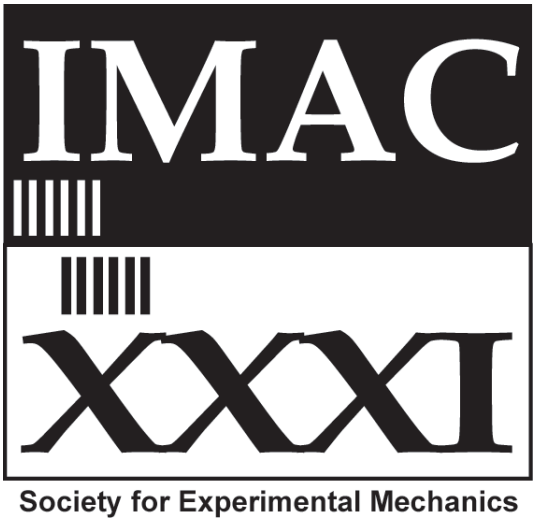
IMAC-XXXI
Conference and Exposition on Structural Dynamics

Engineering Nonlinearities in Structural Dynamics

Hyatt Regency Orange County
Garden Grove, CA USA

February 11 - 14, 2013



IMAC is organized by the
Society for Experimental Mechanics, Inc.
7 School Street, Bethel, CT 06801 USA
(203)790-6373 • (203)790-4472 • sem@sem1.com
http://sem.org

The ABRAVIBE Toolbox for Teaching Vibration Analysis and Structural Dynamics

Anders Brandt, Associate Professor

Inst. of Technology and Innovation, University of Southern Denmark, Campusvej 55, DK-5230 Odense M, Denmark

ABSTRACT

Vibration analysis is a subject where students often find it hard to comprehend the fundamental theory. The fact that we have, in general, almost no intuition for dynamic phenomena, means that students need to explore various dynamic phenomena in order to grasp the subject. For this reason, a MATLAB toolbox (the ABRAVIBE toolbox) has been developed as an accompanying toolbox for the recent book "Noise and Vibration Analysis" by the author. This free, open software, published under GNU Public License, can be used with GNU Octave, if an entirely free software platform is wanted, with a few functional limitations. The toolbox includes functionality for simulation of mechanical models as well as advanced analysis such as time series analysis, spectral analysis, frequency response and correlation function estimation, modal parameter extraction, and rotating machinery analysis (order tracking). In this paper, an overview of the functionality is given and recommended use in teaching is discussed. It is also shown how the toolbox can be used for general vibration analysis using data from multichannel measurements. Finally, some laboratory exercises for structural dynamics teaching are discussed.

Keywords: Vibration analysis, experimental modal analysis, order tracking, spectral analysis, ABRAVIBE toolbox

INTRODUCTION

This paper describes a toolbox for MATLAB and GNU Octave which was developed as an accompanying toolbox for the book "Noise and Vibration Analysis – Signal Analysis and Experimental Procedures", [1]. The purposes of the toolbox are at least threefold:

1. to aid teachers in setting up realistic and illustrative examples of the many intriguing things in mechanical vibrations in general, and experimental vibration analysis, in particular,
2. to aid the student's understanding of mechanical vibrations and analysis of it, by being able to go through each step in calculations etc., in an open fashion, and finally
3. to be a tool for students, researchers, and engineers in industry, to use for analysis of vibrations in an open software environment. For this purpose, the toolbox includes the same functionality as a typical high-end commercial software system (except the data acquisition part), and many extra functions not usually available in commercial systems.

All these purposes are supported by many examples supplied with the toolbox, ordered into separate folders for each chapter of the book. The principle of the toolbox is to allow transparency for the student/user into all the steps in the analysis, so that every single step can be investigated to ensure understanding. The toolbox thus contains high-level commands for standard tasks needed in this field. And each and every function is open, so that the student can open it and investigate it.

All functionality can start with recorded or simulated time data, the latter being found by the author to often being easier to comprehend by students, if they themselves can go through all the typical steps of analysis such as converting data to spectra, then perhaps to frequency response functions, and thereafter for example extracting operating deflections shapes for animation. A teacher using the ABRAVIBE [2] toolbox can very easily set up examples to illustrate various aspects of vibration. In assignments, the teacher can also set up example scripts at a level so that the students can accomplish requested results in a reasonable time, for example in time limited lab assignments. Undergraduate students can thus get more "canned" demonstrations, whereas graduate students can be asked to develop more of the tasks themselves.

As an overview, the ABRAVIBE includes, among other things, functionality to

- store data with header information in a standardized format, which allows for easy implementation of, for example, operating deflection shape analysis and experimental modal analysis,
- import and export data in universal file format, to allow import from and export to most commercial measurement systems in this field,
- generate data in the form of frequency response functions (FRFs), or modal parameters, from known mechanical systems described by mass and stiffness matrices and either damping matrices, or modal damping,
- generate simulated time data for the forced response of mechanical systems, which can be used for understanding mechanical vibrations, as well as for investigating signal analysis techniques on data with known parameters,
- define signal analysis operations such as filtering, acoustic analysis (1/n octaves, sound level meter integration), etc.,
- compute statistical functions such as probability density functions, skewness and kurtosis, frame statistics and hypothesis tests for stationarity tests, and data quality assessment,
- estimate spectra of time signals by linear (rms) spectra, spectral densities, or transient spectra (energy spectral density), with time windowing, averaging etc. by the same algorithms implemented in commercial software, and some more sophisticated methods not yet available commercially,
- estimate frequency response functions and coherence functions, either from impact testing (using the enhanced method described in [1, 3]), or from shaker testing, with single-input as well as multiple-inputs,
- perform order tracking functions such as rpm maps, synchronous resampling, order maps etc.
- extract modal parameters using well-known modal analysis methods, simple SDOF as well as the time domain polyreference MDOF method, and
- animation of operating deflection shapes and modal analysis results.

