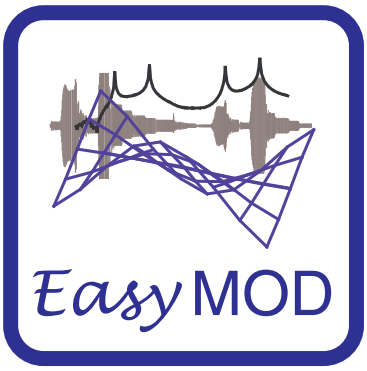
— User guide —



EasyMod — A MatLab/Scilab toolbox for experimental modal analysis

G. Kouroussis
Université de Mons — Faculty of Engineering
Department of Theoretical Mechanics, Dynamics and Vibrations
Boulevard Dolez 31 | B–7000 Mons
e–mail: `Georges.Kouroussis@umons.ac.be`

POLYTECH MONS

UMONS

1 Getting started

This document describes how to start using the EasyMod toolbox for *MatLab* or *SciLab*. It is a user guide to EasyMod version 2.1.0. It is a combination of a reference guide, providing an overview of all functions of EasyMod, and a tutorial, presenting an example to illustrate the use of EasyMod. This document is not meant as a textbook on modal analysis. The theoretical background of experimental modal analysis, as well as the various identification algorithms, are assumed to be known by the user. For interested persons, major of experimental modal analysis methods are presented in other more works:

D. J. Ewins, *Modal Testing: Theory and Practice*. John Wiley & Sons, Great Yarmouth (UK), 1991.

N. M. M. Maia et al., *Theoretical and Experimental Modal Analysis*. John Wiley & Sons, Exeter (UK), 1997.

2 About the EasyMod toolbox

From the last thirty years, many experimental modal analysis software packages have been developed. Initially composed by simple SDOF methods (*peak picking*/*mode picking*, *circle–fit* or *line–fit*), they have been updated to new complex methods, allowing rapid and efficient analyses. In fact, simple methods have been abandoned in favour of MDOF ones. Nonetheless, these simple methods are very interesting for the education and allow a progressive approach in the modal analysis fundamentals teaching.

To fill this gap, the *Department of Theoretical Mechanics, Dynamics and Vibrations* has developed some *MatLab* routines , which have been included in the so-called toolbox EasyMod, working under *MatLab*. The power of matrix manipulation added to a user-friendly platform are elements paving for this choice, in addition to the fact that *MatLab* is a reference program in engineering. The functioning and the possibilities of EasyMod are illustrated in the Figure 1 with an emphasis on the identification and validation methods.

To date, three identification methods have been implemented:

- two SISO (single-input/single-output) methods : the *circle-fit* and *line-fit*;
- one MIMO (multi-input/multi-output) method : the *LSCE* (least-square complex exponential).

Several MatLab/Scilab functions have been developed for various applications in structural dynamics:

- reading and witting of UFF (universal file format) files,
- generating of frequency response function (FRF) from mass $\mathbf{M}$, stiffness $\mathbf{K}$, damping $\mathbf{C}$ matrices defining discrete systems,

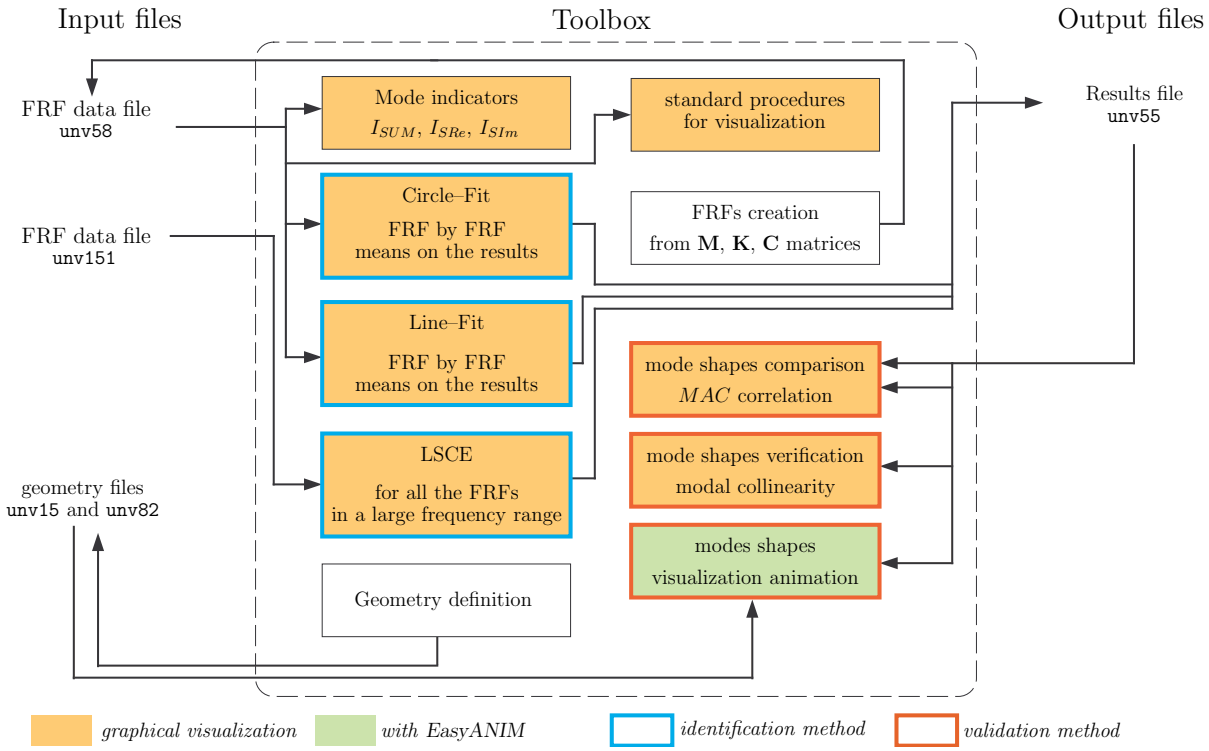


Figure 1: Schematic operating diagram of toolbox *EasyMod*

- mode indicators (sum of FRFs, sum of FRFs real part and sum of FRFs imaginary part) and their visualization,
- MAC (modal assurance criterion) and modal collinearity for a comparison of two sets of analysis.

