do the identification is through the UMPA
% in this case we use a low order model

[nat_freq,InvCond,err,epsilon] = low_order_UMPA(1,3,H,ft,2,2);
\end{verbatim}

In this particular case the code for low order model is utilized.
    Therefore the denominator of the transfer function of the system
    will be of the second order. In this case a study on the
    stabilization of the modal parameters is not possible as \emph{m}
    could vary just between $1$ and $2$.

%\lstinputlisting{/SOME/PATH/low_order_UMPA.m}

\begin{lstlisting}
% This function implements the UMPA method for low order model.
% Therefore m<=2 and in this case also num<=2
%
% It takes as an input:
%                   g: the FRF matrix
%                   w: the frequency vector
%                   m: the numerator order
%                   num:

function [nat_freq,InvCond,err,epsilon] = low_order_UMPA(Ni,No,g,w,m,
```

```
    num)

[rw,cw]=size(w);      if rw>cw,    w=w.';    end
[rg,cg]=size(g);      if rg>cg,    g=g.';    end

w = [w,-w];
g = [g,conj(g)];

FMAX = abs(w(end));

for k = 0:length(w)-1
    W(1:No,(Ni*k+1):Ni*(k+1)) = w(1,k+1)*ones(No,Ni);
    WI(1:Ni,(Ni*k+1):Ni*(k+1)) = w(1,k+1)*eye(Ni);

end

 OM = (1i*W).^m;
    IOM = -(1i*WI).^m;
    for kom = m-1:-1:0
        OM = [OM;(1i*W).^kom];
        IOM = [IOM;-(1i*WI).^kom];
    end

     H = g;
    for kom = 1:m-1
        H = [H;g];
    end

    Dva = OM(No+1:end,:).*H;
    Dvb = IOM((m-num)*Ni+1:end,:);

    D = [conj(Dva);conj(Dvb)];
    LHS = D*D';

    rhs = -OM(1:No,:).*g;
    rhs = conj(rhs);
    RHS = rhs*D';
    RHS = RHS';

    AABB = LHS\RHS;
    AA = AABB(1:m*No,:);
    BB = AABB(m*No+1:end,:);

    I = eye((m-1)*No);
    O = zeros(((m-1)*No),No);
    CC = [-AA'; I O];

    e = eig(CC);
    nat_freq=abs(e);
```

```
70
    % Erreur calculation
72  %- Inverse of conditionnning number
        InvCond = 1/cond(LHS) ;
74  %- Singular normalized  values
        [U,S,V] = svd(LHS,0) ;
76      SingVals = diag(S) ;
        err = 1/(max(SingVals)/min(SingVals)) ;
78      %    - of least squares
        epsilon = norm(RHS-(LHS*AABB));
```

The high order code fails to work in this situation giving a singular matrix to working precision at $m = 3$.

Hereby follows a table with the results for the *Low Order* computation. Its results are compared with the LSCE results and the theoretical ones.

Low Order UMPA				LSCE	Theoretical
m	1	2	2		
num	1	1	2		
Nat. freq	23.00	126.05	122.66	122.69	123.19
	3.80	95.14	90.47	90.47	90.96
	22.99	61.54	51.86	51.87	52.35
error	1.87 E-09	1.92 E-07	6.30 E-05	5.6 E-12	
invCond	5.14 E-05	1.21 E-05	1.15 E-09	1.2 E-06	

It is clearly visible that the code gives better results for $m = 2$ and $num = 2$.

Through this simple example, an overview of the **ModalID** functionalities has been illustrated. Further developments of this toolbox are previewed for the next six months. First of them is to solve the problems linked to the conditioning of the matrix of the system for the UMPA method, second to extend the applicability of this software to a general geometry given as an input through the `.unv` files.

References

[1] Jimin He and Zhi-Fang Fu, *Modal Analysis*, Butterworth- Heinemann, 2001.

[2] R. J. Allemang and A. W. Philips, *The Unified Matrix Polynomial Approach to Understanding Modal Parameter Estimation*, 2001.

[3] G. Kouroussis, L. Ben Fekih, C. Conti, O. Verlinden, EasyMod: A Matlab/SciLab toolbox for teaching modal analysis*, Proceedings of the 19th International Congress on Sound and Vibration*, Vilnius (Lithuania), July 9-12, 2012.