All theory that follows is, of course, described in [1]. This will not be referenced but implicitly assumed. In some cases we will, however, give references to background literature.

ABRAVIBE TOOLBOX FUNCTIONALITY

To simplify the notation below, all toolbox command variables will be written in **bold** letters in the general text, whereas MATLAB code examples will be in `Courier font.`

Data Storage Format

The basis for much of the functionality of the toolbox is the way data is stored. First, data from measurements can be stored in a standardized data format which contains information about measurement DOFs etc., to facilitate easy implementation of, for example, experimental modal analysis. Data can be imported from measurement systems in the universal file format, or by writing a function that stores the data in the ABRAVIBE file format. The principles of this data storage format are that

- Each function is stored in a separate file, to allow processing of as much data as possible without memory limitations. A function can for example be a time history, a spectrum, or a frequency response function (FRF).
- Each file consists of two variables: **Data**, containing the function data in a column, and **Header**, which is a structure containing flexible header information, which means that only as much information as needed must be included. New header fields can be added when needed.

- There are high-level commands for reading data into the toolbox for analysis. An example of such a command, is the **data2hmtrx**, which converts single files of FRF data into one FRF matrix and a number of variables with information about the DOFs measured. An example of this function is shown in Section "Experimental Modal Analysis".

Another important functionality is to be able to store data into matrices containing multiple-input/multiple-output (MIMO) data. For this purpose, 3-dimensional matrices are used in a standardized fashion, **H(f,d,r)**, where

- The first index, **f**, is frequency
- The second index, **d**, is a response location
- The third index, **r**, is a reference location

This means, that if, for example a structure is measured in 35 response points, using two shaker positions, the resulting FRF matrix, **H**, if analyzed with 1025 frequencies, would be a 1025-by-35-by-2 matrix. Information about which DOF is located at which address in this matrix can be stored in separate variables.

Mechanical System Simulation

Teaching vibrations and structural dynamics often starts with the single-degree-of-freedom (SDOF) system. This system is analyzed for steady-state harmonic response, transient response, etc. It is then shown that multiple-degree-of-freedom (MDOF) systems behave as can be referred to as an extended case of the SDOF system. In the ABRAVIBE toolbox, to aid the understanding of this, the frequency response (FRF) or impulse response (IRF) of a mechanical system with any number of DOFs (within reasonably limits) can be computed with high-level commands. Thus, if the system matrices [*M*], [*C*], and [*K*] are all known, the FRF between a number of input (force) positions, and a number of response locations, can be computed by the command **H=mck2frf(f,M,C,K,indof,outdof,type)**, where **indof** is a vector with each input DOF, and **outdof** is a vector of the output DOFs, and **type** is a variable to produce the FRFs in the form of receptance, mobility, or accelerance. With this single command, an entire FRF matrix, or a subset of it, can thus be computed in one call. Similarly, if only the mass and stiffness matrices are known, modal (viscous) damping can instead be added, and the command **H=mkz2frf(f,M,K,z,indof,outdof,type)**, where **z** now is a vector with the modal damping of each mode.

Next in teaching mechanical vibrations, is often a description of (analytical) modal analysis, where modal parameters (natural frequencies, damping ratios, and mode shapes) are computed from the system matrices. Furthermore, the analysis of the modal solutions are often divided into undamped, proportionally damped, and generally damped systems. In the toolbox, this is supported by the command **mck2modal**. This is a complex command, which can be used in various ways, reflecting the form of damping. Thus, called with only mass and stiffness matrices, it computes the eigenfrequencies and normal modes by the syntax **[fr,V]=mck2modal(M,K).** If the damping matrix is known, the command instead uses the syntax **[p,V]=mck2modal(M,C,K)** giving the complex poles, **p**, and either the real-valued normal modes if the damping is proportional, or the complex-valued mode shapes using the state-space system formulation, if the damping in non-proportional. This is done for pedagogical reasons, of course, since the state-space formulation could indeed be used in both the latter cases. Finally, for completion, of course the toolbox also contains commands to convert from modal parameters to FRFs. Also, there is, of course, commands to convert from different modal scaling principles, particularly unity modal mass, and unity modal A [1].

Time Domain Forced Response

A crucial part of teaching vibrations, is to illustrate the transient versus harmonic forced response, as well as the response to random loads. For vibration analysis, it is many times very important to be able to check an algorithm or method on data with known parameters. For both these purposes, the time domain forced response algorithm implemented in the toolbox is very important. The algorithm is based on a ramp-invariant method of designing digital filters described in [4, 5] which has some very important advantages

1. It is (immensely) faster and much more accurate than standard methods such as Runge-Kutta variants [4],
2. It uses a modal superposition formulation, which means it can use either mass, damping, and stiffness matrices, or mass and stiffness matrices and modal damping, or modal parameters, as input, which makes it very flexible.