3 Obtaining and installing EasyMod

EasyMod can be downloaded from the internet at `http://mecara.fpms.ac.be/EasyMod`. It is distributed as a RAR archive.

For *MatLab* uses, the archive should be extracted to a directory on the hard disk, e.g. `C:\Program Files\MATLAB\R2010a\toolbox\` (for Windows OS). After extraction of the RAR archive, this directory contains a number of M-files, classified in several subfolders:

- `general`: this folder contains all general functions (file writing and reading, mode indicator, MAC, FRF generation,...);
- `LeastSquareComplexExponential`: this folder contains user's functions related to the *least-square complex exponential* method (calculation and visualization);
- `CircleFit`: this folder contains user's functions necessary to the *circle–fit* method;
- `LineFit`: this folder contains user's functions necessary to the *line–fit* method;

- `Local`: this folder contains internal functions necessary to the toolbox basic functions.

The EasyMod directory must be added 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 EasyMod directory.
- Save the path and close the dialog window.

For *Scilab* uses, the corresponding archive contains several `.sci` files in the same subfolder classification. Under *scilab* prompt, it is possible to create binary files with the help of the function `genlib` which saves the functions to their corresponding `.bin` file and are mentioned in the path to make the toolbox functions available:

```
genlib("EasyMod",DIRECT+'\EasyMod') ;
genlib("EasyMod_G",DIRECT+'\EasyMod\General') ;
genlib("EasyMod_L",DIRECT+'\EasyMod\Local') ;
genlib("EasyMod_CF",DIRECT+'\EasyMod\CircleFit') ;
genlib("EasyMod_LF",DIRECT+'\EasyMod\LineFit') ;
genlib("EasyMod_LSCE",DIRECT+'\EasyMod\LeastSquareComplexExponential') ;
```

where `DIRECT` is the local folder where the toolbox folders are placed. These commands must be launch at each startup of *SciLab*. To avoid it, they can be placed in the file `loader.sce` used to load modules in Scilab.

4 Terms of use

EasyMod/EasyAnim are free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2, or (at your option) any later version. EasyMod/EasyAnim are distributed in the hope that it will be useful, but without any warranty; without even the implied warranty of merchantability or fitness for a particular purpose. See the GNU General Public License for more details.

Scientific publications presenting results obtained with EasyMod must include a proper reference. In order to refer to EasyMod, please add at least one of the following articles to the list of references:

[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.

[2] G. Kouroussis, L. Ben Fekih, C. Conti, O. Verlinden, EasyMod : du développement d'un toolbox sous MatLab vers l'enseignement des bases de l'analyse modale expérimentale, *3ième Colloque "Analyse vibratoire Expérimentale"*, Blois (France), 20 et 21 novembre 2012.

5 Functions — By format

All the information is saved in files called *universal files*, for whom the format is standard in the field of vibration/dynamic experimentation. They are defined as data files containing measurement, analysis, units or geometry[1], under ASCII format. The following structure is dedicated:

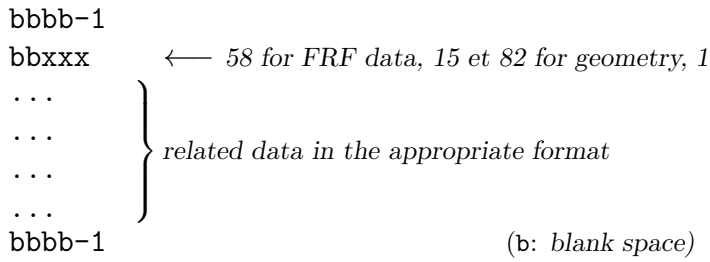


Figure 2: Structure of universal files

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

- the **58** file giving a selected FRF (or time history or coherence),
- the **55** file for the modal analysis and the associated parameters (natural frequency f_k, loss factor η_k and the modal constant/residue B_{ijk}),
- the files **15** and **82** are related to the geometry, for nodes and connexion respectively,
- the **151** file which is a head file and brings together all the aforementioned files.

Methods implemented in EasyMod are presented as functions to be described in the next section. Don't hesitate to use the `help` command of *MatLab/Scilab*.

These functions work with variables of various types. The most usual ones are:

- real numbers as the frequency step `Df`, the number of experimental nodes `No`, their number (`numin` and `numout`), frequencies (`fmin`, `fmax`) or modal parameters (`frequencyk`, `etak` and `Bijk`);
- vectors as those related to the frequency (`f`, `freq`), the circular frequency `omega`, the time `time`, the local parameters (`freq_local`, `H_local` and `H_gen_local`) or those related to the FRF in various formats (`recept`, `mobil` or `inert`);
- a set of vectors like the matrix `H` containing all the FRFs to analyse, or its equivalent `h` in time domain for the impulse response;
- matrices like mass $\mathbf{M}$, damping $\mathbf{C}$ or stiffness $\mathbf{K}$;

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

- strings for filenames (`filename`) and method names (`methodname`) ;
- structures[2] 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

6 Functions — By category

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

6.1 General functions

`[H,freq,infoFRF] = unv151read(filename)`: This function reads the 151 type UFF file in order to extract the information of all FRFs only.

`[coh,freq,infoFRF] = unv151readcoh(filename)`: This function reads the 151 type UFF file in order to extract the coherences only.

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

`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.

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

`unv15and82write(Nodes,Connexions,filename,NodeColor,LineColor)`: This function records the geometry information (nodes and connexions) in UFF file bringing together the file type 15 (for the nodes) and the file type 82 (for the connexions between nodes).

`[Hswitch,freqswitch,infoFRFswitch] = switchfrf(H,freq,infoFRF)`: This function switches the FRF $H_{ij}(\omega)$ to $H_{ji}(\omega)$, according to the Betti-Maxwell theorem.

[2] In the case of MDOF data, it represents an array of structure.

`[H_estimate,freq,infoFRF] = estimate_frf(x,y)`: This function builds the H_1 estimator from time history of input x and output y.