The syntax of the command, which is called **timefresp**, is dependent on which of the input parameters are known, but an example is **y=timefresp(x,fs,M,C,K,indof,outdof,OutType)**, where **x** is the force time history or time histories if more than

one force, and **indof** and **outdof** are vectors, allowing all requested information to be computed in one call to the command. To illustrate the use, let us define a mechanical SDOF system with natural frequency f_r=100 Hz, damping ζ_r=0.05, excited by a half sine force, in [N],

$$F(t) = \begin{cases} 100\sin\left(\frac{\pi t}{T}\right), 0 \le t \le T \\ 0, t > T \end{cases} \quad (1)$$

where the pulse time T = 11 ms. The code to generate the output is found in Example 1 and the result plotted in Fig 1. A more advanced illustration of the use is found in Section "Frequency Response Estimation".

```
wn=2*pi*100; z=0.05;                    % Natural frequency in [rad/s] and damping ratio
m=1; k=m*wn^2; c=2*z*sqrt(m*k);         % mass, stiffness, and (viscous) damping
T=11e-3;                                % Pulse time
fs=1e4;                                 % Sampling frequency in Hz
t=(0:1/fs:.2)';                         % Time axis in column
F=makepulse(length(t),fs,T,'halfsine');
F=100*F/max(F);                         % Scaled force
u=timefresp(F,fs,m,c,k,1,1,'d');        % Transient response in displacement [m]
```

Example 1. Code to generate time forced response of a SDOF system to a transient (half sine) force signal. The result is plotted in Fig. 1.

Time Series Analysis

An important part of vibration analysis is signal analysis, so much more because most students taking a class in vibration analysis are mechanical or civil engineers, with lack of background in signal analysis. The ABRAVIBE toolbox thus contains commands for in-depth illustration of time series analysis. There are illustrations of convolution, as well as the sampling theorem, which we will not cover here for lack of space. Among more advanced functionality, are digital filter examples for acoustic 1/n octave filtering and A- and C-weighting. In addition, the toolbox includes an RMS (root mean square) integration algorithm which can be used to simulate a sound level meter (SLM), or for vibration comfort analysis, with the command **arms**.

Another important functionality is implemented for integration and differentiation of signals in the time domain. Unlike what is commonly thought, these two fundamental applications are not so readily implemented, and common well-known algorithms such as the trapezoidal rule taught in numerical analysis classes are not suitable for vibration analysis with high dynamic range. Instead, the ABRAVIBE toolbox contains state-of-the art digital filter methods for both integration and differentiation with very high accuracy. Finally we should mention the function **psd2time** which can be used to produce Gaussian noise with a specified spectral density.

Statistics and Data Quality Assessment

Statistical functions are used frequently in vibration analysis, particularly when dealing with field measurement data. There is thus functionality for assessing the statistical properties of signals, testing for stationarity, and for quality assessment of measured signals. The latter will be taken as an example here, as it includes most of the other functionality.

Data quality analysis is essentially based on using a set of statistical measures such as RMS value, min- and max values, skewness, and kurtosis, and to compare these values with known values for signals of good quality, what we can call "normal" values. The analysis is typically done using two time scales; the entire time signal, and a shorter time interval. The entire time signal gives overall statistical values, which may indicate important errors. Shorter intervals, for example one-second intervals, may instead reveal time restricted errors due to intermittent errors resulting from, for example, cable issues, electrical spikes, etc.

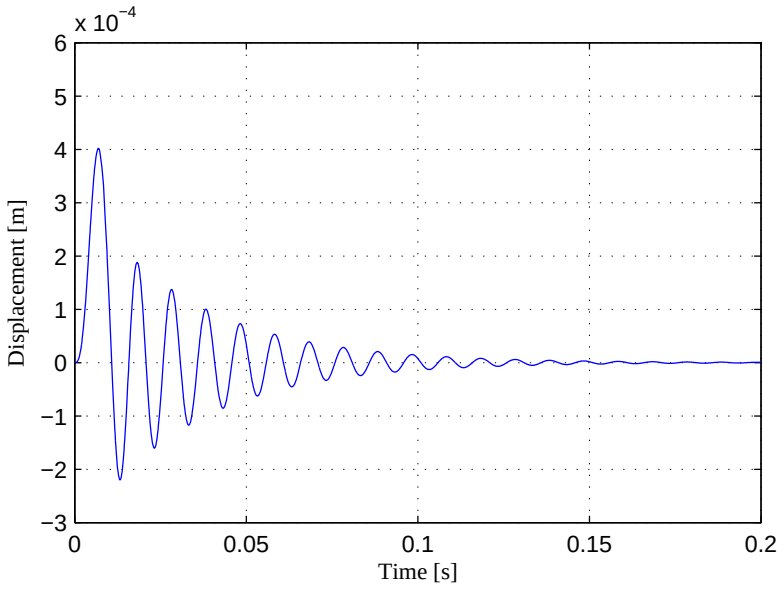


Fig 1. Plot for Example 1. SDOF response to a half-sine input force

To illustrate an example of a data quality assessment analysis, there are data from eight channels from a measurement on a truck supplied with the toolbox. In Example 2 the main code for data quality analysis of these data are shown. The first few lines show a structured way of assessing all files in the data directory, and then looping through all files, which is facilitated by the naming convention of the file names, where the data are stored in files with file names "Truck1.mat", "Truck2.mat",… "Truck8.mat" in this example. For each channel, the standard deviation is computed as a function of time, for 100 frames, and plotted and the reverse arrangements test is run for each channel. The results of the frame statistics plot are shown in Fig. 2. Next the high-level command **statchkf** is used to produce a list of most common statistical variables, and also to log data to two files; a text file called "TruckStats.log", and a MATLAB file "Truckstats.mat". The former is typed into MATLAB to produce the results in Table I. The latter is again opened and used to produce a plot of the kurtosis of all channels normalized to the kurtosis of channel 3, which is plotted in Fig. 3. The normalization to one of the channels is a good "trick", to more easily see if there is a discrepancy from what is "normal".

```
D='..\Data\TruckData\';
DirStruct=dir(strcat(D,'Truck*'));
NoChannels=length(DirStruct);              % Define number of channels
% First, run a frame statistics analysis of all channels, based on standard deviation
for n = 1:NoChannels
    FileName=strcat(D,DirStruct(n).name)   % Create file name 'Truck1.mat' etc.
    load(FileName)
    N=floor(length(Data)/100);             % Use 100 frames in total
    subplot(NoChannels/2,2,n)
    F(:,n)=framestat(Data,N,'std',1);      % This command produces the plot, see Fig. 2
    s(n)=teststat(F(:,n),0.02,'reverse');  % s contains Boolean variables true/false
end
% We then compute a reverse arrangements test on channel 3 only (for space reasons)
% Next, we run some standard statistics and log to a file
statchkf(Prefix,1,8,'TruckStats');         % Prefix is entire directory structure…
type TruckStats.log                        % This lists the results in MATLAB
% Next, we plot the overall kurtosis of all channels, from the analysis just made
load TruckStats                            % This file contains the variables used
Kurtosis=Kurtosis/Kurtosis(3)              % Normalize kurtosis to channel 3
bar(Kurtosis)                              % Results of this is shown in Fig. 3
```

Example 2. Data quality assessment of truck data. Some plot commands are omitted here for the sake of brevity.

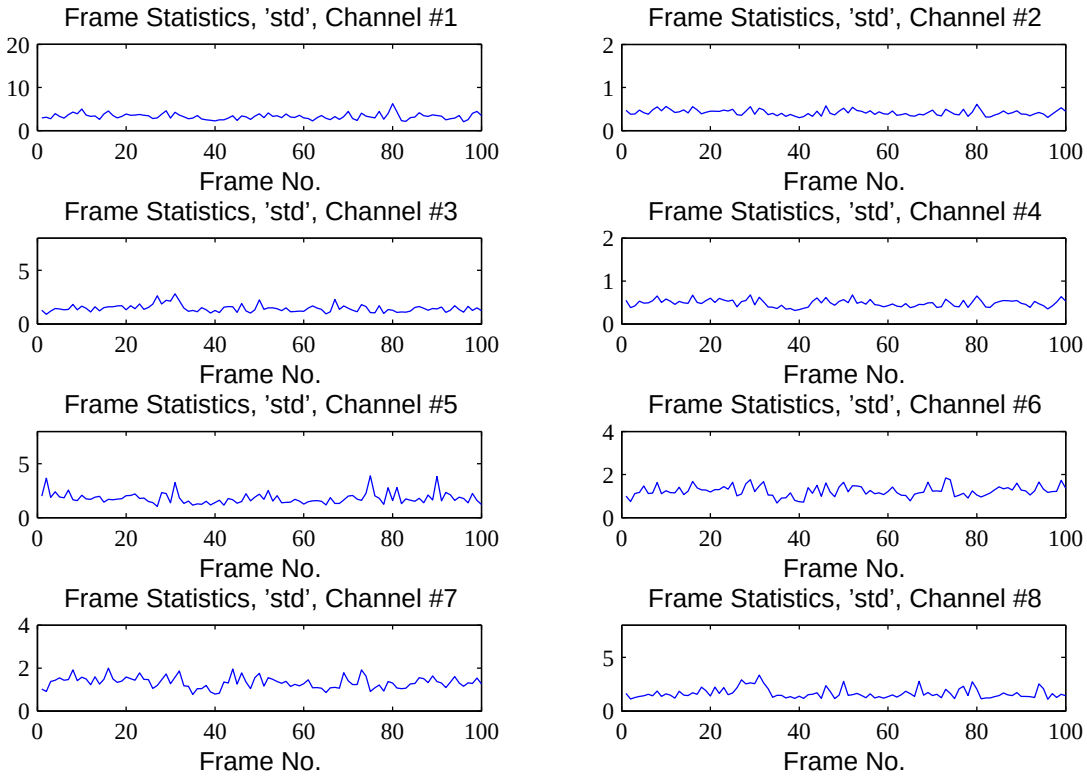


Fig. 2. Results of frame statistics in Example 2. The standard deviation of each channel is plotted versus frame number

File #	Max	Min	Mean	Crest	Std dev	Variance	Skewness	Kurtosis
1	51	-69	1.2	18	3.7	13	0.15	11
2	6.4	-7.2	-0.0035	17	0.42	0.18	0.05	7
3	17	-14	0.2	9.5	1.8	3.3	0.042	4.3
4	4.3	-4.8	-0.014	9.8	0.49	0.24	-0.021	5
5	25	-19	0.047	7.5	3.3	11	0.18	3
6	11	-7.2	0.027	8.4	1.3	1.6	0.7	6.1
7	9.9	-8.1	-0.51	6.8	1.4	1.8	0.25	4.6
8	18	-14	0.28	8	2.2	4.8	0.0062	4

Table I. Statistical parameters from output of the data quality test in Example 2

Spectral Analysis

Spectra are perhaps the most commonly used functions for vibration analysis. The ABRAVIBE toolbox therefore include a number of high-level commands to produce spectra of multi-channel recorded or synthesized data. Spectrum types include linear spectrum (also called RMS spectrum), and phase-spectrum, for periodic signals. For random signals, spectral densities can be computed using the very common Welch's method, but also using the smoothed periodogram method. For transients, the transient spectrum and the energy spectral density functions are available. All functions for random signals can also be computed for cross-spectral densities.