`[I] = ind_mode(H)`: This function generates three mode indicators — sum of FRFs, sum of FRFs real part and sum of FRFs imaginary part.

`[cursor1,cursor2] = plot_ind_mode(I,freq)`: This function displays the three mode indicators — sum of FRFs, sum of FRFs real part and sum of FRFs imaginary part.

`[recep,mobil,inert] = gen_frf(M,D,K,numin,numout,freq)`: This function generates a FRF from mass, damping and stiffness matrices, in various format (compliance, mobility, accelerance).

`infoFRF = add_data(index_FRF,frequencyk,etak,Bijk,infoFRF,index_mode)`: Storage of modal parameters for each FRF and for each DOF.

`infoMODE = save_result_modal(infoFRF)`: This function saves the information related to the modal parameters.

`EM_plot_Bode(freq,H,fmin,fmax)`: This function plots the Bode's diagram of a FRF in a specific frequency range.

`M_plot_Nyquits(freq,H,fmin,fmax)`: This function plots the Nyquist's diagram of a FRF in a specific frequency range.

`[MODE_REAL] = cplxtoreal(MODE_CPLX)`: This function transforms complex mode shapes to real mode shapes.

`[psi1,freq1,psi2,freq2] = appariement(infoMODE1,infoMODE2,coupled)`: This function brings together the correlated modes issued from two sets of data.

`infoMODE = readansys(name,fpts,name_out)`: This function allows to extract natural frequencies and mode shapes calculated by the FEM software *ANSYS* from the result file generated by *ANSYS* (for updating procedures).

6.2 Circle–fit method

`[f_loc,H_loc,H_gen_loc,infoMODE,circ_prop] = circle_fit(H,f,fmin,fmax)`: Identification based on the circle-fit method (SDOF method) giving the natural frequency, the loss factor and the modal constant.

`plot_circ_prop_fit(f_loc,H_loc,H_gen_loc,infoMODE,circ_prop)`: Graphical representation of the results provided by the function `circle_fit`.

6.3 Line–fit method

`[f_loc,H_loc,H_gen_loc,infoMODE,line_prop] = line_fit(H,f,fmin,fmax)`: Identification based on the line-fit method (SDOF method) giving the natural frequency, the loss factor and the modal constant.

`plot_line_fit(f_loc,H_loc,H_gen_loc,infoMODE,line_prop)`: Graphical representation of the results provided by the function `line_fit`.

6.4 LSCE method

`[x,time] = InvFft(H,omega)`: Calculation of impulse response from FRF.

`[lsce_res,infoFRF,infoMODE] = lsce(H,freq,infoFRF)`: Identification based on the least-square complex exponential (LSCE) method (MDOF method) giving the natural frequency, the loss factor and the modal constant in all the frequency range.

6.5 Validation method (in the subfolder `General`)

`[DELTA_FREQ,MAC,coupled] = mac(infoMODE1,infoMODE2,macinf,delfreqsup,graph)`: This function calculates the modal assurance criterion (MAC) of two modal parameters sets and plots, if necessary, the associated chart.

`modegauss(infoMODE)`: This function displays the mode shapes in the Gaussian plane in order to verify the collinearity property.

... and of course the graphical visualization with the help of EasyAnim software (Figure 3).

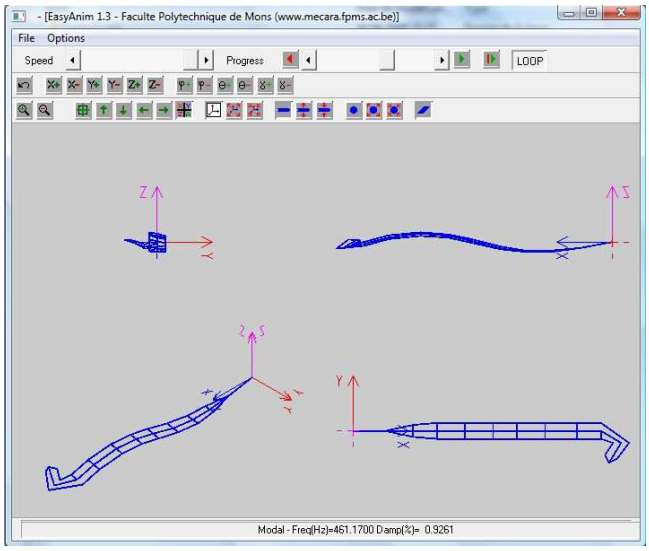


Figure 3: Typical view of EasyAnim

7 Example

Figure 4 presents the illustrative example, consisting of a 3-DOF system, defined by

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

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

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

in the frequency range $[0\,\text{Hz}\,;200\,\text{Hz}]$, the number of samples being imposed to 400.

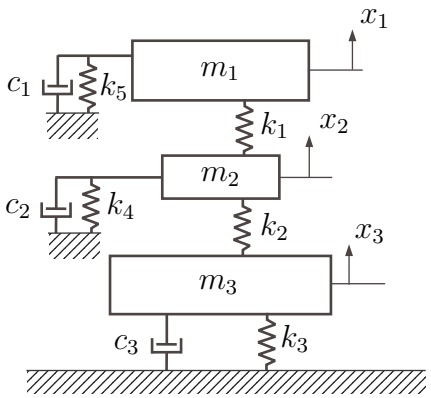


Figure 4: 3-dof discrete model

The generation of FRF related to the proposed system can be easily performed, as illustrated by the *MatLab* code:

```
% The best way to begin
clear all
close all
clc

% Model parameters:
%  mass matrix
M = [1 0 0 ; 0 1 0 ; 0 0 1] ;
%  damping matrix
C = [40 0 0 ; 0 40 0 ; 0 0 40] ;
%  stiffness matrix
K = [237315 -161000 0 ; -161000 398315 -161000 ; 0 -161000 398315] ;

% FRF storage
Df = 200/400;
freq = [Df:Df:200];
```

```
[receptance,mobilite,inertance] = gen_frf(M,C,K,1,1,freq) ;
unv58write(inertance,1,3,1,3,0,Df,'3DL_H11.unv') ;
[receptance,mobilite,inertance]=gen_frf(M,C,K,1,2,freq) ;
unv58write(inertance,1,3,2,3,0,Df,'3DL_H21.unv') ;
[receptance,mobilite,inertance]=gen_frf(M,C,K,1,3,freq) ;
unv58write(inertance,1,3,3,3,0,Df,'3DL_H31.unv') ;
```