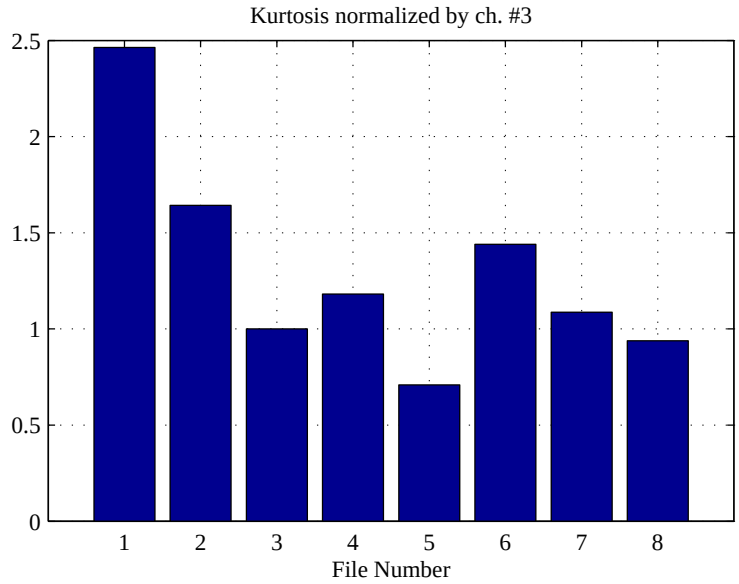


Fig. 3. Kurtosis of the each channels in Example 2, normalized to the kurtosis of channel 3. Assuming the kurtosis of channel 3 is good, the plot reveals suspicious kurtosis in at least channels 1, 2, 5, and 6

Welch's method for computing spectral densities is well-known and is the method implemented so far, in virtually all commercial software. It will be used in Example 3 below. Before introducing this example, however, a few words about the alternative to Welch, the *smoothed periodogram method*, should be mentioned. As described in [1, 6] this method has some advantages over Welch's method that can make it interesting:

1. It can be used with a logarithmic frequency axis, which gives lower random error at higher frequencies, where, usually, a lower frequency resolution is acceptable,
2. It is very practical in cases where unwanted, periodic components present in the noise are to be removed [6].

In any of these cases, the commands **apsdsp, and acsdsp** from the ABRAVIBE toolbox can be used. Both commands operate similarly to the Welch commands **apsdw**, and **acsdw**, which are going to be used in Example 3.

To illustrate the power of the ABRAVIBE toolbox for generating synthetic data, and then performing spectral analysis, we will generate data from a model of a Plexiglas plate [6]. This also illustrates the advanced synthesis models available in the toolbox, and the code for this pedagogic example, developed in the ABRAVIBE toolbox and also using the free open CALFEM [7] toolbox, is available in the ABRAVIBE toolbox. The example is used thoroughly throughout the ABRAVIBE toolbox, and is very similar to the IES plate described in [8]. It is a small, rectangular plate made of Plexiglas (PMMA), and with, essentially, to two first modes at very close natural frequencies. The advantages of using such a simple structure are many, for example

- The structure geometry is trivial, so the student can focus on the topic,
- The structure is easily measured with high quality either by impact testing or shaker testing, in relatively short time, which allows for high-quality experimental modal analysis to be incorporated,
- The plate can be modeled relatively simply using inexpensive shell elements [6],
- More than the first ten modes of the plate can be readily described with a mere 5-by-7 grid, either for EMA or for reduced mode shapes for synthesis.

In Example 3 we assume that we have access to the eigenfrequencies and the analytical (normal) mode shapes reduced to the experimental grid size of 5-by-7 DOFs as shown in Fig. 4. How to obtain such modes are described in [6] (and is, as mentioned, available in the toolbox). We will now show how to use this modal model, by adding some modal damping of 2% to each mode, and then compute time data corresponding to a two-input shaker test using pure random excitation. For this

purpose we call the command **timefresp** with the syntax using poles and mode shapes as input. We divide the example into two parts, Example 3a to create data, and Example 3b to produce the entire cross-spectral matrix.

```
% Part 1 – create data and store in files
indofs=[1 9];             % Force DOFs
outdofs=[1:35];           % Response DOFs
poles=fz2poles(fr,0.02);  % Add 2% damping to all eigenfrequencies, to produce complex poles
fs=round(3*max(fr));      % Set a suitable integer sampling frequency
N=100*1024;               % Use 100K samples
Forces=randn(N,length(indofs));       % Create 2 independent Gaussian forces
Header=makehead(1,Forces(:,1),1/fs);  % Create Nominal Header
Prefix='PlexiTimeData';
Data=Forces(:,1);         % Next few lines saves the first force
Header.Dof=indofs(1);
Header.Dir='Z+';
Header.Unit='N';
Header.Title='Simulation using Plexi FE model results, and 2 inputs in dofs 1,and 9';
FileName=strcat(Prefix,int2str(1));
save(FileName,'Data','Header');   % Save the first force in file PlexiTimeData1
Data=Forces(:,2);          % Next few lines saves the second force
Header.Dof=indofs(2);
Header.Dir='Z+';
Header.Unit='N';
Header.Title='Simulation using Plexi FE model results, and 2 inputs in dofs 1,and 9';
FileName=strcat(Prefix,int2str(2));
save(FileName,'Data','Header');   % Save the second force in file PlexiTimeData2
% Next, save all responses in following files
for n=1:length(GEOMETRY.node)
    Data=timefresp(Forces,fs,poles,V,indofs,n,'a');
    Header.Dof=outdofs(n);
    Header.Dir='Z+';
    Header.Unit='m/s^2';
    Header.Title='Simulation using Plexi FE model results, and 2 inputs in dofs 1,and 9';
    FileName=strcat(Prefix,int2str(n+2));
    save(FileName,'Data','Header');
end
```

Example 3a. Create synthesized time data from Plexiglas plate using a 5-by-7 grid. The example assumes that eigenfrequencies are in the vector **fr** and normal modes in variable **V**. At the time of writing, this takes approx. 14 seconds on the author's laptop. This includes double-differentiating all signals to acceleration.

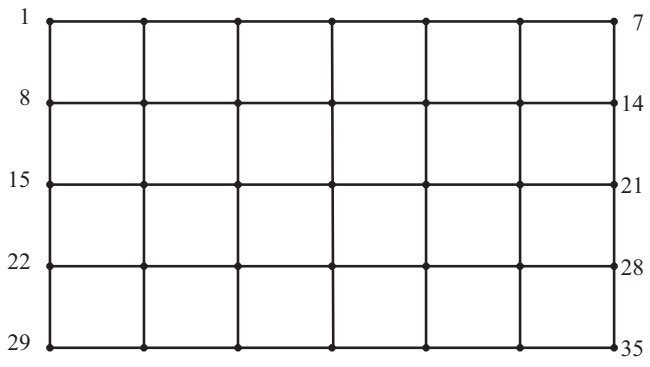


Fig. 4. Grid used for reduced mode shapes as well as for EMA of Plexiglas plate.

In Example 3b we will now first read the reference data (the force time signals) in, and compute the input autospectral matrix Gxx. Then we will loop through all reponse channels and compute cross-spectral densities between the two references and the response channel the 3D-matrix Gyx, and store the output autospectral densities in 2D-matrix Gyy. If we have R(=2) responses, and D(=35) responses, this analysis thus produces the matrices

- **Gxx** N/2+1-by-R-byR, input autospectral matrix
- **Gyx** N/2+1-by-D-by-R, input-output cross-spectral matrix
- **Gyy** N/2+1-by-D, output autospectral matrix (no need for cross-spectral densities between outputs)

```
NoRefs=2;          % Number of references (in first files)
NoResps=35;        % Number of responses (in consec. files)
Prefix='PlexiSynt2Forces';
% Put forces into columns in RefSignal
for n=1:NoRefs
    FileName=strcat(Prefix,int2str(n));
    load(FileName);
    RefSignal(:,n)=Data;
end
fs=1/Header.xIncrement;
% Loop through response channels
for n=1:NoResps
    FileName=strcat(Prefix,int2str(FileNo(n+2)));
    load(FileName);
    RespSignal(:,n)=Data;
end
% Compute all 2-in/35-out auto and cross-spectral densities
[Gxx,Gyx,Gyy,f]=time2xmtrx(RefSignal,RespSignal,fs,N);   % 50% overlap, Hanning window
```

Example 3b. Processing all time data from the 2-in/35-output synthesized "measurement" into auto and cross-spectral density matrices for MIMO spectrum analysis. On the author's computer, at the time of writing, this takes approx. 1 sec.

It should also be mentioned that there are a lot of details in the chapter examples about how to optimize the spectral analysis FFT settings, such as the blocksize, for minimizing bias error etc. It should also be mentioned that there is functionality for many more types of excitation signals, including impact testing, pseudo random noise and burst random. Example 3b is just that – an example. It should also be pointed out that the method described in Example 3, could also incorporate known error signals (extraneous noise) either on the input, or output signals, to analyze bias effects when estimating FRFs, for example. Numerous such examples are available among the chapter examples for chapters 13 and 14 in [1] which deal with FRF estimation.

Frequency Response Estimation

The next natural step after producing auto and cross-spectral density matrices described in the previous section, is to compute the MIMO frequency responses and multiple coherence functions. This can be done immediately after the steps in Example 3b, by using the command **[H,Cm,Cin]=xmtrx2frf(Gxx,Gyx,Gyy)** which computes all FRFs in the matrix **H**, size N/2+1-by-D-by-R, the multiple coherence matrix **Cm**, size N/2+1-by-D, and the input coherence matrix with ordinary coherence between all inputs, in Cin, size N/2+1-by-R-by-R.

There is also functionality in the toolbox for estimating FRFs from impact test data. This is done using an enhanced method based on recorded time signals including all impacts. The method is described in [3].