A preliminary test can be done, by using the `eig` function in order to determine the eigenvalue:

$$f_1 = 52.4\,\text{Hz} \qquad f_2 = 91.0\,\text{Hz} \qquad f_3 = 123.2\,\text{Hz}$$

We dispose then three UFF files related to the system, representing virtual measurement data. The FRF loading can be made, to get a matrix $\mathbf{H}$ containing the three FRF:

```
% FRF loading
[H11,freq,infoFRF(1)] = unv58read('3DL_H11.unv') ;
[H21,freq,infoFRF(2)] = unv58read('3DL_H21.unv') ;
[H31,freq,infoFRF(3)] = unv58read('3DL_H31.unv') ;
H = [H11,H21,H31] ;
infoFRF2 = infoFRF ;
infoFRF3 = infoFRF ;
```

The matrix manipulation that *MatLab* offers can be used to display the FRFs in various formats (Figure 5):

```
% Visualization on various layouts
figure
subplot(4,3,1)
plot(freq,20*log10(abs(H11)))
xlabel('Frequency  [Hz]'), ylabel('H_{11} [dB]')
subplot(4,3,2)
plot(freq,20*log10(abs(H21)))
xlabel('Frequency  [Hz]'), ylabel('H_{21} [dB]')
subplot(4,3,3)
plot(freq,20*log10(abs(H31)))
xlabel('Frequency  [Hz]'), ylabel('H_{31} [dB]')
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]')
subplot(4,3,7)
plot(freq,real(H11))
xlabel('Frequency  [Hz]'), ylabel('H_{11} [real]')
```

```
subplot(4,3,8)
plot(freq,real(H21))
xlabel('Frequency  [Hz]'), ylabel('H_{21} [real]')
subplot(4,3,9)
plot(freq,real(H31))
xlabel('Frequency  [Hz]'), ylabel('H_{31} [real]')
subplot(4,3,10)
plot(freq,imag(H11))
xlabel('Frequency  [Hz]'), ylabel('H_{11} [imag]')
subplot(4,3,11)
plot(freq,imag(H21))
xlabel('Frequency  [Hz]'), ylabel('H_{21} [imag]')
subplot(4,3,12)
plot(freq,imag(H31))
xlabel('Frequency  [Hz]'), ylabel('H_{31} [imag]')
pause
```

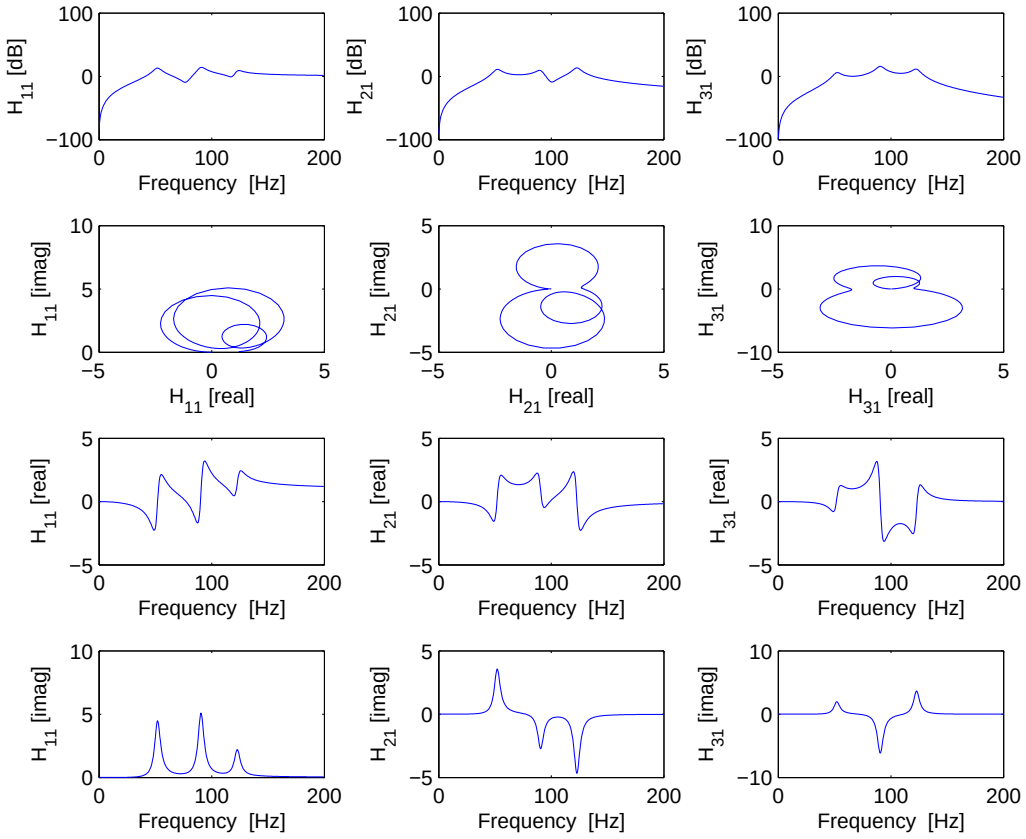


Figure 5: FRFs in various formats

Before using an identification method, it is often interesting to use mode indicators, for checking the local frequency range to analyse (Figure 6):

```
[indicators] = ind_mode(H) ;
[cursor1,cursor2] = plot_ind_mode(indicators,freq) ;

figure
subplot(3,1,1)
plot(freq,20*log10(abs(indicators.ISUM)))
xlabel('Frequency  [Hz]')
ylabel('Indicator I_{SUM} [dB]')
subplot(3,1,2)
plot(freq,20*log10(abs(indicators.ISRe)))
xlabel('Frequency  [Hz]')
ylabel('Indicator I_{S,Re} [dB]')
subplot(3,1,3)
plot(freq,20*log10(abs(indicators.ISIm)))
xlabel('Frequency  [Hz]')
ylabel('Indicator I_{S,Im} [dB]')
```