Experimental Modal Analysis (EMA)

There is functionality for a variety of EMA operations in the ABRAVIBE toolbox. The command **frf2msdof** includes several algorithms for obtaining modal parameters using SDOF approximations. The command **frf2ptime** uses MDOF estimation methods to estimates poles and sometimes modal participation factors (MPFs), using either Prony's method (for one FRF at the time, see [9]), the least squares time domain method (LSCE, see [10]), or by the polyreference time domain method [11]. Mode shapes can then be computed using the command **frfp2modes** which includes a multi-reference least squares

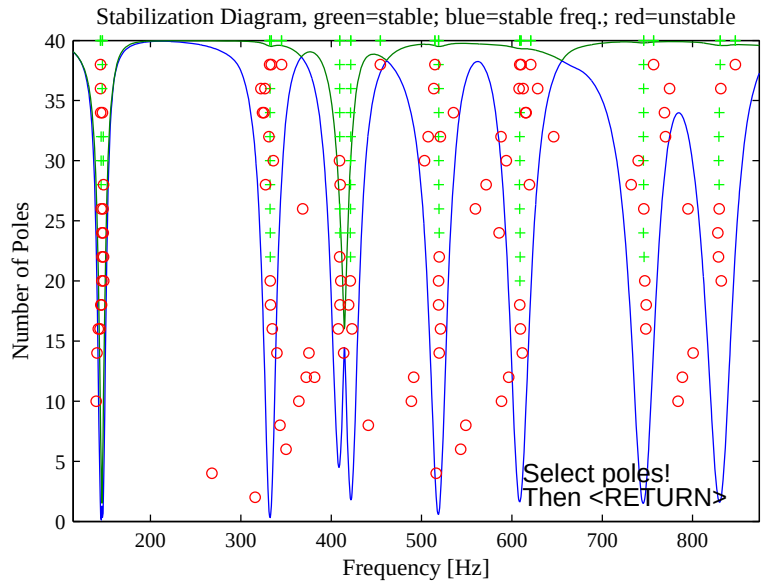


Fig. 5. Stabilization diagram of polyreference time domain estimation for Example 4

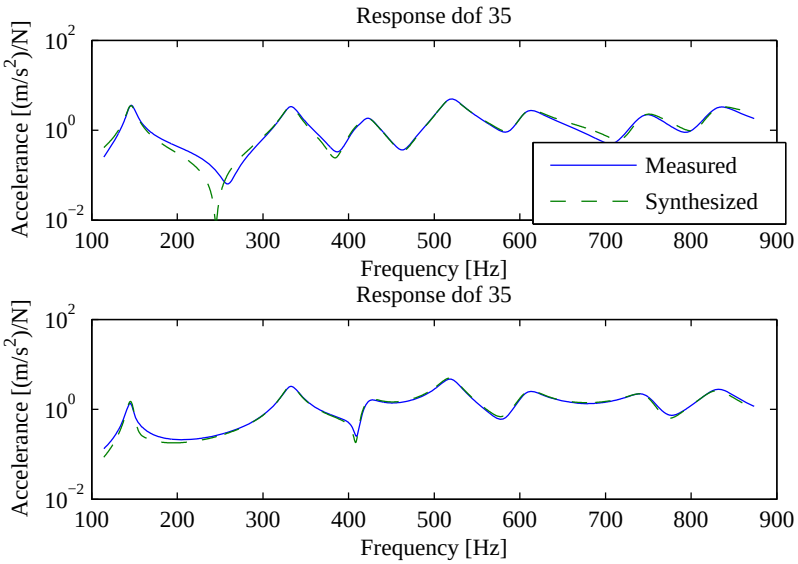


Fig. 6. Measured and synthesized FRFs from estimation of residues (mode shapes) for dof 35

frequency domain estimation of residues (mode shapes). Finally animation of the estimated mode shapes can be accomplished by using a GUI-based tool called **animate**. This functionality was supplied externally, see Acknowledgements.

To aid the analysis there is a high-level command that reads in all data and sorts it into variables, the command **data2hmtrx**. This command also rearranges data if coming from an impact test with roving impacts, so that it looks as if it comes from a shaker test to avoid having to deal with both matrix dimensions. There is also a command for mode indication functions, **amif**, which can produce various MIFs, eg. the normal MIF, and the multivariate MIF. There are also two commands to deal

with modal assurance criterion (MAC) matrices, **amac** to compute MAC matrices (either auto or cross), and plotmac to plot the matrix in Manhattan style display (see Fig. XX).

To illustrate an EMA example, we will use data from an impact test on the Plexiglas plate mentioned in previous sections. Of course, we could use the FRF matrix computed in Section "Frequency Response Estimation", however, that would not allow us to see how actual measurement data are used for EMA. In Example 4 there is thus some code illustrating a multiple-reference analysis using the polyreference time domain method. First the data are read in and sorted using the **data2hmtrx** command. Thereafter the poles and modal participation factors are computed and selected in a stabilization diagram shown in Fig. 5. After selecting the poles in this diagram, the residues are estimated using the least squares frequency domain method, using the poles and MPFs from the polyreference method. The command **frfp2modes** also plots each synthesized result as in Fig. 6, together with the corresponding measured function, if so is requested. After the mode shapes are thus computed, a MAC matrix is computed and plotted, as shown in Fig. 7. All that remains is to save the modal parameters to a file for subsequent animation, which will not be shown here.

```
Prefix='PlateH';
[H,f,Rdof,Rdir,Fdof,Fdir,FillMtrx]=data2hmtrx(Prefix,1,105);
[p,L,fLimits] = frf2ptime(H,f,400,20,'mvmif','ptd'); % Estimate poles and MPFs
idx=find(f>=fLimits(1) & f<=fLimits(2));
V=frfp2modes(H(idx,:,:),f(idx),p,L,0.5,Fdof);     % Estimate mode shapes
MAC=amac(V);                                      % Compute MAC matrix
plotmac(MAC);
```

Example 4. Experimental modal analysis (EMA) example using data from a 2-reference impact test on a Plexiglas plate.

Order Tracking

The final topic we shall discuss is analysis of rotating machinery using so-called order tracking. The ABRAVIBE toolbox includes some fundamental functionality for producing RPM-time profiles from a tacho signal. Using the RPM-time profile, RPM maps can then be computed and from those, order tracks can be extracted. There is also functionality for resampling the time signals and from the resampled signals extracting order maps and order tracks.

SUMMARY

In this paper we have presented a free toolbox for MATLAB or GNU Octave, the ABRAVIBE toolbox, and discussed the toolbox functionality for noise and vibration analysis, with emphasis on teaching vibration analysis and structural dynamics. It was pointed out how the transparency of the open software makes it possible for the student to study the steps involved in the analysis in detail by opening and inspecting the program routines. The flexible nature of the toolbox also enables teachers to customize examples based on the level of the students and the time at hand.

ACKNOWLEDGEMENTS

The author would like to express his gratitude to the original author, Dan Lazor, and Dave Brown, University of Cincinnati, for allowing incorporating and distributing the **animate** GUI-based animation software with the ABRAVIBE toolbox.

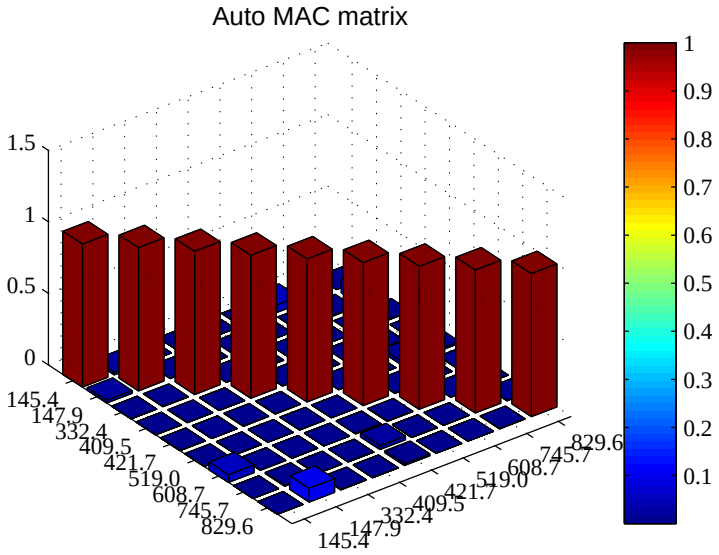


Fig. 7. MAC matrix of mode shapes from least squares frequency domain estimation using poles and modal participation factors from the polyreference time domain estimation for Example 4

REFERENCES

[1] Brandt, A., Noise and Vibration analysis - Signal Analysis and Experimental Procedures, John Wiley and Sons, 2011.

[2] A. Brandt, ABRAVIBE, A MATLAB/Octave toolbox for Noise and Vibration Analysis and Teaching, Revision 1.1, 2011. Available from http://www.mathworks.com/matlabcentral/linkexchange/.

[3] Brandt, A., Brincker, R., Impact Excitation Processing for Improved Frequency Response Quality, in: Proc. 28th International Modal Analysis Conference, Jacksonville, FL, 2010.

[4] Brandt, A., Ahlin, K., A digital filter method for forced response computation, in: Proc. 21st International Modal Analysis Conference, Kissimmee, FL, 2003.

[5] Ahlin, K., Magnevall, M., Josefsson, A., Simulation of forced response in linear and nonlinear mechanical systems using digital filters, in: Proc. International Conference on Noise and Vibration Engineering (ISMA), Catholic University, Leuven, Belgium, 2006.

[6] Brandt, A., Linderholt, A., A periodogram-based method for removing harmonics in operational modal analysis, in: Proceedings of the International Conference on Noise and Vibration Engineering (ISMA), 2012.

[7] Sturesson, P.-O., Brandt, A., Ristinmaa, M., Structural Dynamics Teaching Example – A Linear Test Analysis Case Using Open Software, Proc. 31st International Modal Analysis Conference (IMAC), Garden Grove, CA, 2013.

[8] Austrell, P.-E., Dahlblom, O., Lindemann, J., Olsson, A., Olsson, K.-G., Persson, K., Petersson, H., Ristinmaa, M., Sandberg, G., Wernberg P.-A., Calfem – A Finite Element Toolbox, Version 3.4, Studentlitteratur AB, 2004.

[9] Proakis, J. G. & Manolakis, D. G. Digital Signal Processing: Principles, Algorithms, and Applications Prentice Hall, 2006.

[10] Brown, D., Allemang, R., Zimmerman, R., Mergeay, M., Parameter Estimation Techniques for Modal Analysis, SAE Tech. Papers, 1979.

[11] Vold, H, Kundrat, J., Rocklin, T. G., Russell, R., A Multiple-Input Modal Estimation Algorithm for Mini-Computers, SAE Tech. Papers, 1982.