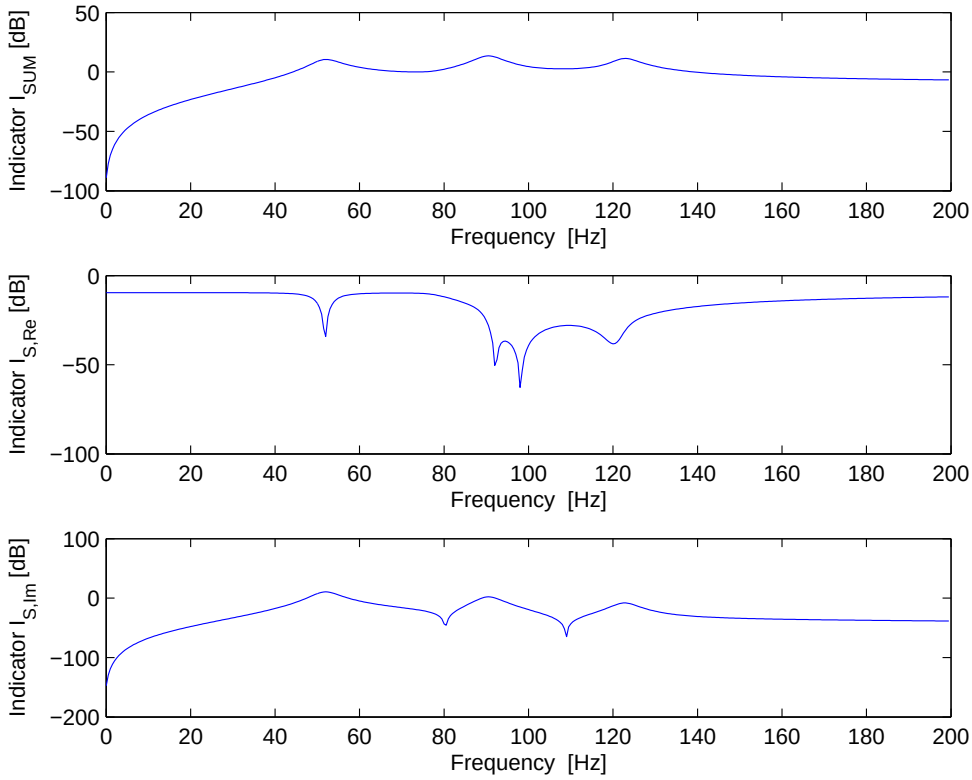


Figure 6: Mode indicators

The identification step can begin, with the circle-fit method (Figure 7):

```
% Circle-fit
N = size(H,2) ;
bornes_min = [40 75 110] ;
```

```
bornes_max = [60 100 130] ;
Nbr_mode = length(bornes_min) ;
for i=1:N
    for j=1:Nbr_mode
        [freq_local,H_local,H_gen_local,infoMODE,circ_prop] = ...
            circle_fit(H(:,i),freq,bornes_min(j),bornes_max(j));
        plot_circle_fit(freq_local,H_local,H_gen_local,infoMODE,circ_prop);
        infoFRF = add_data...
        (i,infoMODE.frequencyk,infoMODE.etak,infoMODE.Bijk,infoFRF,j);
    end
end

% Results saving
infoMODE1 = save_result_modal(infoFRF);
unv55write(infoMODE1,'3DL_circle_fit.unv',1);
```

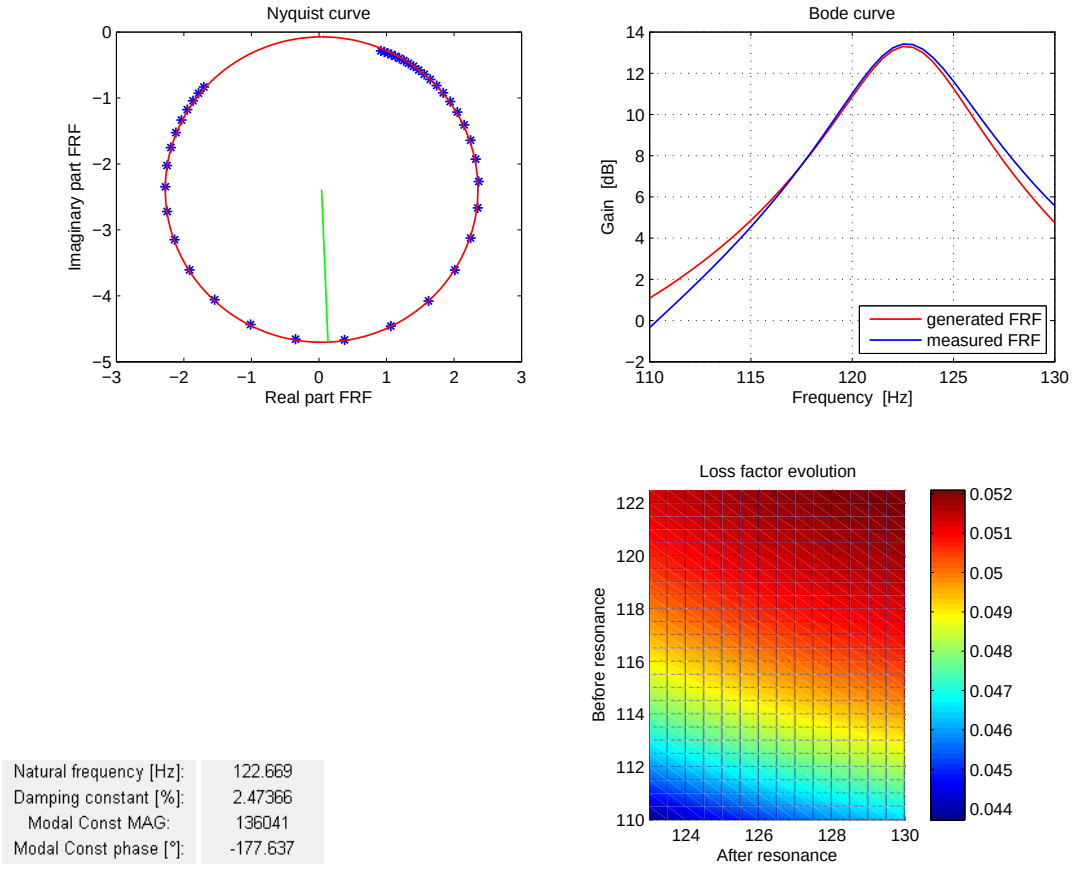


Figure 7: Example of information provided by the *circle–fit* method

or the line-fit method (Figure 8):

```
% Line-fit
```

```
N=size(H,2);
bornes_min = [40 75 110];
bornes_max = [60 100 130];
Nbr_mode = length(bornes_min);
for i=1:N
    for j=1:Nbr_mode
        [freq_local,H_local,H_gen_local,infoMODE,circ_prop] = ...
            line_fit(H(:,i),freq,bornes_min(j),bornes_max(j));
        plot_line_fit(freq_local,H_local,H_gen_local,infoMODE,circ_prop);
        infoFRF2 = add_data...
        (i,infoMODE.frequencyk,infoMODE.etak,infoMODE.Bijk,infoFRF2,j);
    end
end

% Results saving
infoMODE2 = save_result_modal(infoFRF2);
unv55write(infoMODE2,'3DL_line_fit.unv',1);
```

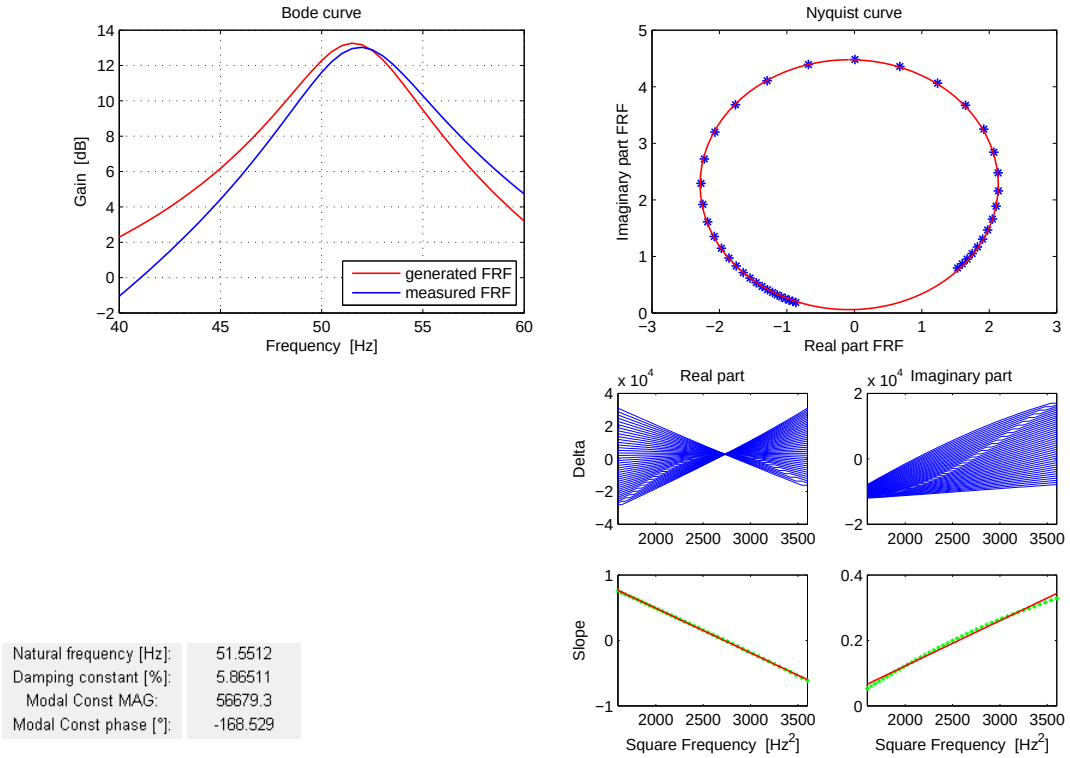


Figure 8: Example of information provided by the *line–fit* method

or the LSCE method which needs a more compact script (Figure 9):

```
% Least-square complex exponential
[RES,infoFRF3,infoMODE3]=lsce(H,freq,infoFRF3);
```

```
% Results saving
unv55write(infoMODE3,'3DL_LSCE.unv',1)
```

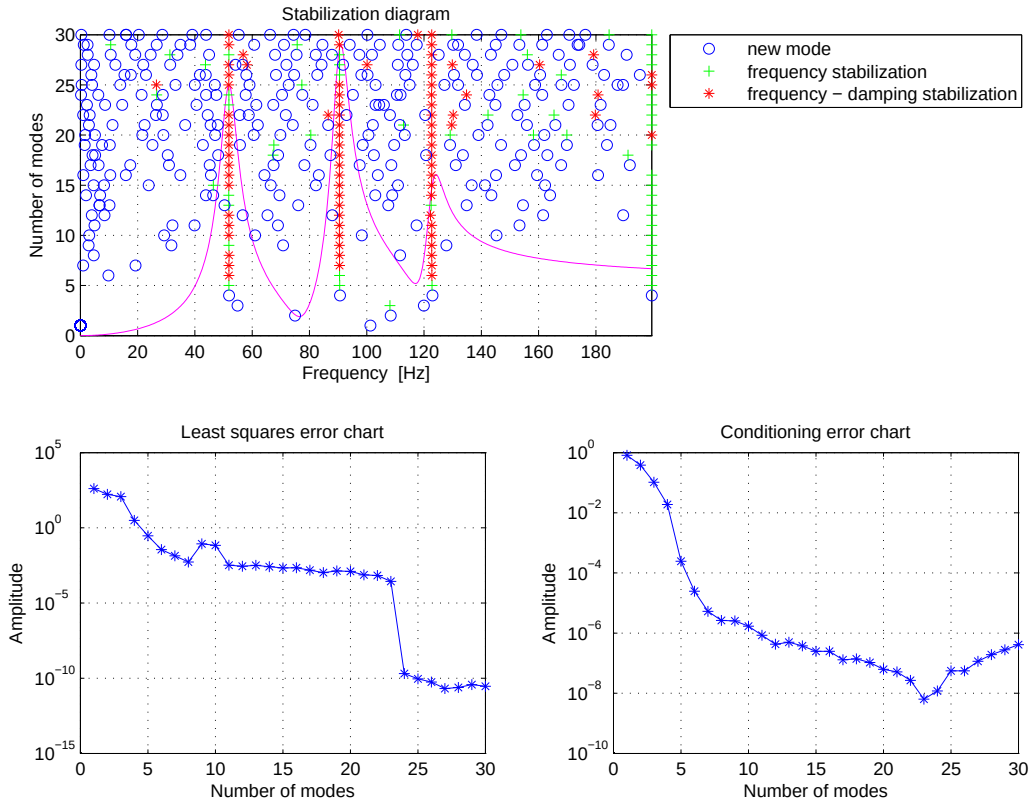


Figure 9: Example of information provided by the *LSCE* method

The visualization can be done in *MatLab*, when the structure geometry is simple (Figure 10), or with the help of EasyAnim (Figure 11)

```
% Geometry definition and saving
Nodes = [1 0 0 1 ; 2 0 0 2 ; 3 0 0 3];
Connexions = [1 2 3];
unv15and82write(Nodes,Connexions,'3DL_geometry.unv',8,8);
pause

% Mode shapes visualization
figure
Methode1 = [real(infoMODE1.Bijk(3,:));real(infoMODE1.Bijk(6,:));...
   real(infoMODE1.Bijk(9,:))];
Methode2 = [real(infoMODE2.Bijk(3,:));real(infoMODE2.Bijk(6,:));...
   real(infoMODE2.Bijk(9,:))];
Methode3 = [real(infoMODE3.Bijk(3,:));real(infoMODE3.Bijk(6,:));...
   real(infoMODE3.Bijk(9,:))];
subplot(3,1,1)
```

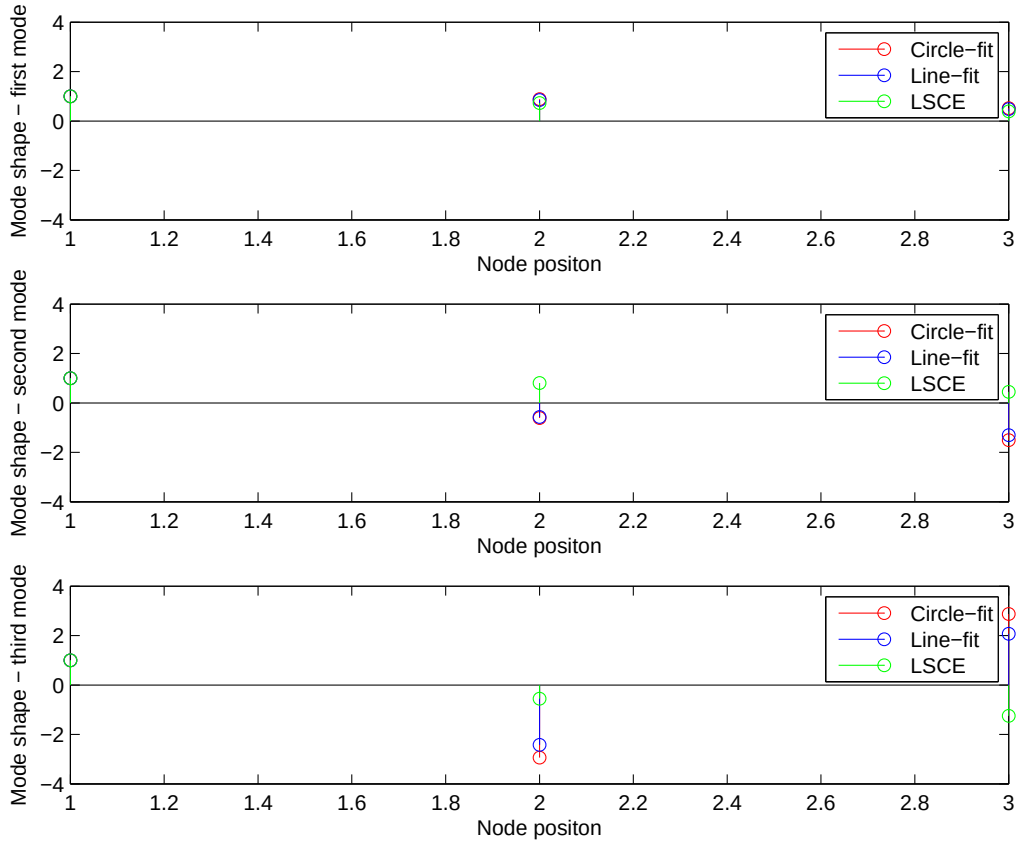


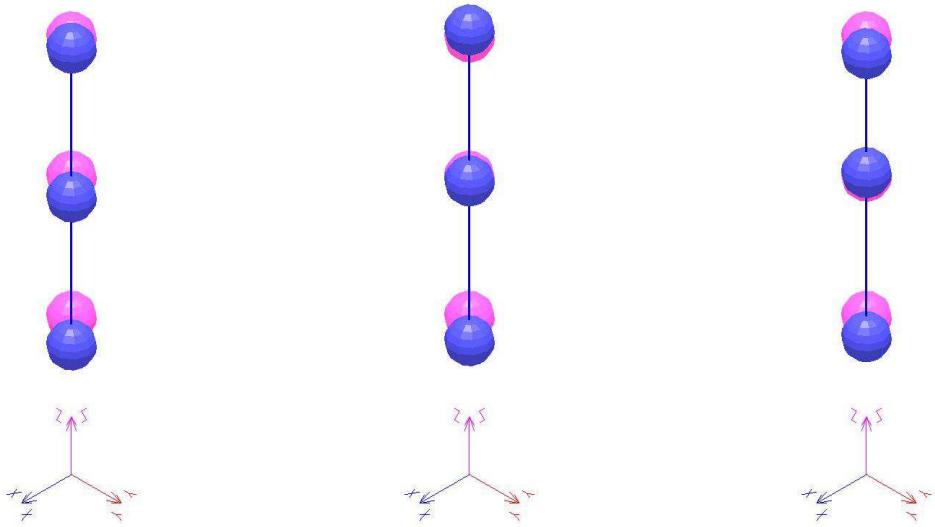
Figure 10: Mode shape visualization in *MatLab*

```
h1 = stem(Nodes(:,2),Methode1(:,1),'r');
hold on
h2 = stem(Nodes(:,2),Methode2(:,1),'b');
hold on
h3 = stem(Nodes(:,2),Methode3(:,1),'g');
legend([h1(1);h2(1);h3(1)],'Circle-fit','Line-fit','LSCE')
xlabel('Node positon')
ylabel('Mode shape - first mode')
axis([1 3 -4 4])

subplot(3,1,2)
h1 = stem(Nodes(:,2),Methode1(:,2),'r');
hold on
h2 = stem(Nodes(:,2),Methode2(:,2),'b');
hold on
h3 = stem(Nodes(:,2),Methode3(:,2),'g');
legend([h1(1);h2(1);h3(1)],'Circle-fit','Line-fit','LSCE')
xlabel('Node positon')
ylabel('Mode shape - second mode')
```

```
axis([1 3 -4 4])

subplot(3,1,3)
h1 = stem(Nodes(:,2),Methode1(:,3),'r');
hold on
h2 = stem(Nodes(:,2),Methode2(:,3),'b');
hold on
h3 = stem(Nodes(:,2),Methode3(:,3),'g');
legend([h1(1);h2(1);h3(1)],'Circle-fit','Line-fit','LSCE')
xlabel('Node positon')
ylabel('Mode shape - third mode')
axis([1 3 -4 4])
```



(a) First mode (b) Second mode (c) Third mode

Figure 11: Mode shape visualization in EasyAnim

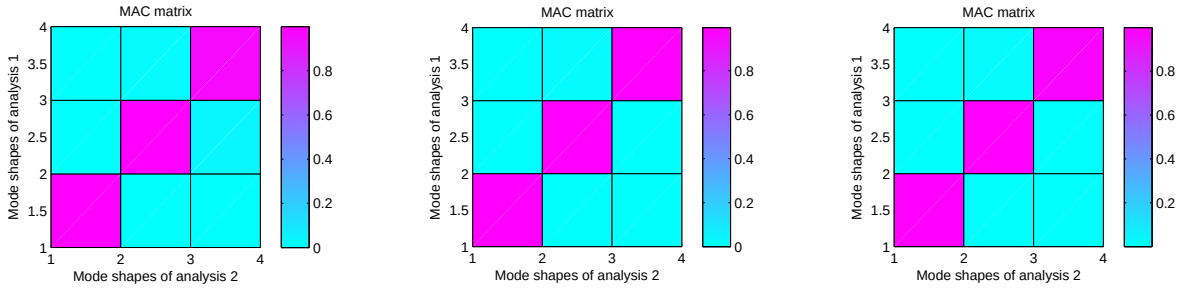
Finally, a comparison between the methods using the MAC criterion is given in Figure 12, obtained with the code:

```
% Modal assurance criterion between the identification methods
[DELTA_FREQ,MAC,coupled] = mac(infoMODE1,infoMODE2,0.8,0.1,1) ;

[DELTA_FREQ,MAC,coupled] = mac(infoMODE1,infoMODE3,0.8,0.1,1) ;

[DELTA_FREQ,MAC,coupled] = mac(infoMODE2,infoMODE3,0.8,0.1,1) ;
```

as well as the collinearity property of the mode shapes:



(a) Between *circle-fit* and *line-fit*

(b) Between *circle-fit* and *LSCE*

(c) Between *line-fit* and *LSCE*

Figure 12: *MAC* analysis of the three methods

```
% Mode collinearity
modegauss(infoMODE1) ;
modegauss(infoMODE2) ;
modegauss(infoMODE3) ;
```

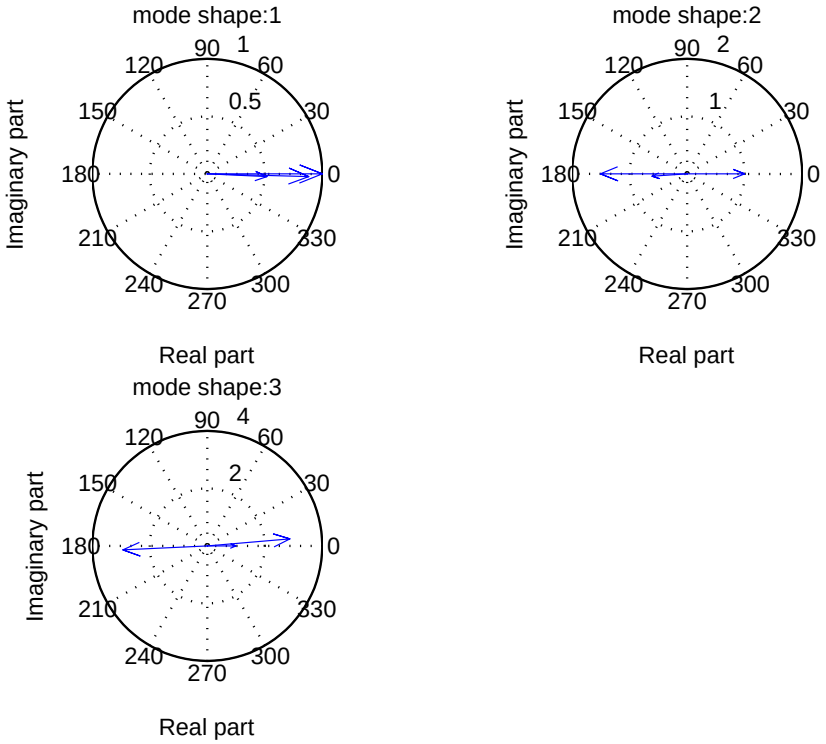


Figure 13: Mode collinearity visualization in *MatLab*

Through this simple structure, an overview of EasyMod functionalities is illustrated. If you want to try with an other structure, you can follow this example, and with measurements files (some examples are given in our website — a bike frame, an alpine ski or a squash racket).